

Software Terminology as a Curse or Blessing: Possible Solutions for Terminological Problems

Matthias Vogel
Universität Halle-Wittenberg
Germany

1. Introduction

In his essay „Glanz und Elend der Übersetzung“ (1937), the Spanish philosopher Ortega y Gasset comments on general features of special languages and terminologies:

Eine Sprache ist ein System von Wortzeichen, mit dessen Hilfe die einzelnen Menschen sich ohne vorherige Vereinbarung verständigen können, während eine Terminologie nur verständlich ist, wenn derjenige, der sie schreibt oder spricht, und derjenige, der sie liest oder hört, sich zuvor und individuell über die Bedeutung der einzelnen Zeichen geeinigt haben. Deshalb nenne ich sie eine Pseudosprache [...] Sie ist ein Volapük, ein Esperanto, die durch eine besondere Übereinkunft unter denen, die ein bestimmtes Fach pflegen, festgelegt wurde.¹ (Ortega y Gasset 1996: 128f.)

Is this really true, we might ask ourselves? Is terminology research nothing but a kind of pseudo science? Is terminology work really as important for the quality of technical documentation and translation as it is often described? Is Ortega right in saying that languages for specific purposes are only just pseudo languages?

¹ In English, this quotation reads as follows: A language is a system of signs that people use to understand each other without previous agreement. In contrast, a specific terminology can only be understood if those who use it have commonly agreed on the meaning of the various signs. Consequently, I am calling terminology a pseudo-language [...], a kind of Volapük or Esperanto, which are artificial languages created by experts of a specific field.

This paper aims at giving an insight into the practical terminology work of the German software company Intershop and underlining the importance of terminology issues. The first chapter of this paper is dedicated to the main differences between software terminologies and “classical” kinds of specialized terminology (cf. chapter two). In chapter two, terminology work is described as a prerequisite for creating technical documents and translations (cf. chapter three and four). Following this, I will introduce a localization tool that Intershop developed to localize templates for Enfinity MultiSite, which is the core Intershop product. Finally, I will make a suggestion for a possible structure of a terminology database (cf. chapter five).

2. Software Terminology and Classical Terminologies

It is commonplace knowledge that languages for specific purposes are becoming increasingly important. In today’s global village, information is circulating ever faster. This applies to general as well as to specific pieces of information. The increasing number of subject-specific fields makes information transfer very difficult. To avoid communication difficulties between experts and non-experts, but also between experts, new terms must immediately be collected, defined, and made accessible. Terms mark off the numerous scientific subjects.

Software terminology, however, plays a special role. It has something in common with other special languages, but it also shows a number of differences.

As regards common features of software terminology and other special languages, these features aim at establishing a clear, precise and successful understanding. Theoretically, this goal can only be reached if one concept has only one designation, and one designation has only one meaning. The linguist Florian Coulmas (1992: 347) describes special languages using an optical metaphor. According to Coulmas, special languages are:

Erweiterungen in einem ähnlichen Sinn wie ein Mikroskop eine Erweiterung des Auges ist, Hilfsmittel also, mittels derer nicht nur Gegenstandsbereiche in größerem Detail durchdrungen werden, sondern auch neue Gegenstandsbereiche perzeptuell und begrifflich erschlossen werden.²

Let me now explain the differences between software terminology and other kinds of special languages. Software terminology, literally understood, is no special language like the languages of medicine, law, or biochemistry. However, the interface of a software program often contains elements of special languages. If you have developed a software program for insurance companies or the automobile industries, the program interface would certainly contain terms from these

² In English, this quotation reads as follows: Special languages are a kind of extensions similarly to a microscope being a widening of the human eye. Special languages not only help to look at things in great detail, but they also help to get an idea of how something is like.

industries (cf. Kemmann 2002). The main differences between software and “classical” terminologies are the following:

- Kind of interaction
Whereas classical terminologies contribute to a better understanding between peoples, software terminology facilitates communication between human beings and machines.
- Target groups
Special languages are addressed to specific target groups that are relatively small and homogeneous. Software terminology is different. Today everybody knows some sort of computer software. Consequently, the target group is fairly heterogeneous.
- Assumed knowledge
Using a special language requires a considerable amount of subject knowledge. Computer and software knowledge, however, has become part of the general knowledge and does not require specific training.

In addition to these differences, the term “software terminology” is fairly ambiguous. On the one hand, there is the actual software terminology, e. g. menus and buttons of a software program. On the other hand, there is the computer terminology, which basically is the language of information technology. On the edge of the field of computer terminology you can find the jargon of software developers. This jargon is a mixture of English and German called “Germish”. It is mainly used in new companies, the so-called start-ups. You can frequently meet this kind of “Germish” in oral communication, but also in e-mails, newsgroups, chat rooms and various other communication forms. “Germish” should always be avoided in any kind of documentation or technical translation. It does, however, often affect technical documents and make a writer’s life more difficult. Such bad words are, for example, verbs like *accepten*, *anpingen*, *committen*, *cutten*, *delivern*, *detecten*, but also complex verbs like *konferenzcallen*, *setuppen* or *webcammen*. This jargon is no language for specific purposes. Instead, it is a language mixture in which simple and complex verbs are adapted to the German grammar. This mixture is a kind of “pidginised” German with a reduced grammar, lexicon and style. Because software users often treat “Germish” and software terminology alike, “Germish” reduces the social prestige of the software terminology as a whole.

In their everyday work, technical writers need to have access to monolingual terms and their definitions. To make a software product linguistically and culturally appropriate to a foreign language, translators and localization specialists translate the necessary pieces of information into the target language.

Dictionaries cannot keep up with the fast development in the various fields. This is why the terminology must be worked on from the very beginning of a project so that terminology lists are ready before any documentation or localization work is

done. The process of creating, defining, collecting and storing terminology is referred to as terminology work (cf. Arntz/Picht/Mayer 2002). The amount of work dedicated to terminology depends on the kind of access to existing terminology lists.

A technical writer needs linguistic as well as subject-specific knowledge in order to create an unambiguous documentation or translation. In addition to that, a translator needs a specific knowledge which is called transfer competence. A translator must know the subject and its terminology in at least two languages. To acquire this knowledge, a translator would have to build up the terminological knowledge himself. Such work requires thorough investigations. If you work on a project, you just do not have the time to dedicate to terminology. The only real solution of this problem is to work on terminology issues systematically, that is to work on terminology from scratch. You must start with terminology work early enough – soon after the scheduling and budgeting phase – and keep an eye on the use of terminology until the product is released.

The following aspects are at the core of systematic terminology work:

- Terminology work is oriented towards the concepts of terms. Monolingual terminology work consists of defining concepts and finding suitable names for these concepts. Bilingual terminology work requires defining concepts in the target language. Concepts solely exist in the minds of writers and readers. Only by means of concepts we can think about real or imagined concepts. Working as a writer or translator, you must always remember the so-called Golden Rule: “Use one designation for one concept.” This designation should then be used in all documents of a document set.
- Terminology work is based on definitions. Terminology experts define the conceptual meaning of a term by providing a definition for this term. This task is often underestimated. Companies do not want to dedicate time and resources to the terminology setup. This is no peccadillo, especially if you know that about 75% of a translation process is used up by terminology research. Terminology work is increasingly becoming an economic factor. Terminological mistakes can lead to competitive disadvantages, miscalculations, and other financial losses (cf. Austermühl 2001: 89).

Consequently, definitions should be short, precise, and they should describe the main features of the *definiendum*. Definitions must be correct (conceptual meaning) and linguistically unambiguous (semantic meaning). In your practical work, you might find various kinds of definitions to meet your needs. Sometimes, various definitions are used within the same set of documents. Readers may then interpret these definitions quite differently although they are actually very similar. The

following examples show some definitions of the term “cartridge”.³ As far as the content is concerned, these definitions are very similar. The different wording, however, is a permanent source of irritation:

- The term “cartridge” refers to an installable software module that contributes a certain piece of functionality to an Enfinity MultiSite server. Cartridges provide a standard mechanism for packaging and deploying program code in order to make the functionality implemented by the code available on an Enfinity MultiSite server.
- Think of a cartridge as a software module that encapsulates specific business functionality. It contains all necessary components to execute that functionality within Enfinity MultiSite. After a cartridge is installed and registered with Enfinity MultiSite, it extends Enfinity MultiSite with its functionality seamlessly. This means that, as an Enfinity MultiSite user, you won't notice whether you are executing native Enfinity MultiSite functionality or an installed cartridge's functionality. That is because Enfinity MultiSite itself is built of several cartridges, storing its main functionality in the core cartridge.
- Cartridges are portable and flexible, because you can install and load them on the systems that need this functionality and you can develop additional functionality as needed, just reloading the updated cartridge after completion.
- A term used within Intershop to refer to a software module. Cartridges contain business and presentation logic and provide a well-defined set of functionality for an Intershop e-commerce application.
- An installable software module that provides additional functionality to an Enfinity system. The term "cartridge" refers to the mechanism that allows developers to package and deploy functional extensions of Enfinity, as well as to the functionality that is delivered with a cartridge.
- A cartridge is a software module that is used to deploy functionality to an Enfinity MultiSite system. It is a collection of objects such as pipelines, templates, images, static content, and java code. The standard functionality of Enifinity MultiSite is contained in cartridges. This functionality can be extended by deploying new cartridges which contain additional functionality.
- A cartridge (in the context of Intershop's software) is a piece of software that consists of java classes and methods, pipelets and pipelines, templates and accompanying data (e.g. images, online help files) It provides certain functionality which is either necessary for the product (i.e. the cartridge is part of the product) or it is added to a product/ installation to extend existing functionality.
- An installable software module that implements business logic into the Enfinity platform.

³ These definitions are taken from different documents of the Intershop Enfinity MultiSite documentation package.

Independent of how to define a new term, the results of research and development must be given a name and an explanation. We always need new terms to refer to new concepts. Only after we have provided concepts with a name, we can talk and write about concepts (documentation) or translate them (technical translation and localization).

3. Terminology Work as a Prerequisite for Technical Documentation

The documentation team in a software company has a great variety of tasks. One of the main tasks of a writer is to create user manuals for end users, online help systems, or developer guides (description of software modules and programming interfaces). But there are yet some more tasks a writer is responsible for, namely terminology management and software localization. In his work, a writer should always follow the established rhetorical and editorial principles, which are accurateness, brevity and terminological consistency. Every document thus becomes easier to read and understand. Akronyms like KISS (*Keep it short and simple*) help us remember this fact. Writing for the readers of a document is an essential part of what is called “usability”. Price und Korman (1993) describe the term “usability” as follows:

A rose is always a rose is always a rose:

- Describe the product accurately (the minimum requirement)
- Apply the same standards to all manuals in a series.
- Don't use ten different names for the same concept.
- Organize all sections in the same way, e.g., definition first, then example, then procedure.
- Apply the same formatting standards throughout the document.
- Beware of extreme gyrations of tone.

So far, a number of technical and formal guidelines have been created for technical documents. In contrast to this, there are just a few linguistic standards specifying the content and linguistic structure of documents. In English, such linguistic standards are called “controlled languages”. A controlled language contains a number of terms including concepts and names. In addition, it provides lists of words that should be avoided (anti-terms), synonyms, and grammatical rules.

If terminology work is ignored, this leads to confusion and misunderstanding because one term might then be used in the catalogue, another term in the user manual, and yet another in marketing papers. If you are lucky, the users will blame you for being naive and careless. However, you can run into judicial or safety problems, too. It is also possible that readers distrust your document if they recognize terminological inconsistencies. A consistent usage of terms improves the comprehensibility of texts as well as the linguistic consistency of texts from different authors. Readers should never be able to detect the author of a document. Besides, terminological consistency ensures the corporate identity by using a corporate language. Developers in a software company and other specialists neglect

the fact that corporate identity is little more than terminological work. Hence, it is very important to collect and categorize terms, their definitions, and provide them for the user.

The linguistic elements of a user interface (UI)⁴ often repeat themselves, especially in software documentation. This does not mean that terms are consistently used throughout the user interface. On the contrary, terminology is far away from being consistent and non-ambiguous. This is because developers define new terms instead of terminology experts. There are no fixed processes how to check and harmonize terminology. Some typical terminological inconsistencies are displayed in Table 1:

Word	Usage
ABORT	Use “end” or quit instead.
Appears	Use “The system displays” instead.
Application	Use “program” in end-user documentation.
Check	When referring to a checkbox, use “select” or “clear” instead. Example: Select the checkbox to activate it.
Checkbox	One word.
Choose	Use “select” instead.
Click	Use with button. Example: Click the Save button. Use “press” only for keys on the keyboard. Do not use “click on”. Use “select” for links.
Drop-down list	Use instead of pop-up menu or pull-down menu. See also “list box”.
Enter	Use “enter” when referring to a user entering information on the screen. Example: Enter the date in the “Date” field.
Execute	Use “run” when referring to a user running a computer program. Use “execute” when the system executes a command as a reaction to a users doing. Example: You can run the Properties Update program after you modified the property settings. Example: After you set the xy properties, the system executes the Properties Update program automatically.
Graphic	Use “Figure” when referring to a graphic in a document.

⁴ All elements of a software application used to interact with the user, such as dialog boxes, menus, and messages.

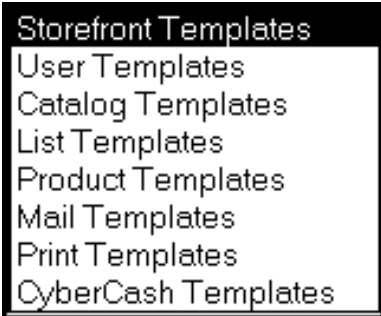
List box	<p>Two words. Used to describe a box that contains selection options that are NOT in a drop-down list. Example:</p> 
Log in	<p>Use “log on” instead. (The GUI overrides this suggestion.)</p>
Screen	<p>Use only when referring to the whole monitor. Otherwise use “window.” Example: If the system displays a window asking for a license key... Example: If you maximize your application window, the screen can only show this one window.</p>
Window	<p>Refers to what is displayed in a browser’s window. Example: If the system displays a window asking for a license key... See also “screen.”</p>

Table 1: Extract from the Intershop Style Guide

As you can see from the examples, terminology work must aim at excluding multiple designations for the same concept. This requires defining terms in advance (before a document is created) and adding them to the corporate style guide.

Without terminological guidelines, it is virtually impossible to use the same terms in the program menu, the online help, and the user manual. A corporate style guide should, therefore, solve lexical as well as syntactic and stylistic problems. Besides, a style guide should provide structural recommendations (templates, page layout), explain the documentations process, and contain checklists for technical writers and editors.

Terminology does, however, not only play a role in documentation issues, but also in the localization process. Localization involves „taking a product and making it linguistically and culturally appropriate to the target locale (country/region and language) where it will be used and sold” (Esselink 2000: 3). The next chapter is about a specific aspect of localization, namely template localization.

4. Terminology Work as a Prerequisite for Software Localization

The term *internationalization* indicates measures that need to be taken before you can localize the product. It is „the process of generalizing a product so that it can handle multiple languages and cultural conventions without the need for re-design“ (Esselink: 2). This involves checking the grammar and orthography of texts before they get translated, as well as processing cultural conventions (e. g., date and currency). The English translation *technical enabling* for the German term *Internationalisierung* makes it clear that internationalization measures are taken before translation. Localization comes afterwards. That is why localization is often used as a synonym of translation (cf. the two English translations for the term *localization: translation* and *adaptation*). Consequently, localization is a mixture of both translation and technical adaptation.

Before localizing a product, you should find out if you can use suitable tools, such as automatic translation systems or translation memory tools. Furthermore, the developers should provide you with the entire set of files to be translated. If possible, you should also have a running version of the software to be localized. The translatable files can be text files (resource files under Windows) or executable files (.dll or .exe files). If you translate text files, you can immediately see your translation in the user interface. If you use executable files, you can create your translation independent of the programming code in a separate text editor. Normally, the translation is based on source code files (text files), or else the texts to be translated are provided in an Excel sheet.

Missing consistency is one of the main localization problems. If you do not use the same words for the same concepts, this may lead to critical questions and changes in the program itself. This costs time and money. Therefore, terminological mistakes must be avoided from the very beginning. Such mistakes are, for example, due to the context dependency and polysemy of terms. A typical example for a context dependent usage of terms is the German term “Bestellung”. In English, this term is rendered either as *purchase order*, *customer order* or *manufacturing order*. To avoid terminological mistakes and ensure a consistent usage of terminology, the following guidelines must be applied:

1. Terminology work must be done in time.

The translator needs to get the right input from the writer. The latter must write for translation or, in other words, write for a global audience. The whole product-specific terminology should be provided in lists and glossaries before localizing the product. This is the only way to cheaply translate many texts into another language in a short period of time. If you talk about writing for a global audience you talk about what to avoid: colloquial expressions, cultural contents in examples and graphics, polysemous expressions, irony and plays on words. A writer should remember that a text gets about 30% longer when it is translated from the English into the German language. And finally, a writer should always try to use terms consistently throughout the different document versions (for a detailed description of these rules, cf. Zerfaß 2002: 209f.).

2. Terminology work must be feasible.

When localizing software, a translator must be able to recognize which elements are localizable texts, and which are part of the program code and hence non-translatable. To distinguish translatable from non-translatable parts, it is necessary to insert a kind of localization tag in the code to indicate the translatable parts. Intershop developed a localization tool (“tLoc”) for the purpose of template translation.

The Intershop product “Enfinity” is based on English templates. To use languages other than English on the Enfinity Websites, the templates must be localized. This process is done by the “tLoc” tool. This tool helps to extract the user actions and list them in a separate Excel sheet where they can be translated. After that, the translations are inserted in the templates again.

4.1 The Localization Tool „Enfinity tLoc“

To find out the translatable parts (strings) of an Enfinity template, proceed as follows:

Preparing the Templates

All translatable parts of a template are enclosed by “isloc” tags. A specific configuration file is used to create so called master templates.

Creating Dictionary Files

To create dictionary files, the translator can choose between two common file formats, namely XML or comma-separated files (.csv files). The format depends on how and where the strings are to be translated. Comma-separated files are processed in spread-sheet programs like StarOffice, Calc or Microsoft Excel whereas XML files are translated using computer-aided translation tools (CAT programs like Star Transit or TRADOS WorkBench).

Translating Marked Strings

This is the actual translation process. The following example requires that the translatable files are provided in XML and comma-separated files. Both the XML and CSV formats are based on a template that has been created for exercising purposes only.

```
<html>
<head>
  <title>Localization Test Template</title>
</head>
<body>
  <h1>This is a Localization Test Template</h1>
  <p>The text outside of html tags is considered as localizable text.</p>
  The value attribute of inputs is considered as localizable if the input is of type
  'button','submit' or 'reset'.
  <form>
    <input type="text" value="" />
  </p>
```

```
<input type="reset" value="Reset" />&nbsp;  <input type="submit" value="OK" />
</p>
</form>
<script>
  var text = "Text inside script tags is currently left unchanged!";
</script>
</body>
</html>
```

After adding the localization tags, you would get the following XML file:

```
<dictionary>
<isloc>
  <field>3ea9317e000517a300000a00180304fc</field>
  <field>html</field>
  <field />
  <field>Reset</field>
  <field>Reset</field>
</isloc>
<isloc>
  <field>3ea9317e000338a200000a00180304ff</field>
  <field>html</field>
  <field />
  <field>The text outside of html tags is considered as localizable text.</field>
  <field>The text outside of html tags is considered as localizable text.</field>
</isloc>
<isloc>
  <field>3ea9317e0004284b00000a00180304cc</field>
  <field>html</field>
  <field />
  <field>The value attribute of inputs is considered as localizable if the input is of type
'button','submit' or 'reset'.</field>
  <field>The value attribute of inputs is considered as localizable if the input is of type
'button','submit' or 'reset'.</field>
</isloc>
<isloc>
  <field>3ea9317e000174fa00000a0018030528</field>
  <field>html</field>
  <field />
  <field>Localization Test Template</field>
  <field>Localization Test Template</field>
</isloc>
<isloc>
  <field>3ea9317e0002348300000a00180304f6</field>
  <field>html</field>
  <field />
  <field>This is a Localization Test Template</field>
  <field>This is a Localization Test Template</field>
</isloc>
</isloc>
```

```

<field>3ea9317e0006b6e900000a0018030531</field>
<field>html</field>
<field />
<field>OK</field>
<field>OK</field>
</isloc>
</dictionary>

```

If the translator works with an automatic translation program, the file can be processed in a suitable CAT editor (cf. Figure 1):

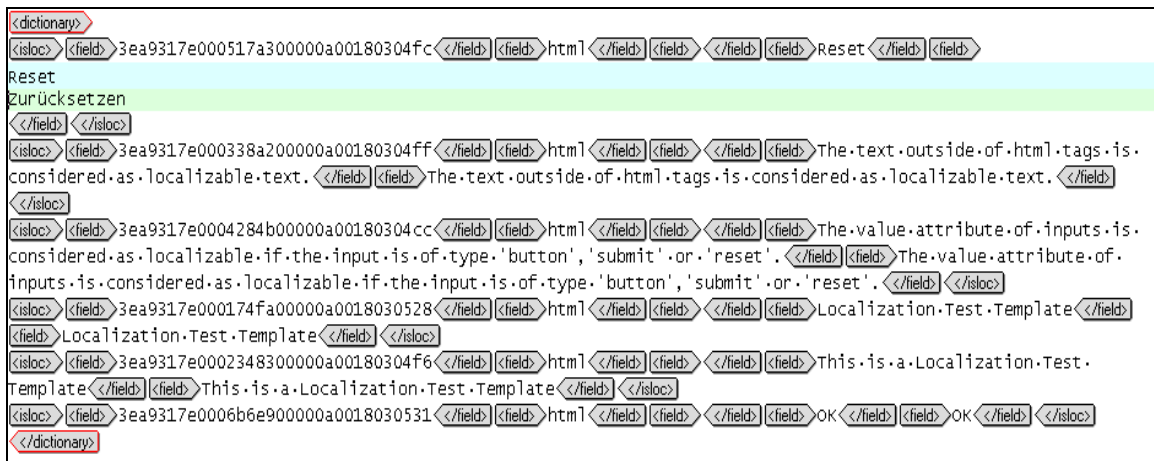


Figure 1: XML based dictionary file in a CAT editor

If the translation is based on a comma-separated dictionary file, the translator can choose any calculation program for processing the file (cf. Figure 2):

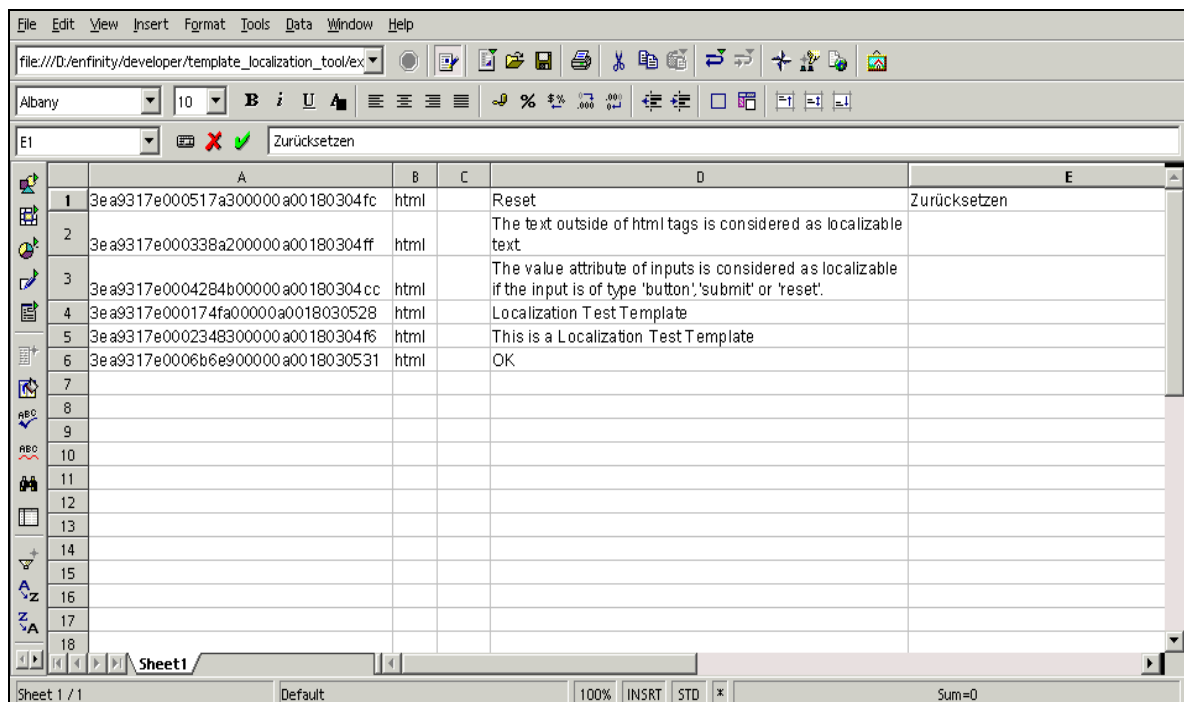


Figure 2: Comma-separated dictionary file in a spread-sheet program

Creating Localized Templates

The master templates created at the beginning of the localization process are now integrated with the dictionary files. This means that the strings enclosed by *isloc* tags are replaced with the translated strings. If you use the new template, the user can see the translated version on the Enfinity Websites.

To completely localize templates, however, you will have to make some additional changes. This includes localizing hard-coded time and date formats as well as address fields (cf. Ottmann 2002, for a detailed description of date and time formatting problems).

In general, variables should never be displayed as non-translatable elements (*hard-coded text*). This is to avoid that German users can enter the date using only the American format. Table 2 shows the various possible formats when localizing a product into English:

Style	Deutschland	USA
DEFAULT	26.9.2003	26-Sept-03
SHORT	26.9.03	26/9/03
MEDIUM	26.Sept.2003	26-Sept-03
LONG	26. September 2003	September 26, 2003
FULL	Freitag, den 26. September 2003	Friday, September 26, 2003

Table 2: German and American date formats

5. Plastic Words in the Software Industry

Languages for specific purposes are a big reservoir for words that seem to be non-ambiguous. However, this contradicts the reality. Software terms are ambiguous and inconsistent. Most of us seem to understand these terms because their meaning resembles the meanings of words in our everyday language. This is why the German linguist Uwe Pörksen called these words “plastic words” (cf. Pörksen 1997).

Many plastic words are software terms, which is illustrated by the following article from the German weekly newspaper „DIE ZEIT“. This article is an extract of an interview that some journalists conducted with the headquarters of software companies:

12, guten Tag.

Hallo, ich habe eine Ihrer Anzeigen gesehen, und ich verstehe sie nicht ganz. Da steht ganz groß auf einer fast leeren Seite: „Value2“. Was bedeutet das?

Das ist unsere Werbeanzeige für Unternehmen, nicht für Privatpersonen.

Aber sie steht in der „FAZ“, und ich lese sie auch.

Das bedeutet, unsere Software bringt den Unternehmen Wert [...] Jetzt haben Sie mich erwischt. Also, dass die ihre Prozesse optimaler [...] Deswegen dieses Value hoch zwei.

Was macht Ihre Software genau?

Sie deckt alle Kernprozesse eines Unternehmens ab. Von der Planung bis zum Verkauf von Produkten. Diese Infrastruktur läuft übers Internet.

Sie verkaufen also ein Programm.

*Nein, verschiedene **Lösungen**.*

Und worum handelt es sich bei „Marketplace-to-Marketplace-Kollaboration“?

Das kann ich Ihnen jetzt nicht erklären. Am besten gehen Sie auf unsere Homepage, www.i2.com.

Oje, ob ich dadurch schlauer werde?

Da steht alles, auf Englisch.

In der Anzeige beinahe auch: „Nur eine umfassende B2B-Lösung, die Marketplace-to-Marketplace-Kollaboration ermöglicht, die gesamte Supply-Chain koordiniert und optimiert und reichhaltige Content-Management-Instrumente bietet, macht solche Ergebnisse möglich.“

Was um Gottes willen bedeutet das?

Ich kann Sie nur auf nächste Woche vertrösten, bis unsere Mitarbeiter aus den Staaten wieder da sind.

Mit wem soll ich sprechen?

*Da müssen wir mal schauen, wer im Büro ist. Am besten wenden Sie sich an unsere BDR-Dame, unsere **Business Development Representative**. Sie ist zuständig für solche Fragen am Telefon.*

Ist Ihre Firma keine deutsche?

Nein, eine amerikanische, sie wurde 1998 gegründet. Wir sind die deutsche Niederlassung.

Und Sie wissen nicht, was „Content-Management-Instrumente“ sind?

So tief stecke ich in der Materie nicht drin.

What can this interview tell us about the characteristics of plastic words?

As you can see from the example, the company representatives cannot always explain what they are doing. They either do not know what they are talking about, or they use definitions that do not convey the meaning of a term. The various misunderstandings between the “experts” and the interviewer are caused by the fact that the terms to be defined (*definiens*) are explained with the help of other terms that need some explanation themselves. Let us take another example, which is not mentioned in the interview:

The term “knowledge management” is generally described as a “discipline that seeks to improve the performance of individuals and organizations by maintaining and leveraging the present and future value of knowledge assets” (cf. Knowledge Management Theory Papers 1999). If you analyze this definition, it becomes clear that the vagueness in meaning is caused by the terms *value* or *knowledge assets*. The terms *knowledge asset* or *artifact* have a multifaceted meaning themselves. They comprise miscellaneous things, such as documents, files, graphics, thoughts, conversations, software, databases, e-mails et cetera. In other words, terms like asset or artifact are broad enough to cause confusion in the reader’s mind.

As plastic words are frequently used in marketing and other public papers, the sources for ambiguity should always be part of a terminological entry in a database.

6. Terminological Entries in a Database

There are several possibilities to avoid terminological problems in a large company. A very common solution is to create a central terminological database or termbase⁵. Bob Clark from the Logos Group used a musical analogy to underline the importance of a database. He said that a database would allow writers and translators „to sing from the same hymnbook“.

The structure of a terminological entry generally depends on the purpose of a termbase and the size of the company. Practical considerations should, however, always be the starting point for creating a database. Terminology work is not just an academic exercise.

The crucial point of a termbase is a variable entry structure. It must be possible to freely select and change data categories and languages (cf. Schmitz 2002). Also, it must be possible to import data to and export data from the termbase.

To create a termbase for a new product, you must first decide which terms are generic (and hence stay outside the termbase), and which terms are product-specific (and hence must be included in the termbase). At this point, it is very important to agree on a compromise between developers and terminology experts. A round table is the best way to solve terminology issues and create simple term lists that can later be imported to the termbase. Figure 3 shows a terminological entry in the termbase of Intershop. It consists of the categories definition, part of speech, related terms, example, and provides the English and French translations of a term. Trados MultiTerm is used as a terminology management system⁶:

⁵ A data collection that defines concepts, generally in a specific specialized subject field, and documents the terms associated with those concepts.

⁶ A terminology management system (TMS) is a software application that stores and encodes terminology resources in dictionaries. Examples of terminology management systems are STAR TermStar and Trados MultiTerm.

The screenshot shows the Intershop Terminology Database interface. At the top, there is a navigation bar with the title "Intershop Terminology" and a search bar containing the word "storefront". To the right of the search bar are buttons for "Search", "Index", and "Target", along with dropdown menus for "English" and "German". On the left side, there is a vertical menu with various terms like "storefront", "structure template", "Structure View", etc. The main content area displays the definition of "storefront" in English, including a "Definition" section and "Related Terms" like "buyer". Below the English definition, there are sections for "German" (Storefront) and "French" (magasin).

Figure 3: Terminological entry in the Intershop termbase

Despite of all technologies, it is always the writer who must anticipate problems and solve them before it is too late. In his practical work, a writer can only compensate for terminological problems that have been caused during product development. Because a product can never be developed a second time, a writer can only make cosmetic changes. This is why a writer should keep an eye on terminology from the very beginning of product development and accompany this process until the product is ready to be released.

Only terminology experts should decide how to classify terms and which names to use. This work can neither be done by software developers nor by a terminology program. This remains manual work. There are various possibilities to categorize terminological data, e. g. the entry structure according to ISO 12620 (1999) distinguishing between a conceptual and a designation level (Schmitz 2002: 189ff.). For the purposes of technical documentation and localization the following data categories are considered to be sufficient for a project glossary (cf. Esselink 2002: 403):

- Industry-specific terminology, e.g. standard terminology for particular industries, such as the automotive, pharmaceutical, and financial sectors
- Equivalents for keywords, both verbs and nouns, appropriate for the product
- Product-related names that should not be translated
- Words or even phrases that are repeated throughout the project, such as "Note", and "Select"

- Names and explanations of essential product concepts, e.g. basic concepts of the Intershop Procurement Solution
- Names of help files or manuals
- Hot keys
- Product name and version
- Category, e.g. button, menu, dialog box title, etc.

7. Summary

If taken seriously, terminology work is not a curse, but a blessing. Terminology is an essential part of software development. Hence, terminology work must start in time and early enough to be successful. If you enforce a consistent style and terminology, you can save time and money. This is especially true for software documentation and localization.

Software developers and terminologists should closely cooperate to work on terminological problems is an essential part of their work, a new product can more quickly be released, and the time to market be shortened. Michael Kemmann (2002: 100) once commented on this topic: „*Terminologiarbeit kostet Geld. Keine Terminologiarbeit kostet noch viel mehr Geld.*“⁷ As a consequence, terminology work should be considered a big challenge to better connect LSP research with practical tasks.

We should pay terminology a similar attention as the Romans did with their God of landmarks, Terminus. When the Romans built the temple on Capitol Hill and removed the smaller sanctuaries, the augurs did not agree to remove the temple of Terminus. This temple stayed where it had been before. The symbol of Terminus, a stone, was included with the temple of Jove, which is the guardian of all landmarks.

Bibliography

- Arntz, R./ Picht, H./ Mayer, F. (2002⁴): Einführung in die Terminologiarbeit. Hildesheim [u.a.]: Olms.
- Austermühl, F. (2001): Übersetzen im Informationszeitalter: Überlegung zur Zukunft fachkommunikativen und interkulturellen Handelns im Global Village. Trier: Wiss. Verlag.
- Bickle, C. (2002): Wege zur Firmenterminologie. In: technische kommunikation. Zeitschrift für Technische Dokumentation und Informationsmanagement; 3/02, S. 40-43.
- Coulmas, F. (1992): Die Wirtschaft mit der Sprache. Eine sprachsoziologische Studie. Frankfurt a. M.: Suhrkamp.
- Esselink, B. (2002): A Practical Guide to Localization. Amsterdam: Benjamins.

⁷ Terminology work terminology is expensive. No terminology work, however, is even more expensive.

- Göpferich, S.* (1998): Interkulturelles *Technical Writing*. Fachliches adressatengerecht vermitteln. In: Forum für Fachsprachen-Forschung; Bd. 40. Tübingen: Narr.
- Gräfe, E.* (Hg.) (2002): *tekom-tagungen. Jahrestagung 2002 in Wiesbaden. Zusammenfassung der Referate*. Stuttgart: Gesellschaft für technische Kommunikation e.V.
- Henning, J. / Tjarks-Sobhani, M.* (Hg.) (2002): Lokalisierung von Technischer Dokumentation. In: *tekom Schriften zur Technischen Dokumentation*; Bd. 4. Lübeck: Schmidt-Römhild.
- ~ (Hg.) (2002): Lokalisierung von Technischer Dokumentation. In: *tekom Schriften zur Technischen Dokumentation*; Bd. 6. Lübeck: Schmidt-Römhild.
- Kemman, M.* (2002): Terminologiarbeit in Software-Entwicklung und – Lokalisierung. In: Henning, J./ Tjarks-Sobhani, M. (Hg.) (2002), Lokalisierung von Technischer Dokumentation. Schmidt-Römhild, S. 88-100.
- Ortega y Gasset, J.* (1996 [1939]): Glanz und Elend der Übersetzung. In: *Gesammelte Werke. Lizenzausgabe für Bechtermünz Verlag im Weltbild Verlag GmbH. Augsburg 1996 (1957 by Revista de Occidente, Madrid)*. Bd. IV, S. 126-151.
- Ottmann, A.* (2002): Software-Lokalisierung. In: Henning, J./ Tjarks-Sobhani (Hg.) (2002), S. 146-163.
- Pörksen, U.* (1997⁵): *Plastikwörter. Die Sprache einer internationalen Diktatur*. Stuttgart: Klett-Cotta.
- Price, J./ Korman, H.* (1993): *How to Communicate Technical Information. A Handbook of Software and Hardware Documentation*. Boston [u.a.]: Addison-Wesley.
- Schmitz, K.-D.* (2001): Terminologieverwaltung. In: Henning, J./ Tjarks-Sobhani, M. (Hg.) (2001), *Informations- und Wissensmanagement für technische Dokumentation*; Bd. 4. Lübeck: Schmidt-Römhild, S. 188-202.
- Zimmer, D.* (1997): *Deutsch und anders. Die Sprache im Modernisierungsfieber*. Reinbek bei Hamburg: Rowohlt.

ABSTRACT

Software Terminology as a Curse or Blessing: Possible Solutions for Terminological Problems

Matthias Vogel
Universität Halle-Wittenberg
Germany

In our everyday life, the various types of user documentation (printed and online manuals) are becoming increasingly important, e.g., to program a video recorder or carry out specific tasks in a computer program. Because very few people read a manual from front to back, it is not as important for the document to be entertaining or to use complicated vocabulary.

Terminology management is a frequently underestimated task in documentation and localization projects. Nevertheless, it plays a crucial role for creating user-friendly documents that are easy to read and understand. Terminological inconsistencies, however, make documents unnecessarily complicated and prevent users from understanding them.

Based on examples from the software company Intershop, this paper contributes to establishing terminology management at the heart of both technical documentation and software localization. First, the paper discusses the impact of terminology issues on how to write and translate technical documents. Second, the paper describes terminology research as a precondition for localization projects. This is illustrated by the process of template localization using the “Enfinity tloc” tool. Third, a suggestion is made about the possible structure of terminological records in terminological databases, including the concept and other information, such as definitions, target language equivalents, grammatical information, and contextual information.
