Terminological recommendations for software localization - By: KLAUS-DIRK SCHMITZ
Posted by Michael Lambarena on June 05 2009 10:16:29

Terminological recommendations for software localization

1. Software localization

After an explosive growth of data processing and software starting at the beginning of the 1980s, the software industry shifted toward a strong orientation in non-US markets at the beginning of the 1990s. Today we see the global marketing of software in almost all regions of the world. Since software is no longer used by IT experts only, and since European and national regulations require user interfaces, manuals and documentation to be provided in the language of the customer, the market for software translation, i.e. for software localization, is the fastest growing market in the translation business.

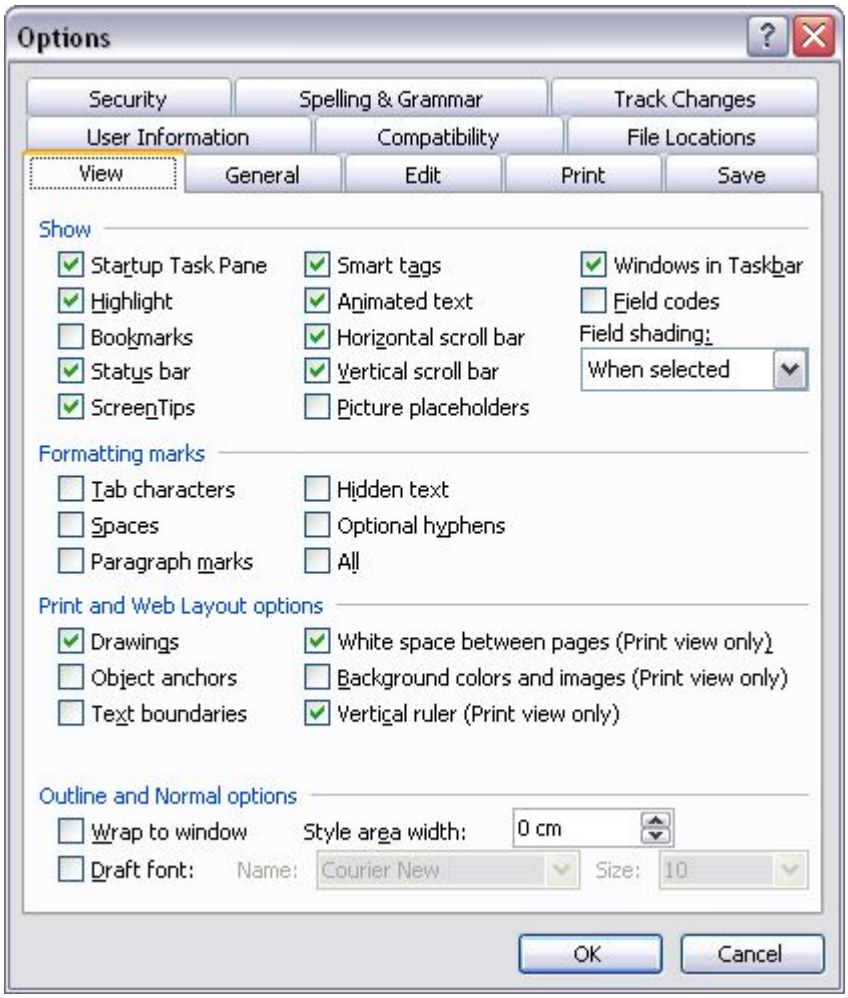
Internationalization and localization comprise the critical components in the effort involved in developing products for multiple regional markets. *Internationalization* concentrates on developing a software product in such a way that it will be easy to adapt it to other markets, i.e. other languages and cultures. The main goal of internationalization is to eliminate the need to reprogram or recompile the original program when localized for a specific regional market. Typical software development errors that run counter to the basic principles of internationalization are e.g.:

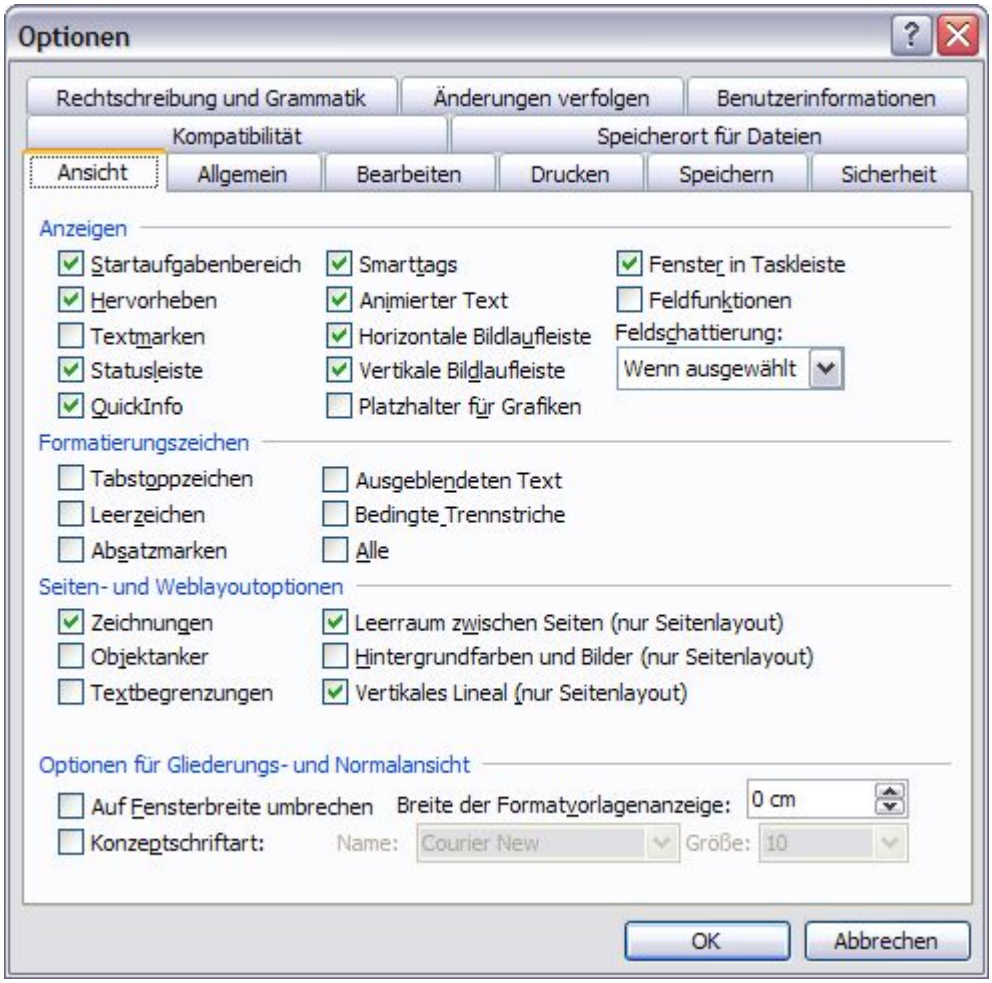
- text embedded in the program code
- length limitations in the text (fields)
- fixed formats for date, currency, units of measure, etc.
- fixed formats for addresses
- textual elements in graphics
- country- and culture-specific icons and symbols

Localization can be defined as the whole process of adapting a software product to a local or regional market with the main goal being the consideration of all appropriate linguistic and cultural aspects. The process of localization is performed by translators, localizers and language engineers and comprises the translation of

the user interface, the online help, the documentation and all packing material including the adjustment of all addresses, examples, units of measure and screen shots.

Internationalization and localization comprise the whole of the effort involved in developing products for several regional markets. While internationalization is "stuff" you have to do only once during the programming of a software application, localization is "stuff" you have to do over and over again for each regional market. Therefore, the more "stuff" you push into internationalization out of localization, the less complicated and expensive the process becomes.





(Figure 1: Software user interface element: original in English and localized German version)

2. Terminology of software products

2.1 Terminology as a means of communication and knowledge transfer

When companies develop software for end-users, they need to ensure that their customers will be able to use the program for the intended purpose. Therefore, each software product needs to be equipped with a user interface, an instruction manual, and other types of documentation. Companies invest a great deal of effort in determining details about typical users of their software, what users really need, and how explicit, detailed and intuitive the user assistance material must be. Depending on the type of the software product involved, this material varies in terms of length, complexity and intuitiveness.

The complexity of the software also influences the extent to which special language is needed to enable the end-user to operate the program in a correct and efficient way. Special language and, above all, the domain-specific terminology involved is not only an essential part of the written user assistance material (e.g. the instruction manual), but also of the interface between the user and the program. Therefore, terminology is the primary means of communication and knowledge transfer between software developers and end-users via the user assistance material. Consequently, avoiding indeterminate, incorrect and inconsistent use of terms and icons must be one of the major goals of software development, quality assurance, and usability testing.

2.2 New terms for new concepts

The Information Technology (IT) industry, like any emerging industry, has seen the development of new technologies, processes, and products. Terminology theory refers to these new entities and processes in the real world as *objects*. When new (concrete or abstract) objects are invented or created, new *concepts*, or cognitive representations of the objects, are established and new *terms* or graphical representations like *icons* are needed to communicate about them.

New terms can be coined by creating new forms, by using existing forms, or by borrowing terms from other languages. Before creating a new term, it is necessary to ascertain whether a term already exists for the concept in question. Additionally, those responsible for creating terms need to respect well-established usage: even if the terms are poorly formed or poorly motivated, they should not be changed unless there are compelling reasons, such as cultural sensitivity or homonymy with other terms within the same domain.

2.3 Transparent terms are easier to grasp

All different types of users need to be able to interact intuitively with the software and understand the user assistance material. A *transparent* terminology enables the user to clearly understand underlying concepts. If a new term needs to be created or selected to express a certain feature or a particular operation of the software, a *morphological motivation* is the best criterion for constructing a new term. For example, terms like *page setup* or *error message* are in most cases easy to grasp because the morphological components of the terms are well known by the user.

As a result, the meaning of the term can be directly derived from the meanings of the parts of the term. Sometimes the relation between the components of a motivated term is indeterminate and may cause problems, especially in languages like English or German: is a *data network identification code* the identification code *of* the network, *for* the network or *within* a network? A translator working from English into German will be untroubled by this distinction because the two languages are equally abstract, but someone translating into a Romance language such as French or Spanish must know precisely what the relationships are between the critical elements in the multi-word term.

Use of *semantic motivation* can create terms that are slightly more difficult to understand. In most cases semantic motivation is associated with term creation procedures such as *terminologization* or *transdisciplinary* borrowing, leading to homonymy across subject fields. Examples from the software industry included terms like *worm*, *virus*, *infected file* or *vulnerability*. Such terms require that the user resolve indeterminacy by transferring the meaning from general language or other subject fields to the new concept as it is used in computing. But if the motivation of the term is understood by the end-user and the usage of the term is established by the community, it becomes transparent and linguistically economical in the user interface and other support materials (e.g. the term *mouse* for a computer pointing device).

The effort of creating transparent and motivated terminology throughout the user interface and all of the user assistance material is one of the major preconditions for user empowerment. If users encounter just one indeterminate term, or even two, in a computing experience, they might not be dissuaded from further use of the program; each terminology problem by itself is unlikely to frustrate users during their experience using the software. However, the cumulative effect of multiple terminology problems (such as lack of transparency or clear motivation, for example), can have an exponential impact on levels of frustration and computer anxiety.

2.4 Appropriate terms will cause less confusion

The language and the terminology in software products need to be *appropriate* for the user group. Appropriateness refers not only to familiarity of terms to the end-user, but also requires that the terms, instructions or messages don't cause confusion or insecurity to end users, including those generally unfamiliar with computers and software products. The following example will illustrate this criterion: During a particular installation process, the user needs to

select either *express installation* (to install only components needed in the current configuration of his or her system) or *network installation* (to install all components, even those not necessary for his or her particular system). In this example, the user is confronted with the indeterminacy of both terms used - he or she could worry about missing something in the case of selecting *express installation*, a decision which would actually be the more appropriate choice. In order to avoid confusion and misguidance the software developer should use terms such as *optimized installation* and *complete installation*, terms that avoid indeterminacy, more appropriately represent the concepts behind the terms and enable the end-user to make the right choice during installation.

Another aspect of appropriateness of terminology deals with connotations of terms. Terms created should be as neutral as possible; those creating terminology should avoid, in particular, choosing terms that have negative connotations. One prominent and controversial example is the pair of terms *master/slave*, which was established many years ago in the IT industry. At that time the negative connotations of *slave* were not taken into account. This instance is a good example of transdisciplinary borrowing, where the concepts of *master* and *slave* are drawn from instrument control technology (hydraulics and pneumatics). In English, this analogy is very strong, and the negative implication of the root meaning of *slave* has long since become a frozen metaphor, whereas it retains its pejorative connotation in languages where other terms are used for control systems. Several software producers are now replacing the term where possible with *master/subordinate* or, if applicable, with *client/server*. In many cases in the past, negative connotations of terms have been discovered only when localizing them, because connotations are very much culturally and linguistically dependent.

2.5 Consistency is the overall prerequisite

Another major objective of terminology indeterminacy that has an impact on end-users is the *consistency* of terminology. In terms of consistency, the main goal should be that only one term should exist for each concept, and no synonymy or homonymy should exist within each domain. This goal is not so easy to achieve in a complex and multifaceted development environment because different developers, product teams and companies all create terms in different places and time periods.

The end-user will be very frustrated if several terms are used for the same concept within the user interface, the help system, the printed documentation, the packaging material and the web

presentation of *one* specific software product. Software developers, user interface designers, technical writers and website authors all have to agree very early during the development process on what a certain feature of the software will be called. When, for example, the enter key is called *enter key* in the user interface and in the first ten pages of the manual, but on page eleven it is called *return key*, the user will assume that this is something different. Thus, inconsistent terminology impedes communication between the end-user and the software product and lessens computer and software ease-of-use.

While terminological consistency is the key to ease-of-use even within one product, as illustrated above, in general, software is not used in its stand-alone form. Therefore, the terminology used in a certain product must be also consistent with the terminology of other software products used together. Terminology management is, in this way, crucial to interoperability. As an example, the terms used in a printer set-up procedure within a word processing application that is embedded in an office package that runs under a specific operating system must all be compatible and consistent with the terms used in the "surrounding" programs.

Consistent terminology increases user confidence by decreasing indeterminacy caused when a single concept is associated with more than one term and enables associative learning (when related terminology reflects a single principle). Consistency also facilitates interoperability among users' many integrated software products. As a result, establishing terminological consistency is one of the most important aspects of user-friendly software products and therefore of user empowerment.

Idioms, colloquialisms, slang and analogies are especially culture and language dependent and often cause similar problems of indeterminacy during the localization process, as do problems encountered with the use of humour and sarcasm. Avoiding these stylistic features in the English version of the software will not only facilitate the localization process, but will also empower end-users who are non-native English speakers who have to use the English version of the software.

3. Terminology management for software localization

3.1 What are the terms to be managed?

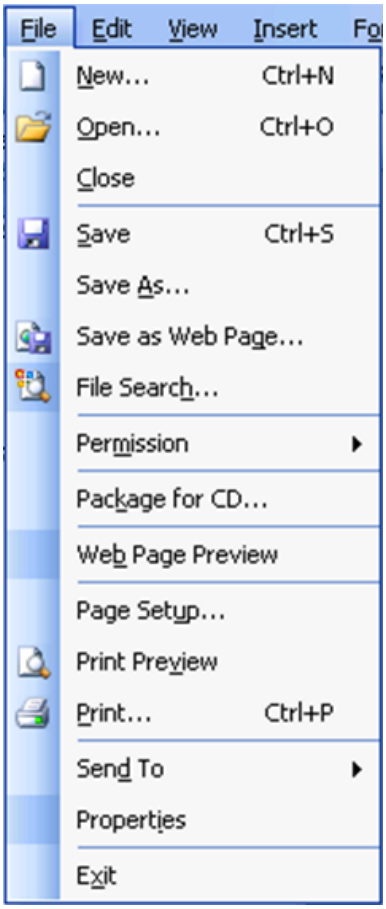
ISO 1087 defines the term as a "verbal designation of a general concept in a specific subject field." The term serves as the representation of the concept. We can write it down, think it, say it

out loud, and use it for communication. Some terms consist of more than one word. These terms are called multi-word terms or compounds. In Germanic special languages such as in English and German, multi-word terms usually consist of several nouns or adjective-noun combinations. The way words combine to form terms varies from language to language.

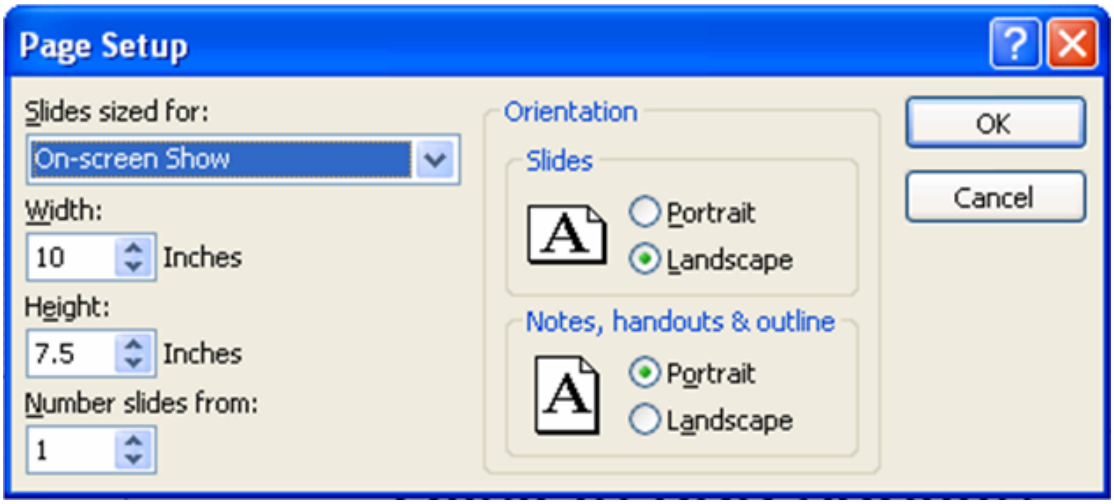
Example 1:	Single-word terms:	*printer*
	Multi-word terms:	*laser printer*
		serial port
		printer with single-sheet feed

There is no doubt that software user interface terms used in menus and dialog boxes like *file* or *options* are representing concepts in the traditional terminological view. But linguistic elements like *open file*, *save as* ... or *insert table* also represent concepts of the user interface although they are traditionally classified as phrases. Similar problems arise with menu items like *templates and add-ins*, *spell check and grammar* or *convert into Adobe PDF and send via e-mail* that are not seen as single concepts but as combination of concepts.

Figures 2 and 3 show examples of terms in menus and dialog boxes of an English user interface.



(Figure 2: Menu)



(Figure 3: Dialog box)

Linguistic elements used in error or system messages are much more problematic. The following list shows examples of this type of messages:

Example 2: *paper jam*

unexpected error in application program

not enough memory for display the graphic file

please check network configuration

file %f could not be opened

All these linguistic units are identified as individual items for the localization process by specific localization tools such as Catalyst or Passolo. They should be managed and documented as individual entries in terminology management systems and therefore also understood as localization concepts.

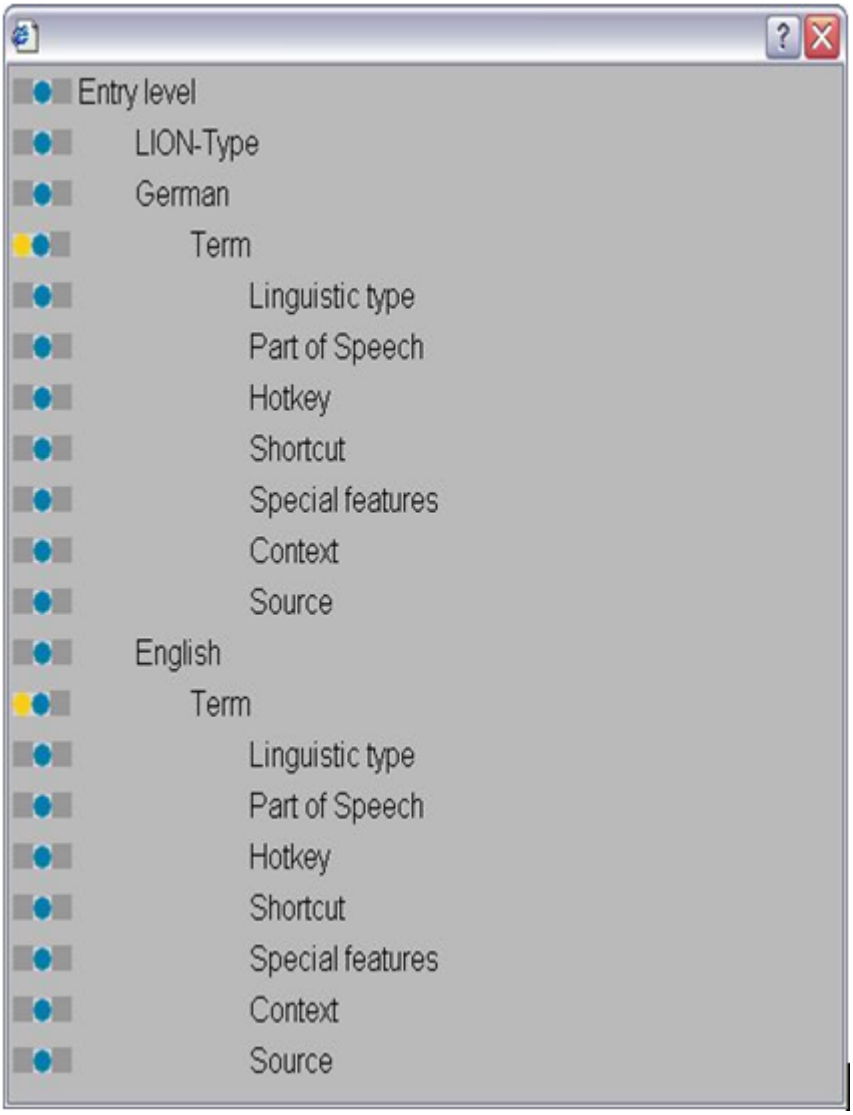
Accepting the view that all these linguistic units of software user interfaces are terms in a broader sense, problems arise when identifying and differentiating the concepts and objects behind these terms. If we have installed the same version of a software product on several computers, we certainly have different objects for one concept represented by an individual term. E.g. the term *Print*... in different installations of MS Word 2003 represents exactly one concept with several objects in each installation. But represents *Print*... in different versions of a software, e.g. in MS Word 2003 and MS Word 2007, or in different software products, e.g. in MS Word 2003 and MS Excel 2003, different concepts? There are arguments supporting this differentiation referring to the different functionality behind these menu items and therefore to the different characteristics of the concepts behind the term. A consequence of this will be that *Print*... has to have several terminological entries in a concept-oriented terminological database.

This more software localization oriented view to terminology theory and terminology management not only influences a specific understanding of term, concept and object for this domain, but also affects other data categories for terminology management such as definition and context (see Schmitz 2008).

3.2 Terminological data modelling for software localization

The previous discussion of terminology management principles and methods supplies the basis for a specific data modelling applicable for terminology in software user interfaces. From June 2005 to May 2007, the national funded research project DANDELION (Data Modeling and Data Exchange for Software Localization) was carried out at Cologne University of Applied Sciences. The main objective of the project was to develop methods for a more adequate documentation and management of software user interface texts in localization-specific tools and formats. One of the results was the design of a data model for software-specific terminology management and the specification of adequate data categories for that model.

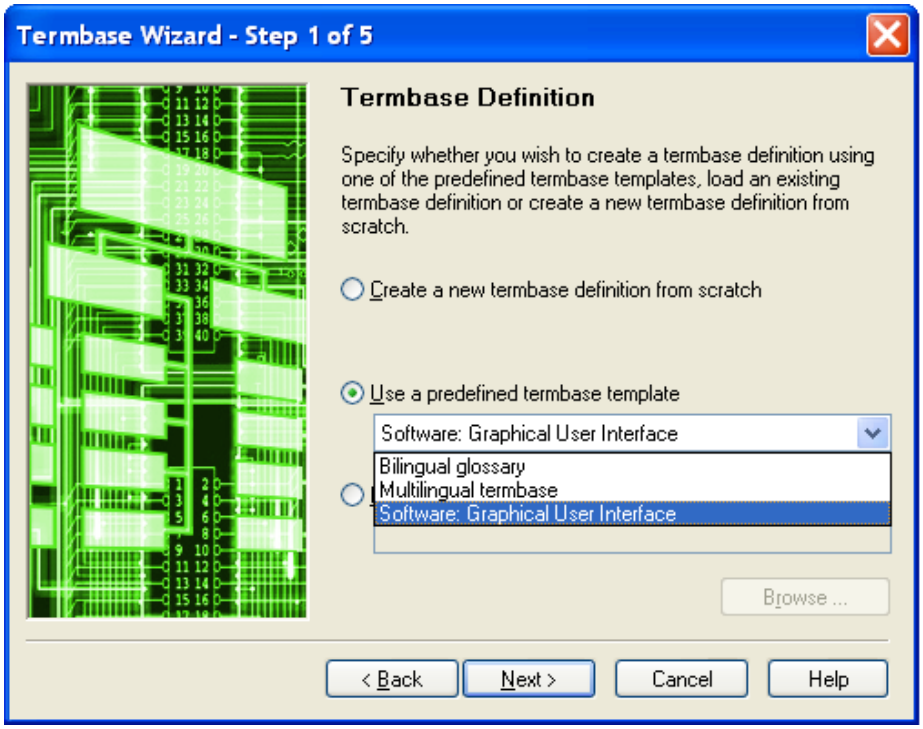
Figure 4 shows the Dandelion data model for localization specific terminology management with German and English as sample languages.



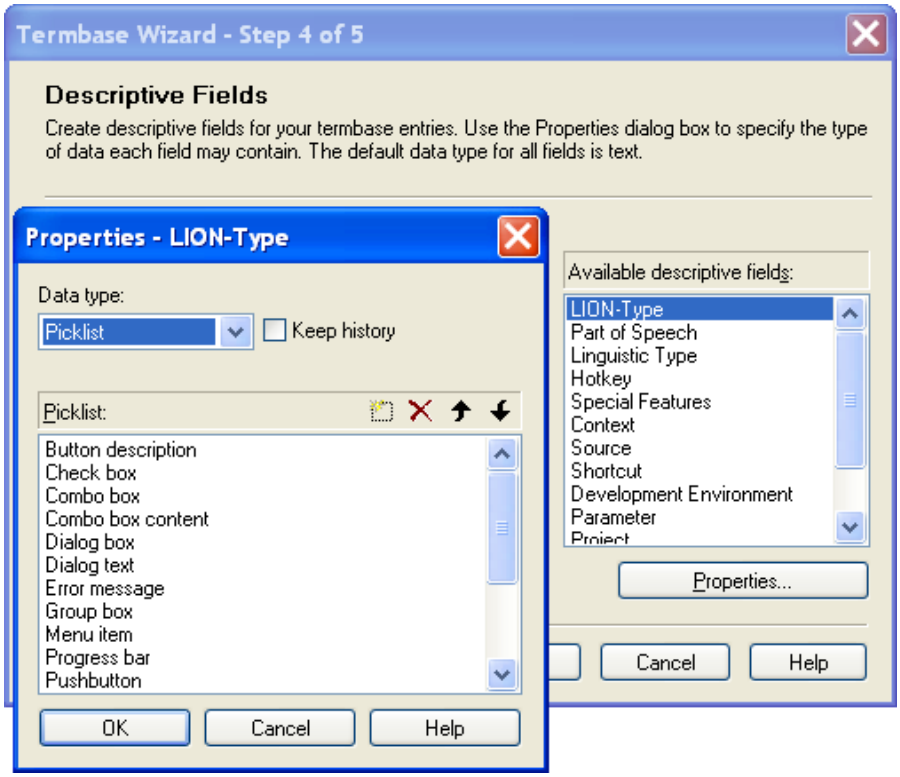
(Figure 4: Dandelion data model)

Special attention was laid on the data category <LION-Type> that indicates the software user interface element the term belongs to. Possible values for <LION-Type> are e.g. menu item, dialog box, check box, radio button or tool tip.

One of the industrial partners of the Dandelion project, SDL-Trados, implemented the data model as a predefined termbase template within the terminology management software MultiTerm 2007 (see Figure 5 and 6).



(Figure 5: Dandelion termbase template in MultiTerm 2007)



(Figure 6: Data categories of the Dandelion data model with properties of <LION-Type>)

5. Conclusion

Innovative domains as well as new concepts, terms and icons are characteristics of software products and their documentation. The development of user interfaces, online help systems, user manuals, websites etc. requires the application of terminological working methods and principles, especially if we focus on software internationalization and localization. Aspects of term creation and term selection, such as motivation, transparency, appropriateness and consistency have to be taken into consideration in order to provide the software user with clear and determinate terminology.

Since the exact definition and consistent use of terms in all parts of the software product is a fundamental precondition, software localization requires appropriate terminology management. Traditional approaches for designing and modelling terminology management solutions have to be adapted to the specific needs of user interface terminology. The Dandelion data model is a first step in the right direction.

Author

Prof. Dr. Klaus-Dirk Schmitz, full professor of terminology studies and language technology at Cologne University of Applied Sciences, President of the International Information Center for Terminology (Infoterm), Vice-president of the German Terminology Association and Chairman of the German National Standards Committee "Systems for managing terminology, knowledge and content".

References

Arntz, Reiner; Picht, Heribert; Mayer, Felix (2004): Einführung in die Terminologiearbeit. Hildesheim: Olms.

Esselink, Bert (2000): A Practical Guide to Localization. Amsterdam/Philadelphia: John Benjamins.

Galinski, Christian; Picht, Heribert (1997): "Graphic and Other Semiotic Forms of Knowledge Representation in Terminology Management". In: Wright, Sue Ellen; Budin, Gerhard (eds.): Handbook of terminology management (Volume I). Amsterdam/Philadelphia: John Benjamins.

ISO 704 (2000): Terminology work - Principles and methods. Geneva: ISO.

ISO 1087-1 (2000): Terminology work - Vocabulary - Part 1: Theory and application. Geneva: ISO.

ISO 12620 (1999): Computer applications in terminology - Data categories. Geneva: ISO.

ISO 16642 (2004): Computer applications in terminology - Terminological markup framework (TMF). Geneva: ISO.

Mayer, Felix; Schmitz, Klaus-Dirk; Zeumer, Jutta (eds.) (2002): eTerminology - Professionelle Terminologiearbeit im Zeitalter des Internet. Akten des Symposions, Köln, 12.-13. April 2002. Köln: Deutscher Terminologie-Tag e.V.

Reineke, Detlef; Schmitz, Klaus-Dirk (eds.) (2005): Einführung in die Softwarelokalisierung. Tübingen: Narr.

Russi, Debora; Schmitz, Klaus-Dirk (2007): DANDELION-Projekt. eDITion, 1/07, 18 - 19.

Schmitz, Klaus-Dirk (2004): „Terminologiearbeit, Terminologieverwaltung und Terminographie". In: Karlfried Knapp et al. (eds.): Angewandte Linguistik. Ein Lehrbuch. Tübingen: Francke, 435 - 456.

Schmitz, Klaus-Dirk (2005): "Terminological Data Modelling for Software Localization". In: Nistrup Madsen, Bodil; Erdman Thomsen, Hanne (eds.): Terminology and Content Development - TKE 2005, 7th International Conference on Terminology and Knowledge Engineering. Kopenhagen: GTW, 27 - 35.

Schmitz, Klaus-Dirk (2006): "Data Modeling: From Terminology to other Multilingual Structured Content". In: Wang, Yuli; Wang, Yu; Tian, Ye (Eds.): Terminology, Standardization and Technology

Transfer, Proceedings of the TSTT'2006 Conference. Beijing: Encyclopedia of China Publishing House, 4 - 14.

Schmitz, Klaus-Dirk (2007a): "Indeterminacy of terms and icons in software localization". Antia, Bassey Edem (ed.): Indeterminacy in LSP and Terminology. Amsterdam/Philadelphia: John Benjamins, 49 - 58.

Schmitz, Klaus-Dirk (2007b): „Die Bedeutung der Terminologiearbeit für die Softwarelokalisierung". In: Thelen, Marcel; Lewandowska-Tomaszczyk, Barbara (eds.). Translation and Meaning, Part 7. Maastricht: Universitaire Pers Maastricht, 29 - 36.

Schmitz, Klaus-Dirk (2008): „Überlegungen zu Begriffen und deren Repräsentationen in Softwareoberflächen". Paper presented at the LSP 2007 Conference Hamburg (forthcoming).

Schmitz, Klaus-Dirk; Wahle, Kirsten (eds.) (2000): Softwarelokalisierung. Tübingen: Stauffenburg.

Wright, Sue Ellen; Budin, Gerhard (eds.) (1997): Handbook of terminology management (Volume I). Amsterdam/Philadelphia: John Benjamins.

Wright, Sue Ellen; Budin, Gerhard (eds.) (2001): Handbook of terminology management (Volume II). Amsterdam/Philadelphia: John Benjamins.