Meta
Journal des traducteurs
Translators' Journal

META

The Intelligent Dictionary Project

Jean-Marie Maes

Volume 36, numéro 1, mars 1991

La terminologie dans le monde : orientations et recherches

URI : https://id.erudit.org/iderudit/003534ar

Aller au sommaire du numéro

Éditeur(s)

Les Presses de l'Université de Montréal

ISSN

0026-0452 (imprimé)

Découvrir la revue

Citer cet article

Maes, J.-M. (1991). The Intelligent Dictionary Project. *Meta*, *36*(1), 182–191.

Résumé de l'article

Description du fonctionnement d'un dictionnaire intelligent; outil électronique qui peut être utilisé comme un dictionnaire unilingue (descriptif), bilingue ou comme un lexique. Ce dictionnaire a été conçu pour la traduction mais se prête à plusieurs autres applications. Le programme construit une liste d'occurrences à partir des termes d'un texte source. Cette liste peut être épurée ou allongée à volonté, et ensuite être mise à jour ou être insérée dans le dictionnaire. Le principal avantage de cet outil est qu'il possède une très grande souplesse d'accès.

THE INTELLIGENT DICTIONARY PROJECT

JEAN-MARIE MAES
PIHO, Gent, Belgium

RÉSUMÉ

Description du fonctionnement d'un dictionnaire intelligent; outil électronique qui peut être utilisé comme un dictionnaire unilingue (descriptif), bilingue ou comme un lexique. Ce dictionnaire a été conçu pour la traduction mais se prête à plusieurs autres applications. Le programme construit une liste d'occurrences à partir des termes d'un texte source. Cette liste peut être épurée ou allongée à volonté, et ensuite être mise à jour ou être insérée dans le dictionnaire. Le principal avantage de cet outil est qu'il possède une très grande souplesse d'accès.

INTRODUCTION: PUTTING ELECTRONIC DICTIONARIES IN PERSPECTIVE

Electronic dictionaries...

Translators and terminologists have always had to find a way to store information that is of particular relevance to their needs. This information often is more valuable and more useful than the information found in dictionaries, lexicons, and other general reference material, precisely because it can be tailored to the needs of the users. In the past such work was carried out on paper. It was time-consuming to develop and maintain such a system. Retrieving information was not always easy or straightforward. Little wonder that people in translation and terminology have taken a keen interest in the development of electronic dictionary systems.

Many people (even in the profession) often think of electronic dictionaries in terms of complex databases on large computer systems that are out of reach for the individual translator. This is no longer a correct description of the situation. The personal computer (PC) now offers a great many possibilities one could not have dreamt of ten years ago. In fact, today's PC's (especially the AT's) are often more powerful than yesteryear's mini systems. As a result electronic dictionaries and dictionary development tools have become available to the individual translator.

There are two distinct dictionary systems on the market: *closed* systems that make use of CD-ROM and *open* systems that allow the translator to compile his own dictionaries. In a *closed* system the user can only consult—but not add to—the electronic dictionary or database. CD-ROM (Read Only Memory) disks, from which information can only be read, are used. There are already a fair number of electronic dictionaries and encyclopaedia available.

They all offer many interesting features and a mass of information to the translator. Undoubtedly they will in time become standard tools for each professional translator.

In an *open* system the user has to compile his own dictionary files. This may initially take a lot of time, but in practice these *personal* dictionaries prove to be

extremely helpful to the translator. Many translators specialize in certain subjects. These open systems enable them to compile specific dictionaries for each of the subjects they specialize in. These dictionaries often prove to be far more useful than the traditional general and specialized dictionaries as they are completely customized to the translator's needs. Ink Text Tools, Termex and Mercury are examples of these open systems.

Most open electronic dictionary systems offer very limited access to their dictionaries, mostly only via the dictionary entry or headword. This is a pity as there are no hardware reasons or restrictions why this should be so.

A translator makes use of bilingual and multilingual (translating) dictionaries, but also of monolingual (explanatory) dictionaries and lexicons. It would be very convenient if these three ways of storing information could be combined and made accessible from a word processor.

As both monolingual and multilingual dictionaries are compiled alphabetically, there is no fundamental problem in combining them. The nature of the information found in a systematically organized lexicon does not differ from what you find in a monolingual (explanatory) dictionary. It is the way the information is presented, the way it is ordered that is fundamentally different. If it were possible to integrate the semantic structure of a lexicon into the electronic dictionary, then the user could access the lexicon or the dictionary on the basis of a single data file.

AND ID

When I made a roundup of systems available a few years ago, I saw nothing that came close to what I envisaged. Existing programs did not offer the flexible access I looked for; some of them were not really suited for any serious terminological work, others were lacking in user friendliness. The latter was very important, as the program would be used at the *Provinciaal Instituut voor Hoger Onderwijs* (PIHO, Henleykaai 84) in Ghent (Belgium) where I work as a computer linguist.

I then decided to develop my own system, ID for *intelligent dictionary*. ID greatly widens access to dictionary-like data. ID builds only one set of data, but the user can access those data via three keys or any combination of them:

– the dictionary entry
– the semantic key
– the grammatical key.

Any of these keys may be incomplete or empty. They cannot all three be empty. In this way the user's possibilities to access the data are greatly enhanced.

He can change the key combination he uses to access the data at any time. There is no limitation as to the number of times any one key can occur. The combination of the three keys has to be unique (which is a logical thing as the three keys together define any concept; within ID they define a single record). A single word may occur various times (no limit) as an entry with either a different semantic or a different grammatical key.

ID can thus be used as a combination of a monolingual (explanatory) and a bilingual (translating) dictionary and as a systematically organized lexicon. The three keys (entry, semantic and grammatical key) offer in fact a highly flexible *zoom* function, allowing the user to access (and bring together) the data the way he prefers.

ID is designed to do a lot more than just tracing terms in a dictionary. The package covers a complete range from source text, over dictionary to target text.

The program builds a frequency list of all tokens in the source text. This list can be filtered: the tokens occurring in the filter (called base list and containing basic vocabulary that poses no problems to the translator) are not taken up in the frequency list. The result is a list of "new" word forms. These can be edited quickly and written straight to any

existing or new dictionary. Words that were not filtered away can be written to the filter. This enables the user to improve the filters he uses in a simple and straightforward way, simply by running ID on a set of texts. There is no limitation as to the number of different base lists or the number of texts to be treated.

The user can return to the source text(s) at any time from the frequency list and from the dictionary. The program takes into account the way the user turned inflected forms into entries, and looks for the maximum stem form in each particular text. Each dictionary entry can hold up to 30 references to different source texts of practically unlimited length. One key takes the user to the first occurrence of the entry in the source text. In a similar way the user can browse already translated material from the dictionary. This way ID is turned into something like a lexicon steered information retrieval system.

Though ID has been designed primarily for use in translation, it has a wide range of possible applications, such as syllabus design, literary studies, documentation, etc.

ID draws up dictionaries starting from one or more texts. Links to the source (and eventually target) texts are established automatically. The user can of course start a new dictionary without any need for or reference to a source text. The flow of the program is illustrated in fig. 1.

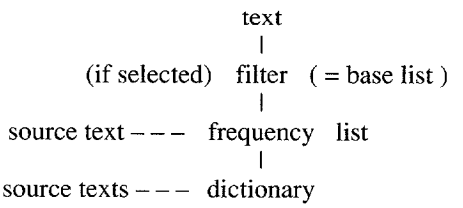

fig. 1: general flow of the program

ID: THE PROGRAM

ID is a complete package offering many possibilities that go far beyond classical dictionary usage. ID is language independent, i.e. it can be used to draw up dictionary files in the most important western languages. It has been tested for French, German, English and Dutch, which are the languages taught at our institute. All special characters of these languages can be used without any problem (e.g. the program distinguishes 'ü' from 'u' but still alphabetizes correctly).

ID offers immediate feedback between texts and dictionaries, at every stage in the development of a dictionary and even when consulting one.

FROM TEXT TO FREQUENCY LIST

Starting from any ASCII-text, ID draws up a frequency list of all tokens occurring in the text. In drawing up this list the user can select a filter, i.e. a list (in database format) of tokens he does NOT want to have in the frequency list and in the resulting dictionary. There is no limit as to the number of such filters. In an educational context he can use different filters related to the level of the students.

The user can quickly and easily edit the word forms. He can at any time return to the original context(s) in the source text. This is an extremely useful feature as it allows him to identify compounds, phrasal verbs and the like using the original contexts.

In order to illustrate this I have run the previous part of the text through ID. The result is a list of tokens. I have captured a sample screen of the list by way of example (see fig.)

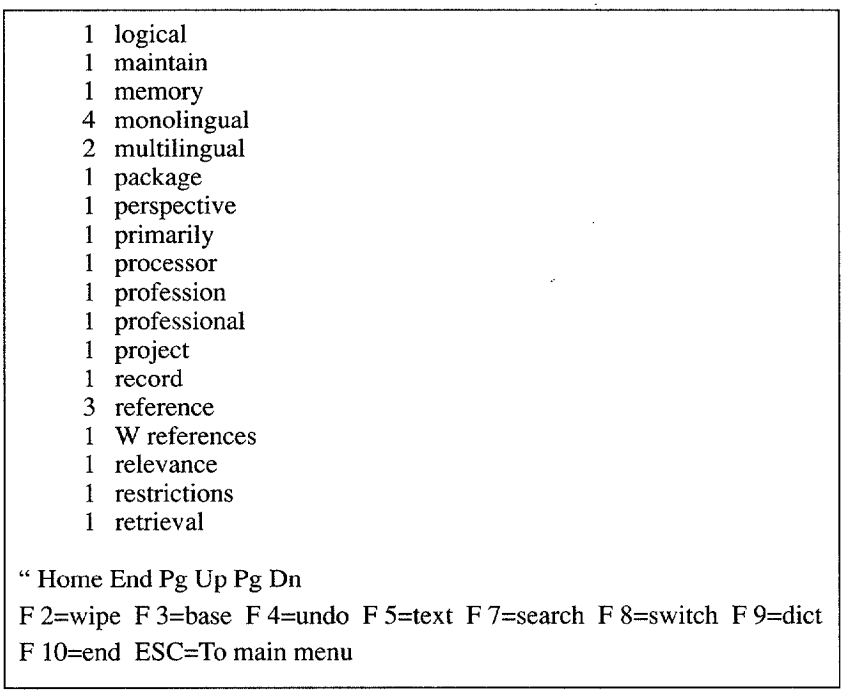

fig. 2: sample screen showing part of a frequency list

The user can move about the list using the cursor and related keys. He can edit each line using common word processor features such as *delete*, *backspace* and *insert*. The function keys are used for a number of specific functions.

F 2 is used to delete a token. Using F3 the user can at the same time delete a token and write it to the filter he used in creating the list. This is a highly useful feature, as it allows the user to enrich the filters quickly, simply by using them on his texts.

F 4 can be used to undo the actions performed by F2 and F3.

F 7 allows him to jump to a specific line or a specific word.

F 8 switches to the field showing the frequency and back.

F 9 ends the session and writes all resulting forms to the dictionary chosen before.

F 10 marks the end of a work session. The user can of course resume the work later if he likes. F5 finally takes the user to the original context of the word form. The first form found is highlighted. He can move on to the next occurrence. In fact the user can move freely through the text, select a number of lines for printing, etc. All this goes in a flash. In our example, we jump from the word record to its first occurrence in the text (see fig. 3).

ID browse function

on key can occur. The combination of the three keys has to be unique (which is a logical thing as the three keys together define any concept; within ID they define a single record). A single word may occur various times (no limit) as an entry with either a different semantic or a different grammatical key.

ID can thus be used as a combination of a monolingual (explanatory) and a bilingual (translating) dictionary and as a systematically organized lexicon. The three keys (entry, semantic and grammatical key) offer in fact a highly flexible zoom function, allowing the user to access (and bring together) the data the way he prefers. ID is designed to do a lot more than just tracing terms in a dictionary. The package covers a complete range from source text over dictionary to target text.

The program builds a frequency list of all tokens in the source text. This list can be filtered: the tokens occurring in the filter (called base list and containing basic vocabulary that poses no problems to the translator) are not taken up in the frequency list. The result is a list of `new' word forms. These can be edited quickly and written straight to any existing or new dictionary. Words that were not filtered away can be written to...

META.ASC Lines: 127-149 Col: 1 Total: 232 Bytes: 9211

fig. 3: sample screen showing part of the source text

This possibility to return to the original text quickly (taking into account the way inflected forms are rewritten to entries) is a typical feature of ID. It greatly adds to the usefulness and efficiency of the program.

A frequency list can of course be printed out or written to an ASCII-file for further use in a word processor for instance.

EDITING A FILTER

A filter is a list of tokens one does NOT want to select in the frequency lists and resulting dictionaries. The user can work on a filter of basic word forms (called a "base list" in the program) in much the same way as with the frequency list. He can add forms to the list, edit existing forms, search for a specific form, etc. The list can be printed or written to an ASCII-file. There is no limit as to the number of filters that can be used. Each filter may contain up to 10,000 entries (it can be larger in fact, but then you can no longer edit it directly).

USING THE DICTIONARY

After editing, a frequency list is normally written into a dictionary. If it is an existing dictionary, only words not yet occurring in the dictionary are added. The context field is automatically updated for the words already in the dictionary. This makes it possible for the user to return to the source text of his choice from the dictionary at all times.

New entries have 'dummy' as semantic and grammatical key. The date is automatically generated. In this way it is very easy to edit new entries systematically. You look for them via the semantic 'dummy' key, edit them and go on to the next new ('dummy') entry. ID remembers the way you consult the dictionary (the key combination) even when you edit the semantic and grammatical keys.

Having selected a dictionary, the user can do three things:

ADD an entry to the dictionary

SEARCH for a specific entry

make use of a number of GENERAL functions

ADD

If the user selects ADD, he will get a 'filing card' with empty fields. The date-field is automatically generated. He can edit all fields as he likes. Three fields must always be filled out, as they define a unique record:

the entry field
the semantic key field
the grammatical key field.

The user can use the list of existing semantic and grammatical keys to select the suitable key (cf. 2.3.2).

SEARCH

The user can look for any particular record by filling out any combination of the dictionary entry, the semantic key and the grammatical key.

The keys need not be complete. The only (logical) restriction is that they cannot all three be empty. The combination of the keys entered in the search is the user's window on the dictionary. The way the user moves through the dictionary is defined by them. If he selected a particular semantic key, he would work with that particular subset of the dictionary data. The user can change the key combination in use for the search at any time.

In the short test dictionary I use, the (incomplete) entry 'id' would result in the screen in fig. 4.

```
ID  D:\ID5\TESTID5                                    Nr. of rec: 29
_____________________________________________________________________

ENTRY: id-project
       SEM. KEY: cptr ai    GRAM. KEY: _count_prop
       Translat_1  :  id-project
       Translat_2  :
       Translat_3  :

Definition :
Project for the development of an 'intelligent dictionary' (ID) system, in which the
functions of monolingual and bilingual dictionaries and of a lexicon are combined.
The dictionary 'remembers' the source texts the entries come from. This enables the
user to return to the original contexts of a term and even to consult already translated
texts. In this way ID has grown into a lexicon steered information retrieval system.
There are plans to further develop the program as an information retrieval system and
for use in semi-automatic dictionary building. Contacts are being laid with institutes
willing to cooperate.
See:"electronic dictionary".
EncY: ID1.ASC                                   Cont: ID1
Sources:   V1:   V2:   V3:   D: jmm   C:
_____________________________________________________________ Search:
N)ext, P)rev, K)ey, E)dit, W)ipe, Q)uit?
Search key = ENTRY
```

fig. 4: example of a dictionary entry

A lot of information can be put into one record. Sources can be specified for the various translations and the definition.

Access to more extensive information is often essential with technical and scientific terms. That's why ID offers the possibility to link up each entry with an outside (ascii-) file.

The 'EncY' field is used to refer to this file. The user can consult this outside text from ID.

The 'Cont' field contains the names of up to 30 text files in which the entry occurs. You can return to the original contexts of the entry in a flash.

Starting from this or any other record, you can do various things:

– N) ext (or down arrow) takes you to the next entry. The result depends on the key or key combination you used when looking up the record. You do not have to remember this, as it is marked at the bottom of the screen. In our example it reads: 'Search key = ENTRY', meaning that the next record will be the next entry in the alphabetical order, independent of the semantic and grammatical categories.

If you used the semantic and grammatical keys in your search, then the next record would be the next entry within the semantic and grammatical (sub)categories specified.

– P) rev (or up arrow) takes you to the previous entry. The result will again depend on the key or key combination used in the search.

–K) ey allows you to change the key or key combination in use for the search, without returning to the previous level. This gives you extensive flexibility in accessing the database, while you are in it. This way you can at any time zoom into a specific grammatical or semantic (sub)category.

E) dit allows you to edit the fields of the record. Most common editing features are available. These even include word wrapping in the definition field.

– W) ipe deletes the record shown and takes you back to the previous level.

– Using the letters that are highlighted in the field names, you can go straight to the corresponding field: '1', '2', '3' take you to the corresponding translation field, 'D' takes you to the definition field, 'C' takes you to the 'Cont'-field and 'Y' to the 'EncY'-field.

When editing the semantic and grammatical fields and when adding a record or searching for one, you can call up the list of existing keys with a function key (see fig. 5).

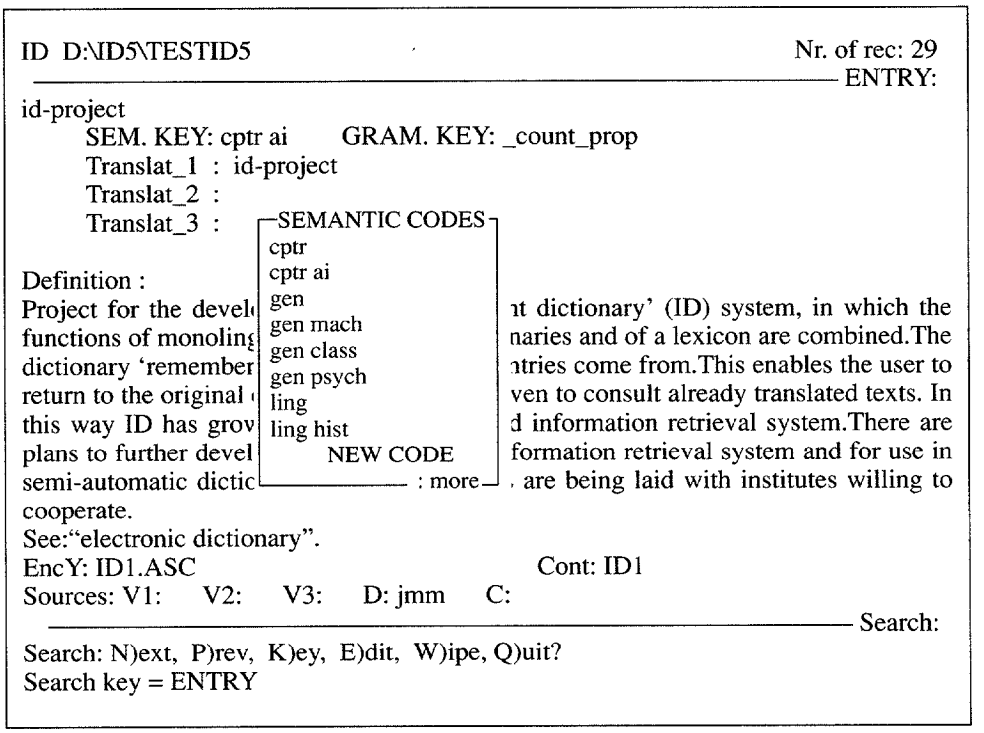

fig. 5: example of a dictionary entry with semantic codes

You can move the highlighting using the cursor keys, or you can type in the first few letters of a semantic (or grammatical) code. If you press the enter-key, the semantic (or grammatical) code will be written into the field and the cursor will move on to the next field. If you enter a new code this list of codes is automatically updated.

Cross references can easily be made throughout the definition field. You only have to mark a cross reference using F2 and F3 for the beginning and the end of the reference respectively. You can go straight from one cross reference to the other.

If the record you want to refer to does not exist, it is automatically created when establishing the reference for the first time. We have the reference 'See: «electronic dictionary» in the definition field of our example. When the cursor is anywhere on the reference, you can jump to that record by entering F5. If you like, you can immediately edit the record or move on to another record. Whatever you do on this level of the program, even if you change the key (combination), even if you use other cross references, you can always return to the original record you started from. There is no limit whatsoever as to the total number of cross references. The only limit is the space available in the definition field. Using this possibility a user can create and explore his own paths through the database without even a chance of getting lost.

From the 'EncY' field, you can jump to an ASCII-file containing extra information using F5.

ID's browser has many features, of which I can only mention a few. You can select lines for printing or writing into a file, you can locate terms, put markers in the text, move

freely to and fro, (etc.) and there is a help function showing you how to do all this (called by F1).

In the same way you can select a source text using cursor keys (up to 30 text references per record can be stored), and jump to the original contexts in which the entry was used. The first occurrence of the entry is highlighted.

Each of these source language texts can have a target language 'countertext' (with the same name, but a different extension). You can call these up in the same way as the source texts using another function key. In this way each record potentially opens up 60 texts of indefinite length, allowing the user to compare translations in the various SL and TL texts.

This system of immediate access to SL and TL texts offering numerous examples of real usage in real context is of course highly preferable to the classical description of contextual usage in (maximum) a few lines.

GENERAL FUNCTIONS

If you give in 'G' for 'General', you can get a general view of what information there is in the database. You can again make use of the zoom function. Suppose you want to know what *nouns* are in the base that have to do with *computers*. You simply enter the (partial) keys you want to select on in the command line. The result is a list of entries with semantic and grammatical codes that answer your criteria.

You can not only get this general information on your screen, the partial or complete contents of those records can also be sent to the printer or to a text file.

IDRES AND UTILITIES

IDRES and IDSWAP are resident subprograms of ID. These programs enable you to use any dictionary from the word processor of your choice. You can jump straight from a word to the record in the dictionary, or you can start at the first level (add, search, general functions). Translations can be pasted into the underlying text immediately or they can be transferred to a paste buffer for pasting later on. You can load another dictionary anywhere on your hard disk without having to leave your wordprocessor. IDRES and IDSWAP offer exactly the *same dictionary functions* as the main program ID, but for the browsing utilities, which cannot be used in IDRES. IDRES takes up less memory than IDSWAP (about 160 KB). Both programs can make use of expanded (LIM-EMS) or extended memory (via a RAM-disk) or they can swap to disk. They then only take up 8 KB of RAM-memory.

They can be unloaded to free memory, so that rebooting is not necessary.

ID has a number of external utilities. The most interesting one allows you to merge dictionaries or subsets of dictionaries. In doing so you can make use of the semantic and/or grammatical codes to select a particular subset of a dictionary for merging.

CONCLUSIONS & PLANS

I have presented the reader with a general overview of the program. Of course, there is much more to it than can be mentioned in this article. It will be clear, however, that ID is a powerful tool for anyone working in the translation field and even for anyone working with languages.

The most important features that set ID off against other programs are:

- the flexibility in accessing the dictionary in an intelligent way using either the headword or the semantic or grammatical keys or any combination of them.

- the highly integrated dictionary functions allowing the user to perform most actions within the same level of the program (even changing the search key while consulting a dictionary entry);
- the possibility to return to the original contexts from the frequency list;
- the possibility to return to the original SL and TL contexts from the dictionary;
- the possibility the user has to refer to additional information in a text file and to access that information in a flash;
- the system of cross references which offers the user unlimited possibilities to travel through the database in his own way;
- the general flexibility and ease of use of the program.

ID needs only a (compatible) personal computer with a hard disk (PC XT minimum) running under DOS. A PC-AT is not necessary, but it speeds up the program considerably. The development of a specific network version is planned for next year.

There are plans to expand ID as a *lexicon-based information retrieval system* on the one hand and as a *semi-automatic dictionary builder* on the other.

We would like to work with others to achieve this. A few examples of what we are thinking of:

- It would be very useful if one could draw up a hierarchically structured semantic and grammatical classification, which would function as a superstructure for the user of ID, so that he himself need not develop the grammatical and semantic classification.
- Developing (or linking up with existing) lemmatizers for the most important languages would greatly enhance efficiency in creating frequency lists. Moreover this would make it possible for the program to generate the correct grammatical code automatically.
- We are considering linking up this hierarchically structured typology (which should basically be language independent) with language specific frames. This would in fact mean that a grammar would be integrated into ID.

Such a project would enable the developers to gather extensive know-how in the field of lexicography, terminology and computational linguistics. The program as it stands now, is a very useful tool for anyone in the language field. Its modularity opens up the way for future extensions. The feedback we get from the experiences users have with the program, can be of great help in its further development.

At this moment ID is only available from the author. Talks are going on with EUROTERM (Maastricht). They have agreed in principle to using ID for bringing ELSEVIER's well known technical dictionaries onto the market in an electronic form. It would be possible for the user to build on those existing dictionaries (editing, adding, etc.) for his own purposes.