Klug, Helmut W., and Roman Weinberger: 'Old English Plant Names Go Cyber: the Technical Aspects of the *DOEPN*-Project.' In: Old Names – New Growth: Proceedings of the 2nd ASPNS Conference, University of Graz, Austria, 6-10 June 2007, and Related Essays. Eds. Peter Bierbaumer and Helmut W. Klug. Frankfurt/Main: Lang, 2009. 181-209.

This volume is dedicated to Carole P. Biggam, Honorary Senior Research Fellow and Visiting Lecturer at the University of Glasgow, who by the foundation of the Anglo-Saxon Plant-Name Survey, decisively revived the interest in Old English plant-names and thus motivated us to organize the Second Symposium of the ASPNS at Graz University.

“What's in a name? That which we call a rose by any other name would smell as sweet …”

Shakespeare, *Rome and Juliet*, II,ii,1-2.

Preface

Whereas the first symposium of the ASPNS included examples of research from many disciplines such as landscape history, place-name studies, botany, art history, the history of food and medicine and linguistic approaches, the second symposium had a slightly different focus because in the year 2006 I had, together with my colleague Hans Sauer, started the project 'Digital and Printed Dictionary of Old English Plan-Names'. Therefore we wanted to concentrate on aspects relevant to the project, i.e. mainly on lexicographic and linguistic matters.

Together with conferences held more or less simultaneously to mark the occasion of the 300th anniversary of Linnaeus' birthday in Sweden, this resulted in fewer contributors than at the first symposium. As a consequence the present volume in its second part also contains three contributions which are related to the topic but were not presented at the conference: the semantic study by Ulrike Krischke, the interdisciplinary article on the *mandragora* (Anne Van Arsdall/Helmut W. Klug/Paul Blanz) and - for 'nostalgic' reasons - a translation of my first article (published in 1973) on the Old English plant-name *fornetes folm*.

The articles in the first part can be divided into three groups:

1. Those directly dealing with lexicographic and linguistic matters: Antonette diPaolo Healey, main editor of the *Dictionary of Old English*, deals with the plant-names *foxes glofa* and *geormanleaf* , illustrating various problems from the point of view of her work for the *DOE*. Inge Milfull, Oxford University Press, looks at the treatment of the Latinate OE plant-names *pulege* and *psyllium* in the *Oxford English Dictionary*. Eric G. Stanley, one the doyens of Anglo-Saxon studies, shows that the Old English names of the cedar tree and of the hyssop are, with the exception of the name *hlenortear* glossing *hyssop*, loan-words and occur mainly in biblical contexts. Prof. Hans Sauer and his assistant Ulrike Krischke describe the Graz-Munich project of the *Dictionary of Old English Plant-Names*, focusing on etymology, word-formation and semantics.
2. Articles dealing with more general plant-related topics: Ann van Arsdall, who came all the way from Albuquerque, New Mexico, shows in her article on the mandrake in Anglo-Saxon England that a great amount of detail of the 'mandrake and dog-legend' was unknown at

3. the time. Maria D´Aronco, the great Italian expert on medieval herbals convincingly argues that in spite of the undoubted merits of de Vriend's edition of the *Old English Herbarium* and of the *Medicina de Quadrupedibus* a new edition of these texts would be desirable. Della Hooke, who specialises in aspects of the Anglo-Saxon landscape, demonstrates in her article on tree names in Anglo-Saxon charters that an enormous amount remains to be understood about early medieval landscapes and arboriculture.
4. My assistant Helmut W. Klug, who is both a trained medievalist and an EDP-expert, and the EDP specialist and trained psychologist Roman Weinberger describe the technical aspects of the project 'Dictionary of Old English Plant-Names', which in the end should be quite a revolutionary 'clicktionary' of Old English botanical terms and might also become a model for similar specialized dictionaries.

I want to thank all participants for coming to Graz, but in particular I would like to express my gratitude to Eric Stanley, first for giving us the honour of coming to Graz, and second for suggesting the very apt name of the present volume. My very special thanks also go to Toni Healey, who over so many years kept my passion for plant-names alive by keeping me informed about the progress of the DOE and by occasionally asking my advice on plant-name matters, and to Maila D´Aronco, who during all those years maintained her interest in my work and remained a good friend and colleague.

I would also like to express my thanks to individuals and institutions who contributed to the success of the conference: The University of Graz represented by Prof. Gernot Kocher, Dean of the Faculty of Humanities, Prof. Helmut Mayrhofer, head of the Department of Plant-Sciences at Graz University, the Governor of Styria, Franz Voves, the Mayor of Graz, Siegfried Nagl, the head librarian of the Special Collections Section of Graz University Library, Dr. Johann Zotter, and the head librarian of the monastic library at Stift Admont, Dr. Johann Tomaschek.

Last but not least our thanks go to my friend and colleague Adolf Sawoff, who accompanied the opening ceremony with his guitar, and to the Knorr-Kohlhofer family, who provided us with excellent food and drinks on very generous terms.

Peter Bierbaumer – Graz, September 2008

As a specialist in German mediaeval studies, until the time Peter Bierbaumer introduced me to Old English plant names and approached me with the idea of republishing and updating his *Der botanische Wortschatz des Altenglischen* I had no idea how fascinating Old English could be. After browsing this special subject on the Internet and in scientific literature, the value of his undertaking was soon evident, both for the strong, active Old English community and for my personal studies in the fields of electronic data processing and mediaeval plant research. Fortunately the Austrian Science Fund (FWF) backed our project ('Digital and Printed Dictionary of Old English Plan-Names') in 2006. Today, we can look back on two years of hard work and number of things we have accomplished. One of those was hosting the 2nd *Anglo Saxon Plant Name Survey* Conference in June 2007, and another was publishing this compilation.

The conference was held at a time when the most tedious work of our project – the digitalisation of all three volumes of Peter Bierbaumer's books – had just been finished. We rushed to implement some of the basic research features and to input some of the data so that we could present a functioning online-platform at the conference. We greatly profited from helpful hints and tips from all participants for the work with and the development of the *Dictionary of Old English Plant Names*. All this input resulted in the idea to apply for funding for a follow-up-project (same title as the dictionary) that will generally broaden the research possibilities and the possibilities of user interaction. Funds were granted in early summer of this year and we received a very positive feedback from the project reviewers. This positive feedback obviously confirms that our project is headed in the right direction. The conference and the papers in this volume show the importance of a holistic approach towards the topic of mediaeval plants and their names: researchers must not be stopped by the borders set by his or her field of study. Risking excursions into and taking on the ideas of neighbouring studies nearly always is worth the effort and the results clearly justify the means.

Since Peter Bierbaumer deals with the organisational details in his introductory remarks, all that is left for me is to express my thanks to the following people: I want to thank Peter for giving me the opportunity to literally turn my hobby into my job with the projects on Old English plant names, and for all the help and encouragement I have received form him in the past. I want to thank Roman Weinberger for doing such a terrific job with designing and programming the *Dictionary of Old English Plant Names* web-site. I want to thank Anne Van Arsdall and Paul Blanz for the chance to co-author the paper on the mandrake in this volume – it was a very instructive and enriching experience. Finally I want to thank all the authors in this volume for their help and, most of all, for the patience they showed and the encouragement I received during the strenuous time of editing.

Helmut W. Klug – Graz, September 2008

Contents

Abstracts

Eric Stanley: 'The Cedar tree that is in Lebanon, euen vnto the Hyssope that springeth out of the wall'

Botany is a difficult subject, and the identification of plants with plant names is beyond my competence. The CEDAR, however, is easily recognized. In the Bible it is mentioned together with the HYSSOP, the mighty tree and the little plant. The wisdom of Solomon is exemplified by his willingness to discourse on great things and little, on the CEDAR and on the HYSSOP. Old English literature is usually the work of monastics, and these two plant names therefore occur often. The exact meaning of Modern English plant and the etymology of CEDAR and HYSSOP are briefly discussed in this paper. A biblical crux involves the HYSSOP, Christ on the cross is given a sponge on a HYSSOP to quench his thirst. Almost all the uses of the two words occur in contexts related to such biblical occurrences. The great, modern Reallexikon of Germanic antiquities has no entry for either plant. Usually the Old English names are merely loan-words based on the Latin, but once the name appears as *hlenortear.* HYSSOP is used in various biblical cleansing rites, and these too are referred to in Old English, and in a number of medical texts.

Maria Amalia D'Aronco: The edition of the Old English Herbal and Medicina de Quadrupedibus: two case studies

In 1984, more H.J. de Vriend published a new critical edition of the *Old English Herbarium* and the *Medicina de Quadrupedibus* for the Early English Text Society. These two tracts are vernacular synopses of various Latin pharmacological texts that circulated throughout western Europe from late antiquity to the middle ages and beyond. They are attested in four witnesses, an extraordinary exception in the history of OE culture where the texts have been generally preserved in sole and unique survivors. It is the very nature of the manuscript tradition of the two Old English pharmacopoeias that prompts me to comment on de Vriend's actual editorial practice. Therefore, the main scope of this paper concerns not so much the undoubted merits of de Vriend's edition as various observations about specific aspects of his edition. In particular, I shall focus on two more general characteristics: his treatment of variant readings, and his handling of scribal emendations

Anne Van Arsdall: Exploring what was understood by 'mandragora' in Anglo-Saxon England

In the Latin and Anglo-Saxon herbals, the mandrake plant appears as a medicinal herb that should be collected using a dog. In fact, the dog and mandrake are ubiquitous in drawings. The purpose of this paper is to show that over the years, editors and art historians have added a great amount of detail about the mandrake and the dog when discussing works from Anglo-Saxon England, or Continental works known there, detail that was most probably unknown at the time.

Della Hooke: Trees in Anglo-Saxon charters: some comments and some uncertainties

Tree names are an important component of early place-names and documents and most native species of tree can be found. A few species, however, remain elusive while other names cannot be accurately or certainly identified. Despite the efforts of place-name scholars, it is also still difficult to be precise about the actual use of some Old English woodland terms and an enormous amount remains to be understood about early medieval landscapes and arboriculture.

Antonette diPaolo Healey: Perplexities about plant names in the *Dictionary of Old English*

In this essay, I first situate DOE's treatment of plant names in relation to other specialized vocabularies, such as etymologies, place names, and personal names. I then suggest the strategies employed by the DOE for handling plant names, including DOE's usual treatment of the morphological type noun in the genitive + noun, such as *foxes glofa*, as a phrasal unit rather than a genitival compound. I next look at three specific problems, all devolving around issues of palaeography, a concern as valid, I argue, as phonology, morphology, syntax, semantics and taxonomy in our discussion of plants and their Anglo-Saxon names. Finally, I describe how the palaeographic issues around the forms *geormenletic*, *gearwan leaf*, and *reosan* have been handled, if not resolved, in the DOE.

Inge B. Milful: PULEGE and PSYLLIUM: Old English plant names in p- in the *Oxford English Dictionary*

After discussing some recently revised plant name entries in the *Oxford English Dictionary* (*OED*), this paper looks at the treatment of two Old English plant names, PULEGE n. and PSYLLIUM n., in particular, and focuses on our treatment of Latinate forms of problematic status. We have decided to include these in our entries, as the entries themselves were transformed by our increasing awareness of a continuity of the use of Latinate forms of these plant names in the history of English, in particular in medical and pharmaceutical use.

Hans Sauer, Ulrike Krischke: The *Dictionary of Old English Plant-Names (DOEPN)*, or: The Graz-Munich Dictionary Project

Although *ca.* 1300 different Old English plant names are attested, no comprehensive Old English plant name dictionary exists. It would be useful to have one, however, because in the extant dictionaries the entries on plant names are scattered and information about them is often brief and fragmentary. Therefore we have embarked on the Graz-Munich project with the aim of compiling *The Dictionary of Old English Plant-Names* (*DOEPN*). It will provide the inventory the plant names as well as their attestations; it will also explain and where necessary discuss the meaning of the names and the identification of the plants; furthermore it will give linguistic information about the names, especially as regards etymology (origin), morphology (especially word-formation) and semantics (meaning and motivation). In the present article we explain the scope of the *DOEPN* (inclusions and exclusions), the structure of the entries and we provide a number of specimen entries.

Helmut W. Klug, Roman Weinberger: Old English plant names go cyber: the technical aspects of the *DOEPN*-Project

The fwf-funded project 'Dictionary of Old English Plant Names' is based on the work on this subject carried out by Peter Bierbaumer in the late 1970's. Our intentions are to update it not only with regard to scientific research but also in technical aspects. The three volumes of *Der botanische Wortschatz des Altenglischen* had to be digitalised: this paper provides a glimpse at how it was done and which problems were encountered. We also want to give a thorough report on the design process that spawned the sql-database which is the solid foundation of the dictionary: there will be an excursus into database and web design theory, a detailed description of the database in relation to its contents, and on techniques for data input and retrieval. This sums up the technical groundwork of the backend of our web application. It is meant to give people normally not involved in technical matters a basic understanding of database theory. The frontend – the future public portal to Anglo-Saxon plant names – is heavily 'under construction': some features are already implemented, the majority, though, is still a bunch of wild ideas. Both present and future applications are dealt with in this context.

Ulrike Krischke: On the semantics of Old English compound plant names: motivations and associations.

Complex plant names reveal a lot about the way the Anglo-Saxons perceived and experienced the natural world. In this paper, the morpho-semantic make-up of the Old English compound plant names that appear in the sections *nomina herbarum* and *nomina arborum* of abbot Ælfric's *Glossary* are examined and morphological aspects, motivation categories and the associative relations holding between source and target concepts are discussed. The alphabetically arranged list of plant names in the appendix provides information on the identification of the plants, on the morphologic shape and structure of the plant names as well as a detailed discussion of the motivation and the associative relations of each plant name.

Peter Bierbaumer: Old English FORNETES FOLM– An orchid.

This contribution is a translation of my article "Altenglisch *fornetes folm* – eine Orchideenart", published under the editorship of Helmut Gneuss in *Anglia* 92 (1974), 172-176. I have included it mainly for the "nostalgic" reason that this was my first publication on an Old English plant name, which already shows my line of reasoning, based on a thorough concern with detail and a lot of enthusiasm for the subject. In this article I argue that the plant name *fornetes folm*, 'hand of Fornet', denotes a kind of orchid because it is used as an aphrodisiac in the *Lǣcebōc* and because the word *folm* points to a plant with a hand-like appearance. These two conditions

apply in particular to orchids, e.g. to *Orchis maculata* L., cuckoo flower, German *Knabenkraut*.

Anne Van Arsdall, Helmut W. Klug, Paul Blanz: The mandrake plant and its legend: a new perspective

This paper demonstrates how the contemporary legend about mandrake plant evolved from classical through early-modern times. A major misconception about the Middle Ages and the era directly preceding it is an assumption that the different elements of the mandrake legend were always widespread and well-known. Our paper stresses the importance of distinguishing different stages in the mandrake legend in the centuries from ca. A.D. 500 to 1500, showing that not all concepts we know today were associated with the plant at any given time or place in the past. We base our research strictly on historical documents (illustrations, literary and botanical/pharmaceutical texts) carefully correlated in time. Our findings bring an important corrective to many folkloristic assumptions about the mandrake legend that have been handed down and accepted at face value for years. In fact, more research is needed to pinpoint when and where various elements of the legend originated and how (and how far) they spread, especially for the time after the 12th century.

Old English plant names go cyber: The technical aspects of the DOEPN-project

Helmut W. Klug, Roman Weinberger

The basis of our[1] work are the three volumes *Der botanische Wortschatz des Altenglischen* by Peter Bierbaumer (*I. Teil: Laeceboc; II. Teil: Lacnunga, Herbarium Apuleii, Peri Didaxeon; III. Teil: Der botanische Wortschatz in altenglischen Glossen*) which were published in 1975, 1976 and 1979 respectively and have since been regarded as a major achievement in the field of Old English plant name research. Almost 20 years later there has not only been an enormous increase in research on this topic but also a technical revolution providing the single PC user with possibilities even the computerised research centres of the 1970's did not have. These two facts encouraged the birth of the *Dictionary of Old English Plant Names*. Our project wants to sum up recent research and compare it to Bierbaumer's conclusions. But it also wants to present both old and new data in an updated and forward looking way: how this is done and what problems we encountered so far will be the main concern of this article. We want to introduce our work flow management, give some detailed examples of the more exciting aspects and want to present the possibilities an on-line dictionary offers both the producer and the user. We not only want to create a product that is up to date in the field of Old English plant name research but a product that also covers the given web standards and takes into account the latest trends in web programming and web design.

1. Data Acquisition

As already mentioned above Bierbaumer's work is the basis for our data pool and since the books were written on a typewriter, no electronic copy is available. Thus, *Der botanische Wortschatz des Altenglischen* had to be digitalised with the help of OCR-technology, which "is the mechanical or

1 'We' and 'our' in this article generally refers to the authors (*e.g.* all passages that explicitly deal with technical details as the OCR-test-series or the basic database design, *etc.*). Only exceptions are passages that deal with the project in general: here the pronouns refer to the whole project team (like this very passage, or *e.g.* decisions on database content).

electronic translation of images of handwritten or typewritten text (usually captured by a scanner) into machine-editable text" (*Wikipedia:* 'Optical Character Recognition'; accessed Dec. 11th 2007, 04:05.).[2] Everybody will know advertisements for OCR-software, where tons of tightly written papers are digitalised in no time ... this clearly is the world of advertising! For some of the more developed programmes the advertised 99.9% recognition accuracy may of course be true – provided the source material is flawless. Our material is all but flawless: we have about 660 pages of text, which were typed on three different typewriters, which all have a different type face: two have more Arial-style letters and the typewriter of the second volume produced Courier-style text, the text also contains a lot of special Old English characters, and underlining,[3] which distorts the individual image of a single character, has been used a lot for pointing out different aspects.

When starting with our work we had scans of the books available, thus we could immediately start training the programme to read our text-material. We used FineReader Pro,[4] which is a high-end consumer product; statistic analysis has shown that the training was and still is the most important part of working with OCR-programmes, especially if you have to recognise a vast amount of text. Although most OCR-software is able to recognize letters on its own due to pre-stored character definitions, each programme should be trained when working with large amounts of text because recognition accuracy will increase. In our case even more so, because we have to work with Old English and its special characters and a

2 This part of our talk has been left out in our presentation on June 8th, because we decided to concentrate on more 'thrilling' examples and information, like the structure of the database or different possibilities of data in- and output, which will be dealt with below. But we want to include it in this paper because there is a lot of practical advice to be gained from.

3 Underlining was used to label Old English text (Bierbaumer, 1975: XIV).

4 For more information see: http://www.abbyy.com/ (accessed June 2008). We started off with version 7.0 of the programme and near the end of our OCR-work we switched to version 8.0, which did not have any striking influence on the recognition process (and the results of our OCR-test). OCR-accuracy might have increased a little bit from v7 to v8 but the problem of poor source material still remained. A recent visit to the Abbyy website revealed that in the meantime version 9.0 of the programme has been released.

lot of underlined characters.[5] For this purpose we chose 2-5 pages of each book to train the software on: the programme tries to recognise each character correctly on its own and the user has different means to correct the result. If the programme is right on the first try, the user verifies the character and moves on to the next one, if the software calculated wrong, the user can either correct the recognition area (the area may be too small, so that the character is only partly covered or it is too large so that there are more than one characters selected) or correct the character the software proposes (when it wrongly recognises the selected area of the text). This also worked fine for two or more characters "glued" together which originated from the use of underlining as a means of denoting Old English Words: This means that for the software 'wyrt' is a completely different character set than 'wyrt' because each single character is combined with the line below it. Since the underlining was of no more use in the electronic version of the text the programme had to be taught that *e.g.* 'w' = 'w' = 'w' = 'w' → 'w'. Another obstacle was that Fine Reader only manages to automatically recognise and print characters of the western alphabet, so the training on special Old English characters like thorn (þ) or even the long vowels (ā, ē, ...) was most important. The software was trained to read those images and substitute them with a certain code built of standard characters, which in the text processing programme could then be replaced automatically (*e.g.* 'th&' → þ). This may seem an elaborate and time consuming task, but on the whole scale these few hours of training can save a great amount of time during the actual recognition process.

When preparing this part of our presentation we were curious how well the OCR-software would work with our source material, and we did a little empirical research which produced really interesting results: the OCR-program offers different recognition patterns which can be combined in different ways. The first part of the following list of combinations is the type-setting and the second part refers to the character patterns used for the recognition process: 'automatic' type-setting means that the software tries to recognise the type automatically, 'typewriter' specifies the use of this device in the source material, 'no user-pattern' *vs.* 'user-pattern' defines if our previously collected character set is used or not – if not the software tries to recognise characters automatically. The combination of both techniques adds the user-created character set to the pre-stored one.

5 The feature of collecting your own user character sets also defines the quality of the OCR-programme. There are only a few which have this feature.

This provides us with following combinations:

- automatic, no user-patterns
- typewriter, no user-patterns
- automatic, user-patterns
- typewriter, user-patterns
- automatic, internal and user patterns
- typewriter, internal and user patterns

The test set-up is as follows: we put together representative text parts[6] from all three volumes each containing 100 characters plus 62 blanks and paragraphs. That is a total of 362 instances which the program has to recognise correctly.

The first count seems to verify advertising slogans with automatic recognition bringing the best results and the trial-runs that make use of the user-pattern having a much higher error-count. But after checking back with the recognised text we encounter a major problem: proof-reading is much more difficult with the automatic text because it has many more misinterpreted characters that are mixed with correctly recognised characters than for instance in the 'typewriter, user-patterns' combination, where only correctly recognised letters are given, while wrongly recognised ones are substituted with a caret (^). It is this distinction that makes the process of proofreading much easier and also a lot faster. To factor this difference in correction time into our test set-up a second error-count was made where characters not recognised counted as one and those wrongly recognised as three error-points. In the resulting diagram (see figure 1) the typewriter / user-pattern combination leads the field. This is the method which provides the least time and energy consuming OCR-procedure. This fact could also be verified by time taking during another OCR-run on a completely different sample (a randomly chosen page). Automatic recognition still recognises most of the characters but searching for wrongly recognised letters is much more demanding on the user.

All known OCR-problems have been verified: poor recognition-accuracy with typewriter-fonts, underlined passages, or problems with special characters. Our empirical study has shown that it is most important

6 We tried to produce a sample which has the overall character set of each book. The scanned pictures were cut up and different lines were pasted together in a new sample picture. This, of course, matched the image quality of the source material.

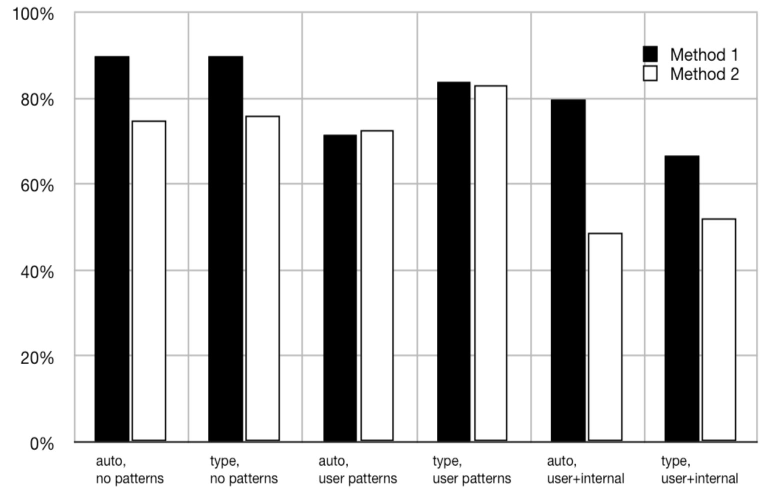


Figure 1: Possible recognition methods: the 'typewriter, user pattern' method is the fastest approach.

to test and train the OCR-software before starting with the recognition-process of a huge amount of text, because only a thorough error-count and analysis of the work flow can show which parts can be optimised, so that the workload can be tackled in the most efficient way.

2. Data Structuring

2.1 Database theory

Once again the starting point are Bierbaumer's books: in this case it is the single 'prototype' plant name entry which was reconstructed from (a) the description of the entries by Bierbaumer (1975: XII, 1976: XII, and 1979: XLIII-XLV), (b) selected entries of each book,[7] and (c) personal experience gained while working with the data.[8] With all this data at hand the process of re-shaping it into a database-fit structure could start. But before we go into

7 We mainly concentrated on the entry *bēowyrt* which can be found in all three volumes and is rather extensive.

this process, there are some technical concepts and terminology to be defined, to make sure that we can start off from a common basis:

> Simply put, a database is a structured body of related information. [...] You can think of a database simply as a list of information. A fine example is the white pages of the phone book. Each listing in the white pages contains several items of information – name, address and phone number – about each phone subscriber in a particular region (information). All subscriber information shares the same form (structure). In database terms, the white pages comprise a table [*i.e.* a concept] in which each subscriber is represented by a record. Each subscriber record contains three fields [*i.e.* properties]: name, address, and phone number. The records are sorted alphabetically by the name field, which is called the key field. [...] You can model and design a database to store anything which can be represented as structured information. (O'Neil 2004: 2-3)

This definition provides us with the basic features every Database Management System needs to support. In addition to that we will now take a closer look at the relations between pieces of information stored in a database. As a simple example we can consider a database that stores information about people (for example name, age). In addition to that it also contains a second table that stores information about countries. Now we want to record which country some person lives in. We have two entries in different tables in our database: one in the 'people' table and another one in the 'countries' table and we want to link them with a 'lives-in' relationship. To be able to define such a relationship, we have to introduce an additional column, a unique identifier, and add that one to our records. Such a column is often called 'id'. Using these id fields we now have an easy way to reference our records. If we take a closer look at the 'lives-in' relation we can see some important properties: each person (*i.e.* the record in the 'people' table) can only live in one specific country (*i.e.* the information this table holds) and each country can have many different inhabitants. This information makes defining this relation rather straightforward: We can simply add an additional column to our record in the 'people' table that references the id field of the desired record

8 This experience is mostly based on ideas of how data retrieval could be made more efficient and interesting: so, for example, we invented the concept 'type' which defines a lemma according to its entity: if it either denotes a plant, a part of a plant or a plant-product.

in the countries table. Doing this we have defined our relation that allows us to perform various database queries like "how many people live in country 'England'" or "is person 'x' an inhabitant of country 'Germany'". This 'lives in' relation is, in database terms, called '1:n' and in natural language it can be said that 'a person belongs to a country' as well as 'a country has many people'. This is only one type of relation. In database design three main types of relation exist. We want to take a detailed look at them now :

The first type of relation is formally defined as 1:1. It is symmetric and defines the relation between exactly two records in different tables. The arguments for using a 1:1 relation are data safety, cleaner programming style and in some rare cases also database performance. It can be seen as a specialized variant of a 1:n relation which we will cover next. It is the most commonly used type of relation and can be expressed as 'has many' or its inverse 'belongs to'. As demonstrated in the example above, it can, for example, be defined by introducing an additional column that references another table (often called 'parent') into some other table (often called 'child').

The third big type of relation is m:n that can be expressed as 'has and belongs to many' and defines a record that can have many child records which in turn can have many child records themselves – and these are stored in the same table as the parent record. This type of relation cannot be defined by adding a column to some entry. Instead, another table, often referred to as 'link table' needs to be defined: this table stores at least the id of a record in one table and the id of a record in another table. This method can be extended by adding additional properties to the link table. An example: two tables, one called 'teas' and the other called 'ingredients'. Now we could define that each tea has-and-belongs-to-many-ingredients, but it would be convenient to have another column: the amount of each ingredient. Doing this we would be able to say 'Tea A' contains '10g' of 'Ingredient C' while 'Tea B' contains '5g' of 'Ingredient C'. What we have done now is to define properties that are specific to the relation but not to one of the records: this information is stored in a column of the link table. The benefits of m:n as well as 1:n relations are, among others, increased data integrity, reduced data redundancy, and increased database performance. There are more complex forms of relations, like recursive definitions, but in this context it is more than sufficient to cover the most common types.

The process of deriving a database structure (also called 'scheme') using various forms of relations is called database normalization. The degree of normalisation of a database is called the 'normal form'. The first normal form (1NF) demands that a table has an 'id' field and that each column represents an 'atomic property' (*i.e.* smallest meaningful entity) and no 'null values' are allowed. An example of an atomic property would be a column 'name' that would violate the first normal form whereas two columns 'first-name' and 'surname' would be valid in first normal form. The second normal form (2NF) demands that the database is in first normal form and that each relation only defines one aspect of data. This is the main step in reducing redundancy. The third normal form (3NF) finally removes last redundant information by demanding that the data must be in second form and that each property in a table is directly and not transitively dependent on the 'id' of its containing table. Additional normal forms (4NF, 5NF…) exist, but those are of interest mostly for transaction-oriented database design and not so much for typical read-oriented web applications. With this short excursus we have covered the cornerstones of relational databases. Now we want to take a look at some of the details and the nomenclature behind it.

The first important concept is the 'key'. Keys in relational databases can occur in various flavours like super keys, candidate keys (minimal super keys), alternate keys, compound keys, and foreign keys. For practical usage, and most important, are the primary keys (here we have natural and surrogate variants) and the foreign keys. The primary key is the property that is most often used in relational databases to uniquely identify a record in a table. Considering our examples above, the introduced 'id' fields would be primary keys. The foreign key on the other hand is a reference to a primary key stored in another table: so coming back to our initial example, the id of a country stored in the people table would be a foreign key.

The next important factor to consider is the column attribute. Column attributes are more or less constraints for what you want to store in your columns. There are many different column types, but the basic ones are: INTEGER columns that are used to store numbers, FLOAT and DOUBLE columns that are used to store floating point numbers with single or double precision. VARCHAR or STRING columns are used for storing size restricted text entries with typical restrictions being less than 1024 characters. TEXT columns are normally used for storing size unrestricted amounts of text. DATE and TIME columns are used for storing just that. The last common column type is called BLOB (for binary large object) and can be used for storing arbitrary data (*e.g.* like images). However, it should be used with caution, as storing binary data in the database provides no real

advantages over storing just reference to files in the file system, but often leads to a huge increase in the size of a record, which causes slow database performance.

A final concept that is very important for practical database usage is the concept of indices. An index is a table that helps optimising the way in which a database accesses its content. Beside the simple index, there are various different types of indices that mostly depend on the type of database used. All have in common that after an index is defined for some column, the database management system will build some internal data structures like binary trees that increase the performance of searching in the indexed column – much like a table of contents speeds up scanning through a book. It is of importance to note that most database management systems enforce restrictions on which keys can be used for a column depending on the column type. It is, for example, normally not possible to define a simple index for a field of type TEXT, whereas it is no problem to define one for a field of type STRING. This is one of the core points to watch out for when designing your tables.

The design of a database should satisfy two goals – we want to be able to store all needed data in our tables and we want to store and retrieve our data efficiently and without anomalies.

To satisfy the first goal, we need to determine what information our field of research produces. The next step is to split up this knowledge into concepts and to define the properties that are needed to further specify these concepts. Though this sounds rather simple, it is often the most difficult part of a good database design, as it depends on expertise concerning the data as well as technical competence. Hence, most of the time vast collaborative efforts are needed to complete this task. The following chapter is a summary of this process and a detailed description of the database that builds the foundation of the *Dictionary of Old English Plant Names*.

2.2. DB layout

Based on database theory as explained above we started to dissect our prototype plant name entry and to design our database. For means of easy data input our database can be accessed via the 'backend' of our website: http://oepn.uni-graz.at/main_entry/list. The result of our analysis was the following structure, which is also summarised in the flow-chart (see figure 2).

A) The concept 'main entry' is the core of our database and all following information is linked to this data (*i.e.* to this entry's ID). It also contains the smallest possible denominator that links to all the concepts we have defined: the **lemma** (or headword, or key word) in an attested spelling. In the main entry a lemma can be described with basic linguistic features: the **type** specifies the entity (plant, plant-part, plant-product). **Word-class** specifies the part of speech, *i.e.* if the headword is a noun, adjective, verb, past participle, or affix. In most cases the **gender** (m., n., f., and combinations) and **declension** (a, ō, i, u, n, r, s, root *etc.*, with subtypes) of the Old English noun can be reconstructed. This information can be the basis for grammatical analysis of Old English plant names. The main entry also contains information on **spelling-variants**, **derivatives** and **compounds**, **place names**, and links to spelling variants.

B) Spelling variants include not only actual spelling variants but also all morphological variants of the lemma (*e.g.* variants for *bēowyrt* are *beovyrt, beowirt, beowyrt, biouuyrt, biowyrt, buuyrt*). Newly added properties like derivatives (*e.g. plant*: *plantian*), compounds (*e.g. āc*: *āccynn*), or place names (*e.g. āc*: Acomb) will not only add to the possibilities of data retrieval and to the means of internal reference (also see C) but an entry-count based on this data will also be a good index of the cultural importance of the plant.

C) The concept 'literature' includes the following properties: **type** (primary texts *vs.* literature of reference), **sigle** (abbreviation), **author(s)**, **title**, **subtitle**, **appendages** (additional bibliographic information), **date of origin** (*i.e.* date of production of a MS), **publisher**, **city**, **year**, **page**, **journal**, **issue**, **signature** (of the library the book / article can be found in), **library** (which has a copy of the book), **notes** (notes on the topic, usefulness, *etc.* of the book / article), **ISBN**, **ISBN-13**, **verified** (all information for a correct reference is stored in the database: yes / no). This data originally has no relation. It is the basis for the project bibliography which, for example, could be easily extended into an annotated bibliography (*i.e.* the property 'notes' would be the place for annotations). While working on the different plant name entries multiple relations are created: different literature entries are linked to certain plant name entries, thus providing the data for a plant-bibliography, which simply is a list of all literature that mentions a certain plant name.

D) 'Internal reference' is a table that stores information on the interrelationship between entries, providing the basis for a wiki-style data output. These entries complement the data of the properties 'derivatives' and 'compounds' and contain links of general reference between entries. (*i.e. ācmistel* refers to *āc* and *mistel*; whereas *āc* and *mistel* both only refer to *ācmistel* via the property 'compounds'.) This table holds information on the **reference ID** and the **lemma** we refer to. It was introduced to meet the needs of the new medium internet.

E) 'Meaning' is a concept which holds those very pieces of information that link the Old English lemma (*i.e. bēowyrt*) to the present: the contemporary plant names. The property **variant** takes care of the fact that an Old English plant name can be associated with more than one contemporary name, variants are listed with capital letters (A, B, ...) according to plausibility of the correlation. **Assessment** holds the information on the the plausibility of the modern equivalent and is expressed by different combinations of question marks: no question marks = safe identification, '?' = probable identification, '??' = hardly tenable identification, '???' = wrong identification (which can be found in reference literature and has to be cited to provide a correct picture of the research done on the plant name so far). '**Bot_name**' gives the botanical plant name according to Linnéan nomenclature (*i.e.* Melissa officinalis L.), '**engl_name**' holds the most popular contemporary English plant name (*i.e.* balm), while '**ger_name**' is reserved for its contemporary German equivalent (*i.e. Zitronenmelisse*). The complete entry looks like this: A: *Melissa officinalis* L., balm, *Zitronenmelisse*.

F) The concept 'synonyms' is in direct relation to either the English or German popular name. This table has been introduced because the vernacular of both languages provides various names for the same plant. We try to provide at least some synonyms for each plant name so that data retrieval via the index or the name-search is bound to be more successful. The property **synonym** stores this data. For *bēowyrt* this would mean that (English contemporary name) balm is a synonym of 'common balm' and 'lemon balm' and (German contemporary name) *Zitronenmelisse* is synonymous with *Melisse* and *Bienenkraut*. Each synonym is not only directly related to the respective contemporary meaning but also indirectly to the lemma. This again increases the possibilities for data retrieval.

G) The table 'occurrences' holds all information on where a certain lemma can be found in an OE text. The properties needed are derived from Bierbaumer's original entries: *e.g.* "HA, 280/12f, asg, *þe man* ACANTON 7 *oþrum naman beowyrt nemneþ*" (1976: *bēowyrt*) or "Erf, 20, APIASTRUM *buuyrt*" (1979: *bēowyrt*). So the properties are **text** ('HA' for *Herbarium Apuleii* or 'Erf' for the Erfurt manuscript of the *Épinal-Erfurt Glossary*), the exact location (**source**: 280/12f respectively 20), the grammatical **case** (asg. *vs.* no case for the gloss) of the occurrence, the Latin plant name (*i.e.* **record_lat** which is intended for glosses only: APIASTRUM) and of course the Old English records (**record_oe**): *beowyrt* and *buuyrt*.

H) Since we plan to bring these citations in line with the conventions of the Corpus of Old English we have to build a 'translation-table' which equals the codes Bierbaumer used for referring to the texts to the correct Cameron number (*i.e.* **cam_no**) and Short title (**short_title**). Using the example introduced above, this table would have to hold the following information: Erf = (short_title:) ErfGl 1 (Pheifer) = (cam_no:) D36.1 and HA = (short_title:) Lch I (Herb) = (cam_no:) B21.1.1.2. This information will also be useful for providing standard links into the Corpus of Old English, so that the occurrences cited in our dictionary can be read in their actual context. (See Data Output.)

I) The concept 'comments' holds the text that explains why a certain meaning was chosen or marked as doubtful or wrong. For providing the correct relation of **comment** to the discussed meaning each analysis is connected to the concept 'meaning' by **meaning_id**. The Meaning "**A:** *Melissa officinalis* L., balm, *Zitronenmelisse*" thus has the following comment: "= Lat. APIASTRUM , APIAGO (*cf.* André s. vv.)" (Bierbaumer, 1979:23).

J) 'Footnotes' are mainly used to point out transcription or printing errors occurring in editions of Old English texts and of course to give reference for quotations. They are used within the concepts 'occurrences' and 'comment.' Due to the structure of Bierbaumer's work 'footnotes' have been organised as an independent concept. The dynamic data structure (the user is able to arrange the content according to his/her needs: *cf.* Data Output.) of our on-line dictionary required this step which now allows us to process the numbering of the entries automatically according to their position in the displayed text.

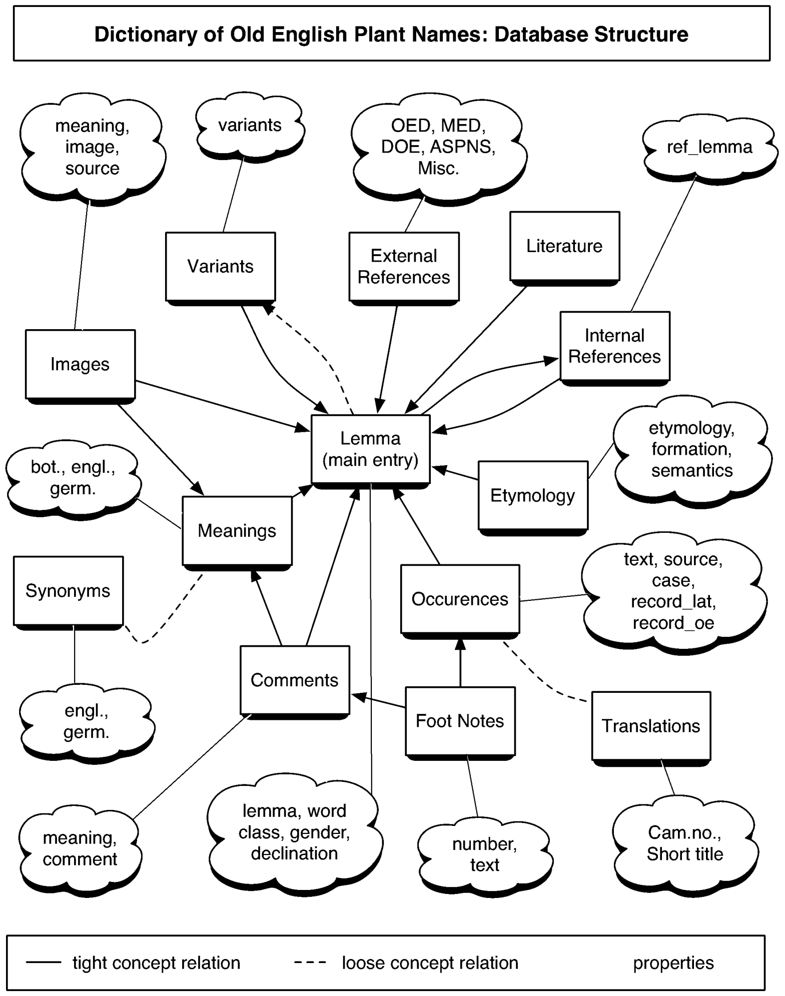


Figure 2.

K) The next concept is headed by the term 'etymology' and the underlying table is structured to meet the research of Hans Sauer and Ulrike Krischke. The main properties are **etymology**, **formation**, **semantic structure**, and **motivation**. The values for these fields are predetermined, input is solved via drop down lists. In case this data needs to be discussed in more detail, each property has an additional **comment**-field for verbal explanations. For *bēowyrt* the data would look like this: etymology: "loan-rendition" and etymology-comment: "OE loan-rendition from L APIASTRUM or APIAGO (itself probably a loan-translation from Gk *melissophyllon*); OLG *bini-wurt*, OHG *bini-wurz* are probably independent loan-renditions"; word-formation (noun/noun) without word-formation-comment.

L) 'External reference' is meant to provide the user with information on how (and where) to find the plant name in a broad variety of other dictionaries: Bosworth/Toller (**bt**) and supplements (**bts, btsc**), Clark/Hall (**clh**), the new Dictionary of Old English (**doe**), the Middle English Dictionary (**med**), the Oxford English Dictionary (**oed**), the Dictionary of the Old Scottish Tongue (**dost**), Scottish National Dictionary (**snd**), Wright's English Dialect Dictionary (**edd**), Holthausen: Altenglisches Etymologisches Wörterbuch (**aew**), the Oxford Dictionary of English Etymology (**odee**), Kluge: Etymologisches Wörterbuch der deutschen Sprache (**ewds**), and Marzell's Wörterbuch der deutschen Pflanzennamen (**marzell**). Special studies (books, articles) relevant for specific plant names only are listed under the heading '**misc**'.[9]

M) Another new concept is 'images', which allows the user to add pictures of plants to a specific meaning. This feature can, for example, be used to visualise our explanations provided in the comment. So, an optical feature of a plant which might have led to the Old English name can be shown: *e.g. cūslyppe*, for which the optical appearance of the decayed plant provided the basis (Bierbaumer 1976:42-43). Pictures can be used to point out the similarity or dissimilarity in optical appearance of different plants associated with the same Old English lemma. What is more, pictures also provide another easy means to link an Old English plant name to the present. The table stores information on the **image-relation** (via the meaning_id), its

9 This last property may be substituted by the functions described in C.

file-name (*e.g.* melissa_offic.jpg) and also **reference-information** (*i.e.* where the picture was taken from or who holds the copyright).

The next step in database design is the database normalisation process (see figure 2). Which level of data normalisation we want to reach is not always simple to answer, as it depends on the way we want to access our data. High levels of normalisation are very important for transaction oriented databases as they increase the performance of insert operations. For rather read oriented databases, like ours, too much normalisation can even lead to a decline in performance. After this step we are ready to implement our database and develop the software needed for data input and output. But before we discuss these topics more thoroughly the process of accessing data in a database has to be dealt with in a little more detail.

2.3 Database Access

To access the contents of a database, some appropriate software needs to be written, of course. You could use the low level database interfaces[10] of your programming language of choice, as it is often the case with web applications. This allows you to send arbitrary SQL queries to the database and get the expected results. For some developers this looks like the most natural way to access a relational database, but it has some serious drawbacks. Firstly, this approach is in most cases not database agnostic, meaning that if you want to switch to another relational database (because of performance or security reasons), you also need to change huge portions of your software in order to handle the different database interface. Secondly, this approach often results in security holes, as parameters coming from the web are sent to the database without appropriate sanitisation. This problem is quite common with a lot of php[11] software available on the web.

A simple solution to this problem was the creation of database abstraction layers. These are software libraries that provide a standardized set of functions for accessing databases without vendor specific functions.

10 Concerning database programming low level interfaces in contrast to high level interfaces produce more work (*i.e.* more code) for the programmer. The next step in programming style are database abstraction layers, and the non plus ultra are object relational mappings.

11 It is a backronym for 'PHP: Hypertext Preprocessor', the original acronym was 'Personal Home Page'.

This, in combination with the use of standard SQL queries, solves the problem of vendor dependencies mentioned above. Some of these libraries also provide support for input sanitisation. Still, this is optional and the database access is rather low-level. So-called 'Object Relational Mapping' (ORM) libraries are an advanced solution. These provide a bridge between object oriented programming languages and relational databases, by *e.g.* mapping database tables to 'classes' and database rows to 'objects' of these classes. Furthermore, 'functions' of the objects are used to map database relations. A detailed explanation of these concepts would easily fill another paper, so just compare these two examples – both represent a database query, the first one using some simple database abstraction, the second one using object relational mapping (ORM).

Example 1:
```
X = layer.query("SELECT * FROM table WHERE id=#{id}
LIMIT 1")
a_table_row = x.fetch
```

Example 2:
```
a_table_row = TableRow(id);
```

Without referring to technical details, the visual differences between Example 1 and 2 should provide enough evidence as to why the use of ORM is to be preferred to standard SQL queries. The first language to introduce ORM to a wider audience of developers was Java and especially the Java Enterprise (JEE) Platform. Platforms like WebObjects have featured this type of database abstraction for many years. Today most Java web applications use ORM through the Hibernate Library. Though hibernate definitely is an excellent library, it requires a rather massive amount of xml set-up for each object and property which sometimes does not support the agile nature of many web development projects. Fortunately Martin Fowler (2002) presented the so called ActiveRecord type of ORM in *Patterns of Enterprise Application Architecture*, which is the most important source for object oriented design theory and practice. This pattern was later implemented as an integral part of the web development framework Ruby-on-Rails and allows simple and flexible mapping of database tables.

The Active Record implementation and the existence of a handy MVC[12] abstraction were the main reasons why we chose Ruby on Rails as the software platform for implementing the *Dictionary of Old English Plant Names*. What is more, this framework has some additional 'goodies' to offer. First of all, the programming language Ruby allows for short and clean programming that comes very close to natural language. Secondly Ruby on Rails already supports basic AJAX[13] (see below) out of the box and thirdly Ruby on Rails has a rapidly growing community that should ensure a long-term availability of updates for this platform as well as a growing collection of extensions and plug-ins.

3. Data Input

Considering the huge amount of existing data, we did of course try to make data input as efficient as possible. The analysis of our prototype plant name entry not only provided us with the structure of our database but also enabled us to work out strategies for easy and efficient data input. We can distinguish between static, dynamic, and parsing controls. For a closer look you can access the input mask of our database via the following url: http://oepn.uni-graz.at/main_entry/list. The login details below[14] will grant you read-only access, which will make it much easier and much more interesting to follow the technical explanations.

But data input theory is not only strategies for making data storing easier, we also had to deal with questions of font and formatting. Everybody who deals with computers and Old English texts knows that Old English special characters are one major problem. There are, of course, a variety of standard computer fonts and quite a lot special fonts that are able to display these characters, but hardly one could meet our needs! For web pages it is most important that the font used is available on a maximum number of visitors' computers: this not only guarantees a constant layout but also saves

12 Model-View-Controller based web applications relay on the strict separation of database (models), program code (controller), and publicly displayed content (view).

13 Asynchronous Java Script and XHTML.

14 User = „fwf", and password = „fwffwf". Use user name and password without quotation marks.

the user additional work (*i.e.* installing the new font) which might also discourage a lot of people from using the website at all. The UTF-8 encoding of our database clearly influenced our choice of font: since this form of character encoding is a way to store unicode fonts, this type of font was needed. Unicode is a text-representation and manipulation standard which is designed to set the basis for an international character encoding scheme. It is designed to display and store not only standard ASCII characters but also a great variety of special characters (phonetical, mathematical, diacritical symbols, *etc.*) and character properties like decomposition, collation, and rendering and bidirectional display (*Wikipedia:* 'Unicode'; accessed Dec. 8th 2007, 10:04.). We finally settled on 'Lucida Sans Unicode', a font which we used in previous internet projects dealing with phonetic symbols. Even more important is the fact that it is pre-installed on Windows-PC-systems from Windows 98 onwards. On Macs this font is substituted with 'Lucida Grande' which is the default system-font in OS X and has a similar character set as Lucida Sans. With these two solutions we certainly will have served the majority of our visitors. Users of other systems still have the possibility of downloading the font from various websites for free.

Another problem which has to be dealt with before data can be stored in our database is formatting: our project does not require a broad variety of font formatting, but the means have to be applicable to the medium internet and have to be user-friendly. The technically easiest way of solving this problem would have been using html-tags for the various font formats needed, but this definitely is the most user-unfriendly solution possible. Once again the Ruby-on-Rails platform provided an easy solution: the module 'RedCloth', a ruby implementation of the Textile Human Web Text Generator, provides a broad variety of text formatting possibilities.[15] Textile is a very simple mark-up language that allows the user to write an easily readable text (which would not be possible with html-tags) while applying all kinds of font formatting. It uses a predefined set of symbols that are inserted before and after the text that is to be formatted: _text_ = *text*, *text* is **text**, _*text*_ = ***text***, *etc.* The common mark-ups Textile provides are more than enough for our needs: at the moment we only intend to use the mark-up symbols for italic and bold text.

15 For further information on the use and possibilities of Textile see: http://www.textism.com/tools/textile/ and http://hobix.com/textile/, both accessed June 2008.

With the basic problems solved we could go on to tackle the more elaborate ones: static data (*i.e.* data that is fixed from the beginning and is not likely to change) can be fed into the database via drop-down lists. This device is an element used in graphical user-interfaces and is normally chosen when other means of input (text fields, check-boxes, *etc.*) would either disrupt the consistency and lucidity of the visible screen area or would create a tiresome and recurring workflow. A drop-down list holds a predefined set of values, of which only one is visible when the control is inactive. If activated a list of values is shown which the user can chose from. After clicking the 'submit' button the data is written into the database. We use this means of input for the properties 'type', 'word-class', 'gender' or 'declination' in the main entry. The values for the properties 'etymology', 'formation', 'semantics', and 'motivation' in the concept 'etymology' are also fed into the database via drop-down lists.

Some drop-down lists contain dynamic data: *i.e.* data that is dependent on previous user input. In both cases the newly input data (for the properties 'comment' and 'image') is in direct relation to the concept 'meaning'. The values for meaning are not pre-defined since they differ from lemma to lemma: the various possible meanings for a headword have to be fed into the database by the user. So each time the concept is opened for editing a database query is launched in the background and provides the currently available data.

A similar process is used for AJAX based text-fields which are meant to guarantee easy and even more important error free data input for properties that need accurate spelling. One example would be the concept 'internal references' where internal global links are stored. The links generated from these entries are based on the correct spelling of the stored values. So while typing in the first letters the script checks back with the database and provides the user with a series of probable words to enter into the text field. The user then can choose the right value either by selecting it with the mouse-arrow or the arrow-keyboard.

AJAX is a fundamental concept for our website. Therefore we would like to cover it in more detail. To accomplish this, we have to start at the protocol level with HTTP[16] – the protocol you use when surfing the web. One of the main properties of HTTP (and the most important reason why the world

16 Hyper Text Transfer Protocol.

wide web works as it does these days) is that HTTP is a so called stateless protocol. This means that every time the client asks for content on the server, it has to provide all the information needed, as the server does not store any information about prior request of the client. This is the main reason why workarounds like cookies and sessions were generally introduced.[17] Until recently, it was also not possible to update just parts of a web page in a standardised way. This problem was solved with AJAX, short for Asynchronous JavaScript and XML. Interestingly, the main cornerstone of AJAX, the XMLHttpRequest JavaScript object was first implemented by Microsoft and soon implemented by all other big browser vendors.

So how does AJAX work? In essence it is a script inside a web page that – just like any web browser – opens a connection to the server and requests data. The important part, however, is that this action is asynchronous, which means that you can still use the website while the request is performed. When the exchange of data is finished more JavaScript can be put into action (and this even without user interaction). An example for this would be an AJAX request which queries the server for an image. This image can – after the data exchange between server and JavaScript is finished – be inserted on any desired spot on the current web page without reloading the page as it would be done if simple hyperlinks were used. Another example for this would be an AJAX application which sends search terms to the server while you are typing in a search field and simultaneously provides you with some meaningful completions.[18]

The main reason for using AJAX is efficiency. In a traditional web application (*e.g.* you want to delete an email from your webmail inbox) you send a request to the server saying "Delete message x!" and you have to wait for the server to reload the complete page. This renders your web application useless for some time because the whole data needed to display your page has to be loaded from the server. In an AJAX web application the same query would be sent asynchronous in the background, allowing you to read the next mail while the action is performed.[19] The message sent back from the server is not the whole page, but just a short message like "Ok!" or "Failed!". This greatly reduces the amount of data transferred.

17 Cookies and sessions store frequently needed data.

18 *E.g. Google Suggest*: http://www.google.com/webhp?complete=1&hl=en (accessed June 2008).

19 *E.g. Google Mail*: http://www.gmail.com/ (accessed June 2008).

In the backend of our website AJAX is also used with all in-place-edit enhanced text fields. In-place-edit allows the user to edit the stored data without having to wait for website-database interaction. There is, of course, communication between the browser and the server (the web page and the database), but the amount of data exchanged is kept to an absolute minimum, so that this process is hardly noticed by the user working with the input form.

A completely different process is used for smart parsing: The basic function is that a user can input an arbitrary string in a text field, and the system will try to automatically split this string in data entries and properties, provided that the user follows certain predefined patterns. We use different parsers with different concepts. Here are some examples: the simplest parser splits a text string after commas. This feature is for example used with the properties 'compound', 'derivation', 'place names', 'variants', or 'internal reference'. In the text box the input string for the concept 'variants' looks like this: *„beovyrt, beowirt, beowyrt, biouuyrt, biowyrt, buuyrt*". After being processed by the parsing script each spelling-variant (each word of the list above) is written into a separate database field. This method is also used with the concept 'meaning' but the parsing script that is used here is much more complex. The pattern which has to be used looks like this: 'variant (A ...): assessment (empty, ?, ??, ???) botanical name; English name; German name'. Here different separators define the borders of the properties: the colon marks off the property 'variant', the property 'assessment' only has a limited range of possible entries which are recognised by the parsing script and accordingly fed into the database, the different names are separated by semicolons. According to these specifications a string like "C: ? Acorus calamus L.; sweet flag; Kalmus" is written correctly into the property-fields described in the previous chapter. The concept 'occurrences' can also be filled with the help of a parsing script. The data for this concept is rather complex but in Bierbaumer's books it is still presented in an easily recognizable pattern: the only thing we have to keep in mind is that we have to deal with two different 'data-types' – books one and two use the same structure for presenting occurrence-data, book three differs in some ways. This is why we had to apply two different parsing scripts and according to the origin of the occurrence the correct script has to be chosen from a drop-down list. For books one and two, which do give the grammatical case but do not give a Latin lemma, the parsing script looks for the following pattern: "TEXT: case. record_oe: source1; source2;" The separators are the colon for the property 'text', the full stop for 'case', again the colon for 'record_oe', and the semicolon separates the different 'sources'. This pattern will split a text like "LB: NAsg.: *beowyrt*: 11/29; 21/7; 21/25; 80/3;" (Bierbaumer, 1975: 11) into the correct properties and four occurrences are written into the database. In the third book, in which the Old English glosses are analysed, we have no grammatical case, but we have the Latin lemma, thus the predefined string has to look like this: "record-lat _record-oe_: text source1; text source2;". Programming this script was trickier because we do not have distinctive separating elements for all the relevant properties ('record_lat', 'record_oe', 'text', and 'source'). Since we defined the Latin lemma and the Old English translation as different properties[20] we had to invent dividers on our own: in this case the textilize mark-up (see above) for italic spelling was of great help since they are inserted into the text before the parsing script tries to split it up: the first underscore is the separator between Latin and Old English record. A similar problem is encountered with splitting up the properties 'text' and 'source': here the script uses the first space as divider. Thus the script is able to split the string "APIAGO _*beowyrt*_: AntK 145,9; Laud 161;" (Bierbaumer, 1979: 23) into the correct properties and store them as two different entries in the concept 'occurrences'.

4. Data Output

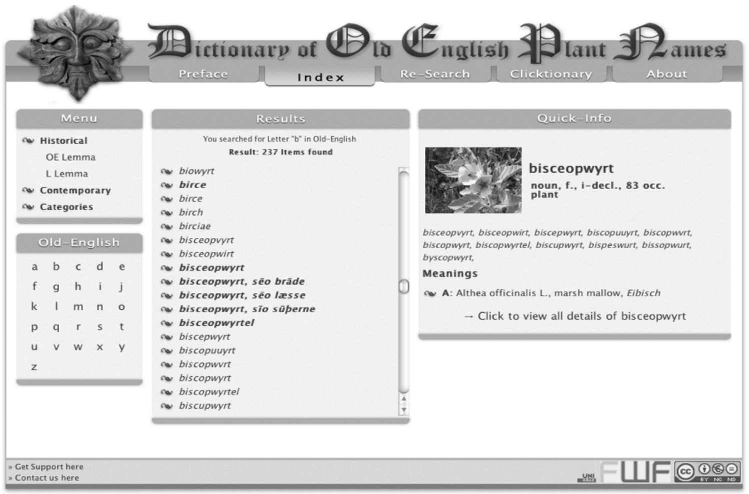
During the last months we have mainly concentrated on database design and making data input as efficient as possible – data output has not yet been dealt with thoroughly, so the 'frontend' of our website is still 'under construction'. The medium internet and the text type dictionary, though, force certain ways of data retrieval and we have of course our own ideas about how to make our data available to the user and especially on how we want to present it. The following link provides access to our 'construction-site' which in the next months will continually be updated with new data retrieval features: http://oldenglish-plantnames.uni-graz.at/.[21]

20 According to the data structure of Bierbaumer's books both Latin lemma and Old English translation present a single entry. We decided on splitting them up because of several reasons, the most important being the increase in different possibilities of data output and the increase in ways of analysing data.

21 For login details, see above.

Normally, the basis of printed dictionaries is a certain (alphabetical) order, which means to present the reader a pre-structured and easy to use text. It would of course be possible to reproduce this framework for our on-line dictionary, but we opted for a completely different approach, which tries to do without the display of the whole (database) content (as it is done in a book, for example). Instead we make the entries available to the visitor via a set of different indices which allow the user to access the database content from various directions. The most important of these is the index of Old English headwords: the list is generated from the property 'lemma'. A second Old English index provides the user with a list generated from the property 'spelling variants', which would allow access to the relevant data even if the user does not know the standardized spelling of the key word. If the user does not know the Old English plant name at all, it is still possible to dig up the correct data: it can be accessed with the indices of the contemporary names, either the English, German or botanical name, which in the end again lead to the Old English headword. Finally we want to provide access via indices on the basis of different categories: type, word-class, gender, declination, etymology, or word formation. Index based browsing is meant to accommodate those visitors who may not have any professional interest but only want to look around, browse the website and get to know this rather specific topic.

Using the indices (see figure 3) is quite simple: it can be accessed through the flag 'Index' on the website. By clicking on it the user opens a box on the left side of the display area which lists the different indices available. The basic menu only shows the superordinate category, another click opens a sub-menu with the entries discussed above. Each of these links opens a new box below, which holds the letters of the alphabet. By clicking on a certain letter the user triggers a database query which provides him with an alphabetical list of adequate entries. The links in the result-box lead to the 'quick-info-box': this is where all the different lines of approach described above finally meet. This box arranges the basic information on the Old English lemma: headword, type, gender, declination, number of occurrences, spelling variants, possible meanings, and if available a picture of the plants denoted by the Old English plant name. (If there is more than one picture they will be continually displayed with the help of a fade-in/fade-out loop, so that the polysemous character of the plant name is not obscured. Images can be enlarged by being clicked on.) The link at the bottom of this box ("Click to view ..") opens the data sheet for the selected headword.



You searched for Letter "b" in Old-English

Result: 237 Items found

- *biowyrt*
- ***birce***
- *birce*
- *birch*
- *birciae*
- *bisceopvyrt*
- *bisceopwirt*
- ***bisceopwyrt***
- ***bisceopwyrt, sēo brāde***
- ***bisceopwyrt, sēo læsse***
- ***bisceopwyrt, sīo sūþerne***
- ***bisceopwyrtel***
- *biscepwyrt*
- *biscopuuyrt*
- *biscopwvrt*
- *biscopwyrt*
- *biscopwyrtel*
- *biscupwyrt*

bisceopwyrt

noun, f., i-decl., 83 occ.
plant

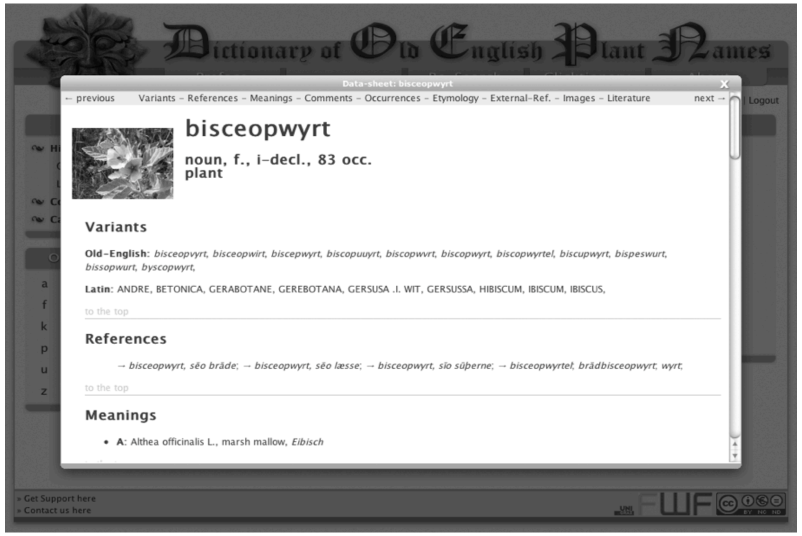
bisceopvyrt, bisceopwirt, biscepwyrt, biscopuuyrt, biscopwvrt, biscopwyrt, biscopwyrtel, biscupwyrt, bispeswurt, bissopwurt, byscopwyrt,

Meanings

A: Althea officinalis L., marsh mallow, *Eibisch*

→ Click to view all details of bisceopwyrt

Figure 3: Screenshot of Index-view: *Dictionary of Old English Plant Names* website at http://oldenglish-plantnames.uni-graz.at/.


Dictionary of Old English Plant Names
Data-sheet: bisceopwyrt
X
← previous
Variants – References – Meanings – Comments – Occurrences – Etymology – External-Ref. – Images – Literature
next →
Logout
to the top
to the top
» Get Support here
» Contact us here

bisceopwyrt

noun, f., i-decl., 83 occ.
plant

Variants

Old-English: *bisceopvyrt, bisceopwirt, biscepwyrt, biscopuuyrt, biscopwvrt, biscopwyrt, biscopwyrtel, biscupwyrt, bispeswurt, bissopwurt, byscopwyrt,*

Latin: ANDRE, BETONICA, GERABOTANE, GEREBOTANA, GERSUSA .I. WIT, GERSUSSA, HIBISCUM, IBISCUM, IBISCUS,

References

→ *bisceopwyrt, sēo brāde*; → *bisceopwyrt, sēo læsse*; → *bisceopwyrt, sīo sūþerne*; → *bisceopwyrtel*; *brādbisceopwyrt*; *wyrt*;

Meanings

- **A**: Althea officinalis L., marsh mallow, *Eibisch*

Figure 4: Screenshot of data-sheet: *Dictionary of Old English Plant Names* website at http://oldenglish-plantnames.uni-graz.at/.

The data sheet (see figure 4) displays all the information available (*cf.* database layout). It is structured in the following order: main-entry data, variants, references, meanings, comments, occurrences, etymology, dictionaries, and images. From here, the user has the possibility to go on in different directions: for one he/she can go back to the index and start over again. He/she also can progress into the database by using the internal links (*i.e.* the reference pointing to other entries, or the wiki-style text links). But the user can even leave the website for further research in the *Corpus of Old English*[22], to which we link in two different ways: one concentrating on the Old English the other on the Latin lemmata. One link into the corpus provides the search result of the corpus website's 'simple search' in combination with our spelling variants. From here on the user can start off with the tools provided on the corpus website. Another link given with each occurrence uses the 'simple search' restricted to a certain Old English text in combination with the occurrence: with this feature it should be fairly easy to access the context of the occurrence. A third link is provided with each Latin lemma where we link to the 'Latin Word Wheel' of the corpus website. This again is meant to make access to the context of the occurrence easier.

This is how far the development of the frontend has progressed until now.[23] The following paragraphs deal with what is planned for the future.

The data will also be accessible through different key word searches. The user can either decide to search for certain plant names (which will be a search that only covers the name-indices) or to carry out a full-text search, which would cover the whole data (*i.e.* all concepts described above that hold relevant text). The search fields are, of course, AJAX-enhanced, which will not only help with spelling problems but can also facilitate the research process as a whole. The results page of the query lists an overview of the located hits according to the concepts they are found in. In a next step the user can choose the category he/she wants to have a closer look at: by selecting the desired concept the whole number of hits is displayed. Based on the principles of Web 2.0 we want to introduce a completely new concept for scientific online dictionaries. In recent years the term "Web 2.0" has become more and more important – it denotes a total change not only

22 Provided that the user has access to the website!

23 *I.e.* December 2007.

in technology but also, and even more important, in user participation and interaction. The idea of the internet as a platform is as old as browser technology (*i.e.* Netscape being one major campaigner), but in the beginning concentrated too much on operation systems and software dependencies to be of major success. Over the past years the interest shifted from the tools needed to the services provided, and one major provider who services independently from this ongoing software race can look back on a continuous rise in market value: Google.[24] The data provided by Google are of course software dependent – both on the company's and the user's side – but "the value of the software is proportional to the scale and dynamism of the data it helps to manage". (O'Reilly 2005: 1) In short, Google fills the space between the user and the content he or she wants to experience, and Web 2.0 provides means for the user to access, control and increase data. The central role of the user has shifted considerably: he/she is not only on the receiving end any more. The internet as a means for publishing content was the core idea of Web 1.0, whereas Web 2.0 is centred on user participation and can be summarised with the following key features: services (if possible, software independent) provided, control over unique data and data sources that become more consolidated with constant user-interaction, trusted and trusting users as co-developers, and channelling collective intelligence. (O'Reilly 2005: 5)

Based on these key features we started designing a module we want to call 'Clicktionary' – the basic slogan for which is simply: 'Click your own dictionary!' Here the user can create his/her own workspace[25] which can be accessed each time he/she re-visits the website. This, of course, requires an elaborate user management, where all relevant data is stored and changes made by users are recorded. It is important to note that we are going to accept only user registrations that can be verified either through personal contact (*i.e.* researchers that are known to the project team personally) or through recommendations of such people or through other official means of identification. The foremost goal of our project is to grant highly scientific information and to make scientific discourse on our topic possible. In our dictionary 'user interactivity' primarily means two things: on the one hand

24 Google is, of course, just one example for a wide range of companies, it was used here because it definitely is the best known.

25 A very good example for this can be found at http://www.protopage.com, where users can create their own start page for surfing the internet (accessed June 2008).

the user can shape his/her research according to his/her needs and on the other hand the user can add data on his/her own. Here we attach great importance to the fact that existing data, *i.e.* data which has been submitted by the project team, cannot be modified but only commented on or complemented. So the dictionary's users can add content (text, photos, links, literature, *etc.*) which then will clearly be marked as an additional entry by a certain user. By applying personal research strategies (*i.e.* Click your own dictionary!) the user is able to use all features of the website on the basis of a modular construction system, the basic principle simply being that all database queries available can be combined in all variations possible and each result can be basis for a 'cross-lemma' comparison, thus opening the database content for future research: so the dictionary data is not static but highly dynamic! Data can be compared, rearranged and recombined according to the actual needs of the researcher.

5. Summary and Perspectives

The *Dictionary of Old English Plant Names* is meant to be the gravitational core in the universe of Old English plant names, with researchers, research and resources more or less loosely connected through our internet platform. From a different point of view it is, of course, just a part of the world wide web solar system of Anglo-Saxon research, but again joining, channelling and interweaving research and researchers. All in all, our dictionary is thoroughly based on the principles of Web 2.0 and its conceptual design may be regarded as a major development in the field of humanities and linguistic research.

It is available at: http://oldenglish-plantnames.uni-graz.at.

Bibliography

A Textile Reference. <http://hobix.com/textile/>

ABBYY FineReader - Professional OCR Software for Document and PDF Conversion Application. 1996-2008. <http://finereader. abbyy.com/>.

Bierbaumer, Peter. *Der botanische Wortschatz des Altenglischen. I. Teil: Das Laeceboc.* Grazer Beiträge zur Englischen Philologie. 1. Frankfurt am Main: Lang, 1975.

- - -. *Der botanische Wortschatz des Altenglischen. II. Teil: Lacnunga, Herbarium Apuleii, Peri Didaxeon.* Grazer Beiträge zur Englischen Philologie. 2. Frankfurt am Main: Lang, 1976.

- - - *Der botanische Wortschatz des Altenglischen. III. Teil: Der botanische Wortschatz in altenglischen Glossen.* Grazer Beiträge zur Englischen Philologie. 3. Frankfurt am Main: Lang, 1979.

Dictionary of Old English Plant Names. Eds. Peter Bierbaumer and Hans Sauer with Helmut W. Klug and Ulrike Krischke. 2007-2008. <http://oldenglish-plantnames.uni-graz.at/>.

Fowler, Martin. *Patterns of Enterprise Application Architecture.* Addison-Wesley Signature Series. Boston: Addison-Wesley Professional, 2002.

Google Mail. 2008. <http://www.gmail.com/>.

Google Suggest. 2008. <http://www.google.com/webhp?complete=1&hl=en>.

O'Neil, M. *Design your own database. Concept to implementation or how to design a database without touching a computer.* Dartmough Colledge. February 2004. Accessed November 2007. <http://www.dartmouth.edu/~bknauff/dwebd/2004-02/DB-intro.pdf>.

O'Reilly, Tim. "What is Web 2.0?" oreilly.com. 30. September 2005. Accessed November 2007. <http://www.oreillynet.com/pub/a/oreilly/tim/news/ 2005/09/30/ what-is-web-20.html>.

Protopage - Free Web 2.0 Personal Start Pages. <http://protopage.com/>.

Textism. *Textile: A Humane Web text Generator.* 2001-2008. <http://textism.com/tools/textile/>.

Wikipedia, the free encyclopedia. 2001-2008. <http://en. wikipedia.org/>.

Contact:

Helmut W. Klug
Department of English Studies
University of Graz
Heinrichstraße 36
8010 Graz
Styria / Austria

E-mail: helmut.klug@uni-graz.at

Roman Weinberger
Oeverseegasse 14/19
8020 Graz
Styria / Austria

E-mail: roman.weinberger@uni-graz.at