

# Single Source Documentation for enCoreXpress

Trond Karl Pettersen

LINGO Project  
Dep. of Humanistic Informatics  
University of Bergen

January 2006

## **Abstract**

The help and documentation available to today's enCoreXpress users is limited and often outdated. There does clearly exist a need for new and improved documentation. This paper looks into some possible ways of maintaining a single set of always up-to-date documentation for the enCoreXpress Interface. First we take a brief look at the pros and cons of a single source documentation, before we, from a technical point of view, explore the technologies at hand. Most of the considered technologies are XML-based, although some non-XML possibilities are also discussed. The report does also try to give a realistic development time estimation for the proposed solution.

## **About**

This report outlines possible single source solutions for the enCoreXpress Interface by and for the LINGO project at the University of Bergen, Norway. It assumes that the reader is somewhat familiar with technical terminology/terms and markup languages.

Comments may be directed to the writer at <st05223@lingo.uib.no>, or to LINGO's project leader, Daniel Jung, at <daniel@lingo.uib.no>.

# 1 Introduction

For this report, what is meant by ‘Single Source Documentation’ is as described on Wikipedia:

Single source publishing or single sourcing allows the same content to be used in different documents and in various formats. (...) Eliminating duplicate content can save translation costs, reduce maintenance costs, improve consistency and reduce errors. (...) Single sourcing also allows the creation of documents in various formats from the same content [8].

Most applications are used by a wide variety of users. In MOOs the available user groups are Wizards, Programmers, Builders, Teachers, etc. All of these user groups are likely to have different wants and needs concerning documentation (e.g. builder versus programmer actions). Ideally, a software project will create and maintain one kind of documentation per user group and make the various documentation types available in multiple formats for the users to choose from, so that the documentation may be consulted both online and offline, using a multitude of software.

One solution to this problem (a rather outdated one) is to create one documentation per type and format; i.e. one PDF-version, one HTML-version, one RTF-version, etc. of each user group’s documentation (i), or a single, large documentation (covering all topics) in various formats (ii). However, these solutions will most definitely result in information redundancy, and is also likely to be inconsistent and maybe erroneous ((i) being the worst-case). A project using this approach will have to make sure that all the different types and formats of the documentation is kept up-to-date and error free - resulting in increased maintainance costs due to the fact that making even a tiny change can be quite time consuming. It may also be difficult to edit the ‘appearance’ of such documentation whenever a decision to change the layout has been made. A better way of keeping multiple kinds and formats of documentation is to maintain a single source documentation only. Single source documentation / documents should not be encoded in HTML or PDF, but in a presentation independent format from which it is easy to extract the wanted information fragments. Of course, there is no guarantee that this will remove redundancy altogether, but here any redundancy will be limited to the kind of redundancy that is a result of describing something multiple times in one source, not multiple times in one source *and* multiple formats.

Plus, by using the right kind of technology, this kind of redundancy may be reduced. Finally, by using a ‘presentation neutral’ approach, it is easier to focus on the important information - the contents of the documentation, instead of the layout.

## 2 Single Source Technologies

### 2.1 XML-based

HTML is well suited for presenting documentation on the World Wide Web<sup>1</sup>. Being an interface to a web-application, web based documentation should be a natural part of enCoreXpress, and so a HTML-documentation should be available to enCoreXpress users. However, to produce and maintain a set of HTML-documents is not recommended. In HTML, *presentation* and typography is in focus - HTML is a *procedural markup language* carrying information (for user agents like web browser) on how to present information to human users. By using a more *descriptive markup language*, we can make the documentation well structured and focus on the different kinds of information in the documentation. Thus, we may also transfer parts of the semantics from the information (documentation) to the resources containing the documentation themselves. To make the single source documentation semantically rich is in itself of course not necessary, but it helps making the information available for alternative use - for both layout and filtering purposes - in case a particular “MOO-community” would want to do so.

The eXtensible Markup Language (XML) is a set of rules for describing markup languages. Using XML one may describe user-defined sets of elements used to mark up / structure information. Any XML vocabulary can be read by any XML-compliant parser and transformed into various formats using XML-technologies like XSL and XSLT. Being a widely used language for sharing information on the WWW, XML is an important part of the World Wide Web Consortium’s (W3C) Semantic Web vision<sup>2</sup>, enabling applications to share and re-use information. The possibility to share and re-use information is in turn strengthened through the use of standardized and/or widely used XML vocabularies. Describing software documentation

---

<sup>1</sup>This may not come as such a big surprise, HTML being 1 of the 2 main building blocks of the WWW.

<sup>2</sup><http://www.w3.org/2001/sw/>

in XML makes it not only easier to share and re-use the information, it also makes it easier to maintain a single source of documentation which may be transformed into multiple presentation formats (our main concern).

### 2.1.1 DocBook

DocBook is a schema (originally a DTD) for describing books, articles, etc. in markup languages such as XML. DocBook XML is not a standardized (ISO or W3C) XML vocabulary, but still a widely adopted XML vocabulary for describing things such as software / technical documentation. To quote *DocBook: the Definite Guide* [7], available online at <http://www.docbook.org/>:

DocBook is a very popular set of tags for describing books, articles, and other prose documents, particularly technical documentation.

Using DocBook XML, an article may be marked as shown in Example 1.

```
<article>
  <arheader>
    <title>My Article</title>
    <author>
      <honorific>Dr</honorific>
      <firstname>Emilio</firstname>
      <surname>Lizardo< /surname>
    </author>
  </arheader>
  <para> ... </para>
  <sect1>
    <title>On the Possibility of Going Home</title>
    <para> ... </para>
  </sect1>
  <bibliography> ... </bibliography>
</article>
```

Example 1: “A Typical Article”, taken from [7].

Example 1 illustrates how XML can be semantically richer than HTML<sup>3</sup>,

---

<sup>3</sup>For our use, the semantics of HTML-elements are not considered important.

which doesn't support elements like `arheader`, `author`, etc. DocBook elements aren't limited to paragraphs, headings and so on, and using DocBook XML one may explicitly state where a chapter, section, etc. starts/stops, what is 'ordinary' text, what is a name, etc. DocBook supports a wide range of elements, and "*guimenu - The name of a menu in a GUI*" and its likes seem to be well suited for documentation purposes.

Whenever information is marked up as above, a program can (relatively) easily extract the kind of information relevant to a given user (based on input, of course). DocBook XML documents may be transformed into formats such as HTML, PDF, RTF using publicly available, Open Source and free, XSLT stylesheets - of which some are available through <http://docbook.sourceforge.net/projects/xsl/>.

Although many XML Editors claim to support DocBook XML, this seems to be limited to the fact that they support XML, and thus DocBook XML. However, some do support 'schema constraint'-editing - i.e. editing files of a specific XML vocabular in accordance to the vocabular's DTD - and styling (see <http://wiki.docbook.org/topic/DocBookAuthoringTools>). Still, being XML, DocBook XML may be created/edited using various open source XML editors. Alternatively, proprietary XML Editors such as epcEdit<sup>4</sup> and `<oXygen />`<sup>5</sup> are also available.

### 2.1.2 Topic Maps and XTM

Originally, Topic Maps was developed in order to facilitate automatic merging of (\*NIX documentation) indices while maintaining consistency. The International Standards Organization (ISO) published the first version of the Topic Maps standard in 1999, and the second version, ISO 13250 [3], in 2002. Although still an important part of the Topic Maps concept, ISO 13250 is not limited to defining a way of merging indices, but defines rules for creating the semantic and associative information / navigation structures that topic maps are. XML Topic Maps [9] is a standardized XML syntax (serialization) conforming to ISO 13250.

A topic map is an information structure consisting of concepts called topics and associations (relations). Topics represents subjects - everything whatsoever (both concrete and abstract 'things') might be represented as a topic in a topic map. If you can talk about it, it can be represented by a

---

<sup>4</sup><http://www.epcedit.com/>

<sup>5</sup>[http://www.oxygenxml.com/docbook\\_editor.html](http://www.oxygenxml.com/docbook_editor.html)

topic in a topic map. Topics may be given different names and variants of those names, occurrences (information relevant to a specific concept; text, image or something else), and participate in associations. Another important feature of topic maps is that one may define scopes under which certain topic characteristics (name, occurrence, association role, etc.) are valid. Using scope, one may for instance say that “This part of the information on XPress\_Object\_Editor is only valid under the scopes ‘Spanish’ (language) *and* ‘Programmer’ (user group)”, and so scope can be use to filter information for various user groups, languages, etc. In topic maps the subject identity of a topic’s subject is also important, because being able to unambiguously identify the subjects of different topics makes it possible to merge topics and indices from multiple topic maps. When merging topic maps (or when two or more topics in a single topic map represent the same subject), topics representing the same subject is merged in a process where a new topic whose characteristics (names, occurrences, etc.) are the union of the characteristics of the individual topics is created. This way, subject identity and topic map merging is a tool for redundancy reduction in knowledge, or information bases (here: documentation and documentation index(es)) [4].

Using topic maps requires the creation of an ontology of the domain under discourse. In effect, this means that one has to create / specify topics for every class/object in the world about which one wants to state something - for instance the different parts of the enCore MOO and the enCoreXpress interface - and relations between them - for example a superclass-subclass hierarchy (taxonomy). This can of course be a time consuming process, but creating any kind documentation is likely to be time consuming. Besides, since enCore MOO is object oriented and supports verb & property inheritance, it ought to be easy to map the (wanted parts of the) enCore MOO ontology to XTM and dump the result to a file (before manual editing). Parts of the hard-coded documentation can also be automatically extracted from the application and added to an XTM document.

Available topic map engines for parsing and presentation (via client programs) of XTM include the TM4J<sup>6</sup>, an Open Source XTM-parser written in JAVA. Sub-projects of TM4J include a Jakarta Velocity based engine (requires Apache Jakarta Velocity and Apache Tomcat server), TM4Web/Velocity, which uses TM4J for Web presentation of topic maps. TM4J may be used without TM4Web/Velocity, though, by creating a JAVA program that can in-

---

<sup>6</sup><http://www.tm4j.org/>

teract with TM4J. TM4J supports both full-text searching and tolog<sup>7</sup>-queries ('SQL-ish' query language for topic maps). For XTM-editing, the Norwegian company Ontopia offers a proprietary editor - the Ontopia Knowledge Suite<sup>8</sup> (OKS). The OKS is reasonably priced for academic development use. There does, however, exist a free alternative in the free Open Source ontology editor Protege<sup>9</sup> by Stanford Medical Informatics. Protege does not produce XTM, but using the TMTab<sup>10</sup> plug-in (not supporting the latest version(s) of Protege, however), Protege-ontologies might be exported to XTM. A list of other XTM-editors is available at <http://www.techquila.com/topicmaps/tmworld/11772.html>. With Topic Maps and XTM being relatively new standards, it is likely that more Open Source alternatives will be available in the not too distant future.

Examples 2 & 3 show how topics and associations may be expressed in XTM.

---

<sup>7</sup><http://www.ontopia.net/topicmaps/materials/tolog-spec.html>

<sup>8</sup><http://www.ontopia.net/solutions/products.html>

<sup>9</sup><http://protege.stanford.edu>

<sup>10</sup><http://www.techquila.com/tmtab.html>

```

<topic id="xpress_obj_editor_01">
  <instanceOf>
    <!-- makes this topic an instance of the xpress_editor-class -->
    <topicRef xlink:href="#xpress_editors" />
  </instanceOf>
  <subjectIdentity>
    <!-- the identity of the topic's subject is indicated through the flash movie -->
    <subjectIndicatorRef
      xlink:href="http://lingo.uib.no/xpress_obj_editor.swf" />
  </subjectIdentity>
  <baseName>
    <!-- this name is used in the unconstrained scope
      i.e. when no (other) scope is specified -->
    <baseNameString>Xpress Object Editor</baseNameString>
  </baseName>
  <occurrence>
    <scope>
      <topicRef xlink:href="#programmer" />
    </scope>
    <resourceData>

      The verb used to display the Xpress Object Editor is...

    </resourceData>
  </occurrence>
  <occurrence>
    <scope>
      <topicRef xlink:href="#enduser" />
    </scope>
    <resourceData>

      In the Xpress Object Editor you may...

    </resourceData>
  </occurrence>

```

```
<!-- perhaps some more occurrences++ -->
</topic>
```

Example 2: A topic representing the enCore XPress Object Editor. An XTM-compatible application like TM4J would be able to extract the occurrence valid in the scope of 'programmer' only, if needed. The same goes for 'enduser' and possible association roles in which this topic participates. Topics referred to by topicRef-elements should (optimally) also exist. Indices of topics in the topic map could be automatically generated and/or one may specify a superclass-subclass association (topic map taxonomy) for a 'Table of Contents' for 'programmers', etc.

```
<association id="oiu23" />
  <instanceOf>
    <!-- makes this association an instance of the part-of-association -->
    <topicRef xlink:href="#part_of_assoc" />
  </instanceOf>
  <member>
    <roleSpec>
      <topicRef xlink:href="#part" />
    </roleSpec>
    <topicRef xlink:href="#trash_bin" />
  </member>
  <member>
    <roleSpec>
      <topicRef xlink:href="#container" />
    </roleSpec>
    <!-- ref. example 2 -->
    <topicRef xlink:href="#xpress_obj_editor_01" />
  </member>
</association>
```

Example 3: This association states that the topic whose id is 'trash\_bin' is a 'part\_of' another topic whose id is 'xpress\_obj\_editor\_01'. Note that association-roles may be scoped (besides roleSpec which specifies the role played by the participating topics), although that has not been done here. All referred to topics - the topics with ids 'trash\_bin', 'xpress\_obj\_editor\_01', 'part', 'part\_of\_assoc' and 'container' - should exist in the topic map.

### 2.1.3 Web Ontology Language (OWL)

OWL is W3Cs “ontology language” for their Semantic Web. OWL is based on RDF(S)<sup>11</sup>/XML (also W3C recommendations). Through OWL one may create ontologies in a way similar to that of XTM. OWL is based on description logics enabling the embedding of inference rules, etc. This makes OWL a very complex language (and a good candidate language for any Semantic Web). Because of the complexity of OWL Full, OWL comes in three variants: OWL Lite, OWL Full and OWL DL. No matter the variant, using OWL seems like ‘overkill’ for a small documentation project like this. The learning-threshold for OWL is likely to be higher than that of XTM - for people with little background knowledge, grasping the central concepts of topic maps is likely to be easier than becoming familiar with OWL, plus there does not exist a need for fully-fledged semantic languages in this project. Even for XTM, only a sub set of the language’s possibilities is likely to be used. For our use, navigation and ease of use outweighs OWL’s complexity. Therefore, this report will not go into details on OWL, but refer to <http://www.w3.org/TR/owl-features/> for further information.

## 2.2 Alternatives to XML

A single source documentation can of course be based on non-XML technologies. As an example, proprietary systems claiming to be tools for, among other things, maintaining single source documentation do exist. One such system is AuthorIT, found at <http://www.author-it.com/>. Although such programs may, at the time of decision-making, be fit for the job, we discourage the use of proprietary software like AuthorIT. An Open Source solution such as enCore MOO ought to be careful in regards to basing its documentation, or other parts, on proprietary solutions alone (future use, etc.). AuthorIT can for instance export documentation to XML, but the result of such a process is AuthorIT-XML, an XML vocabulary as proprietary as the software itself. Just as the Norwegian Technology Council *Teknologirådet* encourages the use of open source technology and open standards [6], so too do we.

Instead of proprietary solutions, an alternative to the XML options outlined in section 2.1 is the use of relational database management systems

---

<sup>11</sup>Resource Description Framework (RDF) and RDF Schema; <http://www.w3.org/RDF/>.

(RDBMS) like MySQL<sup>12</sup>. As an example, one could create a database with tables for each part of the documentation, and let every table's field definitions reflect the different kinds of documentation ('programmer', 'enduser'). Here too, one would have to create an ontology (database schema) - in a way similar to that (roughly) described in section 2.1.2. An example of a documentation-database table may be:

XPress_Object_Editor	
<b>PK</b>	<b><u>ID</u></b>
	Programmer Wizard Teacher End-User

Figure 1: Table for documentation on the XPress Object Editor. If a database was to be used, it ought to be more normalized than this - in order to avoid thousands of NULLs, but this will suffice as an example.

Relations like the part\_of-association from Example 3 could be implemented through "standard" relational-tables in order to 'connect' parts of the database:

Part_Of_Rel	
<b>PK</b>	<b><u>part</u></b>
<b>PK</b>	<b><u>container</u></b>

Figure 2: Part\_of relation.

Such tables (Figure 2) would be populated with values from other tables, e.g. the id of 'trash\_bin' as 'part' and the corresponding id from 'XPress\_Object\_Editor' (Figure 1) as 'container'. The data types may vary: the types in Figure 2.1

---

<sup>12</sup><http://www.mysql.com>

may be TEXT, MEDIUMTEXT, etc. and contain the actual textual contents of the documentation, or they may be of VARCHAR containing references to plain text files (.txt), etc. In order to populate the tables, and as a ‘Web editor’, one could code a Web interface using for instance PHP or Python. Such scripting languages might be used in order to display the documentation too. A PHP application could for instance output HTML, PDF or RTF, possibly (automatically) wrapped up in a ZIP-file. This would be possible either by directly using PHP and it’s built-in functions, existing libraries, or by generating XML (on updates) and transforming XML to the aforementioned formats using XSLT (based on input, ref. section 3).

### 3 Suggested Solution

The biggest advantage of RDBMSs over many XML-based applications is the built-in functions for, and availability of, full text indexing and fast searching - both well tested and optimized by a large community of users and programmers. Also, most programming / scripting languages allow for easy database connectivity and querying. The drawbacks include the fact that RDBMSs are not open systems (hidden to the world outside the server), don’t contain much semantic information and thus do not enable information to easily be shared and re-used. Updating a database ontology is also likely to require more work than updating ontologies in languages like XTM. Combined with the fact that XTM describes an associative navigation structure (and is a step towards the Semantic Web), known to support intuitive navigation in sets of information<sup>13</sup>, this report recommends that future enCoreXpress documentation is based on a central DocBook / XTM ‘Repository’ (XTM-DB).

#### 3.1 Central XTM-DB Repository

Because the learning curve of XTM is likely to be steeper than that of DocBook XML<sup>14</sup>, it is recommended that the documentation is kept in various DocBook XML-documents and that XTM is used for merging of DocBook indices / navigation purposes (in and between documentation). XTM’s could be created using TM4J’s XSLT-stylesheets for DocBook (`indexentry`-elements)

---

<sup>13</sup>See for instance examples in [1].

<sup>14</sup>DocBook XML is “simple” XML, while XTM is based on more abstract concepts.

→ XTM (from TM4J.org). How to create DocBook XML is described in online tutorials (see [2]) such as [10], and of course in [7].

The most apparent solution / ‘division of documents’ is to maintain one document - alternatively, a set of documents, e.g. articles describing the various parts of the enCoreXpress interface - per user group, and one set of such documents per language, assuming that all or parts of the documentation is to be multilingual (using XTM and scoping, so that things were kept in one place, would be preferable, but probably harder to do if multiple people of varying backgrounds were to get involved). This might result in inconsistencies between the different languages, but generating XTM from DocBook indices could assist in uncovering such inconsistencies. XTM indices could also serve the function of navigation aids (at least for the HTML version of the documentation) in that the XTM would be an “abstract” layer, a kind of a map, referring to external resources (from DocBook). As mentioned in section 2.1.1, PDF and HTML versions of DocBook documents may be created using publicly available XSLT stylesheets.

### 3.1.1 Communication enCoreXpress - XTM-DB

First off, all the official XML documents (both DocBook and XTM) should be kept at an advertised location (Web server), so that individual enCore MOO installations could access them directly if needed / wanted.

Because of possible, and experienced, difficulties in including external data in dynamically generated, frame-free HTML-pages<sup>15</sup> and the possible time lags caused by such processes, it is recommended that all output is generated on the XTM-DB side. This means that the XTM-DB server should output (X)HTML, PDF, RTF which would simply be linked to from objects displayed in enCoreXpress. A particular object’s lightbulb would hence link to a script on the central XTM-DB, and the URL used would include parameters specifying object type, preferred language, etc. This may cause the layout and feel of the help (e.g. CSS) to differ from that used in the particular enCore MOO-installation, but this kind of presentation-problems are just considered minor problems, or maybe not problems at all (one work-around would be to allow linking to external CSS, so that every enCore MOO may control the look and feel of the HTML-help). When using specialized objects one would have to include a link to local help (as a URL-parameter),

---

<sup>15</sup>I.e. opening network connections, sending and fetching data to/from external sources, parsing the retrieved information and including parts of it in the MOO-side output.

or override the “lightbulb html”-verb.

If it is wanted that local documentation / help - that is, documentation written by and for a specific enCore MOO community - may be included / 'integrated' in the response from the XTM-DB, the best solution would probably be to install / set up some kind of Web Service on the XTM-DB server which would take DocBook formatted documentation as part of the input and include this in the result. Such a feature would probably be useful to various communities and may be considered (if and when time / resources).

A diagram of the communication process between different end users / enCoreXpress and the XTM-DB server is depicted in Figure 3.

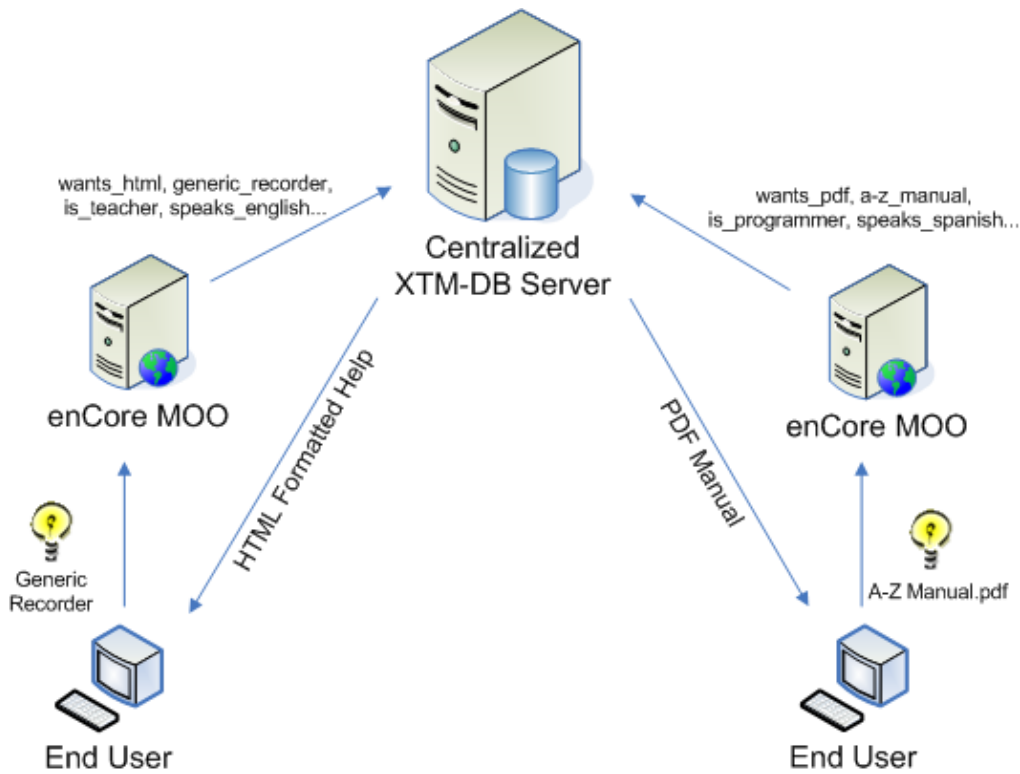


Figure 3: One Single Source Documentation Server for multiple enCore MOO installations and user needs.

### 3.1.2 Use of Existing Documentation

Re-inventing the wheel is (almost) never necessary, so too for enCoreXpress documentation. The enCore v4 User's Guide available from <http://encore-consortium.org/Documentation/> seems to have been made by using Macromedia Dreamweaver templates. Although Dreamweaver is known to at times produce quite 'interesting' HTML, the HTML documents in the enCore v4 User's Guide seems to be quite consistent, and because its author has used headings, etc. in a consistent way it should (hopefully) be possible to relatively quickly / automatically extract information from the HTML documents using for instance PHP5 (simpleXML) + tidy<sup>16</sup>. The extracted data could then be dumped to DocBook XML-files, which in turn would have to be checked manually in order to ensure correctness.

It is less likely that it would be possible to quickly extract information from the Wizard Basics at <http://moo.kcc.hawaii.edu/~moo/MOO/files/documentation/wizbasics.html> using automated methods. If information from this guide is to be used (instead of re-writing it), it would probably be quicker to manually copy the information into DocBook documents.

It ought to be relatively easy to extract the hard coded 'help' / information from enCore MOO by writing a verb which would dump the contents of classes' help verbs to DocBook XML document(s).

## 3.2 Estimated Development Time

In this section, we outline the *expected* development time of the solution proposed in section 3. Although a realistic time frame has been in mind, the estimations described below are not to be taken as absolutes. Issues that might pro-long the development are as likely to occur in this project, as in any other software development project<sup>17</sup>. Naturally, the most time consuming task is to write and maintain the documentation, but since this is an everlasting / iterative process, writing the actual documentation is not considered part of the project.

---

<sup>16</sup><http://no.php.net/manual/en/ref.tidy.php>

<sup>17</sup>Although this has - to some extent - been taken into account, the fact that predicting the future is never an easy job to do, remains true.

### 3.2.1 Slim / Expandable Solution

To get the project started, the development of a somewhat reduced help database based on the solution proposed in 3 is recommended. By “reduced” it is *not* implied that the help database should be anything less than 100% working, but that some of the more “sophisticated” features (i.e. the use of XTM) may be left out. Still, it should always be possible to add any such feature at a later time - without having to rebuild the entire database.

The first step for both the slim and the “full” solution is to set up a server environment in which all files related to the project may be stored - the easiest way, from a ‘technical’ point of view, would here be to merely create a folder structure on a server (preferably with CVS or SVN for versioning control) and give access to a chosen mass of editors and administrators. Alternatives, like DocBookWiki<sup>18</sup> might be explored, in order to establish if any good / usable collaborative DocBook editor exists, or not.

Extracting information from existing sources would be the second step in building the central XTM-DB repository. First off, the information needs to be dumped to DocBook XML documents. Thereafter, someone will have to manually check and mark up the result, in order to ensure that given fragments are marked properly (*guielement*, *indexentry*, etc.).

The third, and perhaps the single most important, step is to create an interface between the DocBook documents and end-users / enCoreXpress. What is needed, is to write scripts that can extract [parts of] the documentation and present the information as HTML, PDF, RTF, etc. However, the slim version would most likely not include relational links because such are to be included in the full version, through the use of XTM. This will most likely involve some coding - hopefully not a lot (although issues may emerge). The script(s) should be able to take input parameters such as object type (generic name), user group, preferred language, search query string, etc. Another aspect of this step is to develop a usable layout for the different kinds of formats and pages.

A search function should be included, and is the fourth thing on our list for the slim version of the help database. Here, the XML files need to be searched, but it is recommended that only a simple text search, based on regular expressions or even sub string matching, is built because the full version is likely to include robust pre-built (e.g. in TM4J) search capabilities.

The development time for the work sketched out in this section may be

---

<sup>18</sup><http://sourceforge.net/projects/docbookwiki/>

roughly estimated to 1 - 1.5 month's of work (one developer). It is not likely that a better estimation can be made until the actual work on the help database has begun.

### **3.2.2 “Full” Solution**

The full version of the help database should contain all features mentioned above, and hence be an expansion of the slim version. In addition to this, it would use XTM for relational linking. By creating XTMs from the Doc-Book documents, one can get an overview of the information related to a specific concept. For instance; when viewing the topic “Generic Recorder”, all indexed and related entries of that particular concept will be listed [for the user to choose from].

Also included in the full version is a better search function with ranked results. This might come as a built-in feature of e.g. TM4J. If not, it is suspected that some time will be spent on building such a feature.

The final (identified) item to add to the full version of the help database is a Web Service which enables the inclusion of local help in the result from the XTM-DB. By “inclusion” it is meant that the help is not merely linking to the local help (after receiving a URL as part of the input), and that the local help is shown next to the “centralized”/official help. This step is likely to be one of the more time consuming steps.

For the full version it is expected that the development time would be an additional 1 - 2 month's of work. Here too, it is unlikely that a better estimation may be given until the actual work has started.

### 3.3 Screenshots

This section presents some screenshots (or rather, “mock-ups”) of a possible layout for the suggested enCore help system described in this report. Note that these are merely suggestions for layout etc. - changes are likely to appear.

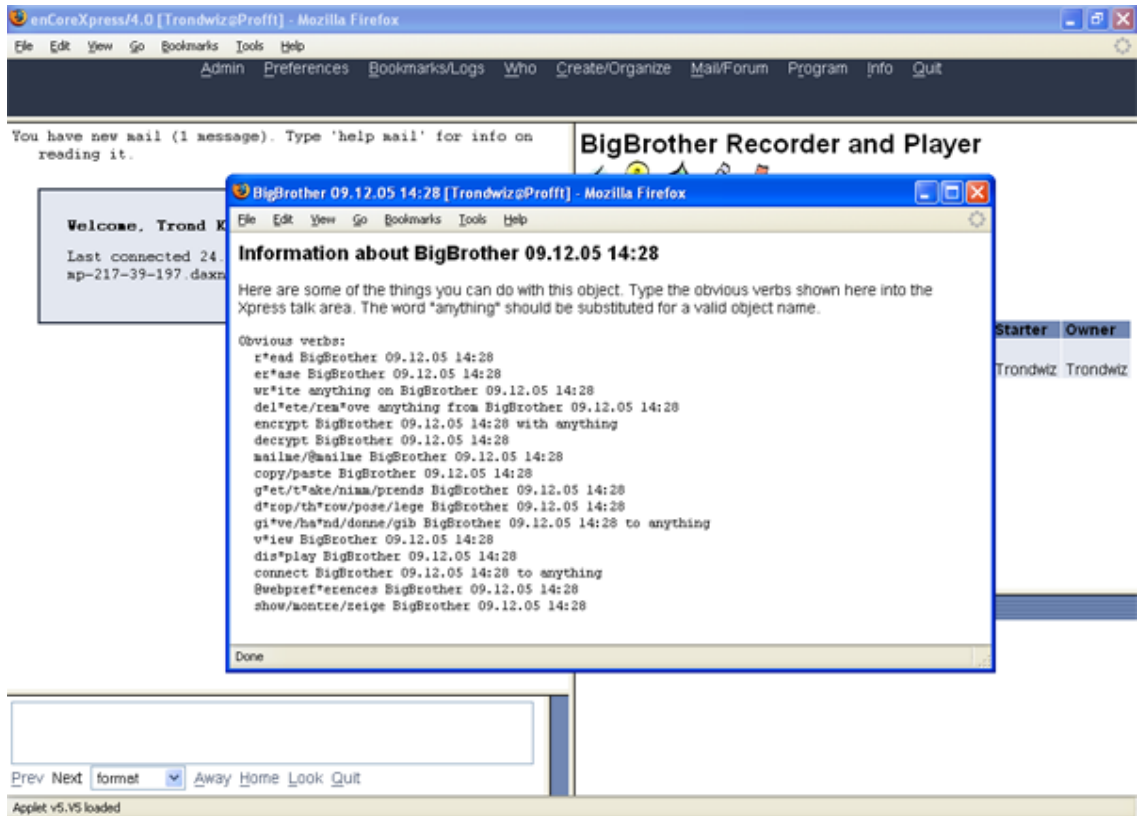


Figure 4: Current help when clicking a recorder transcript’s lightbulb. When working with GUI-centered users (which is the case for LINGO and most likely most current-day enCore MOOs) this may be considered highly user-UN-friendly since we are not doing what the users are familiar with [5].

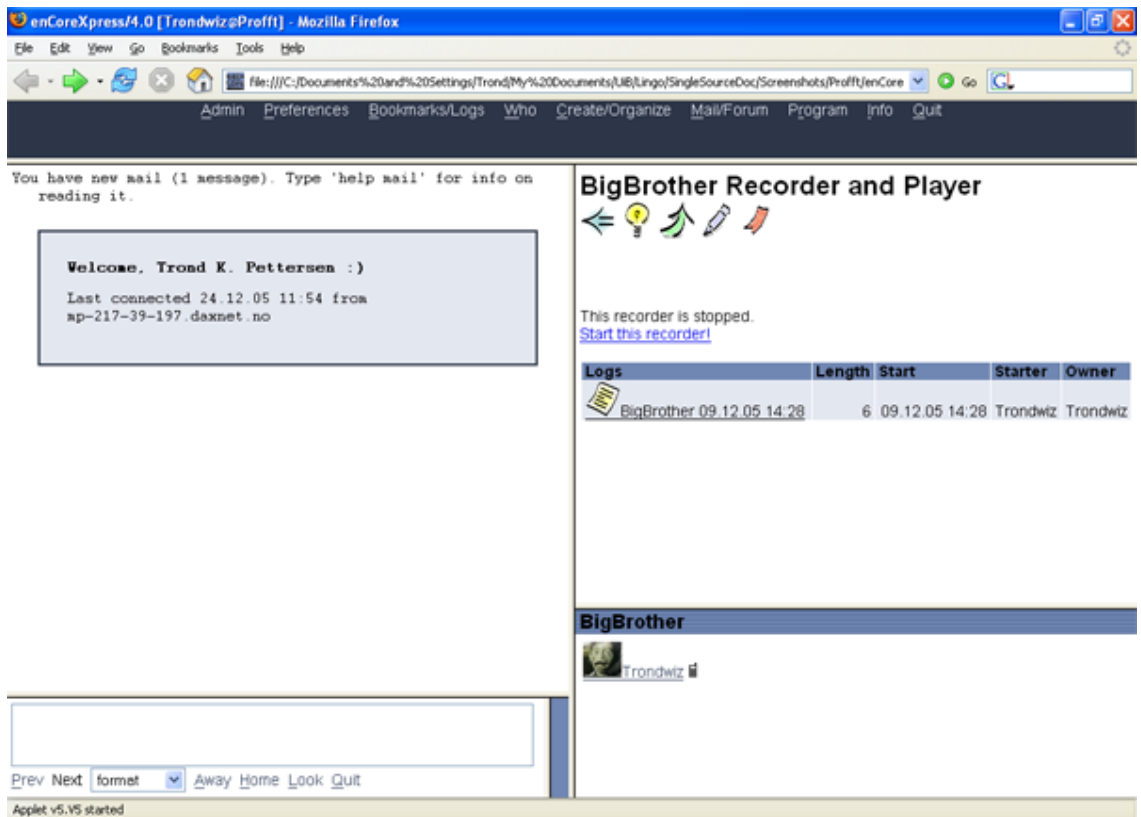


Figure 5: Viewing a Generic Recorder (enCoreXpress v.5 beta) . In need of help on the recorder, but before clicking the lightbulb.

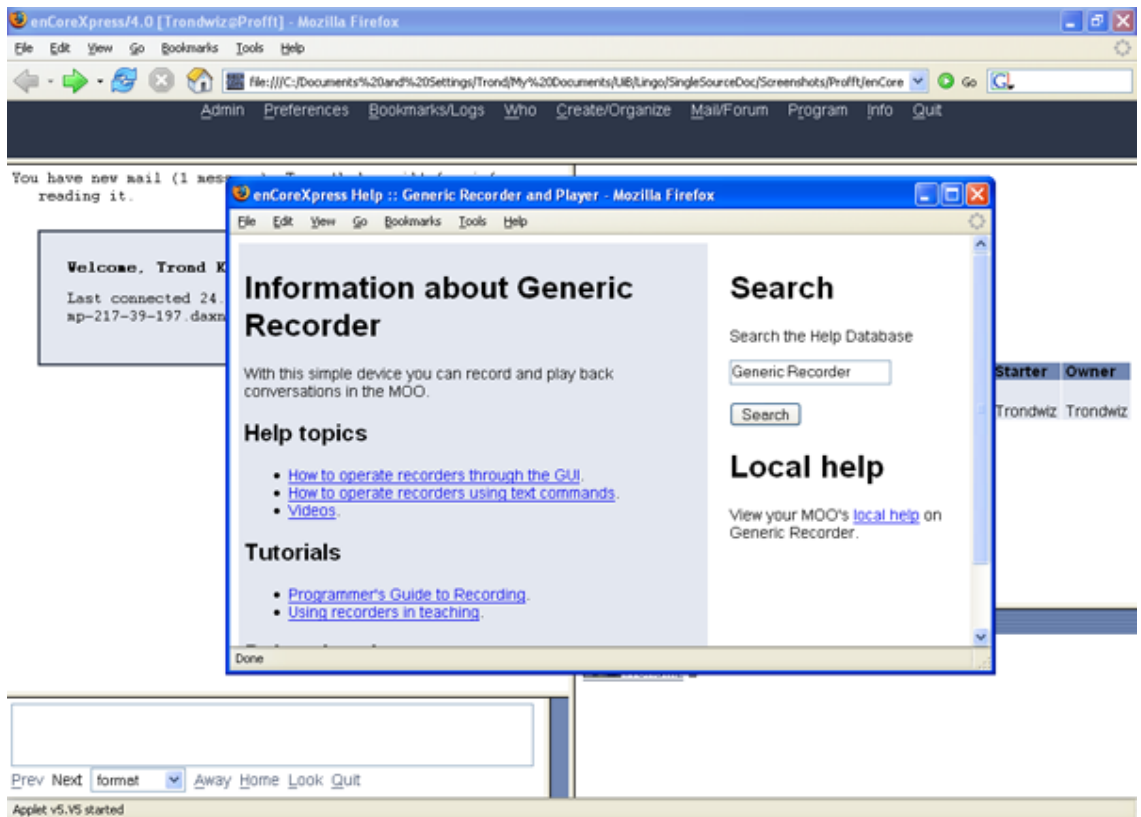


Figure 6: After clicking the lightbulb in Figure 5. XTM-DB generated help on Generic Recorder. The obvious help topics with videos (if available) are listed at the top, then some tutorials - depending on the current user's level, and possible related topics (objects). Also available is a search function and a link to local help (if the XTM-DB was provided with a "local" URL).

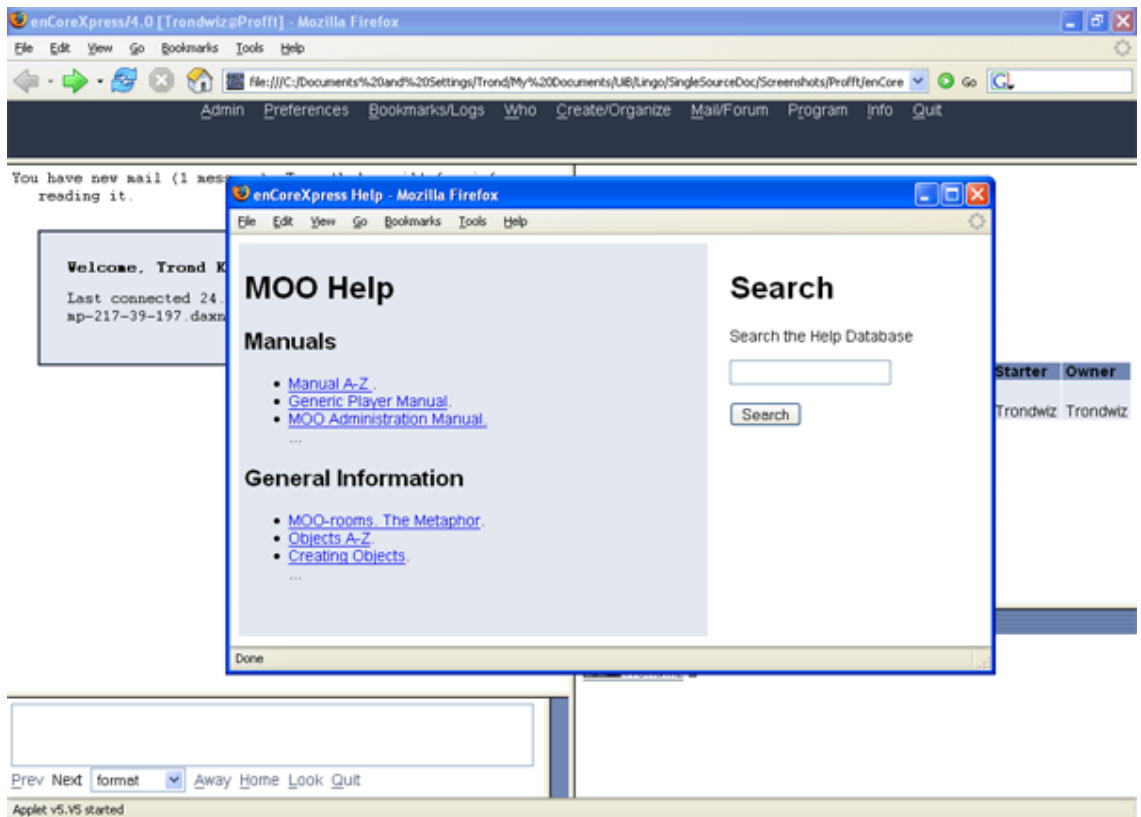


Figure 7: The general help would include links similar to those in Figure 6, but without the perspective of “Generic Recorder”. Here, no URL to a local help has been provided, and the ‘Local help’-link is therefore excluded from the result.

## 4 Conclusion

This report has discussed some possible single source solutions for an enCoreXpress help system. The considered options were highly XML-based, but we did also briefly discuss the pros and cons of a RDBMS-driven approach. The suggested solution is based on a centralized DocBook XML [7] & XML Topic Maps [9] repository (XTM-DB), and accepts input in the form of HTTP GET / POST parameters. A “slim” and expandable solution was proposed built in order to get the project off the ground, with a “full” solution in mind.

It is always difficult to try to give a realistic development time estimate of a software project like this, and the time estimations given in this report

might be proven wrong - either by being too long (which is hoped) or by being too short. It is, however, suggested that work on the XTM-DB is started before enCore version 5 (full) is released.

## References

- [1] K. Ahmed. Topic maps. a practical introduction with case studies. Powerpoint presentation from XML Europe 2002, available at <http://www.idealliance.org/papers/xml02/slides/Ahmed/ahmed.ppt>, 2002.
- [2] Docbook wiki. Online DocBook Help / Wiki: <http://wiki.docbook.org/topic/DocBookTutorials>.
- [3] ISO. Iso/iec 13250 topic maps. International standard, International Organization for Standardization, 2002.
- [4] J. Park and S. Hunting, editors. *XML Topic Maps. Creating and Using Topic Maps for the Web*. Addison-Wesley, 2003.
- [5] J. Preece, Y. Rogers, and H. Sharp. *Interaction Design*. Wiley, 2002.
- [6] T. Storsul, M. S. Dag W. Schartum, L. Sjørgård, H. W. Lie, E. Aas, M. Espedal, and G. Viksaas. Programvarepolitikk for fremtiden. Report, Teknologirådet, 2004.
- [7] N. Walsh. *DocBook: The Definitive Guide 0.0.12 for DocBook V5.0b1*. O'Reilly & Associates, Inc., 2005.
- [8] Wikipedia.org. Single source publishing. Available at [http://en.wikipedia.org/wiki/Single\\_source\\_publishing](http://en.wikipedia.org/wiki/Single_source_publishing), 2005.
- [9] Xml topic maps 1.0. <http://www.topicmaps.org/xtm/>, 2001.
- [10] D. York. Single-source publishing with docbook xml. Available at <http://www.lodestar2.com/people/dyork/talks/2002/ols/docbook-tutorial/>, 2002.