

OAXAL: Open Architecture for XML Authoring and Localization  
by Andrzej Zydron  
February 21, 2007

## Introduction

XML is now acknowledged as the best format for authoring technical documentation. Its wide support, extensible nature, separation of form and content, and ability to publish in a wide variety of output formats such as PDF, HTML, and RTF make it a natural choice. In addition, the costs associated with implementing an XML publishing solution have decreased significantly. Nevertheless, there are some clear do's and don'ts when authoring in XML, some of which are detailed in [Coping with Babel](#), a paper from the XML 2004 conference.

XML, thanks to its extensible nature and rigorous syntax, has also spawned many standards that allow the exchange of information between different systems and organizations, as well as new ways of organizing, transforming, and reusing existing assets. For publishing and translation, this has created a new way of using and exploiting existing documentation assets, known as Open Architecture for XML Authoring and Localization (OAXAL).

OAXAL takes advantage of the arrival of some core XML-related standards:

- [DITA](#) (Darwin Information Typing Architecture from OASIS)
- [xml:tm](#) (XML-based text memory from LISA OSCAR)

DITA is a very well thought-out way of introducing object-oriented concepts into document construction. It introduces the concepts of reuse and granularity into publishing within an XML vocabulary. It is having a big impact on the document publishing industry.

xml:tm is also a pivotal standard that provides a unified environment within which other localization standards can be meaningfully integrated, thus providing a complete environment for OAXAL. OAXAL allows system builders to create an elegant and integrated environment for document creation and localization.

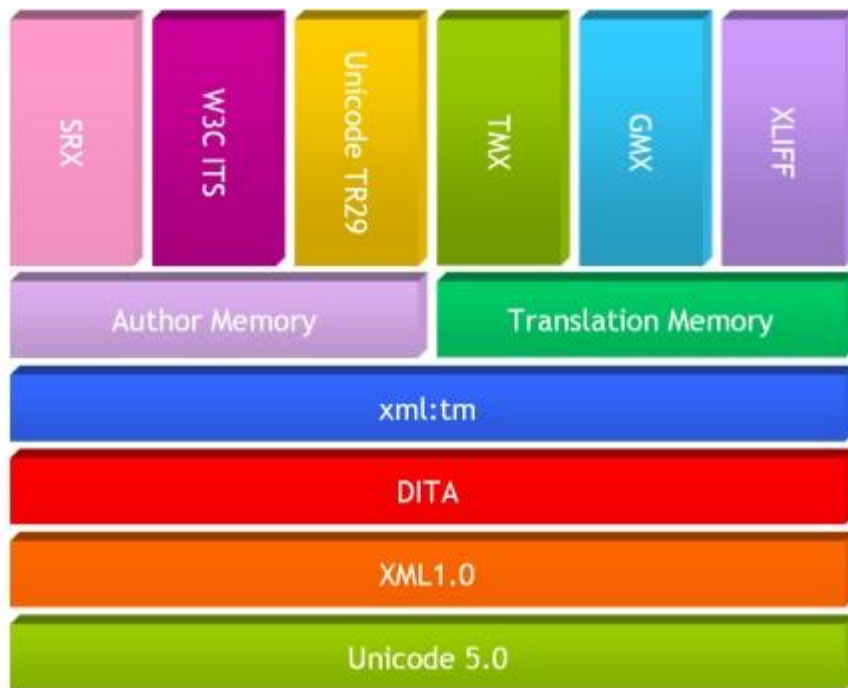


Figure 1. Object 1

- [W3C ITS](#) is an XML vocabulary that defines the rules for translatability for a given XML document type. It states which elements have translatable text, which elements are "inline" and thus do not cause segment breaks, which form subflows, and which attributes are translatable.
- [Unicode TR29](#) is part of the main Unicode standard that defines word and sentence boundaries.
- [SRX](#) (Segmentation Rules eXchange) is an XML vocabulary for defining segmentation rules for a given language.
- [GMX](#) (Global Information Management Metrics Exchange) is a new LISA OSCAR standard for word and character counts, as well as a vocabulary for exchanging metrics information. It is part of a three-part standard that will also tackle complexity and quality.
- [XLIFF](#) (XML Localization Interchange File Format) is an OASIS standard for exchanging localization data in an XML vocabulary.
- [TMX](#) (Translation Memory eXchange) is a LSIA OSCAR standard for exchanging translation memories.

## DITA

The OASIS DITA (Darwin Information Typing Architecture) Standard has certainly raised the profile of XML in the technical authoring field. It has done so for two reasons:

- It is a very intelligent and well thought-out approach to the authoring and publication of any technical manual.
- It substantially reduces the costs associated with introducing a component-based publishing system.

The original architects of DITA looked at the most effective way of writing technical documentation and came to the conclusion that a topic-based authoring approach was the best

way of achieving this. Rather than writing a publication as a monolithic set of chapters, they found that deconstructing a publication into a set of topics was a far better way of creating technical documentation. It meant that authors could work independently on their own topics without impeding one another. It also meant that topics could be reused across many different publications. DITA is also an excellent way of authoring web-based and other types of electronic documents, such as Help topics, as it encourages greater granularity. The core messages of DITA are granularity and reuse.

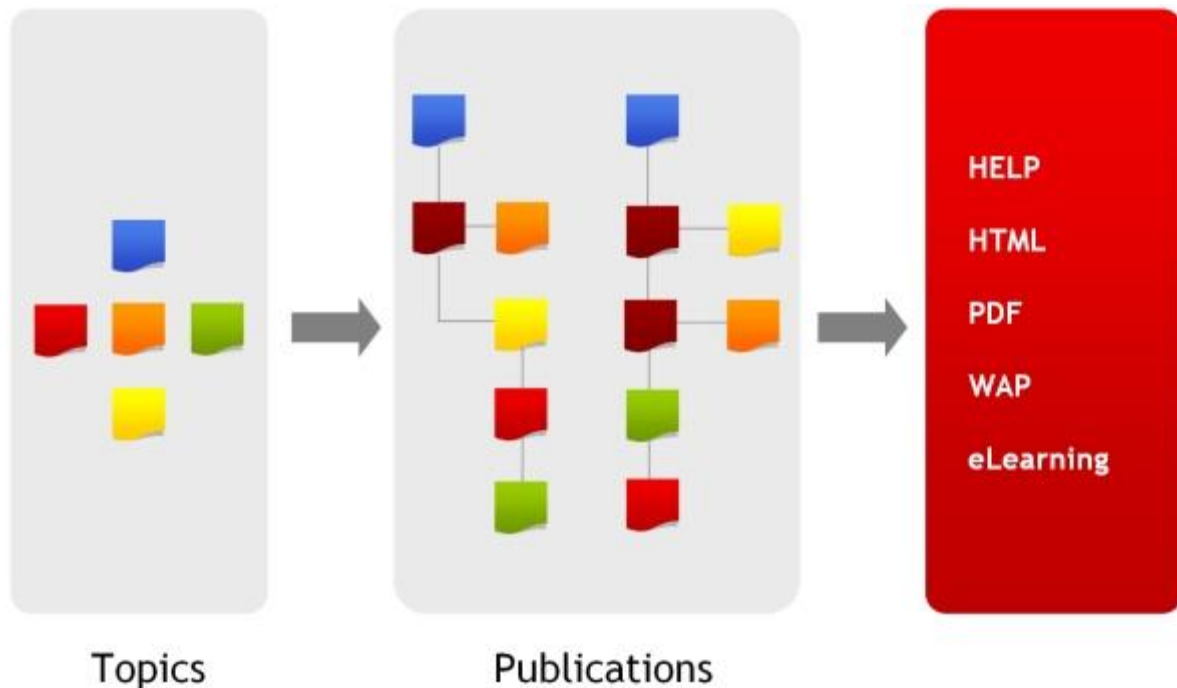


Figure 2. Topics

The topic concept provides DITA with a built-in component model that can be readily understood and implemented: write once, translate once, reuse many times. There are many analogies to this approach in the automotive industry, where the same components are reused across many model ranges. This substantially reduces costs and increases the availability of spare parts. Having a DITA document for discrete operations, such as the removal of a cylinder head for a given engine type, means that this procedure can be reused across all of the publications relating to models that share that engine.

DITA also comes with a built-in concept of extensibility. The original DITA architects realized that topics can have different formats, and that not all topics are equal, so they worked out an extension mechanism that allows topics to be specialized. The standard DITA "topic" is available in the form of the three most common implementations:

- Reference
- Task
- Concept

These can then, in turn, be specialized further, so "Task" can have "Maintenance Task" and "Repair Task," "Disassembly Task" and "Assembly Task" specializations using the simple DITA extension mechanism.

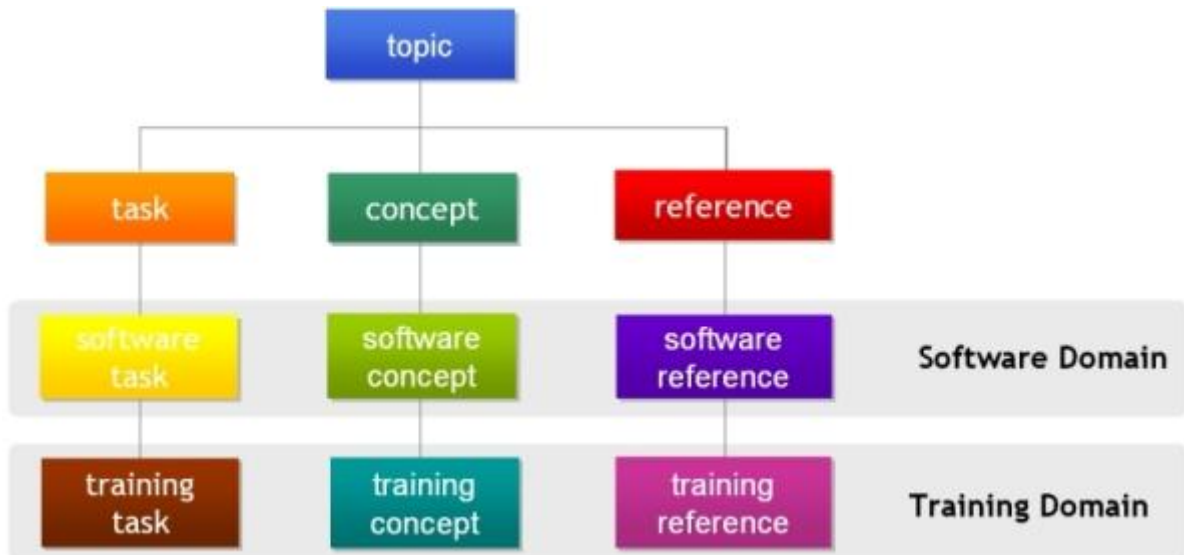


Figure 3. Object 2

Elements in the new specializations can share typographical characteristics of their forebears even though they may be called by a different name.

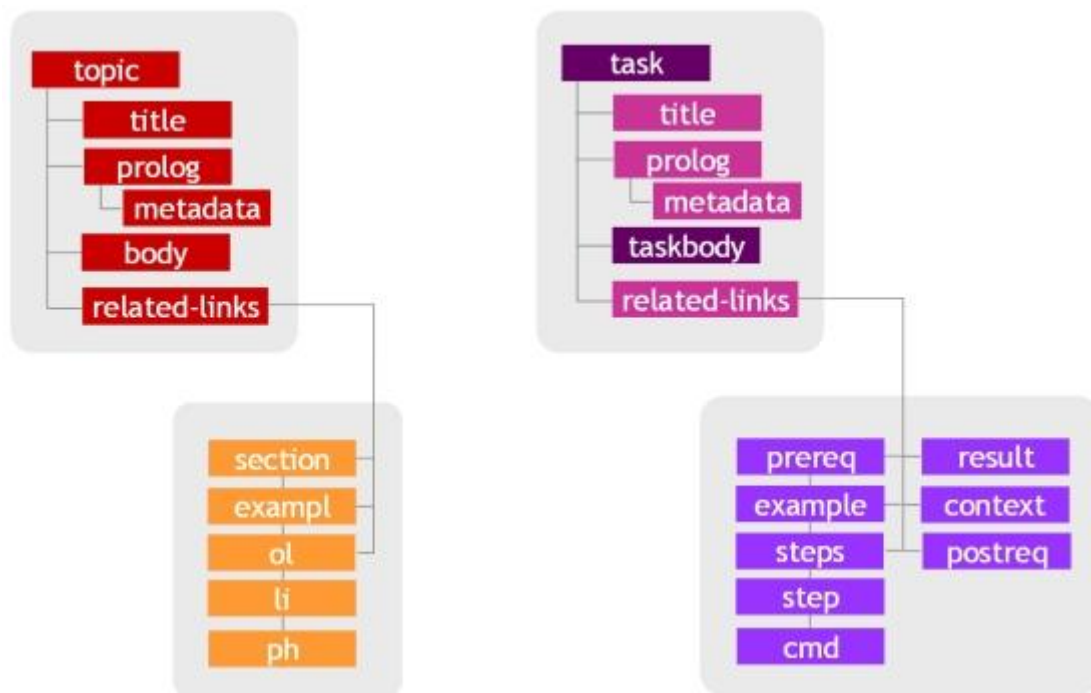


Figure 4. Inheritance

By providing a ready-built toolkit for the construction of technical documentation, DITA substantially reduces the cost of implementation. DITA comes with a prepackaged set of tools to construct books and manuals as well as complex web-page structures for free from the [DITA Open Toolkit SourceForge site](https://dita.sourceforge.io/).

In order to put together a publication from individual topics, DITA uses the "bookmap" concept. A "bookmap" allows the publisher to decide which topics go into a given publication.

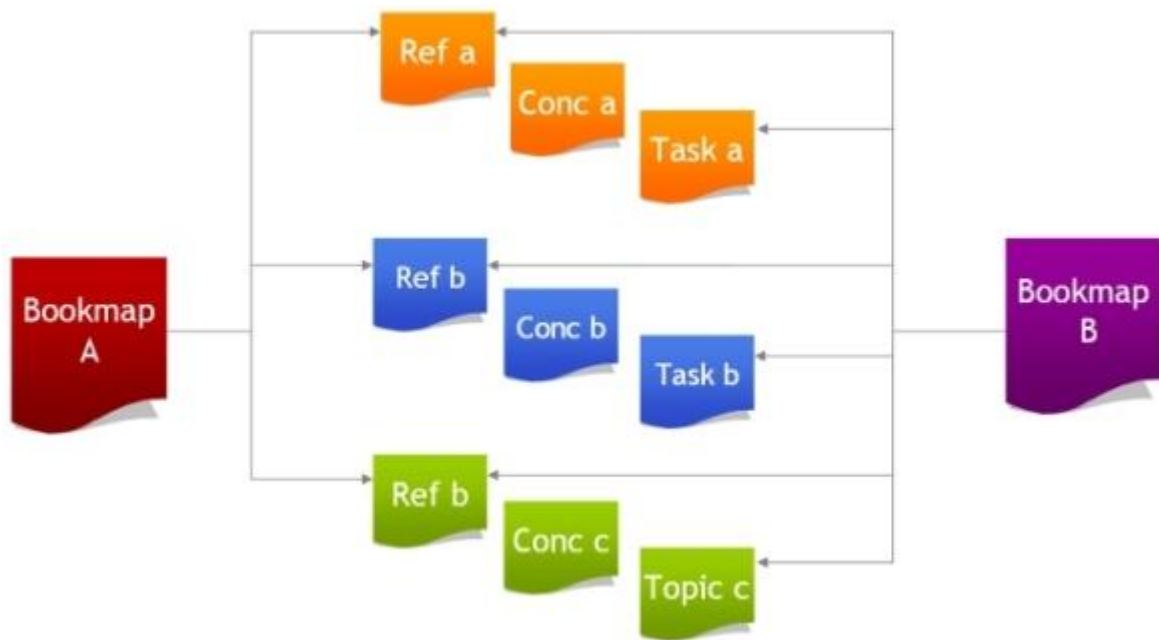


Figure 5. Bookmap

DITA also has some very powerful mechanisms for conditional processing, depending on environmental settings for such items as model name, etc. The following example shows how conditional processing can be used:

```

<p audience="administrator">Set the configuration options for
administrators:

<ul>
  <li product="extendedprod">Set foo to bar for extended product
support</li>
  <li product="basicprod extendedprod">Set your blink rate for basic and
extended products</li>

  <li>Do some other stuff just for administrators</li>
  <li platform="unix">Do a special thing on Unix systems</li>
</ul>

</p>

```

DITA allows conditional processing depending on the attribute values for given elements. In the above example, the "p" element will only be published if the environmental variable "audience" has the value "administrator". Likewise, the "li" elements will only be published if the relevant "product" and "platform" attributes have the required settings.

One of the main benefits of DITA from the standardization point of view is that most of the work regarding the XML Schema construction has already been done for you. Previously, introducing a custom built XML Schema would have entailed lots of work and the associated costs of hiring a consultancy or specialist. DITA provides a ready-made, pre-built generic XML environment for authoring and publication. The full [DITA Open Source Toolkit](#) provides transformations for HTML, PDF, and RTF output as well as many other utilities and

examples. The other benefit of standardization is that there is extensive support now available from XML editors and content management systems for DITA, making implementation simpler and cheaper. In addition, there has been a rapid build up of Documentation, Training, and Consultancy resources for DITA. All of these make for easier and cheaper implementation.

If your existing documentation is already in a "topic" or "task" type format, then moving to DITA should be relatively simple as long as there is direct mapping between the existing elements used and DITA elements. It may even be possible to develop a specialization of the existing DITA into the incumbent element names and attributes. Another factor that makes the move to DITA easier is if your existing documents use the CALS table model that DITA supports.

DITA is not only for technical manuals. It can be easily adapted to other subject matter such as eLearning, warranty, or service bulletin publications.

As with all good things, there are some aspects that need to be considered carefully. DITA does have some potential issues that you should bear in mind:

- Element nesting. DITA has unfortunately followed the HTML principle that nearly every element can appear in another element. This can cause horrible problems with publishing and translation, where segmentation and typography can go badly awry.
- Many DITA elements can occur inline with text, which can cause substantial problems for both composition and translation if not restricted.
- The linking mechanism for conditional text can cause problems with localization.
- Topic-based publishing requires the use of a content management system, especially if you are also translating into other languages; otherwise, control of versions and synchronization with different language versions becomes very difficult.

## **XML-based Text Memory—xml:tm**

[xml:tm](#) is a new LISA standard that is directly compatible with DITA. It takes the DITA principle of reuse and applies it at the sentence level. Integration with DITA is seamless.

Whereas traditional translation memory systems have only concentrated on the translation aspect of the document lifecycle, xml:tm goes deeper into the process, establishing the concept of "text memory." Each sentence (text unit) in the document is given a unique identifier. This identifier remains immutable for the life of the document. xml:tm uses the XML namespace mechanism to achieve this.

xml:tm is comprised of two core concepts:

- Author memory
- Translation memory

Figure 6 shows how the xml:tm namespace coexists within an XML document:

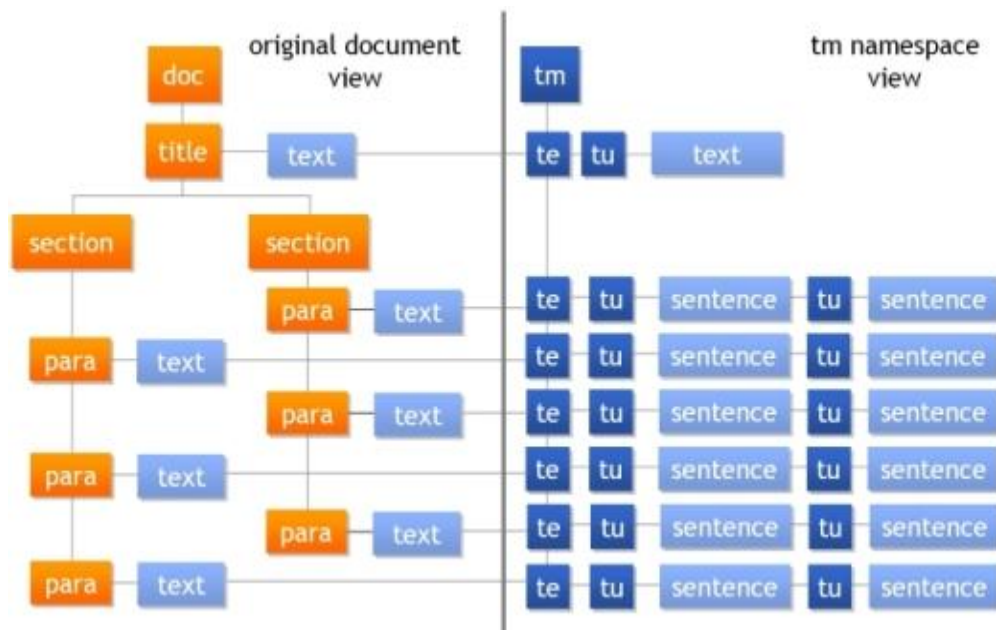


Figure 6. xml:tm tree

XML namespace is used to map a text memory view onto a document. This process is called segmentation. The text memory works at the sentence level of granularity—the text unit. Each individual xml:tm text unit is allocated a unique identifier. This unique identifier is immutable for the life of the document. As a document goes through its lifecycle the unique identifiers are maintained and new ones are allocated as required. This aspect of text memory is called author memory. It can be used to build author memory systems, which can be used to simplify and improve the consistency of authoring. A detailed technical article about xml:tm has been published on [xml.com](http://xml.com).

The use of xml:tm greatly improves upon the traditional translation route. Each text unit in a document has a unique identifier. When the document is translated, the target version of the file has the same identifiers. The source and target documents are therefore perfectly aligned at the text unit level. Figure 7 shows how perfect matching is achieved:

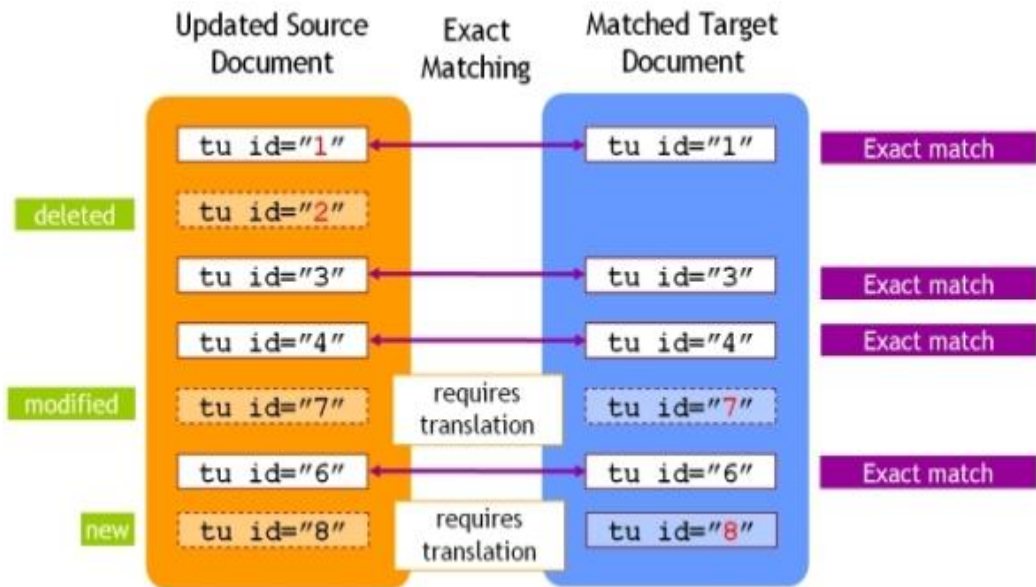


Figure 7. Matching

In xml:tm terms, this type of perfect matching is known as *in context exact* (ICE) matching. The xml:tm concept has been successfully proven within large-scale technical publishing applications.

For authoring, xml:tm represents a systematic way of identifying and storing all previously authored sentences. This repository can then be appropriately indexed and serve as input to authoring tools, to encourage authors to reuse existing sentences where appropriate rather than coming up with different ways of saying the same thing.

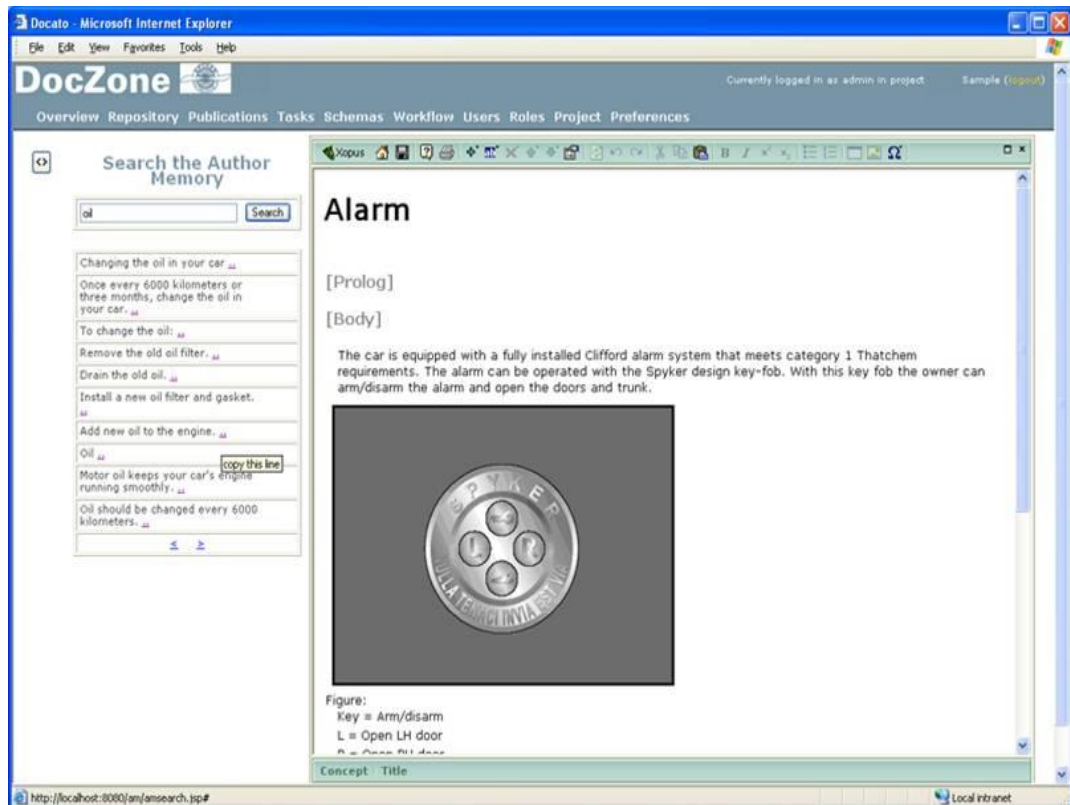


Figure 8. Author Memory

The author can enter keywords and search the author memory for appropriate sentences. If a sentence has already been used, then it is very likely that it has also been translated, in which case a leveraged translation match will be found, further reducing translation costs.

## DITA and xml:tm

DITA and xml:tm are a perfect fit. While DITA enables reuse at the topic level, xml:tm complements this by enabling reuse at the sentence level. The xml:tm namespace can be seeded transparently into a DITA document. As the document goes through its lifecycle, the unique identifiers of each text unit are maintained, as are modifications to the document. In effect, xml:tm provides a change tracker for the document. The easiest way of implementing this is on the back of a content management system (CMS). Using DITA invariably means investing in an appropriate CMS.

The OAXAL document lifecycle for a DITA document looks like this:

1. Check out document for authoring. Check if xml:tm namespace is present; if not, seed first instance.

2. During authoring, the writer can make use of Author Memory from the leveraged Translation Memory database from within his authoring editor.
3. Check in document from authoring. Compare the previous version of the xml:tm namespace with updated document and identify changes. Allocate new text unit identifiers as required.
4. Check out document for translation. Check out previous source and target versions of the document and complete xml:tm-based matching. Next, do any database translation memory matching. Create output XLIFF file and DITA skeleton file for translated text.
5. Check in document from translation. Populate skeleton file with translated text from the XLIFF file, thus creating the target version of the document. Load translated text into the leveraged memory database.

The xml:tm namespace can be made transparent to any editing, printing, or transformation tools by means of a very simple technique.

### xml:tm and the Other Open Standards

xml:tm provides a pivotal role within OAXAL, allowing all the other related standards to interoperate within one elegant architecture.

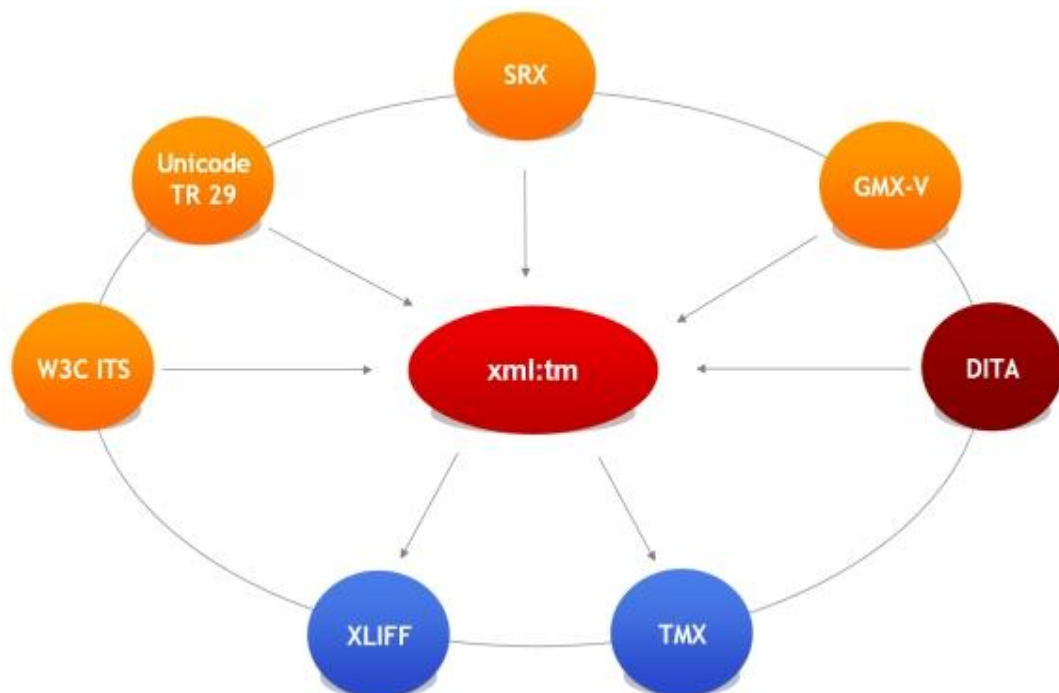


Figure 9. xml:tm

- xml:tm mandates the use of the [W3C ITS](#) document rules to define which elements contain translatable text, which elements are "within text" and so do not constitute a segment break, which "within text" elements are subflows (such as footnotes) that do not form part of the linguistic integrity of the surrounding text, and, finally, which attributes are translatable. W3C ITS document rules are used by xml:tm when seeding the namespace into the DITA document.
- xml:tm recommends the use of [Unicode TR29](#) to enable effective tokenization

of text into words. This is a prerequisite to segmentation and word and character counts.

- xml:tm mandates the use of the LISA OSCAR [SRX](#) segmentation rules standard to define how blocks of text are segmented into individual sentences. Heartsome and XML-INTL have donated their respective rule sets to LISA OSCAR in order to establish an industry-wide set of SRX rules for given languages:
- xml:tm mandates the use of the LISA OSCAR [GMX](#) word and character count standard for maintaining authoring and translation statistics.
- xml:tm mandates the use of the OASIS [XLIFF](#) standard for extracting the text for the actual translation process. Documents seeded with the xml:tm namespace are in effect "pre-digested" for extraction. The use of XLIFF allows translators to choose from competing translator tools that support XLIFF—one of the many benefits of Open Standards.
- xml:tm enables the easy creation of LISA OSCAR TMX files for the exchange of translation memories between localization service suppliers and customers. TMX prevents vendor lock-in that was prevalent in the localization-tools community prior to the introduction of Open Standards.

## **The Benefits of OAXAL**

If we look at the authoring aspect of publishing, OAXAL offers the following significant benefits:

- DITA provides an excellent framework for reducing costs by introducing granularity into the authoring process.
  - Multiple authors can work on different topics at the same time, thus improving delivery times.
  - Topics can be reused many times in different publications.
  - xml:tm Author Memory takes the reuse principle down to sentence level.
    - Authors are encouraged to write consistently. Left to our own devices, we will quite often write the same sentence in different ways. This is compounded when there are multiple authors.
    - If a sentence has been reused, then it will already have an entry in translation memory for other languages. This will reduce translation costs.

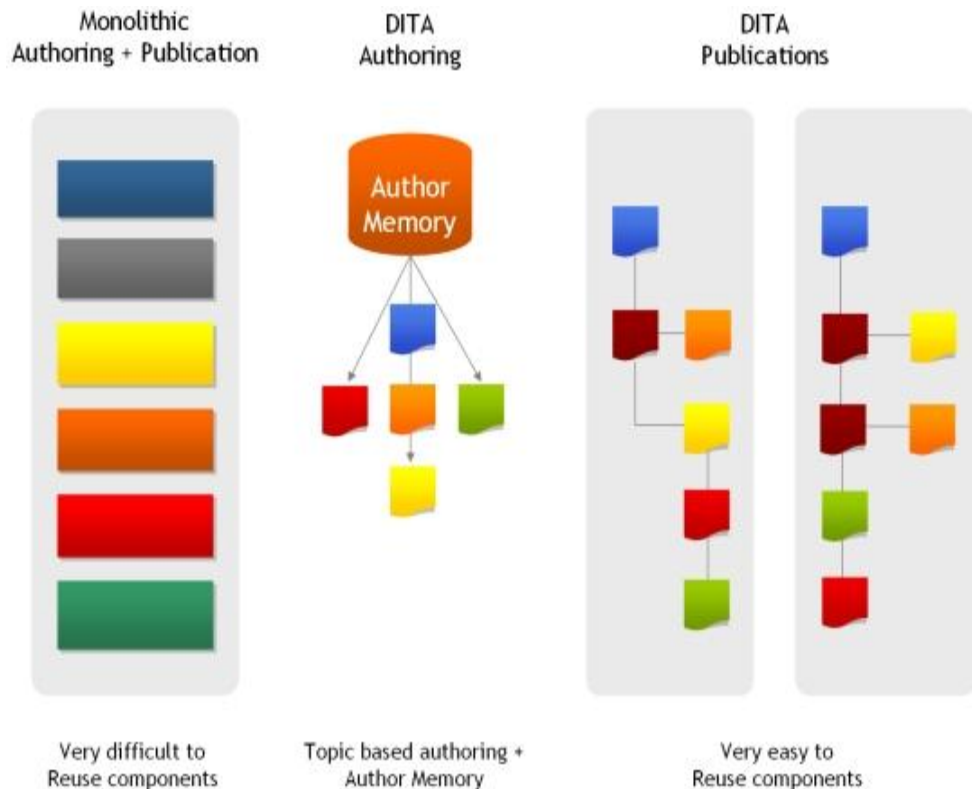


Figure 10. DITA benefits

From the translation point of view, OAXAL offers the following benefits:

- Once a topic has been translated it can be reused multiple times, as long as the source document has not been changed.
- Greater granularity means that translation does not have to wait for the completion of a publication. Individual topics are ready for localization as soon as each one is ready.
- xml:tm adds the ability to significantly increase the scope and effectiveness of translation memory:
  - xml:tm allows for standards-based implementation of ICE matching. Exact matches virtually eliminate the need for proofing these types of matches.
  - The xml:tm structure of a document allows for a much more "focused" approach to leveraged and fuzzy matching:
    - In-document leveraged matching: xml:tm enables matching of identical sentences within a document. These matches will be of a higher order, as they are from the same document.
    - In-document fuzzy matching: xml:tm allows for the easy identification of previous versions of modified sentences and can present these as "in-document fuzzy matches." These will perform better of a higher order than database sources fuzzy matches, as they are based on the previous version of the same sentence.
    - Database source leveraged matching: the xml:tm view of a document represents a predigested version that is ready for translation memory operations.

- Nontranslatable text: xml:tm flags nontranslatable text as part of the namespace seeding, thus making it easier for any matching software to automatically process such text accordingly.

Figure 11 demonstrates the various translation memory matching approaches available via xml:tm.

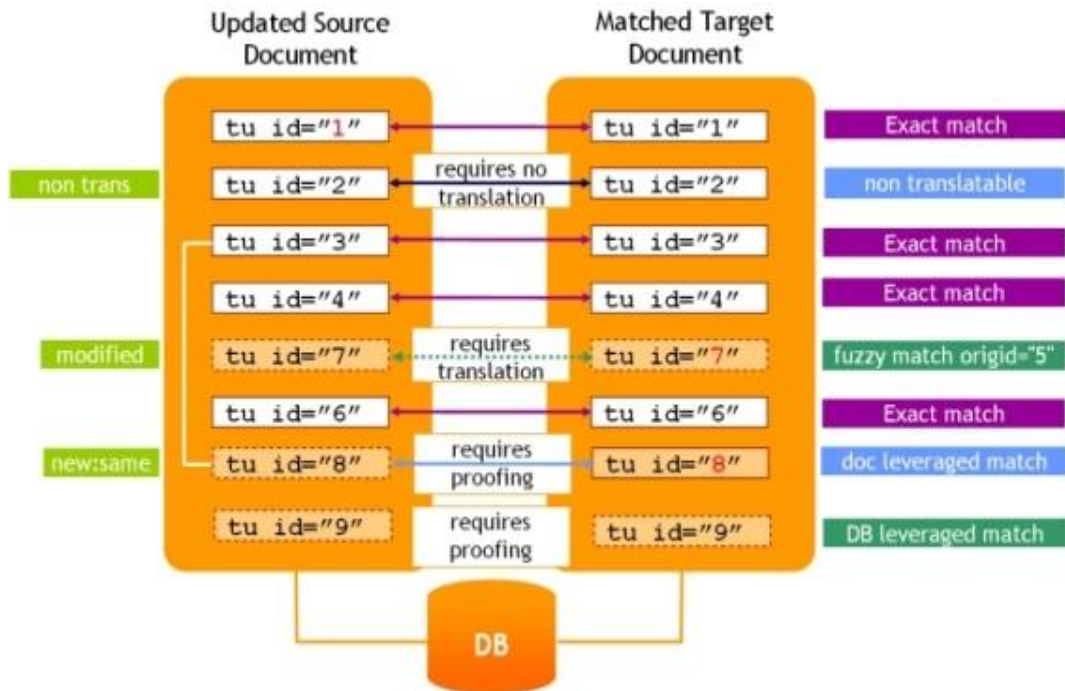
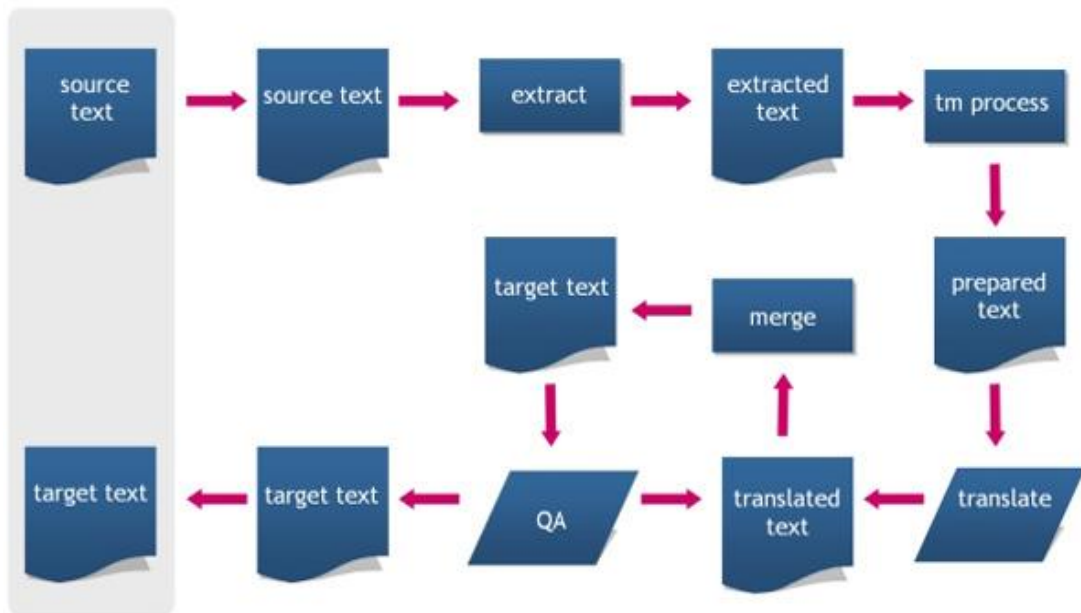


Figure 11. xml:tm focused matching

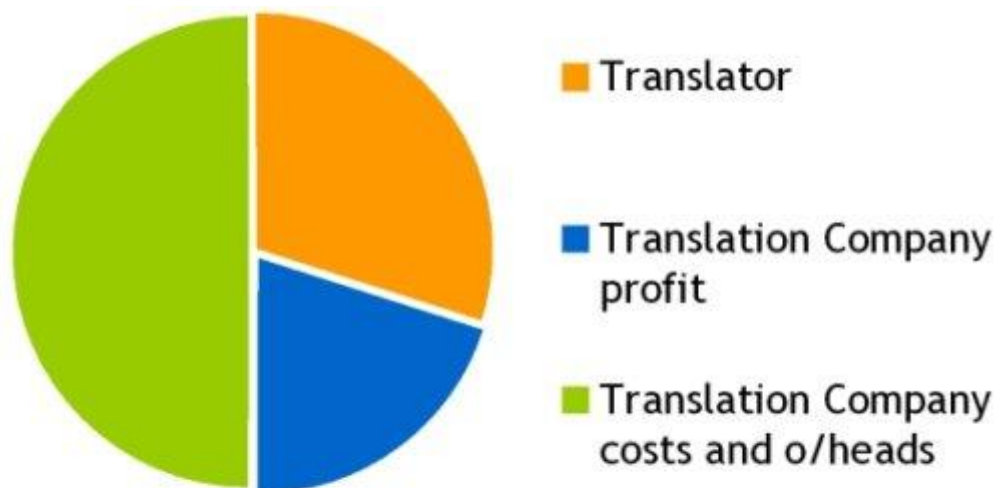
The most significant benefits that OAXAL brings to the translation process are the automation of what previously was a very manual process. Figure 12 illustrates the traditional approach to localization:

## Translation



*Figure 12. Traditional translation process*

This approach is very expensive. Every red arrow is potentially a point of failure in the workflow. Running such a process requires a lot of project management time and effort, as well as complex manual steps. The costs of running such a system are shown in Figure 13, which demonstrates the typical financial model for localization tasks.



Source Professor Reinhard Schäler LRC - ASLIB 2002

*Figure 13. Traditional translation process financial model*

The actual translation in this model accounts for only one third of the localization

process. Nearly half of the cost can be eliminated using the OAXAL model, as shown in Figure 14.

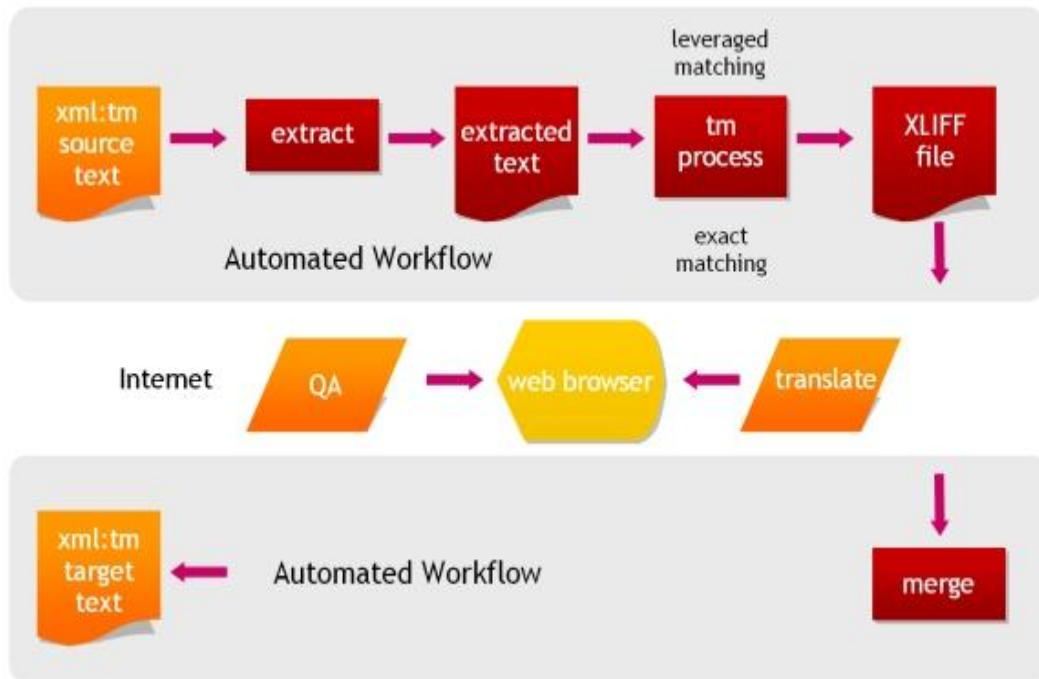


Figure 14. OAXAL-based translation process

The OAXAL model provides full process automation, right up to delivering matched files to the translator. Automation eliminates the costs associated with project management and manual processes. Data gets processed faster and more efficiently and without the costs associated with a traditional localization workflow.