Reference Model for Open Architecture for XML Authoring and Localization Version 1.0

Committee Draft 01 / Public Review Draft 01

20 March 2009

Specification URIs:

This Version:
http://docs.oasis-open.org/oaxal/V1.0/cd01/oaxal-v1.0-cd01.html (Authoritative)
http://docs.oasis-open.org/oaxal/V1.0/cd01/oaxal-v1.0-cd01.pdf

Previous Version:
N/A

Latest Version:
http://docs.oasis-open.org/oaxal/V1.0/oaxal-v1.0.html
http://docs.oasis-open.org/oaxal/V1.0/oaxal-v1.0.pdf

Technical Committee:
Open Architecture for XML Authoring and Localization (OAXAL) TC

Chair(s):
Andrzej Zydroń

Editor(s):
Andrzej Zydroń
Derek Saldana

Related Work:
This specification replaces or supercedes:

Declared XML Namespace(s):

N/A

Abstract:

The Open Architecture for XML Authoring and Localization (OAXAL) provides a comprehensive, efficient, and cost-effective model for building an XML lifecycle production framework based completely on Open Standards from ic trademarked names, abbreviations, etc. here] are trademarks of OASIS, LISA OSCAR and W3C.

Status:

This document was last revised or approved by the OAXAL TC on the above date. The level of approval is also listed above. Check the "Latest Version" or "Latest Approved Version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/oaxal/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/oaxal/ipr.php.

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/oaxal/.

Notices

Table of Contents

B. Acknowledgments

1 Introduction

The Open Architecture for XML Authoring and **Localization** (OAXAL) represents a comprehensive, efficient, and cost-effective model regarding the authoring and translation aspects of XML publishing. OAXAL encompasses the following key Open Standards:

- XML - Extensible Markup Language (XML) is a simple, flexible text format originally designed to meet the challenges of large-scale electronic publishing. XML also plays an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.
- Unicode - A character encoding scheme that encompasses all character sets.
- W3C ITS - An XML vocabulary that defines translatability rules for a given XML document type.
- SRX - Segmentation Rules eXchange, a LISA OSCAR standard defining text-subdivision rules for each language.
- xml:tm - XML-based text memory, a LISA OSCAR standard for author memory (a history of segments and revisions) and translation memory (a history of translated segments).
- GMX - Global Information Management Metrics Exchange, a LISA OSCAR standard for word and character count and metrics (for volume, complexity, and quality) exchange.
- TMX - Translation Memory eXchange, a LISA OSCAR standard for exchanging translation memories.
- Unicode TR29 - The primary Unicode standard defining word and sentence boundaries.
- Open Standard XML Vocabularies, including DITA, Docbook, XHTML, SVG, ODF, and others that may emerge as standards.
- XLIFF - XML Localization Interchange File Format, an OASIS standard for exchanging Localization data.

The architectural model for OAXAL is as follows:

Figure 1: OAXAL Standards Component Stack

The following diagram represents the interaction of the key OAXAL standards:

Figure 2: OAXAL Interaction of Standards

This diagram is annotated and described in detail in subsequent parts of this document. OAXAL is designed to cope with the common requirements for XML authoring and Localization. The authoring plus Localization aspects of OAXAL are most effective

within a Content Management System (**CMS**) environment. For a translation-only **workflow**, OAXAL can be implemented without a CMS system.

OAXAL is designed to integrate tightly and transparently within the document-life-cycle workflow model which includes:

- document creation
- the authoring cycle
- Localization

- subsequent document updates and the Localization thereof

For the translation-only environment, OAXAL provides an elegant and open architecture for processing XML documents for translation.

1.1 What is a reference model

A *reference model* is an abstract framework for understanding significant relationships among the entities of some environment. It enables the development of specific reference or concrete architectures using consistent standards or specifications supporting that environment. A reference model consists of a minimal set of unifying concepts, axioms, and relationships within a particular problem domain and is independent of specific standards, technologies, implementations, or other concrete details.

As an illustration of the relationship between a reference model and the architectures that can derive from such a model, consider what might be involved in modeling important aspects of residential housing. In the context of a reference model, we know that concepts such as eating areas, hygiene areas, and sleeping areas are all important in understanding what goes into a house. There are relationships among these concepts and constraints on their implementation. For example, there may be a physical separation between eating areas and hygiene areas.

The role of a reference architecture for housing would be to identify abstract solutions to the problems of providing housing. A general pattern for housing, one that addresses the needs of its occupants in the sense of, say, noting that there are bedrooms, kitchens, hallways, and so on is a good basis for an abstract reference architecture. The concept of "eating area" is a reference model concept; a kitchen is a realization of "eating area" in the context of the reference architecture.

There may be more than one reference architecture that addresses how to design housing; for example, there may be a reference architecture to address the requirements for developing housing solutions in large apartment complexes, another to address suburban single family houses, and another for space stations. In the context of high-density housing, there may not be a separate kitchen but rather a shared cooking space or even a communal kitchen used by many families.

An actual – or *concrete* – architecture would introduce additional elements. It would incorporate particular architectural styles, particular arrangements of windows, construction materials to be used, and so on. A blueprint of a particular house represents a specific architecture as it applies to a proposed or an actual constructed dwelling.

The reference model for housing is, therefore, at least three levels of abstraction away from a physical entity that can be lived in. The purpose of a reference model is to provide a common conceptual framework that can be used consistently across different

implementations and is of particular use in modeling specific solutions.

1.2 A Reference Model for Open Architecture for XML Authoring and Localization

The goal of this reference model is to define the component parts of XML publishing with respect to the authoring and Localization aspects of the process. It provides a normative reference that remains relevant for OAXAL as a comprehensive model.

The OAXAL standards components stack shows how the reference model for OAXAL is constructed from its constituent Open Standards. The concepts and relationships defined by the reference model are the basis for describing the reference architecture.

Architecture must account for the goals, motivation, and requirements that define the actual problems being addressed. While reference architectures can form the basis of classes of solutions, concrete architectures will define specific solution approaches.

Architecture is often developed in the context of a pre-defined environment, such as the protocols, profiles, specifications, and standards that are pertinent.

OAXAL implementations combine all of these elements, from the more generic architectural principles and infrastructure to the specifics that define the current needs, and represent specific implementations that will be built and used in an operational environment.

1.3 Audience

The intended audiences of this document include (non-exhaustively):

- Architects and developers designing, identifying, or developing a system based on OAXAL
- Standards architects and analysts developing specifications that rely on OAXAL
- Decision makers seeking a "consistent and common" understanding of OAXAL
- Users who need a better understanding of the concepts and benefits of OAXAL

1.4 Guide to using the reference model

New readers are encouraged to read this reference model in its entirety. Concepts are presented in an order that the authors hope promote rapid understanding.

This section introduces the conventions, defines the audience, and sets the stage for the rest of the document. Non-technical readers are encouraged to read this information because it provides background material necessary to understand the nature and use of reference models.

- The Open Architecture for XML Authoring and Localization section introduces the concept of OAXAL and identifies some of the ways that it differs from previous paradigms for authoring and translation systems. This section offers guidance on

the basic principles of OAXAL. This section can be used by non-technical readers to gain an explicit understanding of the core principles of OAXAL and by architects as guidance for developing OAXAL-based architectures.

- The Reference Model section introduces the Reference Model for OAXAL.

- The Conformance Guidelines section addresses compliance with this reference model.

The glossary provides definitions of terms within the reference-model specification but does not necessarily form part of the specification itself. Terms that are defined in the glossary are marked in bold at their first occurrence in this document.

Note that while the concepts and relationships described in this reference model may apply to other "service" environments, the definitions and descriptions contained herein focus on the field of software architecture and make no attempt to completely account for use outside of the software domain. Examples included in this document that are taken from other domains are used strictly for illustrative purposes.

1.5 Notational Conventions

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in RFC2119.

1.6 Normative References

[W3C ITS] Internationalization Tag Set (ITS) Version 1.0 http://www.w3.org/TR/2007/REC-its-20070403/ 03 April 2007

[SRX] SRX 2.0 Specification http://www.lisa.org/Segmentation-Rules-e.40.0.html OSCAR Recommendation, 7 April 2008

[xml:tm] XML Text Memory (xml:tm) 1.0 Specification http://www.lisa.org/XML-Text-Memory-xml.107.0.html 26 February 2007

[GMX] Global Information Management Metrics Volume (GMX-V) 1.0 Specification http://www.lisa.org/Global-information-m.104.0.html Version 1.0, 26 February 2007

[TMX] Translation Memory eXchange format (TMX) Draft Specification http://www.lisa.org/fileadmin/standards/tmx2/tmx.html Version 2.0, October 15, 2007

[Unicode TR29] Unicode Standard Annex #29 http://unicode.org/reports/tr29/ Unicode 5.1.0 2008-03-25

[DITA] Darwin Information Typing Architecture http://docs.oasis-open.org/dita/v1.1/CS01/overview/overview.html Verion 1.1 31 May 2007

[DocBook] DocBook http://www.docbook.org/specs/cs-docbook-docbook-4.2.pdf Committee Specification 4.2, 16 July 2002

[XHTML] XHTML™ 1.1 - Module-based XHTML - Second Edition http://www.w3.org/TR/xhtml11/ W3C Working Draft 16 February 2007

[SVG] Scalable Vector Graphics http://www.w3.org/TR/SVG12/ Working Draft 1.2 13 April 2005

[ODF] Open Document Format for Office Applications (OpenDocument) http://docs.oasis-open.org/office/v1.1/OS/OpenDocument-v1.1-html/OpenDocument-v1.1.html v1.1 1 Feb 2007

[XLIFF] XML Localization Interchange File Format (XLIFF) http://docs.oasis-open.org/xliff/v1.2/os/xliff-core.html Version 1.2 1 February 2008

1.7 Non-Normative References

[LISA] Localization Industry Standards Association

[LISA OSCAR] LISA Open Standards for Container/content Allowing Reuse

[OASIS] Organization for the Advancement of Structured Information Standards

[W3C] The World Wide Web Consortium

2 Open Architecture for XML Authoring and Localization

Open Architecture for XML Authoring and Localization (OAXAL) is a reference model of how to construct an effective and efficient system for XML authoring and Localization based on Open Standards. OAXAL comprises the following standards:

- XML -- Extensible Markup Language (XML) is a simple, flexible text format originally designed to meet the challenges of large-scale electronic publishing. XML also plays an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.
- Unicode - A character encoding scheme that encompasses all character sets.
- W3C ITS W3C ITS - An XML vocabulary that defines translatability rules for a given XMLdocument type.
- Unicode TR29 - The primary Unicode standard defining word and sentence boundaries.
- SRX - Segmentation Rules eXchange, an XML vocabulary defining segmentation rules for each language.
- xml:tm - XML-based text memory, a LISA OSCAR standard for author and translation memory.

- GMX - Global Information Management Metrics Exchange, a LISA OSCAR standard for word and character count and metrics exchange.
- TMX - Translation Memory eXchange, a LISA OSCAR standard for exchanging translation memories.
- Open Standard XML Vocabularies, including DITA, Docbook, XHTML, SVG, ODF, and others that may emerge as standards.
- XLIFF - XML Localization Interchange File Format, an OASIS standard for exchanging Localization data.

This Reference Model will demonstrate the integration of the standards listed above to present a complete automated package from authoring through translation; additional standards may be added by the Technical Committee (TC), and the TC may elect not to include (or to make optional) any of the standards listed above that prove, upon review, not to be feasible or useful to integrate in its profiles. Authors are provided with a systematic way to identify and store all previously authored sentences. OAXAL allows for some variation in how the standards are used and integrated. OAXAL variants may not use all of the standards enumerated above.

2.1 Authoring and Localization Reference Workflow

Key to the concept of OAXAL are Authoring and Localization. Authoring in OAXAL implies XML-based source and the concept of a document lifecycle centered around some form of content management. Content management may be achieved by means of a fully fledged Content Management System (CMS) or a Source Control System (**SCS**). The document lifecycle implies one or more of the following stages:

1. Initial document creation
2. Authoring, Editing, and Validation/Approval
3. Publishing of the source-language version
4. Localization/Translation and Post-editing of the translated content
5. Publishing of the target-language material
6. Subsequent changes to the source material that require re-publishing and Localization/translation.

OAXAL is designed to provide an effective and elegant solution to these requirements within an authoring/localization workflow:

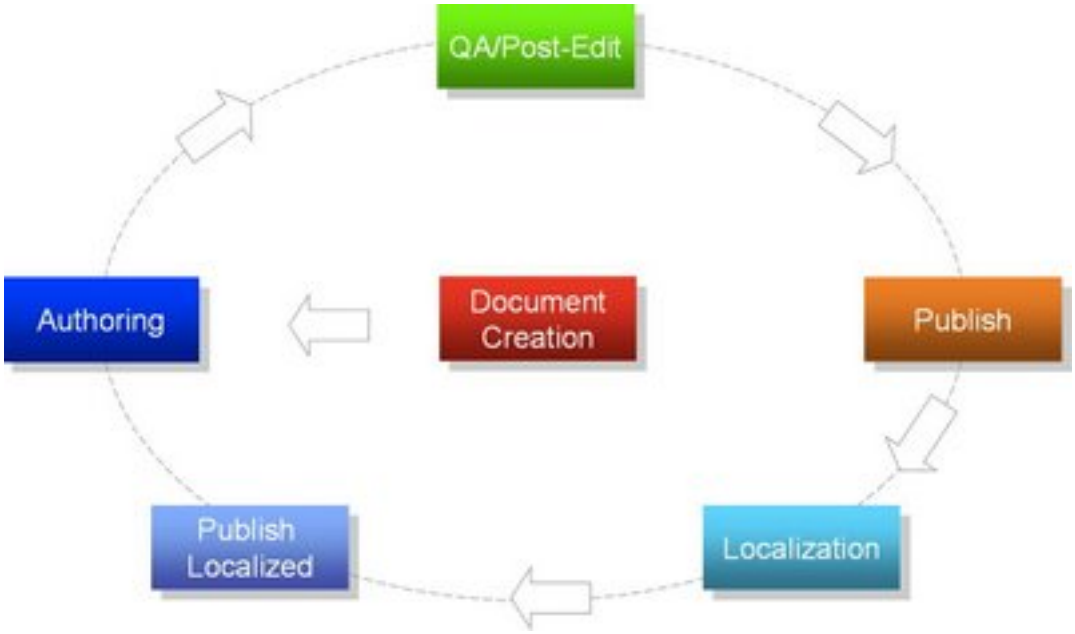

Figure 3: Source Document Lifecycle

OAXAL can also be used to design Localization-only solutions. In this instance, a document is submitted for Localization. The format of the document may not be XML; nevertheless, OAXAL assumes a conversion to an XML form of the data prior to processing and a conversion back into the original format on completion. Translation/Localization comprises the following steps:

1. Identification of the text to be localized
2. Segmentation of the text into sentences if required
3. Matching of the sentences with previous translated versions of the document if possible
4. Translation of the text
5. QA/Post-editing of the translated text
6. Merging/recreation of the original document in the target language

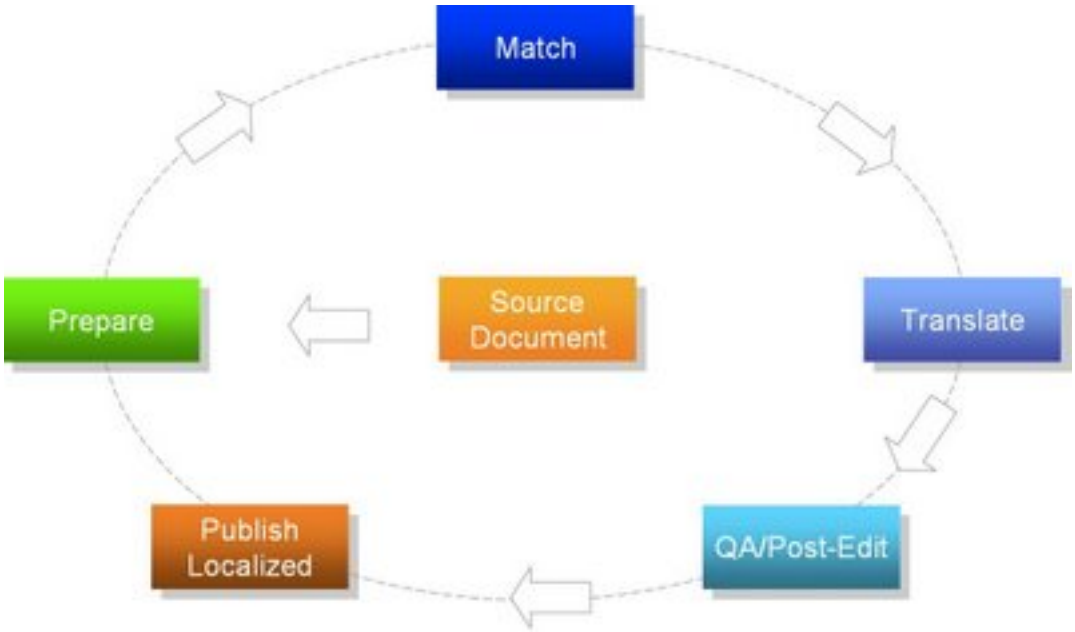

Figure 4: Document Localization Lifecycle

2.2 OAXAL Components

2.2.1 Unicode

Unicode provides the underlying character encoding for OAXAL. Although XML allows various encoding schemes based on 7 and 8 bits, OAXAL mandates full Unicode encoding, preferably using UTF-8 or UTF16 encoding. The benefits of using Unicode character encoding for OAXAL are as follows:

- There are no character restrictions. Even if the text does not require non-ASCII characters, Unicode allows for the use of special-width spaces, hyphens, and other frequently-used characters that fall outside of the scope of 7- and 8-bit character-encoding schemes.
- The same encoding scheme is used for both source- and any target-language character requirements, thus greatly simplifying all aspects of editing and publication.

2.2.2 XML

The key characteristic of OAXAL is the use of an Open Architecture based on Open Standards with XML as the source format for both the document format and, in most cases, the vocabulary of the standards. XML underpins the foundations of OAXAL. The XML source content provides semantic and structured text that can be localized. XML provides many benefits regarding authoring and Localization:

1. The separation of form and content provides an elegant and convenient way of identifying text and markup.
2. The extensible nature of XML allows the creation of specialist vocabularies that can be shared and adopted by interested parties.
3. The syntax of XML allows for the quick and easy validation of XML document instances against specific rules.
4. The widespread adoption of XML means that there are many tools to validate and transform XML documents.

At the center of authoring and Localization is the actual XML document text to be authored and/or localized. OAXAL encompasses all publishing-oriented Open Standard XML vocabularies such as DITA, Docbook, XHTML, SVG, ODF, and others that may emerge as standards. OAXAL may also be used with proprietary XML vocabularies or with non-XML based documents that are converted into a non-Open Standard XML format.

2.2.3 W3C ITS

W3C ITS is the Internationalization Tag Set Recommendation. ITS allows for the declaration of Document Rules for Localization. In effect, it provides a vocabulary that allows the declaration of the following for a given XML document type:

1. Which attributes are translatable
2. Which elements are 'in line'—that is, which elements do not break the linguistic flow of text (for example: 'emphasis' elements)
3. Which inline elements are 'sub flows'—that is, although they are inline, which inline elements do not form part of the

linguistic flow of the encompassing text (for example: 'footer' and 'index' markers)

W3C ITS provides many more features, including a namespace vocabulary that allows for fine-tuning Localization for individual instances of elements within a document instance. W3C ITS is therefore at the core of XML Localization processing.

In OAXAL, W3C ITS stipulates the rules by which an XML document is localized in terms of its translatable content.

2.2.4 Unicode TR29

Unicode TR29 is the Unicode standard defining word and sentence boundaries. It allows for a uniform way of defining word boundaries for OAXAL and, as such, is used by SRX and GMX/V to tokenize text into individual words. It plays a fundamental role in OAXAL.

2.2.5 SRX

SRX - Segmentation Rules eXchange is an Open Standard XML vocabulary for defining the segmentation rules for a given language published by LISA OSCAR. Segmentation is an important aspect of both the authoring and Localization processes. SRX allows OAXAL to have a sentence-level granularity. SRX depends on Unicode TR29 in order to provide the basis for tokenizing text into individual words.

2.2.6 xml:tm

xml:tm is a namespace vocabulary providing a LISA OSCAR standard for author and translation memory. xml:tm is a key component of OAXAL. xml:tm introduces the concept of XML-based text memory that encompasses both in-document author memory and translation memory. In the xml:tm scenario, author and translation memory are embedded within the XML document, providing both an edit-change history of the document as well as the mechanism for 'In Context Exact' (ICE) translation-memory matching. ICE matching guarantees that the text-unit matches are from exactly the same source as the previous iteration of an updated document, as opposed to leveraged matching which cannot guarantee the provenance of a 100% match.

Given a W3C ITS rule set for a given XML vocabulary and the SRX segmentation rules for a given language, it is possible to construct a totally generic process for embedding the xml:tm text-memory namespace within the source document. xml:tm relies on W3C ITS and SRX. xml:tm allocates immutable unique identifiers to each translatable text content or a subdivision of such text content, resulting in identifiable individual sentences.

The key role of xml:tm within OAXAL is in preparing an XML document for further processing as well as providing the syntactical basis for ICE matching.

2.2.7 GMX

GMX - Global Information Management Metrics Exchange is a LISA OSCAR standard for word and character count and metrics exchange. GMX is a tri-partite set of standards:

- GMX/V - Volume, detailing word- and character-count metrics

- GMX/C - Complexity, detailing the level of complexity of a given document in translation terms
- GMX/Q - Quality, detailing the level of quality required for the translation of a given document

Currently only GMX/V has been defined. GMX/V is a key component of OAXAL in terms of providing a uniform and consistent way of calculating the word- and character-count metrics for a given document or set of documents, as well as providing a way of embedding and exchanging such information. GMX/V depends on Unicode TR29 in order to provide the basis for tokenizing text into individual words. GMX/V also uses XLIFF as the canonical form for counting.

2.2.8 TMX

TMX - Translation Memory eXchange is a LISA OSCAR standard for exchanging translation memories. TMX is a key component of OAXAL, allowing for the free exchange of translation memories.

2.2.9 XLIFF

XLIFF - XML Localization Interchange File Format is an OASIS standard for exchanging Localization data. Within OAXAL, the previously described standards help prepare the XML document for translation. The xml:tm version of the document contains all of the information required for extraction and both ICE and in-document leveraged and fuzzy matching. The transformation and matching process that goes into creating an XLIFF version of the document creates a document that can be processed and exchanged by any software that can read and understand an XLIFF file. XLIFF provides an important element of protection regarding the original XML document as well as a means to embed matching information.

2.3 Interaction of Standards

The key concept of OAXAL concerns how to build an efficient and effective systems architecture based on its constituent standards. The most important aspect of this architecture is how the standards interact with one another.

Figure 5: OAXAL Interaction of Standards

2.3.1 Unicode TR29

Unicode TR29 is used by SRX and GMX/V to tokenize text into white space, words, and punctuation. This tokenization is key to processing text for segmentation (SRX) and metrics (GMX/V).

2.3.2 SRX

SRX is used to segment text into individual sentences by xml:tm.

2.3.3 W3C ITS

W3C ITS provides the rule set and in-document namespace directives for identifying translatable text within document elements and attributes. It is sufficient to create a W3C ITS rules file for a given XML vocabulary such as DITA or ODF to allow all such documents to be processed by OAXAL. There is no need to write separate filter programs for each XML vocabulary. W3C ITS is used by xml:tm to identify translatable text and segment it using SRX.

2.3.4 xml:tm

xml:tm provides the basis of sentence-based text extraction by XLIFF as well as the foundation for ICE matching and all in-document leveraged and fuzzy matching. xml:tm can also be used to create TMX files from the aligned source and target versions of the document.

xml:tm relies on:

- W3C ITS to identify the translatable text content of a document
- SRX (which requires Unicode TR29) to segment text into individual sentences, and
- GMX/V to produce authoring and document statistics

2.3.5 GMX/V

GMX/V is used to provide all of the metrics for XLIFF extraction and matching, as well as xml:tm authoring metrics during the document life cycle.

2.3.6 XLIFF

XLIFF is used by GMV/V as the canonical form for metric-counting purposes, as well as providing the basis for TMX files based on the source and translated segments. Within OAXAL, XLIFF uses xml:tm to identify **text units** requiring translation, as well as GMX/V for metrics in terms of how many words/characters require translation.

2.4 Major Processing Features

The true benefits of OAXAL accrue from the ability to produce a generic processing model for XML Authoring and Localization. The xml:tm and XLIFF operations are conducted by general-purpose programs which are completely parameter driven by input from the other standards. This parameterization allows for an elegant and very efficient process that is totally generic and easy to maintain. This benefit can be extended to non-XML document formats by converting them to an XML form and then processing them via OAXAL.

2.4.1 The Traditional Localization Workflow

In the traditional Localization scenario, there is little or no automation of the Localization process. A file, or group of files, is handed over to a Localization facility, and the subsequent workflow is made up of the following activities:

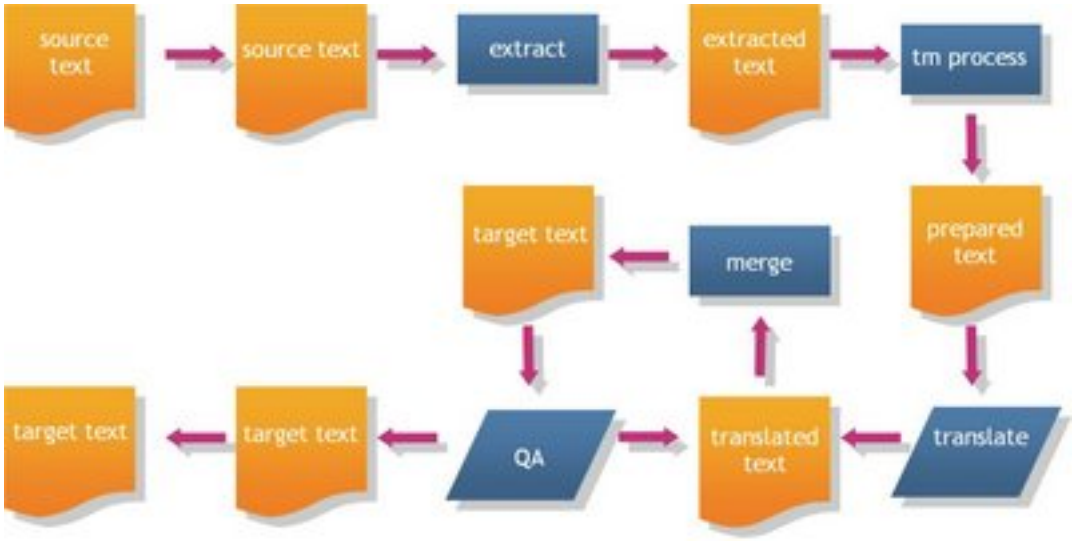

Figure 6: Traditional Localization Workflow

Each of the arrows in this workflow model represents a potential point of failure as well as manual intervention. Not only is this process very error prone, it also adds significantly to the cost of Localization. The following cost model for this scenario was presented by Prof. Reinhard Schäler of the Limerick University Localisation Research Centre at the Aslib Conference in London in 2002:

Figure 7: Prof. Reinhard Schäler ASLIB 2002

The lack of an automated workflow has a very detrimental affect on the Localization process. Without automation, considerable manual intervention is required, as is evidenced in the figure Traditional Localization Workflow. This lack of automation accounts for up to 50% of the total cost of Localization.

2.4.2 OAXAL Localization Workflow

The Localization workflow using OAXAL significantly reduces the processing costs:

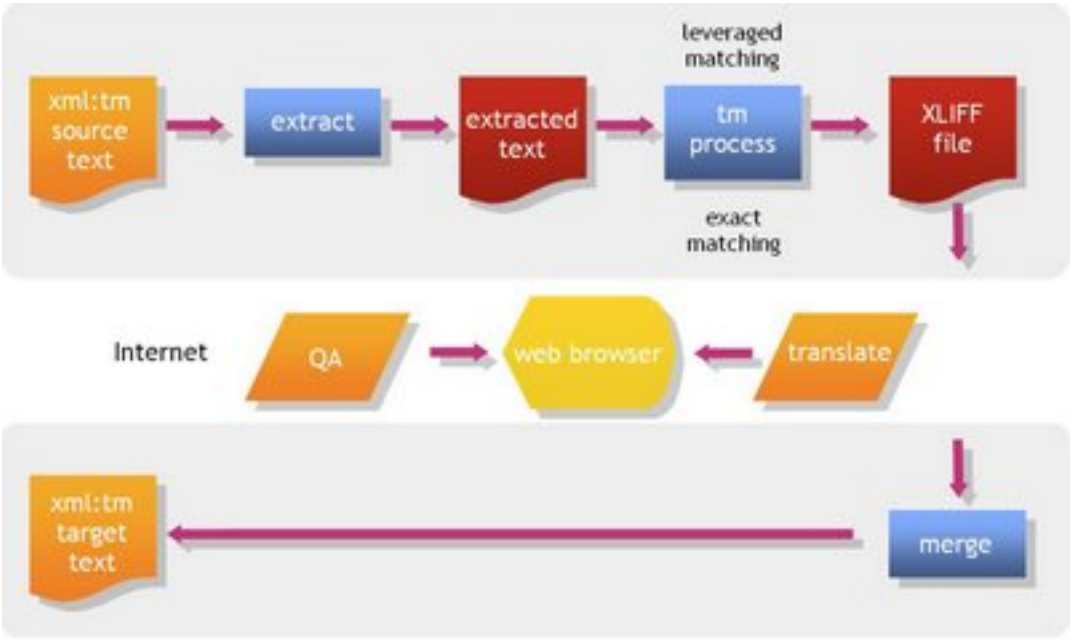

Figure 8: OAXAL Automated Localization Workflow

All of the processing steps, apart from the actual translation and QA activities, are completely automated. In addition, the use of XLIFF as the interchange standard means that translation can be presented via a browser interface, thus significantly simplifying the whole process.

3 The Reference Model

Figure 9: OAXAL Reference Model

There are two prime use cases for OAXAL:

1. Handling authoring and Localization consistency within a CMS environment

2. Handling Localization consistency within a translation-only environment

The essence of OAXAL is to facilitate the following:

1. Consistency in the authored source-language text
2. Consistency in the translated text
3. Substantial automation of the Localization process

These results lead in the long term to reduced translation and authoring costs as well as improvements in the quality of the documentation.

3.1 OAXAL within a CMS environment

OAXAL is fundamentally rooted in the concept of a document life cycle. The life-cycle steps comprise the following:

1. Document creation
2. Authoring cycle, including review
3. Localization

4. Update cycle
5. Localization of the updated text

Thus, a document is created. It is authored by one or more writers and submitted to editorial review and correction. The document is subsequently published in the source language and localized into one or more target languages for publication. The document is then subjected to further modifications according to the requirements of the business unit that is charged with maintaining it. The updated document then requires localization again to translate any new or modified text, and so on during its existence. This paradigm is typical of the vast majority of technical documentation life-cycle processes.

The unit of granularity defined by OAXAL is the text unit. A text unit is either of the following:

- an identifiable sentence within an XML non-inline element
- the whole text content of a non-inline element, if no further subdivision into sentences is possible (also known as a standalone piece of text)

OAXAL can be viewed in terms of a workflow comprising a series of processes that interact with the source text:

1. The XML source content provides the semantic and structured text that needs to be consistently authored and localized.
2. The W3C ITS rules provide the XML syntax to define which parts of the XML document are translatable (including translatable attributes) and additional important information, such as which XML elements do not break the linguistic integrity of surrounding text (so-called 'within text' elements, commonly known as in-line elements).
3. Unicode TR29 provides the definition of what constitutes text tokens within text. Tokens are words, punctuation, white space, and so on. Tokenization is an essential process for segmentation and for classifying text units.

4. SRX provides an XML vocabulary for defining the detailed segmentation rules required to identify individual sentences.
5. The xml:tm namespace provides the means to manage the text at the sentence level. Each individual text unit (identifiable sentence or standalone piece of text) is allocated, among others, the following:

    i. An immutable unique identifier
    ii. A classification type (normal text, numeric, measurement, and so on)
    iii. A key based on the CRC of the text

6. XLIFF is an XML vocabulary that provides the means for packaging the text to be translated in a specific format especially designed for exchanging localizable data. XLIFF also provides the means for adding matching data, terminology, and metrics, as well as protecting the original XML document from accidental damage during the localization process. The xml:tm namespace version of the XML document provides all of the information required to easily and efficiently identify translatable text, but also for 'in context exact' (ICE) matching based on previous source and target versions of the document.

7. GMX/V provides an open, verifiable, and well-defined way of calculating localization metrics based on the XLIFF canonical form, Unicode TR29, as well as an XML vocabulary for exchanging localization metrics.

8. TMX is an XML vocabulary for exchanging translation-memory data. XLIFF files can be generated from translated XLIFF as well as source and translated xml:tm versions of the same document.

The whole OAXAL environment is best viewed in terms of an authoring and localization workflow, encompassing the following:

1. Check out for authoring
2. Check in from authoring
3. Check out for translation
4. Check in from translation

3.1.1 Authoring Workflow

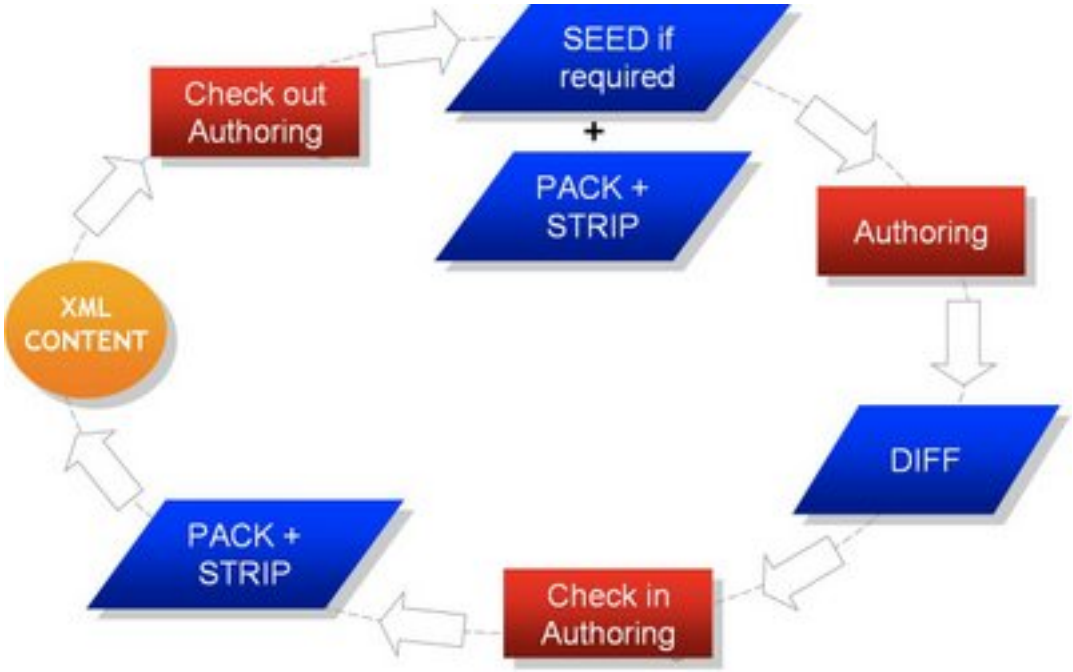

Figure 10: Authoring Workflow

1. The XML document is created and checked out for authoring.
2. As part of the check out for authoring, the document is inspected to see if the xml:tm namespace exists within the document. If not, the xml:tm namespace is seeded into the document.
3. The seeding process includes the following:
    i. Read in the W3C ITS document rules.
    ii. Identify all translatable text.
    iii. Tokenize the text according to Unicode TR29.
    iv. Read in the SRX segmentation rules for the language.
    v. Apply the segmentation rules to the tokenized text stream.
    vi. Identify all text units.
    vii. Allocate unique identifiers and CRC key and category data to the individual text units.
    viii. Update the xml:tm document's overall housekeeping and administrative data.
    ix. The xml:tm version of the document is either stored in the CMS system or is zipped, base-64 encoded, and stored as a processing instruction within the document itself.
4. The document is authored and checked in again.

5. The new document is seeded with the xml:tm namespace as detailed above.
6. The original and new xml:tm versions of the document are compared with each other, and the text units that have not changed are identified. The immutable identifiers from the original document are carried over to the new version, while new or modified text units are allocated new immutable identifiers.
7. The updated xml:tm version of the document is either packed as a zipped base-64 processing instruction within the same document as detailed above or is stored as a separate entity in the CMS system.
8. The document is checked into the CMS system.

3.1.2 Localization Workflow

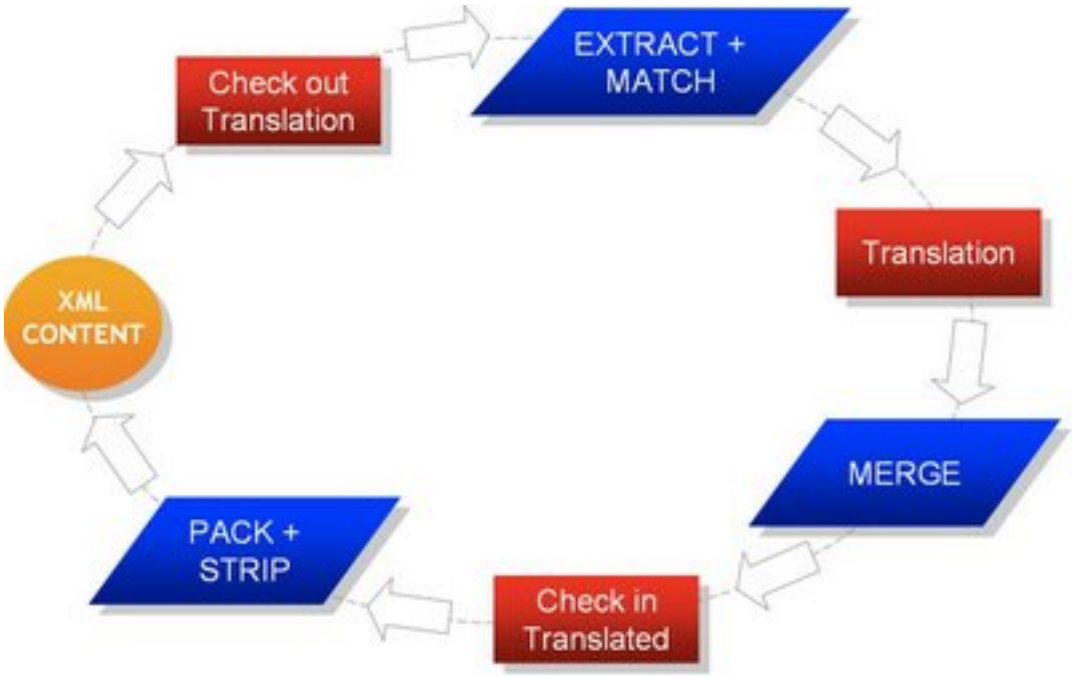

Figure 11: Localization Workflow

1. The document is checked out for translation. A target-language object is created in the CMS system for this version of the source document.
2. The previous target version of the document is located, and the xml:tm version is either unpacked or located in the CMS system.
3. The current xml:tm version of the source document is produced in a similar manner.
4. The XLIFF version of the document is produced along with a skeleton file by navigating through all of the xml:tm text units in the document. In-Context Matching, which provides a higher quality of translation-memory matching, is achieved by using the translated text from the previous localized version of the document, by means of text unit unique identifiers. The previous translated version of the document also allows for more focused leveraged-memory matching using the CRC keys for each text unit, as well as fuzzy matching based on modified text-unit identifiers. A skeleton file is created with placeholders where the translated text units are to be placed.
5. The XLIFF file is handed off to the localization process.
6. Once the XLIFF file has been translated, then the placeholders are used to 'merge' the localized text with the skeleton file.
7. The xml:tm version of the target document can either be zipped and packed as a base-64 processing instruction into a

version of the target document that has had the xml:tm namespace stripped out, or stored in the CMS system as a separate object.

8. The localized target version of the document is checked into the CMS system.
9. The leveraged translation-memory database is updated with the source and translated text-unit data.

3.2 OAXAL within a Localization-only Workflow

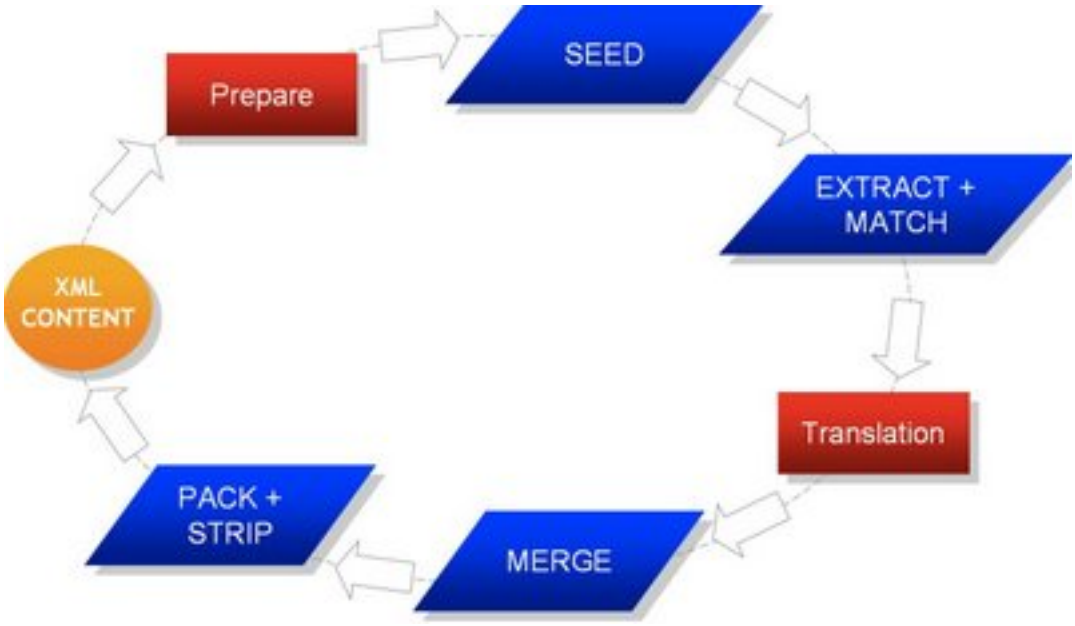

Figure 12: OAXAL Localization Only Workflow

Within a translation-only workflow, the same basic steps apply:

1. Introduce the xml:tm namespace into the current source and any previous versions of the source and target documents.
2. Align the previous source and target xml:tm versions of the documents.
3. Compare the differences between the source versions at the xml:tm level.
4. Carry 'backwards' the unique xml:tm identifiers onto the previous source and target xml:tm versions of the documents
5. Produce the XLIFF version of the document, along with a skeleton file, by navigating through all of the xml:tm text units in the document. In-Context Matching, which provides a higher quality of translation-memory matching, is achieved by using the translated text from the previous localized version of the document, by means of text-unit unique identifiers. The previous translated version of the document also allows for more focused leveraged-memory matching using the CRC keys for each text unit as well as fuzzy matching based on modified text unit identifiers. A skeleton file is created with placeholders where the translated text units are to be placed.
6. Hand off the XLIFF file to the localization process.
7. Once the XLIFF file has been translated, then use the placeholders to 'merge' the localized text with the skeleton file.
8. Zip and pack the xml:tm version of the target document as a base-64 processing instruction into a version of the target document that has had the xml:tm namespace stripped out.
9. Update the leveraged translation-memory database with the source and translated text unit data.

3.3 OAXAL Alternative Translation-only Workflow

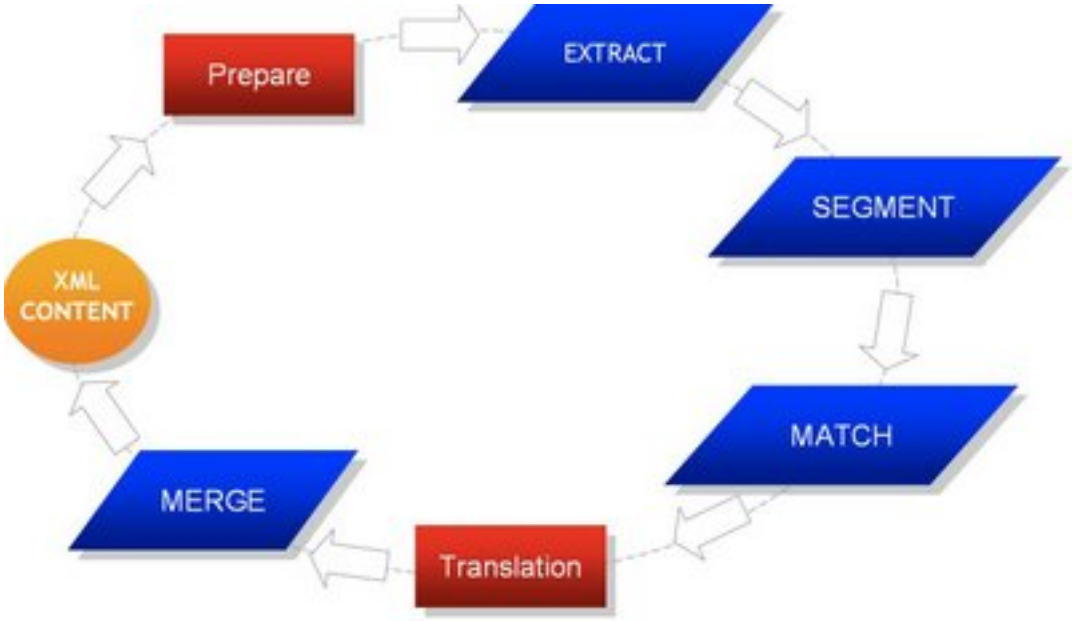

Figure 13: OAXAL Alternative Translation-only Workflow

An alternative translation-only workflow is possible, without the use of the xml:tm namespace:

1. Using the W3C ITS Document Rules, extract the text into XLIFF format and create a skeleton file with placeholders for the translated text.
2. Segment the XLIFF translation units by using SRX and Unicode TR29.
3. Match the segmented XLIFF sentences by using traditional database matching.
4. Hand off the XLIFF file to the localization process.
5. Once the XLIFF file has been translated, then use the placeholders to 'merge' the localized text with the skeleton file, and into paragraphs.
6. Update the leveraged translation-memory database with the source and translated text-unit data from the segmented XLIFF file.

3.4 OAXAL Operations

The OAXAL processes described in the above-mentioned use cases can be broken down into the following fundamental operations:

1. SEEDING - The introduction of the xml:tm namespace into a document. This operation involves ***updating*** an XML document with the xml:tm namespace. The required standards used are Unicode TR29, SRX, and W3C ITS. The xml:tm namespace is used to allocate a unique identifier to each translatable sentence or individual translatable standalone text segment. These are referred to as text units. The identifier is immutable for each text unit for the lifespan of the document. SEEDING takes place in the following operations:
    i. Check out for authoring, where no xml:tm version of the document exists
    ii. Check in from authoring, for the updated document
    iii. In the translation-only environment, for all relevant versions of the source and target documents
2. DIFFING - The comparison of the current and previous versions of a document, and maintaining the xml:tm namespace during the authoring lifecycle. For each update stage of a document, the original version of the document in its xml:tm

form is required, as well as the updated version with a fresh xml:tm namespace. The two documents are compared, and any unchanged xml:tm text elements inherit the identifiers from the original version, thus maintaining the immutable identifiers. DIFFING takes place in the following operations:

i. Check in from authoring
ii. Comparing current and previous versions of the source document in the translation-only environment

3. EXTRACTION - The creation of an XLIFF file and commensurate skeleton file, including document-centric matching based on the previous target and source versions of the document. Extraction takes place for all situations for the translation stage.
4. MERGING - The creation of the target version of the document, based on the translated XLIFF file and the skeleton file. The extraction process involves the identification of each translatable text unit, transferring it to an XLIFF document, and replacing the text unit with an identifier marking the location for the resultant translated text unit. A skeleton file is thus created, with the placeholders for the translated text. The extraction process also involves extracting all translatable text and implementing all document-centered matching (ICE, leveraged, and fuzzy) as well as database matching, resulting in the creation of an XLIFF file along with a commensurate skeleton file. MERGING takes place for all situations for the translation stage.

5. PACKING - An optional stage that is used to pack the xml:tm version of the document into the existing document as a zipped, base-64 encoded processing instruction. PACKING takes place in the following operations:
   i. Check out for authoring
   ii. Check in from authoring
   iii. Check in from translation
6. STRIPPING - The process of removing the xml:tm namespace from the document. Stripping takes place at all the major stages where a non-xml:tm version of the document is required.

3.4.1 SEEDING

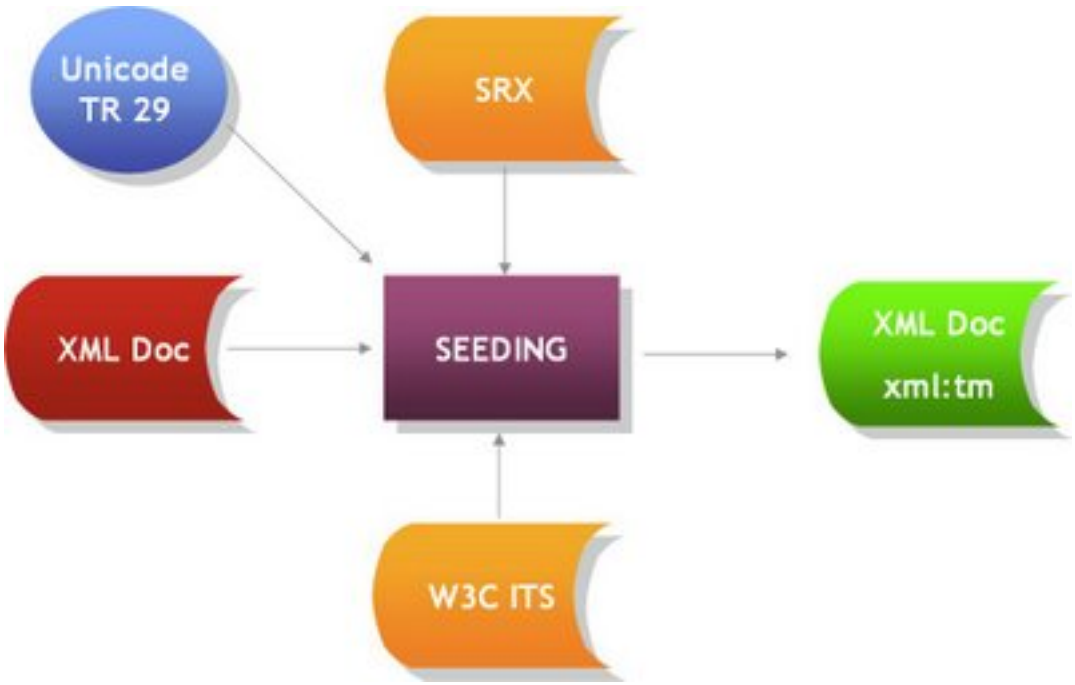

Figure 14: OAXAL xml:tm SEEDING

This operation involves *updating* an XML document with the xml:tm namespace. The required standards used are Unicode TR29, SRX, and W3C ITS. The xml:tm namespace is used to allocate a unique identifier to each translatable sentence or individual translatable standalone text segment. These are referred to as text units. The identifier is immutable for each text unit for the lifespan of the document.

The processing model for SEEDING is as follows:

1. Read in the W3C ITS document rules.
2. Read in the SRX segmentation rules for the language.
3. Read in the XML document.
4. Identify the elements and attributes containing translatable text.
5. Identify in-line elements and sub-flow elements.
6. Segment translatable text into individual text units.
7. Allocate the following to each text unit:
    i. Unique identifiers
    ii. CRC signature
    iii. Text unit classification: normal text, numeric, alphanumeric, measurement, and so on
8. Write out the newly seeded file.

SEEDING takes place in the following operations:

i. Check out for authoring, where no xml:tm version of the document exists
ii. Check in from authoring, for the updated document
iii. In the translation-only environment, for all relevant versions of the source and target documents

3.4.2 DIFFING

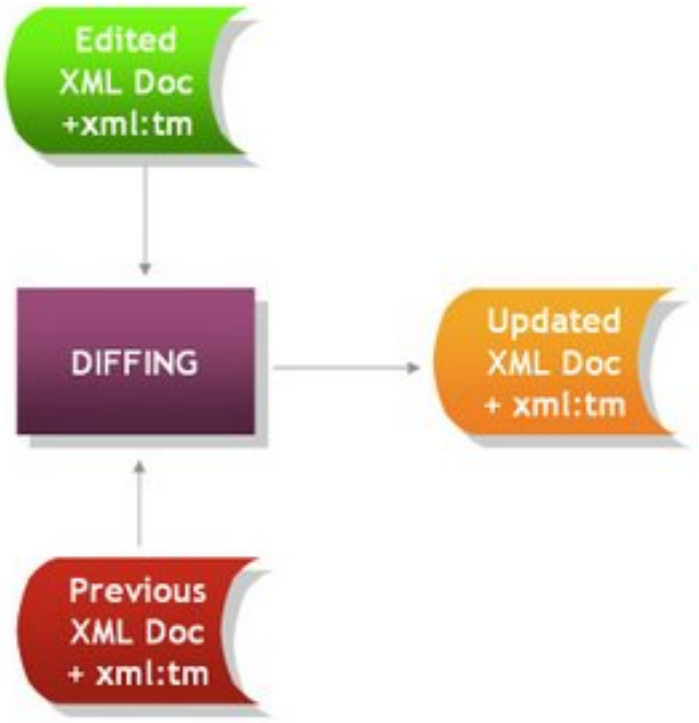

Figure 15: OAXAL DIFFING

For each update stage of a document, the original version of the document in its xml:tm form is required, as well as the updated version with a fresh xml:tm namespace. The two documents are compared, and any unchanged xml:tm text elements inherit the identifiers from the original version, thus maintaining the immutable identifiers.

The processing model for DIFFING is as follows:

1. Locate the previous version of the document by either unpacking the previous xml:tm seeded version of the document or by locating it from storage.
2. SEED the current version with the xml:tm namespace.
3. Open and read in the previous and current seeded documents.
4. Compare the two documents to identify which text units have not changed.
5. Carry over the previous unique identifiers to the new document.
6. Allocate new identifiers to new or modified text units. For text units that have changed, keep a note of the previous identifier to be used for in-document fuzzy matching.
7. Write out the updated file.

DIFFING takes place in the following operations:

i. Check in from authoring
ii. Comparing current and previous versions of the source document in the translation-only environment

3.4.3 EXTRACTION

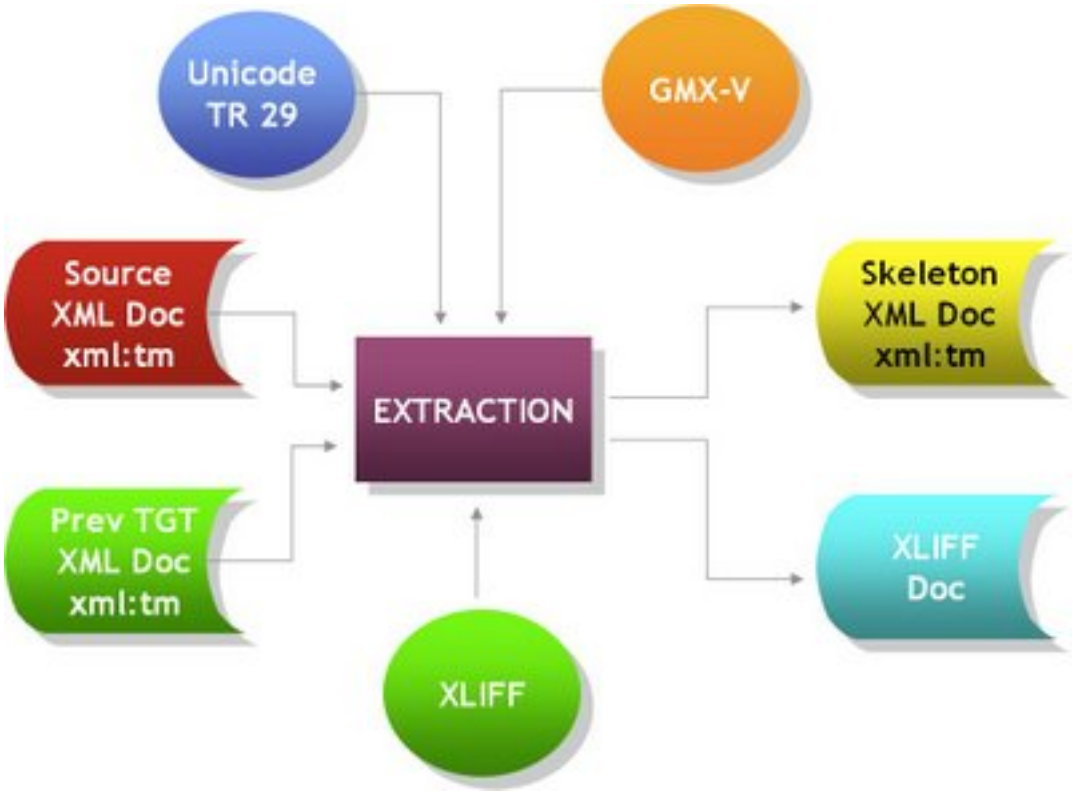

Figure 16: OAXAL EXTRACTION

The extraction process involves the identification of each translatable text unit, transferring it to an XLIFF document, and replacing the text unit with an identifier marking the location for the resultant translated text unit. A skeleton file is thus created with the placeholders for the translated text. The extraction process also involves extracting all translatable text and implementing all document-centered matching (ICE, leveraged, and fuzzy) as well as database matching, resulting in the creation of an XLIFF file along with a commensurate skeleton file.

The processing model for EXTRACTION is as follows:

1. Locate and read in the previous target xml:tm version of the document.
2. Read in the current updated source xml:tm version of the document.
3. Create the basic XLIFF document and the skeleton file based on the updated source xml:tm version of the document.
4. Process each text unit in the updated source document (matching based on identifier) with the previous version, where possible.
5. Attempt in-document leveraged matching based on text unit CRC.
6. Attempt in-document fuzzy matching based on previous identifiers in the case of modified text units.
7. Where no other matching text unit is found, attempt leveraged matching using traditional database-matching methods.

If no database matching is attempted, then the whole extraction process can be implemented as an XSLT transformation.

EXTRACTION takes place for all situations for the translation stage.

3.4.4 ALTERNATIVE EXTRACTION

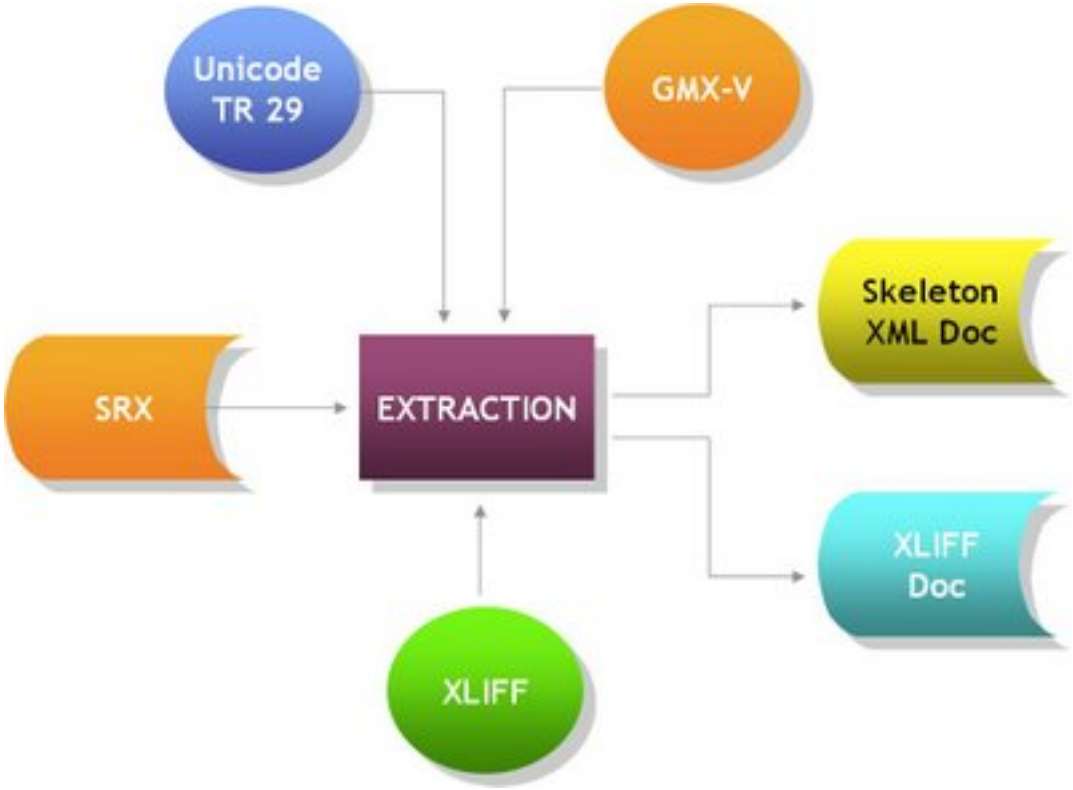

Figure 17: OAXAL ALTERNATIVE EXTRACTION

The alternative extraction process does not use xml:tm versions of the document and does not implement document-centric matching. It involves the identification of translatable text, segmenting the text into sentences, transferring the resultant text units to an XLIFF document, and replacing the text units with an identifier marking the location for the resultant translated text unit. A skeleton file is thus created with placeholders for the translated text. The extraction process also involves extracting all translatable text and implementing all document-centered matching (ICE, leveraged, and fuzzy) as well as database matching, resulting in the creation of an XLIFF file along with a commensurate skeleton file.

The processing model for EXTRACTION is as follows:

1. Read in the current updated source version of the document.
2. Identify translatable text by using W3C ITS document rules.
3. Attempt to segment the text into sentences (text units) if possible.
4. Create the basic XLIFF document and the skeleton file, based on the source version of the document.
5. Attempt leveraged matching on each text unit by using traditional database matching.

If no database matching is attempted, then the whole extraction process can be implemented as an XSLT transformation.

3.4.5 MERGING

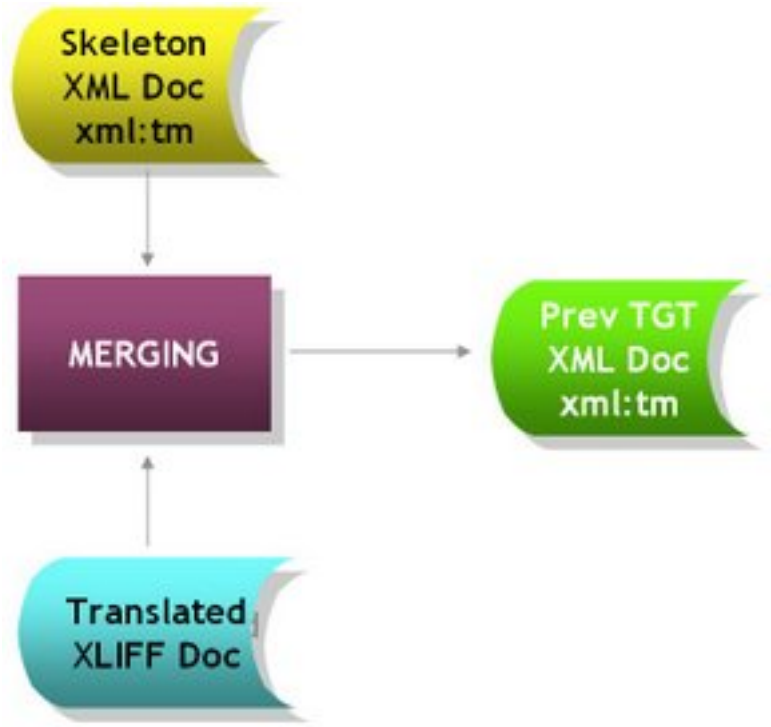

Figure 18: OAXAL MERGING

MERGING involves recreating the target file from the translated XLIFF file and original skeleton file. The translated text is 'merged' with the skeleton file, replacing the placeholders for the translated text.

The processing model for MERGING is as follows:

1. Read in the skeleton file and translated XLIFF file.
2. Transfer the translated text units into the appropriate placeholders in the skeleton file.

The whole MERGING process can be implemented as an XSLT transformation.

MERGING takes place once the translation has been completed.

3.4.6 PACKING

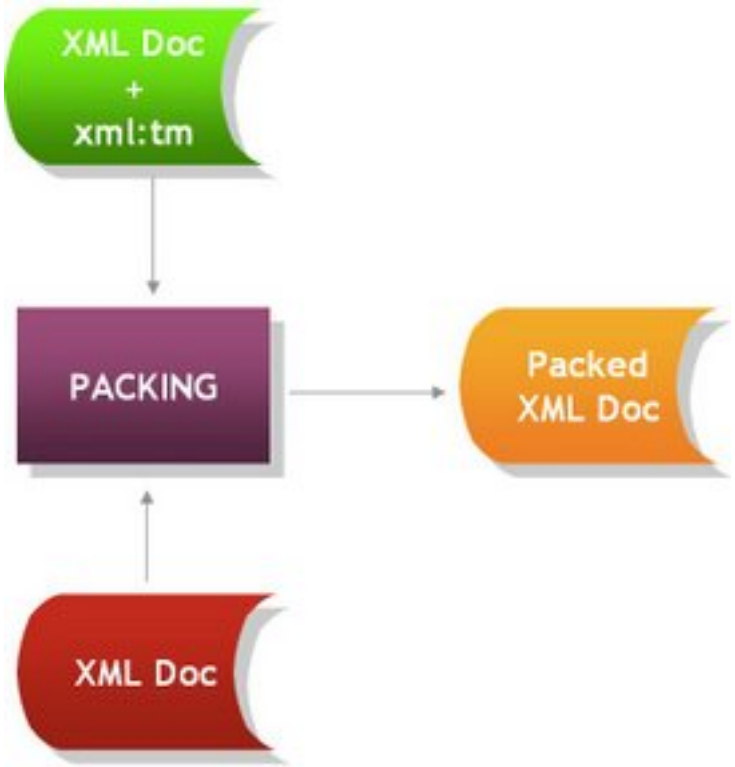

Figure 19: OAXAL PACKING

The processing model for PACKING is as follows:

1. Read in the xml:tm version of the document as a binary input stream.
2. Compress the input stream by using the gzip algorithm.
3. Base-64 encode the compressed file.
4. Insert the compressed base-64 encoded file as a processing instruction into the non-xml:tm namespace version of the document.

An optional stage is used to pack the xml:tm version of the document into the existing document as a zipped, base-64 encoded processing instruction. PACKING takes place in the following operations:

i. Check out for authoring
ii. Check in from authoring
iii. Check in from translation

3.4.7 STRIPPING

Figure 20 OAXAL STRIPPING

STRIPPING is the process of removing the xml:tm namespace from the document. STRIPPING takes place at any stage that requires a non-xml:tm version of the document.

The processing model for STRIPPING is as follows:

1. Read in the xml:tm version of the document.
2. Remove the xml:tm namespace components from the document.
3. Write out the non-xml:tm version of the document.

The whole of the STRIPPING process can be implemented as an XSLT transformation. The result of this process is then suitable for output to a variety of output formats by using XSLT.

4 Conformance Guidelines

The authors of this reference model envision that architects may wish to declare that their work is conformant with this reference model. Conforming to a reference model is not generally an easily automatable task, given that the reference model's role is primarily to define concepts that are important to OAXAL rather than to give guidelines for implementing systems.

We do expect, however, that any given Service Oriented Architecture will reference the concepts outlined in this specification. As such, we expect that any design for a system that adopts the OAXAL approach will:

- Implement the key Open Standards referenced herein
- Follow the key OAXAL concepts and workflow
- Implement an Open and Unencumbered processing model, allowing for the free and easy interchange of data at any point in the processing cycle

It is not appropriate for this specification to identify best practices with respect to building OAXAL-based systems. The ease with which the above elements can be identified within a given OAXAL-based system, however, could have significant impact on the scalability, maintainability, and ease of use of the system.

A. Glossary

[CMS] - Content Managment System

[CRC] AUTODIN II Polynomial Cyclical Redundancy Check singature for a byte sequence. Provides a unique 32-bit signature for a given byte sequence.

[Localization] - Localization is the process of adapting a product or service to a particular language and culture. Translation usually forms a large part of localization, where the target language is different from the source language.

[SCS] - Source Control System

[Text Unit] - A text unit is either the complete text content of a document element or a subdivision of the same into identifiable sentences if possible.

[Workflow] - Workflow is a term used to describe the tasks, procedural steps, organizations, or people involved; required input and output information; and tools needed for each step in a business process.

B. Acknowledgments

The following Technical Committees provided the constituent Open Standards for OAXAL:

OASIS XML Localization Interchange File Format (XLIFF) TC

W3C Internationalization Tag Set (ITS) Working Group

LISA OSCAR TC

Unicode Consortium

The following individuals were members of the committee during the development of this specification and are gratefully acknowledged:

Participants:

Hackos, Dr. JoAnn, *Comtech Services, Inc.*

Domeny, Mr. Doug, *Ektron*

Saldana, Mr. Derek, *Freescale Semiconductor, Inc.*

Schnabel, Mr. Bryan (**secretary**), *Individual Member*

Lommel, Mr. Arle, *Localization Industry Standards Assoc. (LISA)*

Raya, Rodolfo (**secretary**), *Associate*

McRae, Mary, *OASIS*

Jewtushenko, Mr. Tony, *Associate*

Zydroń, Mr. Andrzej (**chair**), *Associate*