Introduction

This chapter is divided into four main sections, starting with the present section which provides a high-level overview of the eXtensible Markup Language (XML). The second section provides a historical perspective of the role played by the various standardization bodies seeking to use XML in the area of translation technology, as well as a full discussion of XML-based formats that can be useful to support translation workflows. The third section presents some of the emerging issues related to XML-based standards in the translation industry while the fourth section provides a summary and lists some future implications.

Overview of XML

XML is composed of two different components. The first one is a metalanguage, with syntactic rules allowing the definition of multiple formats (Savourel 2001). The second component is based on an optional XML schema, which is a language for expressing constraints about XML documents. Examples of XML schemas include Document Type Definitions (DTD), XML XSD schemas or RELAX NG schemas, the latter being popular with document-centric markup languages. Such schemas allow to define a format for a specific purpose using a pre-defined number of keywords (known as the vocabulary, also sometimes referred to as data categories). For example, XSL (the eXtensible Stylesheet Language) is an XML language which is used to transform XML documents into other formats. Nowadays a typical document produced by the Microsoft Word application is also an XML file, albeit in a compressed format, but the author of the document is unaware of the structure that is used ‘behind the scenes’. In the software publishing industry, XML is commonly used to create source documents because once the document has been created, it can be transformed into multiple output formats, such as an HTML page or a PDF file. This process is often referred to as single-source authoring. Examples of popular schemas used for source text authoring include DITA (Darwin Information Typing Architecture), Docbook and oManual. Because the focus on this chapter is on translation technology,
these formats will not be discussed further, but more information on these formats can be found in Closs (2016), Walsh (1999) and Schaffer (2014) respectively. In the translation and localization industries, multiple XML file formats and standards have been defined in the last 25 years or so, such as TMX for Translation Memory Exchange, TBX for TermBase eXchange and XLIFF (XML Localization Interchange File Format) for translatable asset exchange (see Wright in this volume). Due to space constraints, the discussion will focus on formats that have been defined by working groups or technical committees rather than file formats defined by single entities (e.g., RESX by Microsoft to store translatable strings in software applications or Android’s strings.xml format, but more information is available in Ferrier (2007) and Miller (2017) respectively). Standard formats such as TMX and XLIFF will be described in detail once XML itself has been further introduced.

Serialization rationale

As a standard format (XML became a Recommendation from the World Wide Web Consortium as early as in February 1998), XML can be useful in a number of scenarios. For example, a program may need to transform some of its objects state held in memory for storage or transmission (possibly over a network). This process is often referred to as ‘serialization’ and the opposite operation, which extracts a data structure from a series of bytes, is called ‘deserialization’. Using a standard format to achieve this goal may be beneficial, as the XML format will not change from one version of the program to another. This means that internal program changes (e.g., variable name updates) will not affect how information gets stored on disk. Another scenario involves the exchange of program information between two (or more) users. Users may rely on different versions of the program, so having a stable information format should prevent interoperability issues. Finally, a program may need to make some of its information available to another type of program. For instance, a translation management program user may want to perform an additional translation task using a standalone translation quality assurance program. In this situation, it is essential that both programs use a shared information exchange format so that information is not lost or misinterpreted during a roundtrip process (where information goes from program A to program B and then back to program A). Information exchange between two programs, such as Web services, or translation tools from other vendors, may be completely automated, so depending on the scenario, information loss may indeed be undesirable or unacceptable.

Structure of a typical XML document

XML and HTML were designed with different goals: XML was designed to carry data with a focus on what data is, whereas HTML was designed to display data with a focus on how data looks. Another difference between these two languages is that XML elements (or tags) are not predefined the way that HTML tags are. Being a textual data format, XML allows for the creation of content that is both machine readable and human readable. As mentioned in the previous section, two Web services may have to exchange information between each other, so they must be able to process XML documents successfully (hence the machine-readability aspect). Various methods exist to process XML documents in a programmatic manner, but they are beyond the scope of this chapter. Since XML documents are also human readable, they can be opened using simple programs such as text editors, as shown below:
The example above shows that an XML document may start with a prolog, containing version and encoding related information. If this prolog is present, it must come first in the document and possibly specify a character encoding (otherwise the default UTF-8 encoding is used). An XML document contains normal text as well as sequences of characters that have a special meaning. The sequence `<tu>` is known as an element (or tag) whose name is `tu` (as an abbreviation for translation unit). Elements in XML are similar to elements in other markup languages such as HTML. Their opening part starts with a less than (`<`) character, contains a number of characters referring to the name of the element and ends with a greater than (`>`) character. Elements have a number of additional characteristics:

- Some characters have a special meaning. For instance, the less than character must not be used within an element because an XML parser interprets it as the start of a new element. To avoid this issue, it must be replaced with an entity reference (`&lt;` as a shorthand for less than). It is even possible to define custom entities at the start of a document.
- Apart from the prolog, elements must have a starting tag and a closing tag, which includes an additional backslash character after `<`. In the example, the opening `<tu>` tag is matched by the closing `</tu>` tag.
Besides a case-sensitive name, opening elements may contain additional information in the form of attribute/value pairs separated by spaces. An attribute is a sequence of characters such as \texttt{xml:lang}, whose quoted (either single or double) value follows an equal sign: \texttt{en}. In this example, this additional information (known as metadata) suggests that this part of the document pertains to the English (\texttt{en}) language.

Elements may have empty values (e.g., the \texttt{header} element), text-only values (e.g., the first two \texttt{seg} elements of the document), a combination of text and internal elements (e.g., the second two \texttt{seg} elements of the document).

Elements must be properly nested and are arranged following rules that are defined in a schema. This schema, which may be explicitly linked to in the XML document, informs users (and programs) how elements and attributes can be formed, arranged and used. Violating such rules will result in non-valid documents, which are not allowed by the XML standard.

Elements can be represented using a tree structure, whereby a root element contains a number of child elements, which may in turn contain their child elements and so on. In the example above, the root element is \texttt{tmx} and it has two child elements: a \texttt{header} element (containing metadata about the actual content) and a \texttt{body} element, which contains the actual content. More detail about what this content means in relation to the TMX standard will be provided later in this chapter.

Element names are defined by developers, so this may result in a conflict when trying to combine XML documents from different XML applications. Name conflicts can be avoided using a name prefix, as shown in the example with the \texttt{xml:lang} attribute of the \texttt{tuv} elements. In this specific example, the namespace whose name is \texttt{www.w3.org/XML/1998/namespace} is bound by an implicit definition to the prefix \texttt{xml:}. Typically, however, when using custom prefixes in an XML document, a \texttt{namespace} for the prefix must be explicitly defined using the \texttt{xmlns} attribute whose value is a Uniform Resource Identifier (URI). A namespace refers to a specific set of elements defined by a specific entity (e.g., an automotive company may have defined a \texttt{make} element).

When XML schemas such as Relax-NG, Schematron or W3C XSD are used instead of DTDs, elements and attributes can be characterized using specific data types (e.g., strings, integers) and be assigned default and fixed values. This type of granularity can be extremely powerful to avoid ambiguity (e.g., what are the possible values accepted by a specific attribute?).

This section provided a very brief overview of the structure of XML documents using a simplified and contrived example. While XML documents can be more complex depending on the rules defined in the schemas, the principles listed above should suffice to understand the other formats presented in this chapter. A more detailed overview of XML in the context of localization and internationalization is provided by Savourel (2001) and the \textit{Best Practices for XML Internationalization} Group Note (W3C 2008). This group note provides some guidelines for creating XML documents and schemas that are properly internationalized. According to its authors, following such guidelines allows both the developer of XML applications, as well as the author of XML content to create material in different languages.

**Historical trajectory and literature**

In this section, the literature is reviewed as part of a general, critical discussion of the evolution of various XML-based standards in use in the translation industry. The previous
section provided an example of a document following the TMX standard. However, not all XML specifications automatically become standards. So before proceeding further, the standardization process must be briefly explained.

**Standardization bodies (also see Wright in this volume)**

According to Filip (2012), ‘real working standards, open or not, must have been driven by a representative industry consensus’. But one may wonder where these standards originate. DePalma (2012) explains that the standards stack spans individual conventions, collaborative workgroup group conventions and corporate conventions. These conventions complement industry conventions, government regulations and technology standards. In the last 25 years, multiple standardization bodies have indeed been involved in the creation of standards related to or based on XML. One such body is the World Wide Web Consortium (W3C), which is a global community involving member organizations, full-time staff and the public, all working together to develop Web standards. The W3C was founded in 1994 by Tim Berners-Lee and standardization work on XML started in 1996 one year after the work on HTML started. Multiple recommendations and standards have been published by the W3C in relation to XML so it is neither practical nor relevant to list them all in this section.

Another body which played a very important role in standardizing formats related to translation technology is the Localization Industry Standards Association (LISA). This industry body was formed in 1990 and involved most of the large information technology companies of the period. One of LISA’s working groups, the OSCAR (Open Standards for Container/Content Allowing Reuse) group, focused specifically on formats pertaining to translation technology and term bases. LISA also collaborated closely with other standardization bodies such as the International Organization for Standardization (ISO), which resulted in having the TermBase eXchange (TBX) format defined as ISO 30042:2008 in 2008. This terminology-related standard will be covered in its own section. However, LISA ceased its activities in 2011 and designated the European Telecommunications Standards Institute (ETSI) Localization Industry Standards (LIS) Industry Specification Group (ISG) as its successor organization for its standards portfolio (Guillemin and Trillaud 2012). However, this relationship was short-lived as explained in further detail in Wright in this volume.

Finally, OASIS (the Organization for the Advancement of Structured Information Standards) has a number of technical working groups focused on translation and localization activities. One of these groups is the XLIFF committee. XLIFF is an OASIS standard that enables translatable source text and its bitext to be exchanged between various tools within localization and translation workflows. The OASIS XLIFF technical committee prepared version 2.0 of the XLIFF specification (XLIFF 2.0 2014), which was then adopted under a special fast-track procedure by Technical Committee ISO/TC 37 under ISO 21720:2017. XLIFF will be described in more detail in a later section.

With so many formats and standards available, some meta-standard sometimes become available. For instance, during the 2000s, OAXAL (the Open Architecture for XML Authoring and Localization) was also formed as an OASIS initiative to provide an overarching framework for all localization-related standards (e.g., TMX, TBX, XLIFF). One of the deliverables produced by this initiative was a reference model with suggestions on how to best make use of such standards (OAXAL 1.0 2009). This model was used effectively in the Okapi Framework, which provides an open-source set of applications for localizing and translating documentation and software.
The TMX standard

Translation Memories (TMs) are the most widely used translation technology among professional translators. In order to effectively exchange data between two different translation memory systems, a standard format must be used. TMX was initially developed in the mid-1990s by OSCAR. When LISA ceased to exist, however, its standards projects were all handed over to the European Telecommunications Standards Institute (ETSI), which until August, 2015, maintained a working group called the Localization Industry Standards (LIS) Industry Specification Group (ISG). While the standard can still be downloaded from ETSI, the latest version of the TMX specification is now visibly available from the GALA (Globalization and Localization Association) Website. The TMX format can be used to export the content of a translation memory database into another application. This scenario is likely to occur when (i) there is no requirement to link the translation units back to their original documents and (ii) multiple stakeholders are involved. Some of these stakeholders may have different preferences with regard to translation memory applications that should be used during the translation process. In order to reuse previous translation units stored in a different application, however, one needs to be able to seamlessly export and import them. TMX provides a standardized container which is understood by most modern translation memory applications, or larger applications (such as translation environments) that have a translation memory functionality.

The TMX specification

TMX is XML-compliant and uses various ISO standards for date, time, language codes and country codes. TMX elements are divided into two main categories: the structural elements (e.g., the <tu> element that consists of aligned segments of text in two or more languages) and the inline elements (e.g., the contents of a <seg> element containing an individual segment of translation-memory text in a particular language). TMX offers two levels of implementation. Level 1 offers support for the container only so the data inside each <seg> element is plain text, without content markup. This level is enough when the data do not have inline codes (e.g., software strings that do not rely on HTML), but it is often insufficient for documentation-type formats. Level 2 offers support for both container and content. It is worth pointing out, however, that TMX files are not limited to a single document since a TMX resource may reflect a gradual compilation of translation units from hundreds or thousands of documents.

A TMX document is enclosed in a tmx root element. The header element of a TMX document contains metadata about the document. For instance, its segtype attribute can specify the kind of segmentation used in all of the tu elements if these do not individually contain such information. The segmentation type information is important, as a translation memory tool may need to perform additional segmentation to effectively make use of any translation unit contained in a TMX file it is importing. Specifically, if translation units were created using a paragraph-level segmentation, they are unlikely to be of any use for a user expecting sentence-level segmentation. The TMX standard specifies which segmentation types are allowed: paragraph, block, sentence and phrase. However, the rules used to implement such segmentation have to be defined by each tool, possibly using the SRX format (as discussed later). The TMX specification recommends using sentence-level segmentation to maximize portability, but specific use-cases (e.g., subtitling, software string translation) may be best suited by other segmentation types (e.g., phrase, paragraph).
In addition to its attributes, the *header* element can also store document-level information in *note* and *prop* elements. The *prop* element is used to define the various properties of the parent element (or of the document when *prop* is used in the *header* element). These properties are not defined by the standard. In a TMX document, the *<body>* element contains the set of translation units (the *<tu>* elements) in no specific order. These translation units are usually the representation of the contents of a large translation memory that may have been populated with segments originating from many different documents, all stored together in no particular order. Each *<tu>* element contains at least one translation unit variant, the *<tuv>* element. Each *<tuv>* contains the segment and the information pertaining to that segment for a given language. The text itself is stored in the *<seg>* element, while the *<note>* and *<prop>* elements allow storage of information specific to each *<tuv>*.

A segment can contain markup content elements: The *<bpt>*, *<ept>*, *<it>* and *<ph>* elements allow the inclusion of original native inline codes. The *<hi>* element allows for extra markup that is not related to existing inline codes. Finally, the *<sub>* element, used inside encapsulated inline code, can delimit embedded text.

**TMX and translation technology tools**

Since TMX was defined as a standard to exchange data between translation tools (e.g., translation memory systems) it is obviously widely implemented in applications used by translators wishing to leverage previous translations (e.g., from previous projects or from colleagues). A review of multiple tools (including open-source tools and proprietary tools) conducted by Gough (2010), found that all tools supported TMX but lacked collaborative features to share translation memories in real time. TMX may also be used to import translation data into machine translation (MT) processing pipelines, but MT systems’ support for TMX is yet under development. Dealing with internal markup can indeed be a challenge for MT systems, so various methods have been proposed to address this issue in a statistical machine translation (SMT) framework (Du, Roturier and Way 2010).

One of the other potential limitations of TMX is that it does not allow by default for the leveraging of smaller segment parts which might be desirable in specific scenarios (e.g., advanced leveraging or hybrid solutions making use of both translation memory and machine translation technologies). To address this limitation, Forcada (2014) proposes a solution to ‘enrich TMX-encoded translation memories with information about subsegment equivalence, which may be obtained from external sources such as machine translation systems, term bases, glossaries, etc., or by using a statistical word aligner followed by phrase extraction’. (also see Simard in this volume) Going forward, future standardization work may be required to address these requirements in a more systematic manner. Such work might be carried out within the framework of the XLIFF standard.

**The XLIFF standard**

XLIFF is popular in the localization industry as it allows for the exchange of bitext between various stakeholders (such as content developers and translation providers). For example, a software publisher or language service provider may look after the extraction of translatable content from source files (including code and documentation). However, the actual translation may be done by a translator, so content must flow from one stakeholder
to another as smoothly as possible (without information loss). XLIFF may be used in this context to allow the transport of the information from one system to another. In the past, systems that made use of the XLIFF standard sometimes needed to extend it to add system-specific information. To achieve this, the namespace mechanism was used, to support vocabularies from several schemas in a single XML document. Having this flexibility, however, could sometimes add complexity because systems had to be able to recognize these extensions (and possibly needed to rely on custom filters to make full use of all the information present in the XLIFF document).

The XLIFF specification

To address this problem, the XLIFF standard evolved to provide its own set of additional modules, based on common use cases (e.g., providing translation candidates for translators or translation validation rules to reviewers). The XLIFF 2.0 specification therefore splits the format into a base namespace called the Core that all implementations must support and several specialized optional modules. This separation ensures the stability of the format while offering support for future enhancements. At the time of writing, the latest approved version of the XLIFF schemas was 2.1. In a typical XLIFF document, a unit element contains the segment elements and each segment holds the source and target elements. There is also a canResegment attribute that indicates if tools can change the existing segmentation in a given part of the document. The segment element also holds two attributes to indicate its status: state and subState. The state attribute can have the values initial (the default), translated, reviewed or final. If a process needs additional values, each state can be refined using a custom value specified in subState. The value of subState must start with a prefix that identifies the authority defining the values, in order to avoid conflicts between tools. The state attribute allows a minimal level of interoperability, while subState provides some flexibility for customization.

Besides the Core, XLIFF 2.0 provides eight specialized modules:

- Translation Candidates, which allow the source text of a document to be pre-processed against various translation resources such as translation memories or machine translation systems to provide translation candidates.
- Glossary, which allows an XLIFF document to embed a list of terms with a definition or translation (now compatible with TBX 2.0).
- Format Style, which allows an XLIFF document to contain information to generate a quick HTML preview of XLIFF content.
- Metadata, which provides a mechanism for storing custom metadata using elements that are part of the official XLIFF specification.
- Resource Data, which offers a mechanism to reference external resource data that may need to be modified or used as contextual reference during translation (e.g., a screenshot).
- Change Tracking, to store revision information for XLIFF elements and attributes.
- Size and Length Restriction, which has support for storage size restrictions and general size restriction.
- Validation, which may define a set of validation rules that can be applied to target text both globally and locally, possibly depending on conditions in the source text.

Each module has its own namespace and can evolve independently from the Core and the other modules. This modular aspect of 2.0 is important because it allows tools to
continue working with future versions of XLIFF without any change, as long as the Core and the modules they use do not change. However, this modularity has a cost since default features from version 1.2 are not supported in the Core. For instance, translation candidates are now supported by a specific module whereas they could be included in the alt-trans element in version 1.2. Such translation candidates may be obtained by pre-processing the source segments against various systems (e.g., TM, MT).

**XLIFF and translation technology tools**

Many commercial or open-source translation technology tools now support all (or some parts) of the XLIFF standard. Such wide adoption was made possible by earlier work performed in specific areas such as Statistical MT (SMT) and localization (e.g., the M4LOC solution designed by Hudík and Ruopp 2011) or post-editing (e.g., for the collection and analysis of post-editing data in the ACCEPT framework (Roturier, Mitchell and Silva 2013).

This section focused on the XLIFF format, which is an extremely powerful and customizable format used to exchange information during various types of localization projects. In early 2018, the latest specification (XLIFF 2.1) was approved as standard by an active technical committee from OASIS. Previously, this OASIS committee had also worked in very close collaboration with ISO Technical Committee 37 (Language and Terminology) Sub-Committee 5 (Translation and Interpreting) to reach a wider audience by issuing the XLIFF standard as ISO 21720:2017. The next section focuses on additional XML formats, some of which being intrinsically linked to XLIFF (e.g., ITS and TBX).

**Other XML standards**

Multiple translation-related XML formats and standards have emerged over the years and it is not possible to review all of them in detail, such as the XML-based Text Memory format (xml:tm) (refer to Wright in this volume). This section focuses on those whose popularity gives them a standard status: GMX-V to count various items in a translation project such as words or characters, ITS to prepare source documents for translation, SRX to segment a source text into fragments and TBX to store, retrieve and exchange terminology resources.

**GMX-V**

The Global Information Management Metrics – Volume (GMX-V) specification is a European Telecommunications Standards Institute Localization Industry Standards (ETSI LIS) specification, even though it was originally developed within LISA OSCAR. Although ETSI LIS is no longer active, GMX-V can still be downloaded from the ETSI website. GMX-V version 2.0 was published in 2012 and tackles the issue of quantifying the workload for a given localization or translation task. Very often, word counts are used as a proxy to quantify this kind of workload, but defining ‘what a word is’ is not always straightforward. For instance, some mappings are sometimes required (e.g., for converting Chinese, Japanese, Korean and Thai logographic character counts to word counts). For instance, GMX-V suggests that 3 Japanese characters are required to make a word, whereas 6 Thai characters are used to make a word (Zydroń 2014b). The goal of GMX-V is therefore to provide an unambiguous way of counting as well as classifying word and character counts (e.g., translatable words, non-translatable words) for all languages and scripts in order to enable an accurate definition of the scale of a translation task. It also
Johann Roturier

offers an XML vocabulary for exchanging data on localization metrics. This specification addresses two important issues: how words and characters are counted unambiguously and verifiably for a given localization task, and how these counts are exchanged rigorously between systems.

One of the core problems when calculating word and character counts is the large number of proprietary file formats containing both form and content data (e.g., a Word document may contain some text that is automatically generated, such as a Table of Contents, so some words could be counted twice). To work around this issue, GMX-V relies on the XLIFF representation of the original file format and expects all characters to be counted in their Unicode representation. Word boundaries are defined with reference to Unicode Technical report 29 on text segmentation (Davis 2018). From an implementation point of view, GMX-V is actually designed to co-exist as a namespace within other XML documents, such as XLIFF documents. It is worth highlighting that word counts do not cover the full range of possible metrics that may be used to assess the cost of localizing a document (e.g., number of screenshots for a software localization project, or page counts for a document layout task). Metrics do indeed fall into two categories: directly verifiable from a file's content (words and characters) and unverifiable (pages, lines, etc.). GMX-V does not forbid the presence of unverifiable counts, but is mostly concerned with defining precise rules for verifiable counts based on the file unit that is being counted. This flexibility seems to be appreciated by tool publishers, since according to Zydroń (2014b), GMX-V is used in a number of proprietary and open-source translation environments.

**ITS**

The Internationalization Tag Set (ITS) is a standardization effort led by the W3C. This format provides (among other things) translatable rules so that both human translators or MT systems can recognize whether some part of a document must be translated (ITS 2.0 2013). According to the creators of ITS, such rules may be added by content authors or information architects. Parts of a document that must not be translated are assigned a translate ITS data category with a value of no, so that tools or people who manipulate this content are made aware that this particular phrase must not be translated. The ITS data categories go beyond specifying what is translatable or what is not translatable. For example, the latest set of ITS data categories include identifiers dealing with terminology, notes (e.g., for translators), language information, or the type of characters allowed in certain elements.

Examples of such data categories are the *Localization Quality Rating* and the *MT Confidence* data categories. The first is used to express an overall measurement of the localization quality of a document or an item in a document. This feature allows for the annotation of XML and HTML content using specific quality-related markup. In this use case, the *mrk* element delimits the content to annotate. This element holds a *locQualityIssuesRef* attribute that refers to the *locQualityIssues* element, where quality issues are listed. The second data category indicates the confidence score from an MT system for the accuracy of one of its translations (between 0 and 1). These features should significantly help standardize the exchange of quality annotations between the various stakeholders involved in the translation quality processes of localization projects. For instance, the localization quality rating may be provided by a translation reviewer who is tasked to judge the quality of a translated document or segment based on pre-defined criteria. Over the years the ITS technical group has worked very closely with the XLIFF technical committee to ensure some compatibility between the two
standards. This collaboration has led to the inclusion of an ITS module within the XLIFF standard in version 2.1.

**SRX**

Text segmentation rules can be manually defined using the SRX Segmentation Rule eXchange Standard, which is an XML-based standard that allows for the exchange of rules from one system to another. This standard originated in LISA/OSCAR and SRX version 2.0 was officially accepted as an OSCAR standard in April 2008 (SRX 2.0). SRX is defined in two parts: a specification of the segmentation rules that are applicable for each language, represented by the `languagerules` element and a specification of how the segmentation rules are applied to each language, represented by the `maprules` element. Using this standard, two types of rules can be defined: rules that identify characters that indicate a segmentation break and rules that indicate exceptions. For example, one breaking rule can be defined to identify a full stop followed by any number of spaces and a non-breaking rule can be defined to list a number of abbreviated words. SRX rules can be created using regular expressions conforming to the ICU (International Components for Unicode) syntax so that specific patterns can be searched for in a text. Rules will obviously vary from one natural language to another. While a common set of rules may be reused for certain languages, language-specific exception rules will be required.

**TBX**

Most translation environments and tools suites feature some sort of terminology management system designed to support translators with appropriate terminology choices. If translators never leave a proprietary format, they probably never think of an interchange format. If they want to share terminology with other applications or store it in a non-proprietary form, however, they can benefit from such a format. Being able to share and store terminological data vastly increases quality and productivity, because a great deal of time would otherwise be spent mining terminology files or entries to extract relevant information. To standardize terminology transfer, various formats have been proposed over the years. One of these standards is the XML-based Term Base eXchange (TBX) standard, which had been defined by LISA (TBX 2008). After temporarily residing with ETSI, TBX was picked up by LTAC/TerminOrgs for further development. The TBX standard (2008) is available online and has also been approved as standard by ISO (ISO 30042:2008).

Versions of the TBX format are featured by numerous applications and used by various entities to export terminological data. For example, this is the format used by Microsoft to export the following information for each of their terms: a terminology concept ID, a definition, a source term, a source language identifier, a target term and a target language identifier (Microsoft 2018). TBX (2008), however, has numerous drawbacks: implementations are inconsistent and many tools do not comply with the standard, but even so, true interchange only works if termbases use the same data model and the same or mappable data categories and permissible values. The old LISA-based TBX-Basic dialect of TBX forged the way to the future by providing a universal framework for sharing terminological data within localization environments, although some users want less information, and others want more, which has prevented TBX-Basic from becoming a one-size-fits-all solution for terminology interchange (TerminOrgs 2014). The new 2018 version of the standard addresses these and other drawbacks by introducing: interoperability with the XLIFF glossary module; an option to shift from
the sometimes unpopular TEI attribute-based serialization (DCA style) to a simpler
tag-based style (DCT style); introduction of multiple TBX dialects, accompanied by
the adoption of models for creating Relax NG schemas to define specific TBX dialects
(Wright 2018); and introduction of xmlns notation identifying dialects and providing
access to associated data category selections. It is hoped that these upgrades will enable
the standard to fill the role for which it was intended as a vehicle for representing ter-
minological data in cross-platform environments.

Emerging issues

Up until this section, the discussion may have presented XML as the de-facto exchange
format to use when passing information between two systems. However, many other types
of file formats exist (e.g., semi-structured text such as CSV, JSON, or even binary, etc.). One
example of semi-structured text format is the tab-delimited UTX format that originated
from the Asia-Pacific Association for Machine Translation (Okura et al. 2011). The goal of
this format is for users to create, reuse and share glossaries to improve translation quality.
As far as human translators are concerned, UTX is a compact, easy-to-build glossary that
reduces the time and cost required to check terminology. As far as terminology-based
machine translation and terminology tools are concerned, UTX can be used as glossary
data that does not require any modification. So, what are the main benefits and drawbacks
of XML compared to other file formats such as character-delimited text?

One benefit is that it allows for the presence of additional metadata. For instance,
structured concept-oriented terminology formats that can be represented in a format such
as TBX cannot be easily transformed into UTX. However, it is worth pointing out that
the UTX group does coordinate directly with TBX-Min and that conversion tools are
available. Another benefit is consistency. According to Zydroń (2014a), since ‘over 90% of
data for translation is already being generated in XML, it makes sense to base the internal
data structure on XML. Having one consistent electronic form makes for a very clean,
efficient and elegant design’. Another main benefit of XML is the structure it enforces
via the use of schemas. Because rules are used, syntactic ambiguity is not permitted (i.e.
unlike HTML, XML documents cannot be ill-formed) and semantic ambiguity is reduced
to a minimum (e.g., the use of a specific element with a specific value should be clear to
all actors involved in the processing of the document). When ambiguity might still be pre-
sent, comments can also be used to clarify how the metadata should be interpreted. Being
able to rely on such an unambiguous structure is a big advantage over semi-structured
formats such as character-delimited text formats that tend to rely on conventions (e.g., the
first row of the file contains field names).

However, the structure that makes XML so powerful is also one of its drawbacks. Since
XML documents can be complex, they tend to be quite large (thus utilizing more resources
such as network bandwidth and disk space) and their processing can be costly from a com-
puting perspective. From an end-user’s perspective this can also result in perceived delays.
Another drawback of XML is that it often does not match the data model of most pro-
gramming languages, especially the data structures of object-oriented languages such as
JavaScript. For this reason, the JavaScript Object Notation (JSON) format has emerged
as an alternative format that is being widely used by Web applications (Bray 2017). An
example of such an application is Babelnet, a multilingual encyclopedic dictionary with
lexicographic and encyclopedic coverage of terms (Navigli and Ponzetto 2012). The emer-
gence of JSON as a viable serialization format has been facilitated with the availability
of complementary technologies such as JSON schema allowing for the annotation and validation of JSON documents. Finally, it is worth pointing out that XML (as a technology rather than a format) can be affected by security issues, especially in terms of confidentiality and availability. Since XML documents may reference external or custom entities (e.g., a DTD or a character entity), an XML application can be subject to XML Entity Expansion (XEE) vulnerabilities, which can be exploited to retrieve files on a remote system (Arnaboldi, 2018). Also, since XML documents must be well-formed, the amount of resources required to process ill-formed documents (that may have been submitted by an attacker) could result in successful denial-of-service attacks, whereby some application or service could become unresponsive to legitimate users. This is not to say that alternative formats (e.g., JSON) are exempt from security issues (Petty 2017). However, application developers who rely on XML formats in the translation industry should be aware of these potential pitfalls and use relevant libraries when dealing with potentially untrusted user input. For instance, the defusedxml library in the Python Programming language provides several workarounds and fixes for various vulnerabilities in Python’s XML libraries (Heimes 2017).

Conclusions

Various translation-related XML formats and standards have been presented throughout this chapter. Some of them are being updated frequently (e.g., XLIFF through OASIS and ISO, or TBX through TerminOrgs and ISO) while others have not been actively refreshed in a number of years (e.g., SRX). Since standardization work is based on consensus, changes can sometimes take what seems like a very long time for users of these standards. Perceived delays are one of the reasons for which new standardization initiatives are started. Two initiatives, which have not been mentioned so far but are worth monitoring going forward as they seem to be gaining some momentum, fall in this category: The Language Interoperability Portfolio Project (Linport) and the Translation API Class and Cases Initiative (TAPICC). Linport, is a collaborative project developing an open, vendor-independent format that can be used by many different translation tools to package translation materials such as XLIFF files (Melby et al. 2012). Linport specifies two kinds of containers: a package container for describing one bilingual task and a portfolio container for describing an entire translation project. The former is based on XLIFF:doc (a reference guide for representing documents in XLIFF files) and the latter is known as the Translation Interoperability Protocol Package (TIPP). This project is worth monitoring as Stewart (2018) explains that ‘TIPP is now being updated by the newly created Linport working group. The group has also joined the new initiative TAPICC, which will consider adopting TIPP 2.0 as payload container and object model for a payload application programming interface’. TAPICC is another collaborative, community-driven, open-source project to advance API standards for multilingual content delivery.

Another trend is that for a long time, XML formats were needed and used for offline work (e.g., exporting a glossary in TBX format or using the XLIFF Glossary module). However, with more and more translation-related work being conducted online (e.g., in cloud-based applications), XML formats may not necessarily be as ubiquitous as they once were. For instance, a new OASIS technical committee (the XLIFF Object Model and Other Serializations, OMOS) was recently set up to explore the possibility of using non-XML-based formats to serialize the underlying XLIFF object model. If this initiative succeeds, it is possible that the JLIFF format (based on JSON) might become more
widespread in the future, especially if conversion between XLIFF and JLIFF is lossless (Tingley 2017).

Finally, one can easily imagine that, in a not-too-distant future, formats such as the Predictive Model Markup Language (Guazzelli et al. 2009) might play a more important role in the translation industry as translation tools have to interface with new modalities and technologies. For instance, PMML is the de facto standard language used to represent predictive analytic models and data mining models. Going forward some translation tools may have to rely on the exchange of such models, especially if these models are explicitly or implicitly created by translation tools users (e.g., a post-editing model). For example, one could imagine an interactive MT application relying on suggestions from other users clustered using a model built using a different application. Regardless of whether these predictions prove to be accurate, one thing is almost certain: open standards in the translation industry are here to stay as their benefits have been clearly communicated to a wide range of users, who have in turn adopted them.

**Further reading**


**Related topics**

2. Standards for the language, translation and localization industry
4. Terminology extraction and management
5. Building and using parallel text for translation
12. Technology, technical translation and localization
20: Translation technology evaluation research

**References**


