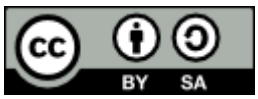
# Open Translation Tools: Disruptive Potential to Broaden Access to Knowledge

**Prepared for Open Society Institute by**
**Allen Gunn**
**Aspiration**
**September 2008**





## Table of Contents

## Executive Summary

The first-ever Open Translation Tools Convergence (OTT07) took place from 29 November to 1 December, 2007 in Zagreb, Croatia. The event brought together two passionate communities: those creating open source software tools to support translating open content, and those with a need for better tools to support translation of the open content they create. “Open content” was interpreted to encompass a range of resource types available under open licenses such as Creative Commons (CC) and Free Document License (FDL), ranging from books to manuals to documents to blog posts to multimedia. “Open translation tool” was interpreted to encompass any piece of software which supports or performs language translation, and which is distributed under a free or open source software (FOSS) license. This paper describes the learnings and outcomes from the event.

Open Translation Tools was co-organized by Aspiration and Multimedia Institute (MI2), and was supported by the generosity of the Open Society Institute, with additional support for participant travel provided by TechSoup. The OTT07 agenda was collaboratively developed by participants and event organizers in the time leading up to and during the gathering, and the proceedings were directed using Aspiration's collaborative approach to event facilitation.

The event focused on a relatively specific category of translation tools: those which support or enable the human translation of text content. Participants engaged in two parallel paths of learning and mapping: one team documented available open translation tools and technologies, categorizing and differentiating the available offerings, while another team oversaw the enumeration of a set of “use cases” which describe how publishers of open content want to be able to translate and manage that content.

While not intended to be comprehensive, the use cases documented at OTT07 span a broad range of scenarios and needs, and serve as an excellent sampler of the types of translation requirements which exist in the open content community and beyond. The use cases were grouped into 7 categories: content translation, multimedia translation, translation workflow, machine translation, Interpreting, content rendering, and localization. In parallel with enumerating use cases, participants assembled a collection of considerations to be addressed when considering a translation strategy; those are included later in this paper. The complete set of open translation use cases can be found in the Appendix I.

The field of translation is in a state of transition, and software tools to support language translation are evolving with corresponding rapidity. Increasingly available internet resources are quickly expanding the possible and the practical when it comes to translating content, and processes and business models which have remained relatively staid for decades are being rethought. The state of open translation tool offerings reflects the same flux. Real-time access to a global network of translation services and talent is a resource only now beginning to be leveraged by the translation industry, and upstart multilingual projects on the internet are pushing the state of the art by treating translation as an exercise in distributed problem solving. In addition, most open translation tools are only now beginning to incorporate substantial workflow into the software, tracking user roles, permissions and detailed state information for each translation project. From the RSS-enabled platforms like Worldwide

Lexicon, which automates translation requests and submissions, to crowd-sourced tools like dotSUB, which though not open source still employs an open approach to data and translation for subtitling digital videos, open translation tools are demonstrating their ability to not only track but also outpace closed and proprietary counterparts.

The goal at OTT07 was to take a snapshot of the open translation tools that were available, and in turn to analyze the gaps. Participants engaged in a range of brainstorming and knowledge capture activities; collaborative mapping of available tools served as a prelude to a discussion of what a “dream translation tool” might provide. Tools were grouped into 7 categories: PO and XLIFF localization editors, translation workflow, subtitling, machine translation, translation memory, dictionary and glossary, and wiki translation. As with almost any collection of software tools, these categories blur and overlap on a tool-by-tool basis; the categories are somewhat arbitrary and many tools fall into more than one. A complete listing of open translation tools mapped out at the event is provided as Appendix II of this paper.

A primary point of discussion at OTT07 was “what's missing” in open translation tools. While a range of issues were identified, two primary functionality gaps surfaced in a range of conversations: workflow support for managing and tracking the broad range of translation tasks and processes, and distributed translation with memory aggregation, offering remote translators the ability to contribute translations to sites of their choosing and to have those translation mappings stored for use in future translations. Another often-identified gap was the lack of integration and interoperability between tools; different communities have their own toolsets, but it is difficult for a translation project to make coherent use of a complete toolset. Among the interoperability issues which require further attention in the open translation tools ecology are common API's that enable tools to share data and requests, plugins for web content management systems (CMS's) to export content into PO files, better tool integration features including shared glossaries, common user interfaces and subsystems, rich file import/export, and finally generic code libraries for common feature requirements.

As part of the tool mapping exercises at OTT07, participants were asked to envision what their “dream translation tool” might look like. The idea was to specify a feature set for a tool which does not yet exist, but which would meet the broadest range of translation needs in terms of features, supported workflows, and business models. This was a purely theoretical exercise; participants generally agreed that large monolithic tools were not the right course for the future, and that a small, distributed set of tools that work well together was the recommended path for better supporting open translation efforts. The resulting work of software fiction and vision is described starting on page 22.

Several significant emerging tools and technology trends were highlighted during the proceedings at OTT07. One of the most compelling innovations shared at OTT07 was the unique use of RSS (Rich Site Syndication) being made by projects like Worldwide Lexicon to syndicate and distribute translation requests. While the focus of OTT07 was on human translation and the software tools that support those processes, machine translation (MT) was often discussed for its substantial role in the human translation process; participants agreed that MT would not be able to generate publication-quality translations in the foreseeable future, but that with the increasing availability of translation corpora, MT would have an

increasingly important role in supporting the work of human translators and their translation tools. One of the most valuable but arguably under-developed aspects of open translation involves the concept of “translation memory” (TM), the process of storing translated sentence pairs, one in the source language and one in a target language; such memories are rarely shared or collectively maintained, and are instead constantly reinvented. And ironically, FOSS projects do not always take advantage of standards for storing translations; by adopting standard file formats and other data standards, such projects could contribute to shared translation memories and create major forward momentum for better translation and localization.

A number of related discussions took place during the three days of OTT07. One focus was on the lack of visibility for existing tools; while there are compelling open translation tools and technologies available, very few users are aware of their existence. Licensing for open content translation was also a central issue; translation of open content raises several issues regarding how the source content is licensed, especially with regard to the creation of derivative works. Also, the significance of regional and cultural issues in translation work can not be overstated; as norms and values vary, a range of secondary connotations and associations must be considered in crafting appropriate translations. These topics surfaced in a number of sessions on the agenda.

OTT07 generated a range of collaboration ideas and emerging plans. The most exciting project idea presented was to collaboratively author an Open Translation Book that would provide an overview of the emerging field and associated issues, while also explaining how to use open source translation tools to implement various open content translation use cases. On online community called Translator Commons was proposed as a destination where practitioners of open translation could go to find support and guidance for their practice, as well as resources such as regional and language-pair-specific translation knowledge and style guides. “Corpora Commons” was proposed as an initiative to support open source machine translation by aggregating translation memories from documents translated by the United Nations and individual governments, who have copious volumes of translated text in a range of language pairs. And participants at OTT07 called out API (Application Programmer Interface) design and adoption as a critical objective in growing the open translation movement. While many tools use standard data formats such as PO, XLIFF, and TMX, very few expose API's which would allow other tools to easily transfer data or invoke services remotely.

The event also engendered collaborations between participating projects. Meedan and Worldwide Lexicon (WWL) got better acquainted at OTT07, and are now partnered and seeking funding to build social-network-oriented translation tools which will tap the distributed translation talent spread across the internet. The Inkscape project and FLOSS Manuals first met at OTT07, and hosted a Book Sprint sponsored by Google in July 2008 in Paris.

## Acknowledgments

Thanks to all the participants and facilitators at OTT07, whose shared wisdom and knowledge are aggregated in this paper. In particular, thanks to those who took notes during sessions for the wiki, as that material forms the basis for a substantial percentage of this document.

In addition, Aspiration would like to thank the following OTT07 participants and organizations:

**Multimedia Institute of Zagreb**, who co-organized the OTT07 event, managed all local logistics, and served as passionate participants and collaborative partners without equal. OTT07 simply would not have been possible without their leadership and support, and the high quality of participant experiences was a direct result of their exhaustive attention to detail and hospitality.

**Michelle Murrain of NOSI** (the Nonprofit Open Source Initiative), and **David Taylor of Radical Designs**, who led the mappings of open translation use cases and tools respectively. Their facilitation and process design led to invaluable knowledge capture.

**Lena Zuňiga of Sula Batsu**, who oversaw the OTT07 event wiki, and who directed the participant interviews for the OTT07 video.

**Greg Bernstein, videographer**, who created the “Voices from OTT07” video that documented participant reflections from the event.

**Ed Zads of dotSUB**, who shared copiously of his experience and knowledge from the translation industry. Ed's teachings allowed other participants to both understand “old school” approaches to translation and workflow, and to consider those against that which the ever-burgeoning internet has made possible and in fact probable.

**David Sasaki of Rising Voices**, whose blog posts about the event were an excellent secondary source for this paper.

**Cultural Beverage Contributors**, whose collective efforts to amass liquid spirits from their respective countries and regions imbued evening activities with grand beverage diversity.

**Open Society Institute** (www.soros.org), who provided the funding to make OTT07 possible.

**TechSoup** (www.techsoup.org), who provided additional support for travel that allowed key participants to join in the proceedings.

And again most importantly, **all the participants from Open Translation Tools 2007**, whose collaborative spirit contributed to a magical and transformative gathering, and whose strong ethic of sharing enabled so many others to learn so much in a few short days.

We hope to find opportunity to meet together again and further strengthen the of this nascent network of practice.

## OTT07 Event Background and Agenda

The first-ever Open Translation Tools Convergence (OTT07) took place from 29 November to 1 December, 2007 in Zagreb, Croatia. This 3-day event brought together two passionate communities: those creating open source software tools to support translating open content, and those with a need for better tools to support translation of the open content they create. "Open content" was interpreted to encompass a range of resource types available under open licenses such as Creative Commons[1] (CC) and Free Document License[2] (FDL), ranging from books to manuals to documents to blog posts to multimedia. "Open translation tool" was interpreted to encompass any piece of software which supports or performs language translation, and which is distributed under a free or open source software[3] (FOSS) license.

The event was co-organized by Aspiration and Multimedia Institute (MI2), and was supported by the generosity of the Open Society Institute, with additional support for participant travel provided by TechSoup. It was hosted in the Student Center at the University of Zagreb.

The event was convened to:

- Document the open source translation tool landscape – What tools are out there? And what should we create to fill the gaps?
- Inventory "open content translation use cases" - What translation support is needed?
- Strengthen the community of practice around open source translation tools for open content, with a particular focus on delivering value to nonprofit and non-governmental organizations (NPOs and NGOs).

All event proceedings were captured on an event wiki[4], and ongoing discussions continue to take place on the event mailing list[5].

The agenda followed a user-driven approach to map out translation use cases and tools, assessing what is supported by currently available open source software tools and services, and identifying pressing needs. Primary focus was placed on tools and technologies which can support and enable distributed human translation of content, but the role of machine translation was also considered at many points.

The event targeted three complementary outcomes:

- A mapping of the open source translation tool landscape, enumerating tools and tool categories as well as services, projects and resources, and assessing gaps and opportunities for development. There is currently very little in terms of a directory of open

1 http://creativecommons.org
2 http://www.gnu.org/copyleft/fdl.html
3 http://en.wikipedia.org/wiki/FOSS
4 http://opentranslation.aspirationtech.org
5 http://lists.aspirationtech.org/lists/info/opentranslation

translation tools for open content publishers, and this event served to create such an initial inventory.

- An enumeration of “open content translation use cases”, with open content creators and publishers describing scenarios where they would like open source software tools and technologies to support their translation needs. These use cases covered a range of tasks (“I need to translate a document into a second language”) and usage scenarios (“I need a widget for my blog that links to open content translation request services and lists available translated versions of my content”).

- A strengthened community of practice around open translation tools for open content for NPO and NGO needs. While many inspiring projects are in play all around the globe, there are few opportunities for practitioners in the field of open content translation to meet and collaborate as a community. OTT07 provided such a venue.

This paper describes the outcomes associated with these objectives, as well as related learnings and observations.

**Event Agenda**

The OTT07 agenda was collaboratively developed by participants and event organizers in the time leading up to and during the gathering. The event was directed using Aspiration's collaborative approach to event facilitation[6].

The first day of the event focused on community building and agenda refinement. Participants introduced themselves at opening circle and then engaged in interactive debates on issues of philosophy and technology. Attendees then broke into small groups to review the agenda and make suggestions for additions and changes. The afternoon saw the beginning of the mapping processes for open translation tools and use cases. Day two was a knowledge sharing day, with participant-driven sessions on a range of topics. Day three allowed participants to finish work on tool and use case mappings, engage in a range of additional sessions, and share reflections during the closing circle.

The following were session highlights from the agenda:

**Spectrogram**: Designed to catalyze dialog and sharing of perspectives, spectrogram[7] statements were read and participants were invited to weigh with their opinions by positioning themselves along a line on the floor with one end labeled “completely agree” and the other end labeled “completely disagree”. The two spectrogram statements were:

- “The open content community needs one unified platform that meets all needs”
- “Anyone can be a translator”

Opinions on the first assertion tended toward agreement that a collection of small, well-

6 http://facilitation.aspirationtech.org
7 http://facilitation.aspirationtech.org/index.php/Facilitation:Spectrogram

defined and interoperable tools was preferable to any monolithic platform. The second statement proved more catalyzing and divisive; some participants took the position that only those with specialized skills could serve as translators, while others asserted that anyone with knowledge of two languages and a desire to offer support could be a translator. A complete transcript of opinions was documented on the event wiki[8].

**Agenda planning break-outs**: Participants broke into groups of 4 or 5 people, and used "Post-it" sticky notes to brainstorm topics, questions, and conversations they wanted to see included in the agenda. Each note was allowed to hold one sentence, either a statement or question addressing an agenda topic. After small-group note creation, participants reconvened and posted all notes on a large wall. They then undertook an interactive grouping process, gathering related notes and categorizing each group. Categories then served as candidates for additional sessions and themes to add to the agenda.

**Open translation tools and use case mapping sessions**: Addressing one of the central goals of the event, these sessions sought to document both what is available and what is needed in terms of open translation tools. Parallel sessions allowed developers and open content publishers to map out both scenarios where technology-based translation support was needed, as well as the inventory of currently available open source translation tools. Outcomes from these sessions are documented in the "Open Translation Tools" and "Open Translation Use Cases" sections of this paper, as well as in corresponding appendices.

**SpeedGeeking sessions**: Participants used the SpeedGeeking format[9] to demonstrate projects to each other in a fast-paced, interactive setting. In two rounds of ten projects each, participants rotated from station to station, enjoying 5-minute demonstrations and spirited question-and-answer with various open translation projects.

**Peer-to-Peer Discussions**: The bulk of the agenda at OTT07 focused on peer-driven sessions, many of which were identified during the agenda brainstorm on the first morning. These sessions spanned a broad range of open translation topics, and included:

- Translation 101: the basics
- Regional and cultural issues in translation
- Managing distributed volunteer communities
- Data standards for translation
- The role of RSS in translations
- Managing quality in translation projects
- Translation workflow and project management
- Inside the professional translation industry
- Envisioning a "dream translation tool"
- Facilitating multilingual online discussions
- The state of machine translation today
- Business models for translators
- Translation memory

8 http://opentranslation.aspirationtech.org/index.php/Spectrogram
9 http://facilitation.aspirationtech.org/index.php/Facilitation:SpeedGeeking

- Licensing issues for translated works
- Preparing content for translation
- Software localization
- Measuring success in translation projects

**Final plenary discussion**: At the end of the third day, participants came together to assess the progress made at OTT07 and consider possible next steps. Outcomes from this discussion are documented in the "Event Outcomes and Ideas for Next Steps" section of this paper.

Complete details of the agenda, including detailed session notes, can be found on the OTT07 wiki[10].

10 http://opentranslation.aspirationtech.org/index.php/Event_notes

## Translation Concepts

OTT07 focused on a relatively specific category of translation tools: those which support or enable the human translation of text content. The following section defines key concepts which will be used in the course of the paper.

Translation refers to the act of taking content from a "source language" in an initial "source format", and recreating that content in the required "target language" and "target format". In terms of task terminology, there is a critical distinction between localizing, which is usually technology-related, such as when one localizes a piece of software, and translation, which is purely content-related and more generic in format. Internationalization is the act of preparing something (for instance, a software application or a document) to be translated and/or localized into other languages. Interpreting is the verbal translation of dialog.

Professional translation is a $15 billion (USD) industry[11], and by most accounts a highly fragmented one. Of all the content in the world, less than 1% is professionally translated. Translation companies function as brokers between freelance translators and companies with translation needs; at least 80% of work is farmed out to independent contractors, and the revenue split is about 50/50 between the company and the translator. The split has historically been more favorable to the companies, but the industry is changing rapidly, with new tools emerging, experienced translators charging more and customers becoming better educated and thus paying less.

Most translations are done in teams of three, referred to as TEP: Translator, Editor, and Proofreader. Translators do the actual translation, while editors focus on style, consistency and qualitative considerations, and proofreaders address fundamental spelling, grammar and quantitative correctness. Most translators translate from their second language into their native language, and most focus on a single topic, such as health care, technology, or poetry. Freelance translators typically use nothing more than Microsoft Word to do their translations, regardless of what the end product will be. Industry convention has it that content to be translated is extracted from where it resides, put into Word documents, and emailed as an attachment to the translator. Translators will often utilize machine translation tools to save time and generate more income. Translated texts are then sent back in Word to "translation engineers" who insert the translated content into final products, whether it be HTML content for the web, video subtitles, print layouts, or other.

To give some indication of the range of translation demand in terms of languages, every piece of software that Microsoft releases is localized into 42 different languages, and their most important content sets are translated into 60-70 languages. Personnel requirements for content that needs to be translated to that many target languages can obviously be quite substantial.

Translators can make use of technology in a number of ways. Computer Aided Translation (CAT) tools provide translation memory, which offers suggestions for sentences that have

11 All translation industry metrics in this section were provided by Ed Zads of DotSub.net.

already been translated in the past. Machine Translation (MT) is often employed by translators to generate a “first pass” translation, which is then corrected and improved. And translation workflow tools, which include project management and tool integration support, provide those overseeing translation projects with an overview of progress and task status.

A number of data format standards exist for the purpose of enhancing translation workflow and maximizing re-use of translation knowledge. XLIFF (XML Localization Interchange File Format) is a standard specification which enables decoupling translated text from associated formatting information, and allows inclusion of additional metadata to support workflow and preservation of context. PO (Portable Object) is a file format that stores both the source language and the target language translation, and is a commonly used format in localization and other “small grain” translation tasks where the text units are strings or short text passages. MO (Machine Object) files are compiled, machine-readable PO files. TMX files (Translation Memory Exchange) enable translation memory datasets to interact, while TBX (Term Base Exchange) enables the sharing of glossaries.

Central to any translation process is the issue of quality; the accuracy and semantic nuance of translations can vary widely. Translators need an understanding of the relevant domain and the technology, coupled of course with a deep understanding of the languages between which they are translating. Processes based on iterative methods, where multiple people verify accuracy and quality, and where knowledge is captured in translation memories and glossaries, yield the best results over time. Editors need to assure that varying styles and conventions of different translators are unified in final products; style guides are combined with memories and glossaries to realize this goal. And all of these quality factors depend ultimately on the development of translation communities of practice and associated resources, where the ability to track reputation and strengths allows for optimal matching of translators to tasks.

Overall, the translation industry is trying to catch up with the internet. The processes and personnel models which have stood fast for decades are increasingly brittle and obsolete in a world where distributed translation talent and collective knowledge can be tapped and marshaled in new and increasingly innovative ways. While advanced translation platforms exist, they are almost always prohibitively expensive, or proprietary to individual companies and thus unavailable to meet the needs of more general audiences and communities, in particular those of open content movement.

OTT07 was an opportunity to survey the possible, the available and the yet-to-be created in the field of open translation tools. This was done by first surveying the state of need, through the eyes of those producing open content, and capturing use cases which describe situational needs for translation. Available open source translation and localization tools were then enumerated and categorized in terms of functionality. The following sections describe these two processes.

## Use Cases for Open Content Translation

Too often when considering the role of software technology in social change and civil society contexts, the dialog takes a “tools first” approach, with an explicit focus on the features and capacities of the software applications rather than on the needs and desired workflows of those who will employ the tools in their efforts.

At OTT07, participants engaged in two parallel paths of learning and mapping. One team documented available translation tools and technologies, categorizing and differentiating the different offerings. Another team oversaw the enumeration of a set of “use cases” which described how publishers of open content would want to be able to translate and manage that content.

While not intended to be comprehensive, the use cases documented at OTT07 span a broad range of scenarios and needs, and serve as an excellent sampler of the types of translation requirements which exist in the open content community and beyond.

In many of these use cases, the “open” nature of the content is a secondary consideration; phrased differently, many of these use cases describe more general content translation needs. That said, a particular emphasis was placed on selecting use cases most germane to the needs of those working in open content projects.

Note as well that these use cases are deliberately simple and smaller-grain. While there is strong temptation to make them more comprehensive (e.g., each translation use case could include editing, proofreading and other work flow sub-tasks), the purpose of this inventory is to map out the range of cases rather than to exhaustively detail the nuances of each one.

There is also a “transitional” nature to these use cases; as indicated elsewhere in this document, translation and localization are quickly evolving from primarily offline to collaborative online processes. Use cases below that describe the translation of “offline” documents will increasingly be supplanted by counterpart online processes; the challenge lies in converting to online formats and toolsets. In addition, the advent of “smart” open translation infrastructure tools will allow synchronization between rich collections of assets and one-to-many format conversions (e.g., outputting a translated page as an office document, a PDF, and a JPEG snapshot).

### What actually is a use case?

The notion of a “use case” means different things to different audiences, with substantial variances in both formality and format. For the purposes of OTT07, participants defined a use case to be an example of an interaction between a user and a software system which generates some form of value to the user. Such use cases help to capture and scope out what a particular system needs to be able to do in order to support its target users.

At OTT07, use cases were defined as having four primary attributes:

- A title
- A short background description for context
- The description of the use case
- The desired outcome for the user

The following is an example use case from the OTT07 set to illustrate the format:

**Title**: Translate blog post

**Background**: A web site has bloggers from around the world posting content which needs to be viewed in more than one language.

**Use Case**: A translator is assigned to translate a specific entry, logs into the website to find that entry, and enters into a translation interface, allowing them to translate the blog entry. The translator submits the entry is placed in the queue for approval.

**Outcome**: Blog entry is translated, and ready for approval.

While "real world" use cases tend to include specific details (e.g. specific languages, situations, technologies), the use cases mapped out at OTT07 have been authored to be more generic and thus hopefully relevant to a broader range of audiences.

**The use case mapping**

Iterating through several phases, the team of participants focused on mapping open translation use cases in a range of categories. Some use cases stemmed from the specific needs of individual participants, while others were amalgams of collective participant needs, and still others represented general and well-known functionality requirements.

After iteration and refinement, the use cases were grouped into seven categories:

- **Content translation**: Translation of text content in various forms.
- **Multimedia translation**: Translation of audio and text content in videos, audio streams, and graphics.
- **Translation workflow**: Management capabilities needed for oversight and realization of translation processes.
- **Machine and computer-aided translation**: Translation done by software programs instead of humans.
- **Interpreting**: Simultaneous translation of spoken words.
- **Content rendering and presentation**: Use cases for tools that render and display

translated content.

- **Localization**: The process of converting a piece of technology, such as a software program, to be available in new language or locale.

The complete set of open translation use cases can be found in the Appendix I, Open Translation Use Cases.

**Additional considerations**

In parallel with enumerating use cases, participants assembled a collection of considerations to be addressed when considering a translation strategy:

Content:

- Is the content model chronological (blog or web site), versional (wiki), or real time (chat) translation?
- Is web content published in a dynamic or static model?
- How many languages are you translating from? How many languages are you translating to?

Human resources:

- How many translators, editors, and proofreaders are available to the project?
- Will translators, editors, and proofreaders be paid or serving as volunteers?
- What is the level of experience among the translators, editors, and proofreaders?

Workflow and quality:

- Will the translation process be collaborative, such as on a wiki, or segmented, with individual documents delegated to different translators?
- What roles will be filled in the translation process? Is there an editor, a proofreader, and/or a project manager?
- Is there a rolling deadline or specific deadline for translations?
- Can the quality to be "good enough", or does it need to be professional and exact, such as for medical and legal contexts?
- Does machine translation exist in your source/target language pair? Do you need a human eye on machine translations as opposed to human-only translation?

Technical and logistical:

- What are the source and target text encodings? How many scripts are required? (e.g Arabic, Cyrillic, etc.)
- What media format and file type are you translating to and from?
- Does your target medium support your target language? (e.g., consider the lack of Flash support for right to left text orientation)

Such questions and considerations are put forth to encourage those creating content and requiring translation to consider all their needs before selecting tools and strategies for their translation solution.

## Open Translation Tools

The field of translation is in a state of transition, and software tools to support language translation are evolving with corresponding rapidity. Increasingly available internet resources are quickly expanding the possible and the practical when it comes to translating content, and processes and business models which have remained relatively staid for decades are being rethought. Even in the so-called “broadband” era where substantial parts of the globe enjoy ubiquitous high-speed access, most translators and translation firms have employed rather rudimentary technology processes in their translation workflow, generally copying and pasting text between Word documents and transmitting their work artifacts as email attachments that lack all but the most basic version control or metadata. But new online tools and innovative new workflow models are turning the translation field on its head.

The state of open translation tool offerings reflects the same flux. Real-time access to a global network of translation services and talent is a resource only now beginning to be leveraged by the translation industry, and upstart multilingual projects on the internet are pushing the state of the art by treating translation as an exercise in distributed problem solving. In addition, most translation tools are only now beginning to incorporate substantial workflow into the software, tracking user roles, permissions and detailed state information for each translation project. From the RSS-enabled platforms like Worldwide Lexicon, which automates translation requests and submissions, to crowd-sourced tools like dotSUB, which though not open source still employs an open approach to data and translation for subtitling digital videos, open translation tools are demonstrating their ability to not only track but also outpace closed and proprietary offerings.

The goal at OTT07 was to take a snapshot of the open translation tools that were available, and in turn to analyze the gaps. Participants engaged in a range of brainstorming and knowledge capture activities; collaborative mapping of available tools served as a prelude to a discussion of what a “dream translation tool” might look like and offer.

Open translation tools, then, fall into a range of categories:

- **PO and XLIFF localization editors**: This encompasses offline, online and distributed localization tools that read and write data in PO, XLIFF and related formats. These serve as the essential tools for many translators and localizers. Examples of these tools include Pootle, Poedit, gtranslator, Transolution, and KBabel.

- **Translation workflow**: These tools manage roles, tasks and other project information, and often interoperate with other translation tools and version control systems. Workflow is a critical area for the growth of open translation, and there exists a range of un-met needs in terms of workflow support. Examples of these tools include Transifex, Translate Toolkit, Pootle, Launchpad Translations and Worldwide Lexicon.

- **Subtitling**: As video becomes a more pervasive web offering, tools for adding translated subtitles to videos are ever more in demand. Examples of such tools include GNOME Subtitles and DotSub (which has open data but not open source code).

- **Machine translation**: These tools, which at present are primarily hosted as web sites like translate.google.com and BabelFish, perform algorithmic translation of text from one language to another. Examples of these tools include Apertium and Moses.

- **Translation Memory**: These Computer Aided Translation (CAT) tools store small discrete language fragments, passages, and terms in order to assist human translators as they perform their work. Examples of these tools include QT Linguist and OmegaT.

- **Dictionary and Glossary**: As their names imply, these CAT tools store definitions for terms in a given language, and support translators as they map from one language to another. Examples of these tools include CollaboDict and Transolution.

- **Wiki translation**: These modules and extensions enhance and augment existing wiki platforms with tools for performing and managing translation of wiki content. Examples of these tools include Cross-Lingual Wiki Engine and BetaWiki.

As with almost any collection of software tools, these categories blur and overlap on a tool-by-tool basis; the categories are somewhat arbitrary and many tools fall into more than one.

A complete listing of open translation tools is provided as Appendix II of this paper. The Open Translation Tools Appendix enumerates the primary FOSS tools in each category. Unless otherwise indicated, all tools described in the appendix are distributed under Free and/or Open Source (FOSS) licenses.

**Related tools and resources**

In addition, there are number of related tool categories and resources which are worth mentioning in the context of open translation:

- **Code libraries and packages**: While the focus of OTT07 was on tools for end users in various translation workflows, code libraries are an essential and core element of the open translation ecology. Most ubiquitous among the libraries is gettext, the API used by a wide range of localization and translation tools to read and write PO files and other translation-related data. Additional libraries are enumerated in Appendix II.

- **Content Management Systems (CMS)**: FOSS CMS platforms offer a range of multilingual capabilities. While no current CMS readily support a true multilingual web site (i.e., either a single site synchronously available in multiple languages, or alternately a site on which separate pages can contain text in multiple scripts), many CMS platforms offer good support for translating site content. These include Drupal, Plone, Joomla!, Twiki, and FLOSS Manuals.

- **Operating systems**: End-user support for multi-lingual operating systems is very much the exception; users of Windows, Macintosh, and most Linux distributions install for a given locale, and must often reboot to properly run in a different locale. A

noteworthy variant in this regard is Linguas OS, a distribution of GNU/Linux operating system adapted for professional translators and those working in software localization.

- **Guides and online resources**: While too numerous to enumerate here, a number of guides and online resources are available to those working in open translation. Several of the most noteworthy include the UNDP Localization Primer[12], LISA publications which provide best practices and primers from Localization Industry Standards Organization[13], and the wiki at translate.sourceforge.net[14]. A resource specific to GNOME is Damned Lies[15], which is a hub for translation workflow for the GNOME project.

### Open Translation Feature Gaps

A primary point of discussion at OTT07 was “what's missing” in open translation tools. While a range of issues were identified, there were two primary functionality gaps that surfaced in a range of conversations:

- **Workflow support**: Though a number of open translation tools provide limited support for translation work processes, there is currently no tool or platform with rich and general support for managing and tracking a broad range of translation tasks and workflows. As documented in the Appendix I (Open Translation Use Cases), the internet has made possible a plethora of different collaborative models to support translation processes. But open source tools to manage those processes, tracking assets and state, role and assignments, progress and issues, are few. While tools like Transifex provide support for specific workflows in specific communities, generalized translation workflow tools are still few in number. As described in the “Dream Translation Tool” section below, an open translation tool which understands the range of roles played in translation projects, and provides appropriate features and views for users in each role, is still at best in the concept phase. As is, most open translation tools provide workflow support for the single type of user which that tool targets.

- **Distributed translation with memory aggregation**: As translation and localization evolve to more online-centric models, there is still a dearth of tools which leverage the distributed nature of the internet and offer remote translators the ability to contribute translations to sites of their choosing which request the same. As of this writing, Worldwide Lexicon is the most advanced platform in this regard, providing the ability for blogs and other open content sites to integrate distributed translation features into their interfaces. In addition, there needs to be a richer and more pervasive capture model for content translated through such distributed models, in order to aggregate comprehensive translation memories in a range of language pairs.

Other open translation technology gaps identified by OTT07 participants included:

12 http://www.iosn.net/l10n/foss-localization-primer/
13 http://www.lisa.org/Industry-Data.512.0.html
14 http://translate.sourceforge.net/wiki/
15 http://l10n.gnome.org/

**Interoperability**: Lack of integration and interoperability between tools means both frustration for users and feature duplication by developers. Different communities have their own toolsets, but it is difficult for a translation project to make coherent use of a complete toolset. Among the interoperability issues which require further attention in the open translation tools ecology:

- Common API's for tools to connect, share data and requests, and collect translation memories and other valuable data.
- Plugins for CMS's to export content into po-files, so that content can be translated by the wealth of tools that offer PO support.
- Better integration between different projects, including shared glossaries, common user interfaces and subsystems, and rich file import/export.
- Generic code libraries for common feature requirements. “gettext” stands as one of the most ubiquitous APIs in the open translation arena, but many more interfaces and services could be defined and adopted to maximize interoperability of both code and data.

**Reviewer Tools**: Tools for content review are lacking; features for quality review should be focused on distributed process and community-based translation. As such reviews can be a delicate matter; the ideal communication model when there are quality problems is to contact the translator, but timing can be an issue. In systems with live posts and rapid translation turnaround, quick review is important and it may not be possible to reconnect with the content translator in a timely fashion.

**Tool Futures**

Several open translation projects were represented at OTT07 who have not yet released corresponding code. These projects include:

The **Meedan Platform** is being designed to enable communities of individuals to sort, share, and translate user generated and mainstream web content. Meedan is an early effort to leverage social networking and community building technologies with the explicit goal of contributing to cross-language distribution, indexing, and access to content. Meedan combines aggregation and clustering services (specifically geared toward extracting “event-oriented” clusters) with user-editable machine translation towards a vision of creating social translation marketplace. The first iteration of the site will enable Arabic and English speaking communities to better understand the diverse narratives that describe emerging events in the Middle East/North Africa region.

**FLOSS Manuals** is working on a plugin for TWiki that will manage copying material from other TWiki installations, basic translation workflow, and inline editing comments. Features will include translation of PO files, online editing of CSS, and basic management of image

translation. The plugin is in beta form, and can be found at http://en.flossmanuals.net/bin/localize .

**Passiflora** will be a Free and Open Source, web-based cooperative document writing system. It is designed to prevent data version conflicts (for example: user 1 loads file, user 2 loads file, user 2 saves, user 1 saves, work of user 2 is lost or incorrectly mixed with work of user 1) without any locking (multiple users can simultaneously work on the same document). Passiflora will have special support for open content licensing: it will feature a license selector, and it will warn users when they combine content with incompatible licenses (in which case it will offer to request permission for use of a different license - if the author of the original work did not opt out of being asked for permission).

OTT07 remote participant **Asgeir Frimannsson** is working on a Java API for manipulating XLIFF 1.2.

## OTT07 “Dream” Translation Tool

As part of the tool mapping exercises at OTT07, participants were asked to envision what their “dream translation tool” might look like. The idea was to specify a feature set for a tool which does not yet exist, but which would meet the broadest range of translation needs in terms of features, supported workflows, and business models.

It is important to note that this was a purely theoretical exercise; participants generally agreed that large monolithic tools were not the right course for the future, and that a small, distributed set of tools that work well together was the recommended path for better supporting open translation efforts.

That said, the generated feature set was both expansive and impressive in its ambition to meet a wealth of translation needs. The following sections describe those desired features, grouped into three sets: core features, workflow support, and additional features.

While most of these capabilities are available in various proprietary and open source tools, there is not currently a FOSS tool or tool set that comes close to offering the features enumerated below.

### Core Features

The following were considered requisite for any “dream” functionality. These are primarily features associated with the translation of a single text source; higher-level features are described in subsequent sections.

Ideally the following would all be simultaneously available in the user interface for the tool:

- **Original text display** would show the source text, using color and iconography to denote progress, commentary and other relevant metadata.
- **Output/preview display** would render the translated text, maintaining layout from the original and supporting detailed linkage between the source and translated versions of the text.
- **A commenting/annotation feature** would allow users to select and annotate text in both the source and translated text in order to add comments and other useful annotations to the core data.
- **Machine translation support** would enable users to generate a machine-translated version for all or selected parts of the source text, in order to obtain a first-pass rendering of the target translation.
- **Terminology/glossary translation** would provide support for translating specialized terms from translation memory.

- **Dictionary widget** would provide definitions for terms in both the source and target languages.

Other desirable core features included:

- **Pervasive Unicode support** for all input and output text, with rich conversion support in both directions. Unicode is a “superset” character encoding, with the ability to store any language or character set. Many existing tools are not Unicode-aware, creating limitations and interoperability problems.

- **Ability to view alternate source text**, in situations where the source has already been translated to another target language. In these situations, the tool would enable translators to view and utilize prior translations as secondary “source” for clarifying meaning and keeping translations consistent.

**Workflow Features**

The following features would address support for the actual processes, or workflow, of text translation.

- **Progress and state management**: The core worklflow features would enable definition of milestones, assignment of tasks, and entry of time estimates for pending work. For both individual documents and collections of documents, the tool would provide the ability to track translation, editing, and proofreading status. The tool would also support progress estimation in both objective terms (“document translation is 80% complete”) and subjective ones (“this is high quality translation”).

- **Role-based user features**: The dream tool would expose different feature sets for different types of users in the translation process:

    - **Project managers** would have a dashboard of all translation activity and status.

    - **Translators** would view their pending translation documents and tasks, in concert with tools to progress on those tasks.

    - **Editors** would view the queue of documents and document segments awaiting review, as well as the status of documents in editorial process.

    - **Proofreaders and reviewers** would view the queue of documents and document segments awaiting proofreading, as well as the status of documents in proofreading process.

    - **Original authors** would be able to track the translation status of documents they had created and made available for translation.

    - **End users** would be able to track the availability of translations they had

requested.

- **Status change notification**: The platform would enable all stakeholders to be notified of changes in status to any document in the system, as well as the arrival of new documents into the system. Notification could be done via email or RSS (Rich Site Syndication).

- **Accounting**: The tool would be able to track hours and completed tasks for each project member, allowing managers to both assess productivity and track compensation.

- **Collaborative document mark-up**: Users could make annotations – e.g., "I had a problem with this phrase"– at any level of detail or scope, and invite others to give feedback. Such markup could also be tied to shared online discussions such as chat rooms or instant messaging.

- **Review process**: As each translation was ready for review, the tool would support assignment of review tasks, and track both editorial and proofreading reviews. An additional component would provide support for peer review, where fellow translators could assess the work and comment on semantics, nuance, and other subtleties.

- **Reputation management**: Hand-in-hand with a review process would be reputation tracking for each user of the system, especially translators. Such a subsystem would track the quality of each user's work, in both objective terms (100% of assigned tasks completed) and subjective terms (editors, proofreaders and peers could evaluate translators on various criteria). Such a system would ideally enable translation managers to select the most suitable translators and other personnel for specific translation tasks.

- **Import and export of source documents**: The tool would be able to handle the broadest range of document formats and encodings, allowing easy import of source texts from Microsoft Word, HTML, PDF, raw text and other editing tools. Translated texts could be exported in all of the same formats.

- **Segmentation of larger texts**: Large documents often need to be broken down into smaller units in order to be delegated to different translators or parceled out in manageable units. Segmentation support would allow breaking large documents into such units, provide tracking of each segment's status and task ownership, and enable eventual re-assembly of the translated segments into a final unified document. Additional functionality would allow prioritizing the segments, so that important sections were done first, and less important sections could be deferred and potentially delegated to less experienced translators.

- **Version tracking**: Translated documents go through a number of versions, both in translation as well as during subsequent editing and proofreading. The tool would archive all versions of each document using a subsystem such as Subversion, and

then provide the ability to compare any two versions to see differences and changes.

- **Cross-lingual change tracking**: While version tracking would maintain history for individual documents, cross-lingual change tracking would enable project managers and translators to be notified when a source document was changed, in order that other dependent language versions of the document could be flagged for pending updates. Such a feature would enable multi-language sets for a particular document to remain synchronized.

**Additional Features**

- **License tracking**: A dream tool would be able to track licensing for imported documents, and ensure that appropriate licensing was assigned to any translated works in a system that supported human overrides to reflect the broad range of intellectual property agreements under which translations can happen.

- **Offline use**: While internet-based features would be critical to the realization of any "dream tool", just as essential would be the ability to enjoy rich offline functionality. The tool would need to launch and operate when no connection was available, supporting translation and editorial tasks, and storing edits and progress updates for synchronization the next time the user connected.

- **Unified translation memory**: This feature would provide local translation memory combined with access to external translation memories. There are a range of memories available, but it would be useful to have one central repository. Similar functionality could be provided for glossaries.

- **Multi-lingual comparison**: For documents translated into multiple target languages, this would allow translators to review how translation was done for related languages. For example, when translating to Serbo-Croatian, a translator could be aware of other Baltic language translations, and could see the work other translators had done in those similar languages.

- **Pledge bank**: Funding the translation of open content is often problematic, because it is not usually institutionally driven. Pledge bank functionality would allow translators to post estimated costs for translating particular documents, and allow parties interested in seeing the document translated to pledge monies they would contribute if the document was actually translated. The document would only be translated and pledges collected once the pledge total reached the projected translation cost.

- **Translation of SVG graphics**: Scalable Vector Graphics (SVG) are images where the data stored includes any text contained in the graphic. A dream translation tool would support translation of the text within SVG files, in order to offer a more complete translation solution.

## Technical Innovations and Considerations

Several significant emerging tools and technology trends were highlighted during the proceedings at OTT07. The following passages summarize the most significant of those topics.

### Using RSS for Translation Tool Interoperability

One of the most compelling innovations shared at OTT07 was the unique use of RSS (Rich Site Syndication) being made by projects like Worldwide Lexicon.

RSS was never specifically designed to be used in translation tasks, but some simple tricks can be employed to make it work effectively for such purposes. Every RSS item contains a "GUID" field, or Global Unique Identifier, which allows the feed item to be uniquely identified. Feeds can be enhanced for use in translation workflow by concatenating language or locale ID's to the GUID in RSS feed items. A site requesting translation can then syndicate a feed with items to be translated and indicating the target language by modifying the GUID. Remote systems can then aggregate such feeds, perform translation in whatever fashion, and then re-syndicate a link to the result. This is a particularly flexible approach to interoperability because the publishing system does not need to know anything about the source material. And whoever is getting the translated feeds does not have to know what the process of translation was.

RSS in this context is not being used in its normal mode, as a consumer service, but rather as a quick and simple way to move data between systems, as an internal tool. As such it can be employed in at least 3 use cases:

1. **True multi-lingual site**: The feed goes out to be simultaneously translated into a number of languages, with translations published back into the site based on one or more return RSS feeds.

2. **Asynchronous translation of new content for a mono-lingual site**: For a site published in a “home” language, but wanting to offer on-demand translation to other languages (as opposed to a full parallel site), the feed can serve as a request list for translation, and community members can be invited to help with translation tasks by watching the feed.

3. **Automated aggregators**: Sites which pull information from other site feeds in multiple languages and then aggregate, translate, and re-syndicate them out in different language feeds.

There are definitely issues with such approaches. Not all RSS feeds or aggregators use Unicode, and thus can only publish or aggregate data in more limited encodings. Planet, for instance, the popular feed aggregator for virtual communities, does not use Unicode, but instead uses Latin1. This implies the need to educate users as to why they might not be able to read text that is in Unicode if they are using older browsers and operating systems.

Another issue is one of workflow support, and it is indicative of the very new nature of these mechanisms in translation. While RSS-based translation requests can easily be aggregated into simple queues, there is a need to build other tools around the feed data to provide queue and task management. Basic tools are available, such as online newsreaders, but for a task such as dividing an RSS feed among volunteers, or letting them select and indicate items they plan to translate from that feed, tool sets are only now emerging.

A specialized version of this translation model could tap domain-specific knowledge by sending out phrases to be translated to a small set of people who are domain experts with specific technical language expertise. SMS (Short Message Service for cell phones) could also be utilized in such a model, enabling cell phones to serve as the editing interface for small translation requests. Such methodologies would however run up against interesting real-world boundary cases: in many languages, technical terms do not yet exist, and the ways in which new terms are coined vary widely. Some translators unilaterally make up terms or phrases, while others defer to national or regional conventions. The tradition in Serbia is to borrow technical words, while Croatians make new domestic words. And it is outright illegal to use English terms as new translated terms in Iceland.

One interesting refinement to such RSS models is the idea of using machine translation to translate just the first paragraph of the RSS summary field in order to provide an abstract in a target language, to enable readers to evaluate whether a full translation is warranted. It might also be possible in such a framework to provide parallel feeds of original and machine translation of the corresponding content.

**Machine translation**: While the focus of OTT07 was on human translation and the software tools that support those processes, machine translation (MT) has a substantial role in the human translation of text content. Translators often pre-translate text using machine tools, and then “clean up” the results. MT can also support “screening” of content to identify which documents merit higher-quality translation processes.

Many users have experience with MT from services such as Google Translate and Babelfish, which are well-known examples of machine translation services. Users of those tools paste in text to a web form or submit a URL for translation. But in general such experiences are not entirely positive; the tools do not work very well. MT is still far from mature or impressive; machine-translated texts are often a source of amusement, and successful content “round trips”—translating from one language into a second, and then correctly translating back to the original text—are the exception, not the norm.

Machine translation software falls into two genres, “statistical” and “rule-based” translation. Rule based systems rely on extensive human effort to construct language-specific grammars, lexicons and other resources which are then used to parse input text and generate translations. Statistical translation systems have been implemented to replace human effort with extremely large volumes of parallel text data which are utilized to statistically derive the most likely translations for a given text chunk.

Both modes of MT are resource constrained in their evolution; in rule-based systems, extensive labor is required per-language-pair to establish the necessary resources for high-quality translation, and very few good open grammars exist for all but the most common language pairs. Statistical translation relies on parallel corpora, but very few comprehensive corpora pairs exist, and again primarily for the most prevalent languages.

In the current state of the field, the output of machine translation is not useful for publishing when the paired languages are very different; for languages that are quite different (for example English and Chinese) you will find errors, words incorrectly sequenced, and incorrect terms being used. But such translations will often allow one to understand what are known as the "5 W's and 1 H" (Who, What, Why, When, Where and How). The distinction between publishing quality and comprehensible quality translation is generally phrased in terms of "translation for assimilation" – understanding quality – and “translation for dissemination” – publishable quality.

For close languages (for example Spanish and Catalan), much higher accuracy (up to 95%) is possible, resulting in a faster translation. And after such machine translation, it is not difficult to perform final edits to increase readability and correct residual errors.

Machine translation is funded by many governments and multi-national organizations. A consistent pathology is that governments often fund the development of translation software, but they let it be implemented as proprietary technology, instead of developing a corpus of work that is freely licensed or in the public domain (PD). Open and free licensing is necessary for distribution and further dissemination of the translation, as well as the tools, and open approaches to both minimize litigation. An additional issue is that some systems can make use of dictionaries, but these are generally copyrighted.

It was the opinion of participants at OTT07 that future trends in FOSS machine translation tools would see advances in translation of distant languages for assimilation, but not so much in translation for dissemination.

**Translation Memory**

One of the most valuable but arguably under-developed aspects of open translation involves the concept of “translation memory” (TM). The phrase refers to the process of storing translated sentence pairs, one in the source language and one in a target language. These data sets are used in subsequent translations to speed up the process of conversion by identifying already-translated passages and suggesting the appropriate mapping. Sentences in new texts to be translated are matched against those in TM's maintained by the translator or their organization. Translation memory is distinct from machine translation, in that the TM software makes suggestions to a human translator, rather than handling the whole translation as an automated task.

To create TM's, one takes existing source and translated files, aligns the files by matching strings by positions and paragraphs, and then does a one-to-one match to create the memory. TM can generate 20-25% matching for new projects, though matching at the

paragraph level yields much lower percentages; 10% is very good for matching rate for paragraphs.

Applying TM works by comparing 2 sentences, the one to be translated and one already translated, and seeing how many characters are different between the two. The degree of similarity can be adjusted in different tools to provide for different precisions of matching. TM does not work that well for small sentences, and single-word matches cause problems due to missing context: does the word "click" refer to a noun or verb? TM is very useful for "rough" translations, and TM's tend to be more efficient in technical contexts which are terminology heavy, but less so for documentation and freely written text where nuance can abound.

TM's in their current state for open translation are vexing in several regards. The first is stylistic; while the obvious intuition is to pool TM's for the greater good, TM pairs can often be personal to an individual translator, who might have their preferred way of converting specific phrases to the target language. In addition, very few professionals are willing to share their TM, as it represents their unique intellectual property. And open source projects which localize their software – which theoretically would be an excellent source of translation memory – have not traditionally demonstrated a commitment to sharing their translation memory data sets.

**Standards and Data Formats**

Participants at OTT07 agreed that data formats and standards are a primary consideration in all open and interoperable translation technology strategies. The open source community has a great opportunity to offer input on how these formats are going to evolve, and to ensure that they are usable in common open source and open content localization and translation workflows.

A substantial percentage of translated content is maintained in one of two file formats, PO and XLIFF.

The PO (Portable Object) format is a de facto standard. It is a very simple bilingual format, tracking source and target strings. It is primarily designed to support localization of software and make that task easier.

XLIFF, the XML Localization Interchange File Format, represents an initiative to create an open container format for localization resources, building on the way in which PO formats are used in the open source software world as a container format. It also has the ability to support translation memory; it is designed to translate any language encoding. XLIFF also provides ability to track workflow information, including translation history, tool-specific metadata, segmentation, comments and annotations. While XLIFF publishes many features, there are few open source tools offering support for this standard yet. Wordforge is one such tool.

Two additional data standards are published by LISA (Localization Industry Standards Association). These are the TMX (Translation Memory Exchange) and TBX (Term based exchange).

TMX enables translators to exchange translation memory. TMX operates at two levels: a primary source-to-translation mapping, as well as referencing of markup.

TBX enables translators to share terminology, and is a standard for glossaries. TBX differs from translation memory in that TBX is a glossary of specific terminology, where as translation memory is a set of the previous translations for arbitrary text.

Online environments such as Pootle exist to allow people to translate with PO or XLIFF files, and can be integrated with content management systems. But there are few open translation memory repositories, and attempts to establish the same raise substantial legal issues about reusing content.

A general concept put forth in OTT07 discussions was the need for standards that support "lowest common denominator" formats. This is critical for both backwards compatibility with less comprehensive standards and maximizing interoperability. Such standards should ensure compatibility with PO formats for text strings, RSS for moving data between systems, as well as appropriate web API conventions (e.g. REST vs. SOAP) where APIs are published.

**Standards support in FOSS communities**

FOSS projects do not always take advantage of standards for storing translations. By adopting PO file formats and other compatible standards, such projects could create major forward momentum for better translation and localization. As it is, few applications allow easy export into PO or PO-convertible formats.

**Translation Support in Content Management Systems**

A large amount of open content is published using various FOSS Content Management Systems (CMS), many of which are open source themselves, such as Drupal, Joomla! and Plone. But such platforms leave users very tied to the workflow within the CMS, and integrating translation support is not often easy or advisable. It can often be better to use two different systems and let the CMS manage and negotiate the content while the translation management system handles the translation.

Plone can work multi lingually "out of the box" but when most CMS are advertised as being multilingual, that simply means that they can support sites deployed and managed in different languages, rather than powering a single site with multiple languages. In other contexts, claims of multilingual support mean that the user interface can be translated, but such claims rarely refer to platforms which can easily display and manage content that is in multiple languages at the same time for a single site.

Emerging RSS models, such as those published by Worldwide Lexicon, provide the potential to greatly enhance the translation capabilities of traditional CMS platforms.

**Software Localization**

While not central to OTT07's focal theme of open content translation, the topic of software localization – converting an application's user interface and inner workings to support second and additional languages – was a frequent topic of conversation at the event. Many of the tools and workflows required to successfully localize software are equally relevant or required for more general content translation tasks.

Some best practices shared among developers at the event:

- Be internalization-aware from start of a project, designing code and documentation to be ready for localization.
- Think internalization first, localization second, and finally in terms of actual translation.
- Do not embed plural forms in strings to be translated; the complex variants associated with plurality in different languages lower the likelihood of successful translation.
- Avoid string concatenation; strings built out of smaller sub-strings are rarely assembled in the same sequence or with the same syntax in other languages.
- In parallel with concatenation, avoid embedded variables; placing counts and other context-specific data within interface language only complicates localization, especially when such variables appear at different locations in phrases from language to language. Where variables must be used, document variable usage and considerations.
- Expect and respond to feedback from translators; don't be afraid to ask translators if you need help.

Issues that had English- and US-centric origins were also addressed. Using jargon in interfaces limits the ability for tools to be utilized by other than professional users, and makes it harder for translators to localize. In English, technical people often coin or make up words, and there is a corresponding lack of equivalent terms in local languages. English has a strong “office culture” component, so translation can be tricky and does not always translate well. Concepts of “inbox” and “outbox” do not necessarily come across in other languages. Literal attempts to translate jargon do not always fit into allotted screen real estate, and icons can help with conveying meaning without strings. Further complicating matters is the fact that software terms are not always consistently used between applications.

A general learning in this regard is that if you enforce the use of native words in a translation, there is a better chance that the new term will be adopted. IT terminology has a small time window in a language to be adopted; it either sticks or never becomes commonplace. In addition, once a term is used you should strive to avoid changing its usage. Term creation is generally drawn from English words but sometimes is pulled from neighboring language IT terminology to create a new word or phrase.

There is a also difference between hosted applications and installed applications in UI design fluidity. Game users and web site users are accustomed to interface changes and more artistic endeavors that have more flexibility, while users of client applications must often wait for major releases to enjoy better localization support and usability. It is also not advisable to use jokes or humor in documentation or interfaces, as such items rarely translate consistently.

A surprising example in the localization arena is that it is very hard to localize Firefox, Thunderbird and Open Office.  The import/export process is buggy and broken, and Mozilla and Open Office are not using standard tools for translation, but rather their own custom solutions. There are campaigns to get them to switch to the gettext API, which have not yet been successful.

## Related Issues and Gating Factors

While the focus of OTT07 was on open source tools which support the translation of open content, a number of related discussions took place during the three days of the event. The follow summaries of those conversations are included to provide a sense of both the diversity of the event agenda as well as the richly textured nature of the translation/localization landscape.

**Lack of visibility for existing tools**: While there are compelling open translation tools and technologies available, very few users are aware of their existence. FOSS projects are generally not inclined to concerted marketing efforts, and are usually resource constrained even if they do want to do outreach. While some tools, like Worldwide Lexicon, have the ability to enjoy viral marketing by appearing on other web sites, most open translation tools are stand-alone applications. Efforts to raise the visibility of these tools will draw more users and support, in turn enhancing the quality, features, and relevance of these tools.

**Licensing for open content translation**: Translation of open content raises several issues regarding how the source content is licensed. The first is with regard to the nature of any translated version of open-licensed content: to the extent that most practitioners consider translations to be derivative works, this implies that open content distributed under “No Derivatives” variants of Creative Commons and other licenses can not be translated without violating terms of the license. A more nuanced licensing issue is the question of whether grammatical annotations to an open work (such as marking verbs and other parts of speech) qualify as derivative works under the license; this is a significant question with regard to translation memories and machine translation.

Further complicating these considerations is the frequent difficulty in contacting original authors of open content. Many blogs and other online publications which have open-licensed content fail to publish appropriate contact information. Email addresses are rarely published for a range of reasons, and short of commenting on individual blog posts, those wishing to translate open blog content for which they need permission can find themselves at a loss in seeking ways to contact the original creator.

The question was raised as to whether Creative Commons should develop a “CC Translation” license to signal “please translate this work, but do not create any other derivative works”, as the current practice of obtaining permission from the author does not scale well. The fundamental counter-argument to any translation-specific license is the ongoing problem of license interoperability. Current CC licenses do not mix well; it is difficult and often impossible to combine content licensed under different CC licenses and distributed the resulting work under a license that satisfies the terms of both source licenses.

The corresponding take-away for those publishing open content that they wish to see in broadest distribution is to use the most permissive license in order to maximize the ability of other parties to translate and redistribute the work.

**Regional and Cultural Issues**: While it may go without saying to those with experience in

translation and localization, the significance of regional and cultural issues in translation work can not be overstated. As norms and values vary, a range of secondary connotations and associations must be considered in crafting appropriate translations. Some of these considerations include:

- **Non-universal metaphors**: English-language software tools are often based on western office life. But in Serbia, for instance, there is no notion of inbox or outbox in the work world, so software users have a hard time using such metaphors in those office cultures. As another example, in the UK each house has a mail slot, but in Syria such slots do not exist, while American mail boxes often have a flag that is raised to show there is new mail. Such variances have a clear impact on translating and localizing “mail” and “office” interface metaphors.

- **Contextual meanings for colors**: In the west red generally means 'dangerous', but in China it means 'safe'. There are myriad other examples of variations of color meaning.

- **Concepts of time**: In central Africa the 24 hour day is strange, and the time cycle is based on the crop cycle and harvest, so one "year" can vary. The “AM” and “PM” of North America are by no means universal, with many parts of the world denoting time based on 24 hours instead of 12. In addition, Jewish and Asian calendars are different from western ones. And of course holidays are almost always regionally, culturally, or locally dependent.

- **Iconography**: A “thumbs up” or “hand-stop” sign from western iconography can be offensive in other regions. This implies that “just translating” strings is not sufficient; the CSS (cascading style sheets), colors, images, and positions of elements must be considered as well.

These issues are further exacerbated when translations go through multiple languages, such as from English to French to Arabic. Ideally translation tools would provide support for marking literal and non-literal translations, so that further translations could reference the original source for such passages. English is a highly idiomatic language; when viewing subtitled English films, things like jokes and other highly language-dependent exchanges often need to be redone. In films, these exchanges are sometimes color coded to show where the subtitles are not faithful to the original.

Approaches to the above issues manifest in different philosophical schools of thought; some translators prefer to "make it new and relevant" while others prefer to "be faithful to the original". Being faithful can affect the usefulness, as translating idioms like "break a leg" presents a real challenge. These issues are particularly acute with on-line translation and free software localization, because strings are translated independent of a larger context. Variances in translations can also reflect political agenda. This can be observed in regions like Croatia, Serbia and Bosnia, where translators may obfuscate a translation to make things very specific to their dialect, or nationalist agendas may be enforced by translating institutions.

**Education and awareness-raising for software engineers and content producers**: Many

of the learnings and best practices documented in this paper are not well known to those creating the open source translation tools and open content assets which make up the open translation universe. Efforts should be made to educate stakeholders about proactive design steps they can take in order to make their tools and text translation-ready and translation-friendly. In addition, such awareness-raising could also focus attention on the value of sharing translation memories and other intellectual property which otherwise is siloed per-project.

**Open-licensed documentation**: There are a wealth of gaps in documentation for open translation. These include style guides for each language, proper preparation of content for translation, best practices guides for the various translation roles, and overall process definitions and descriptions. Ideally these documents would all be available under open license and collaboratively maintained via wikis or other shared document editing systems.

## Event Outcomes and Ideas for Next Steps

OTT07 generated a range of collaboration, ideas, and emerging plans. The following section details the most interesting and compelling of these outcomes.

As has already been noted, many of the gaps identified below reflect the fact that the translation field as a whole is only beginning to leverage the resources the internet can bring to bear. Further discussion is needed to map out best paths for ensuring a flexible and open future for the open translation tool ecology.

**Meedan/Worldwide Lexicon collaboration**: Meedan and Worldwide Lexicon (WWL) got better acquainted at OTT07, and are now partnered and seeking funding to transpose the existing translation marketplace in the context of an informal and distributed setting. In the current market model, translation houses are primarily providing value in project workflow (distribution, reassembly, and delivery) and in quality assurance (reputation). The challenge of workflow in a distributed setting has been initially addressed by the WWL toolset. Reputation is a much more difficult undertaking, requiring a durable social networking/CMS platform for tracking translator work, versioning, lexicons, and translation memory tools.

It is clear that "subscription" mechanisms that allow translators to form a community or "translation network" around a given blog or feed offer great potential for affiliation-based near-real time translation solutions. This is particularly relevant in the case of NGOs and other groups that have motivated translation-capable constituencies. Meedan's Hybrid Distributed Natural Language Translation (HDNLT) contemplates an architecture that would allow for platform and device flexible (RSS, SMS, email) distribution and reassembly of translation problems to a discrete set of users.

There is a great deal of work to be done with portable reputation systems in distributed translation. An algorithm that combines quantitative and qualitative approaches and then allows users with higher reputations greater revision permissions may be a useful approach to evolving quality translations in a distributed open setting. Loren Siebert (www.linguastep.com) has written code that ties correction and recommendation permissioning to reputation in a language learning setting; the Meedan project seeks to take this approach and apply, per the architecture defined in HDNLT, to an open distributed setting. And as we consider distributed translation services, we should recognize that the role of machine augmentation plays a critical role. Meedan's relationship with IBM research enables their project to leverage their Machine Translation engines and toolsets. A critical advantage here to other available machine translation services is IBM's commitment to open licensing of linguistic data.

**Open Translation Book**: The event generated substantial momentum around the idea of authoring a book explaining Open Translation concepts, processes and best practices. The vision is to provide an overview of the emerging field and associated issues, while also explaining how to use open source translation tools to implement various open content translation use cases. A general overview would describe what a translatable project is, and delineate roles in such an effort. In addition, the book would describe how to create content that is ready and optimized for translation, and explain how to define translation workflows for

specific tasks. Such an endeavor would also provide an opportunity to apply best practices as documented at OTT07. Book text would be prepared in the most translation-friendly fashion, and the process of getting versions of the book translated would utilize the best of existing tools to manage both translation and workflows.

**Inkscape Book Sprint**: The Inkscape project and FLOSS Manuals are hosting a Book Sprint sponsored by Google in July 2008 in Paris. The projects first met at OTT07, and as the Inkscape team discussed the new manual requirements on the inkscape-docs list, it was agreed that FLOSS Manuals was the best option for keeping an official manual up-to-date while encouraging non-technical users to contribute. This was based on the simple interface, the ready availability as a web application, and the existence of translation tools being actively developed to provide a usable translation interface. In addition, the ability to organize different manual configurations and how-tos by re-arranging existing content, along with the ability to generate manuals in different distributable formats (e.g. PDF, HTML, print online with www.lulu.com) proved compelling. The conclusion was that these powerful capabilities create an exciting opportunity to build and maintain a dynamic manual with reasonable effort.

**Translator Commons** – Translator Commons was proposed at OTT07 as a destination where practitioners of open translation could go to find support and guidance for their practice. There is not currently any such online community for open translation, and the vision behind Translator Commons is to establish a venue where translators could discuss issues and challenges, including regional and language-pair-specific translation problems. Such a site would also maintain style guides and metastyle guides, and links to related resources. An additional desired feature would be a maintained mapping of other translation communities, providing context and contact information for those wanting to network more broadly. Ideally, such an entity would function as a social networking platform for translators, and provide not only support but opportunities for paid work and other engagement.

Other potential benefits of a Translator Commons could include:

- An opportunity to bring smaller networks of translators together, such as those working on “small” and less ubiquitous languages.

- Increased communication between people who translate the same languages for different tools. Those translating documentation or localizing different software applications could then share resources, learnings, translation memories, and general support.

- A collection of case studies describing uses of particular tools or sets of tools would illuminate their use while also helping developers identify what is missing, what is difficult, what is easy, etc.

- An entry point for those new to translation requirements and wanting to correctly design their projects for translation: Such an entry point would seek to answer the fundamental question "I'm starting a project, where do I start with translation?" Ideally such a resource could enumerate the available tools, references cases where they have been used, and provide recommendations and best practices.

**Corpora Commons** – One of the greatest barriers to higher-quality open source machine translation tools is the limited availability of rich corpora, which are used to map translation pairs between languages. Corpora Commons was proposed at OTT07 as an initiative for aggregating translation memories from which all practitioners could benefit. This would include a good collaborative system of dictionaries in a common format. Identified sources for such a commons included documents translated by the United Nations and individual governments, who have copious volumes of translated text in a range of language pairs. Global Voices was proposed as an additional resource because they provide translations of informal speech. But a gating factor in utilizing all these resources is that not all content is licensed using Creative Commons or in the public domain. An initial proposed thrust of the project would be to identify and raise awareness about available public domain content which could be utilized for such translation needs.

**Better API's**: Most translation tools operate as stand-alone technologies, and have not been designed with interoperability as a primary consideration. Participants at OTT07 called out API (Application Programmer Interface) design and adoption as a critical objective in growing the open translation movement. While many tools use standard data formats such as PO, XLIFF, and TMX, very few expose API's which would allow other tools to easily transfer data or invoke services remotely. The goal of an open translation API initiative would be to make the individual tools secondary, and shift focus to standardized services and functionality, thus providing translators with maximum power and flexibility. One particular use case for API-based integration was to make Transifex, Damn Lies and Pootle work as a tool suite.

## Appendix I: Open Translation Use Cases

As a means of inventorying requirements for open translation tools, OTT07 participants mapped out a broad range of use cases that described the needs of various open content projects.

For the purposes of the exercise, participants defined a use case to be an example of an interaction between a user and a software system which generates some form of value to the user. Such use cases help to capture and scope out what a particular system needs to be able to do in order to support its target users.

At OTT07, use cases were defined as having four primary attributes:

- A title
- A short background description for context
- The description of the use case
- The desired outcome for the user

The use cases generated at OTT07 were grouped into the following categories:

- **Content translation**: Translation of text content in various forms.
- **Multimedia translation**: Translation of audio and text content in videos, audio streams, and graphics.
- **Translation workflow**: Management capabilities needed for oversight and realization of translation processes.
- **Machine and computer-aided translation**: Translation done by software programs instead of humans.
- **Interpreting**: Simultaneous translation of spoken words.
- **Content rendering and presentation**: Use cases for tools that render and display translated content.
- **Localization**: Localization refers to the process of converting a piece of technology, such as a software program, to be available in new language or locale.

### Content Translation

Content translation use cases are a potentially boundless category; there are myriad publishing models, content architectures, and document formats which may require unique steps in the process of translation. The following use cases represent a range of content types, modes of creation, and frequency and granularity of content objects.

**Title:** **Translate text document**
**Background**: Translation of text documents is one of the oldest of the technology-backed use cases.
**Use Case**: A translator is assigned a long document to translate. They segment the document into smaller translatable units, and then translate each one sequentially. At the completion of each segment, it is submitted for proofreading and quality control before being assimilated back into one large document for final delivery.
**Outcome**: Document is available in target language

**Title:** **Translate a page on a static web site**
**Background**: Static web sites are usually maintained in an "offline" manner, with pages edited in programs like DreamWeaver or nVu. See additional use cases in Translation Workflow for site-level translation tasks.
**Use Case**: Translator obtains text of web page, performs translation, and uploads newly translated page.
**Outcome**: Page becomes available in the new language version.

**Title:** **Translate a page on a dynamic web site**
**Background**: Dynamic web sites are published using a content management system (CMS). A CMS can automate the task of maintaining a site in multiple languages, and some CMS platforms provide decent features in this regard.
**Use Case**: Translator logs into CMS and selects page for translation, indicating target language for translation. A new copy of the page data is created to be converted, and when complete, the translation is submitted to the CMS and the page indicates available translations when viewed.
**Outcome**: Page becomes available in the new language version

**Title:** **Translate a blog post**
**Background**: A web site has bloggers from around the world posting content which needs to be viewed in more than one language.
**Use Case**: A translator logs into the blog and views a list of pages with translation requests. After selecting an article to translate, they enter into a translation interface, allowing them to translate the blog entry. They click "submit" when finished with the translation, and the entry is placed in a queue for approval.
**Outcome**: Blog entry is translated, and ready for approval.

**Title:** **Translate a blog comment**
**Background**: Translation of blog posts means that speakers of multiple languages may wish to read comments on the post, or comment themselves.
**Use Case**: When a comment is posted to a blog, readers make requests for translations into their language. A translator answers an RSS request for a comment to be translated, and posts the translated text to the

appropriate location on the blog.
**Outcome**: Comment becomes available in requested language.

**Title:** **Translate an RSS feed**
**Background**: RSS feeds notify subscribers of new content available on an associated web site, and include title and summary for each new page.
**Use Case**: Translator aggregates RSS items from a feed, translating the title and summary, and then re-syndicating each translated item in a parallel feed for the target language.
**Outcome**: RSS feed items are available in new language version.

**Title:** **Translate a wiki page**
**Background**: Wikis allow site visitors to edit all site pages. This presents a dual translation challenge: not only must the page be translated when it is created, but changes to the page must also be tracked and translated.
**Use Case**: After a wiki page is created, a translator answers a request for translation of that page to another language. The content is translated and posted in a new page on the wiki. In addition, the translator subscribes to the RSS feed for the page and tracks changes in the source page on the target page.
**Outcome**: The page becomes available in the requested language, and stays current with the original.

**Title:** **Translate wiki user comments**
**Background**: Most wikis enable discussion regarding the content of each wiki page, where users can collaborate on as well as debate the corresponding content.
**Use Case**: Users debating some particular page content use the discussion tab on that page to exchange their views on what it should say. Comments are translated semi-synchronously to other languages to facilitate a "live" discussion between people without a common language.
**Outcome**: More inclusive debate allows a broader range of contributors to weigh in on the page content.

### Multimedia Translation Use Cases

There is a fundamental difference between translating text content and translating audio, video or graphics. Different skills, tools and processes are required. The following are four canonical use cases for multimedia translation.

**Title:** **Transcribe and subtitle a video**
**Background**: Videos are, by their nature, available in the language(s) of the filmed participants. While voice-over is one option for making such content available to different audiences, a more common and cost effective mechanism is to sub-title the dialog in additional languages.
**Use Case**: Translator watches video, transcribes dialog, and translates. Translated

dialog is then recombined with original video to create subtitled video.

**Outcome**: Video made available in additional language via subtitles.

**Title:** **Translate an audio file**

**Background**: Audio files are ubiquitous assets on web sites, games, and other interactive media.

**Use Case**: Translator transcribes audio into text and translates into target language. Additionally, audio may be re-recorded in target language.

**Outcome**: Transcript is then available in target language in parallel with audio asset.

**Title:** **Translate a graphic**

**Background**: Graphics present a unique challenge in translation. In addition to translating embedded text, translators must also consider whether imagery and iconography are culturally specific. For example, a yellow triangular "warning" sign from U.S.-oriented automotive imagery would need to be converted to resemble a corresponding warning sign in the target locale. Certain graphic formats, such as SVG, are more translation friendly, as they store each element as separate objects, and text strings can be extracted.

**Use Case**: Translator opens graphic in appropriate editor application. Text items are translated, icons and other imagery are localized. For bitmap images, graphic may need to be recreated from scratch in order to properly overlay text or convert necessary iconography.

**Outcome**: Graphic is available and makes sense in context of target locale.

**Title:** **Translate a slide-based presentation**

**Background**: Projected slide presentations are a combination of text, graphics, and potentially audio/video.

**Use Case**: Translator translates text, graphics, and multimedia into target language, following the appropriate process for each type of asset. Assets in target language are reassembled into slide presentation in target language.

**Outcome**: Presentation is available in the new language version.

## Translation Workflow Use Cases

Translation workflow is the act of stringing together smaller translation use cases into manageable processes. For example, translating a web site is the aggregate outcome of translating individual pages. Translation processes involve not just the translation task, but also proofreading and quality control, personnel management, and version tracking, among other considerations. And managing requests for translation is another process altogether. The following are a representative sampling of translation workflows; many are large-grain, describing process which involve substantial work and management.

**Title**: **Managing translation of a web site**

**Background:** Translating a web-site is a many-faceted process, involving different types of assets and challenging integration tasks.

**Use Case**: A web manager needs to convert a web site from its original language to a second language and deploy it as a separate site. To do this they must inventory the pages and assets needing translation, assign translators for each one, proofread and quality check submitted translations, and integrate them together to launch the second language version of the site.

**Outcome**: Web site is available in target language.

**Title:** **Blog publisher requests volunteer translation of new posts**

**Background**: Many blog authors fundamentally lack the resources or know-how to publish in more than one language, but they still desire to see their content available in other languages.

**Use Case**: A blog publisher places a “Request translation of this page” widget on their site, appearing on each page/post. When a site visitor clicks on the link in the widget, they are able to indicate the target language they desire. This submits a translation request into a central request queue, where volunteer translators can indicate pages they are willing to translate. The volunteer translates the page, and uses the request system to submit the translation and make sure the original publisher and translation requester are notified.

**Outcome**: Page is available in target language from original site, and the volunteer translator is acknowledged.

**Title:** **Distributed translation of document**

**Background**: Translation of large documents often requires multiple translators, as well other other role players.

**Use Case**: A large document needs to be translated from Croatian to Chinese. The project manager segments the document into smaller chunks, and then uploads those to a site to be accessed by other project members. Translators are assigned segments to translate, and as each segment is submitted, the manager can have them proofread. When all segments have completed the translation process, the project manager is able to reassemble the segments to for the target document.

**Outcome**: The document is available in the target language.

**Title:** **Distributed reviewing of a document**

**Background**: Much translation work flow is managed via email; there are very few open source translation tools that enable management of a distributed translation team.

**Use Case**: A translator can post a translated document for review. Reviewers can comment on individual sentences and terms to provide ideas for alternate translations, marking regional varieties or even asking the original author what they mean by something.

**Outcome**: The translator enjoys rich review support by tapping a distributed

network of knowledge.

**Title:** **Translation quality control**
**Background**: While most translation processes include proofreading and editing, it is still difficult to track individual translation issues within a document, content set, or translation project.
**Use Case**: A project manager can solicit and track user feedback on defined aspects (quality, consistency, regional variation, technical uncertainty) for each submitted translation, both to verify when the translation is "ready" as well as to assess the talent of the translator. In addition, issues within translations can be specifically identified and tracked to resolution.
**Outcome**: Overall document quality improves because issues do not get lost and better translators are identified through the tracking process.

**Title:** **Track changes to translated content (version control)**
**Background**: For translations which must be maintained over time, it is important to maintain translation history for content asset, storing each target language and versions in both source and target languages.
**Use Case**: A translator is assigned a document to translate. They check that document to be translated out of a document control system, and translate the document. When they are finished, they can check back in both the source and target documents. Subsequent modifications to both source and target documents are stored as new and separate versions in the document control system.
**Outcome**: The project manager has a very fine-grained ability to compare versions over time. In addition, when managing multiple translations for a document, the version control can help to ensure that all target versions are roughly similar.

**Title:** **Democratization of translation requests**
**Background**: Translation resources are scarce, so not every request for translation, either volunteer or paid, is met. It is often challenging to decide which items should be translated when not all items are able to be translated.
**Use Case**: A site administrator enables user-driven selection of pages for translation. Translation requests for pages on a web site are aggregated, and users are invited to vote on which pages they would like to see translated. In addition, users have the ability to rate the value of pages, and those ratings help to determine which articles should be translated and to what languages.
**Outcome**: Likelihood for translating most useful set of pages is increased and driven by community. Site administrators are relieved of the burden of deciding which pages to translate.

**Title:** **Online community directory of translators**
**Background**: It is often difficult to find someone with the appropriate translation skills

for a given project. While private and proprietary directories exist, there are very few open resources for finding in-person or virtual translation support.

**Use Case**: A user needing translation of a specific document can visit the directory and indicate the type of document to be translated, including any associated specializations, their geographic location, as well as source and target languages. The system's search feature responds with a list of available in-person and virtual translators who may be able to provide service. In addition, each translator can indicate their compensation requirements and indicate their specialties and strengths, and network with other translators with similar or complementary skills, background and interests.

**Outcome**: The user is able to find an appropriate translator for the document.

**Title**: **Accounting support for translation system**

**Background**: Especially when working with a distributed network of translators, it is difficult to track who has put in how much time on each translation task. In addition, different roles in the translation process may be compensated at different rates.

**Use Case**: A project manager assigns translation of content assets to specific translators using a translation workflow system. As each translated asset is submitted, the translator indicates the amount of time they spent on the translation. In addition, editors, proofreaders, commentors and managers are able to indicate how much time they have allocated to each translation project or asset.

**Outcome**: The project manager is able to generate summary reports which can be used to calculate compensation as well as assess which translators have better performance.

**Title:** **Pledge bank system for translations**

**Background**: Funding the translation of open content is often problematic, because it is not usually institutionally driven. Attempts to raise donations to fund translation are problematic because people do not want to donate unless they believe the full text will be translated, such as a book.

**Use Case**: Users could post documents requiring translation, and translators could then post estimated costs for translating each document. Parties interested in seeing the document translated could then pledge monies they would contribute if the document was actually translated. The document would only be translated and pledges collected once the pledge total reached the projected translation cost.

**Outcome**: Funding options for translation are diversified, and users are more likely to succeed in getting their text translated.

## Machine Translation Use Cases

**Title:** **Hosted machine translation**
**Background**: One of the most common translation needs is a quick-and-dirty rendering of a text passage in a target language. This is often using machine translation software.
**Use Case**: A user navigates to a machine translation service site. By first selecting source and target languages, and then pasting text or entering a URL, they are able to obtain a machine-generated rendering of the text in the target language.
**Outcome**: The user can assess the content to see if it merits further study or higher quality translation.

**Title:** **Shared translation memory**
**Background**: A translation needs access to a shared translation memory, including domain specific languages and possibly translation memories in similar languages.
**Use Case**: While doing a translation, a translator is able to go online to a shared translation memory web site. By first selecting source and target languages, and then uploading a document, they are able to obtain a translation in the target language of all the sentences which were found in the translation memory. For unmatched sentences, they can refer to translation memories in similar languages. They are then able to translate the passages for which there was no match, and then upload those text pairs to the shared translation memory.
**Outcome**: Bulk of translation is completed with TM assistance, and remainder is translated and the new results shared for use by others.

### Interpreting

Interpreting differs from translation by virtue of its real-time nature; it is the act of simultaneously translating while someone speaks and others listen. It is considered the most challenging of the translation arts.

**Title:** **Interpret a live audio stream**
**Background**: As more content is streamed live over the Internet, there is a corresponding increase in demand for real-time interpretation.
**Use Case**: A conference keynote is being streamed over the internet. An interpreter listening to the speaker transcribes the speech into text and publishes it in an IRC channel as well as a live blog.
**Outcome**: Speakers of the transcribed language are able to follow the keynote.

**Title:** **Translate a slide-based presentation**
**Background**: Presentations are a combination of text, graphics, and potentially audio/video. They present a dual translation challenge in that ideally, both the document and the narrative can be rendered in the target language.
**Use Case**: A Japanese person with limited English listening skills is attending a live event and has problems understanding the slide presentation. Another

participant “liveblogs” in English and the Japanese participant is able to follow better by reading the notes. Additionally, another Japanese-speaking person provides live "subtitles" into Japanese using IRC and a translation memory to increase speed.

**Outcome**: Non-English listener is able to understand English-language slideware presentation.

### Content Rendering and Presentation

While much of the challenge in translating open content lies in the translation task itself, there are also considerations with the presentation of the translated content. Translated content may require different styling and layout to account for differing amounts of textual data, such as when English is translated to German, which tends to be a more verbose language. In addition, text orientation and flow are locale-dependent, and translation tools need to properly render translated content in the target language/locale pair.

**Title:** **Support multi-lingual CSS**

**Background**: Most modern web sites, and all open source content management systems and blogs, make use of Cascading Style Sheets (CSS) to describe graphic design attributes for web pages, including fonts, colors, graphics, layout, and decoration. But CSS settings, especially font and layout information, will vary from language to language.

**Use Case**: A web site publisher publishes a multi-lingual site, which displays different pages in different languages. As part of the platform architecture, they are able to specify a different CSS style set for each language, and the CMS automatically selects the correct CSS for a page based on the language encoding of that page.

**Outcome**: Web site renders translated content correctly for target locale.

**Title:** **Complete directional text support (right to left, bottom to top)**

**Background**: A range of languages, especially Asian and Arabic, are read from right to left, and sometimes from bottom to top. Most platforms and tools have limited ability to offer complete directional text support.

**Use Case**: When installing a web CMS, an administrator sets a locale code while configuring the system, in order to indicate the language and locale the site plans to publish for. The platform makes according adjustments to page layout and rendering in order to fully support the directionality of the selected language.

**Outcome**: Site administrators can manage web sites with differing text directionality.

### Localization

Localization refers to the process of converting a piece of technology, such as a software program, to be available in new language or locale. While OTT07 focused primarily on translation of pure content such as documents and web sites, localization was frequently

discussed, and so a localization use case is included in this inventory for completeness.

| | |
|---|---|
| **Title:** | **Localize a software application** |
| **Background**: | Well-written software applications store the text strings displayed in their user interface in a separate file or files sometimes known as “string tables”. (Less well-written apps embed the interface text directly in the application code). A particular challenge of software localization is the mundane but never-simple issue of layout; text that fit properly in the menus, messages, and dialog boxes of the original application may be too long to fit properly in the translated version. |
| **Use Case**: | An application localizer takes the PO file for the program and translates each string. The new string data is then recombined with the original application and tested for correctness of content and display. |
| **Outcome**: | Software application is available for use in target language. |

## Appendix II: Open Translation Tools

As described in the Open Translation Tools section earlier in this paper, open translation tools fall into a range of categories.

- **PO and XLIFF localization editors**: This encompasses offline, online and distributed localization tools that read and write data in PO, XLIFF and related formats. These serve as the essential tools for many translators and localizers.

- **Translation workflow**: These tools manage roles, tasks and other project information, and often interoperate with other translation tools and version control systems. Workflow is a critical area for open translation, and there are a range of un-met needs in terms of workflow support.

- **Machine translation**: These tools, which at present are primarily hosted as web sites like translate.google.com and BabelFish, perform algorithmic translation of text from one language to another.

- **Translation memory**: These Computer Aided Translation (CAT) tools store small discrete language fragments, passages, and terms in order to assist human translators as the perform their work.

- **Dictionary, glossary and spell checking**: As their names imply, these CAT tools store definitions for terms in a given language, and support translators as they map from one language to another.

- **Wiki translation**: The modules and extensions which enhance and augment existing wiki platforms with tools for performing and managing translation of wiki content.

- **Subtitling**: As video becomes a more pervasive web offering, tools for adding translated subtitles to videos are ever more in demand.

The rest of this appendix enumerates the open translation tools in each category. As with almost any collection of software tools, these categories blur and overlap on a tool-by-tool basis; the categories are somewhat arbitrary and many tools fall into more than one.

### PO and XLIFF Localization

The category “localization tools” encompasses a broad and diverse collection of applications that manage translation data. It is in this tool space that the concepts of “localization” and “translation” blur; most practitioners associate the term localization with the act of converting software or other technology to operate in a different language or locale by translating the text associated with the user interface. Translations, on the other hand, generally refer to conversions of documents and other text content from one language or locale to another. Many “Localization tools” have features to support both types of lingual conversion.

Localization tools store their data in file formats such as PO and XLIFF. Most localization tools are client applications, but some, like Pootle, provide web-based functionality.

**Translate Toolkit**: The Translate Toolkit (http://translate.sourceforge.net/wiki/toolkit/index) is a toolkit designed to make it easier for localizers to work with various formats while helping in parallel to increase quality. It works with XLIFF and PO as its primary formats and can convert many other formats to these. The toolkit has a number of QA related tools that can perform more than 40 checks on the translations. It also has functionality around Translation Memory management and glossary management. The toolkit is continually being developed and forms the basis of a number of other tools including Pootle (an online translation tool) and WordForge (an offline translation tool). The number of supported formats is steadily increasing over time.

**Pootle**: Pootle (http://translate.sourceforge.net/wiki/pootle/index) is a web-based translation tool that allows one to manage PO or XLIFF translations through a web interface. This enables easier community participation. Pootle supports a range of translation workflow tasks, including assigning rights to different users, defining project goals, checking translations, allowing ad hoc contributions, committing translations back to a version control system, and user terminology and translation memory matching. Pootle is being adopted by a number of localization projects including OpenOffice.org, Creative Commons and others.

**Wordforge**: The Wordforge Off-line Localization Editor (http://sourceforge.net/projects/wordforge2), previously known as Pootling, is an intelligent, platform-independent offline localization tool developed specifically to allow translators to get the most out of the XLIFF file format. The application includes a catalog manager, translation memory manager, and glossary manager. The current 0.5 version supports spell-checking, use of third languages as reference, format conversion to and from XLIFF (including PO, TBX, TMX), good SVN merge support, and the ability to verify the quality of translation immediately upon translation of each string. Version 1.0 of WordForge will also support localization workflow management, including assignment of different roles to users, and maintaining translation state information in each XLIFF. By flagging which strings need to be translated, reviewed or approved, these features will be designed to reduce the amount of work in the review and approval stages.

**Poedit**: Poedit (http://www.poedit.net/) is cross-platform gettext catalog (.po file) editor. It is built with the wxWidgets toolkit and can run on any platform supported by that toolkit. It aims to provide more convenient approach to editing catalogs than launching vi and editing the file by hand.

**KBabel**: KBabel (http://kbabel.kde.org/) is a set of tools for editing and managing gettext PO files. The main feature is a powerful and comfortable PO file editor which includes full navigation capabilities, full editing functionality, ability to search for translations in different dictionaries, spell and syntax checking, display of diffs and much more. Also included is a "Catalog Manager", a file manager view which helps maintain an overview of PO files. It also includes a standalone dictionary application to access KBabel's powerful dictionaries. The platform is designed to enable fast translation and while keeping translations consistent.

**gtranslator**: gtranslator (http://gtranslator.sourceforge.net/) is a GNOME2 application intended to make editing PO files easy for language translators.

**gedit – pomode**: gedit-pomode (http://sourceforge.net/projects/gedit-pomode/) is a plugin for convenient editing of PO files in the gedit text editor.

**Ini Translator**: Ini Translator (http://initranslator.sourceforge.net/) is a utility program to translate ini-style language files, with a look and feel reminiscent of poEdit.

**Translation workflow**

**World Wide Lexicon**: The World Wide Lexicon (WWL, http://www.worldwidelexicon.org/) is an open source project that enables web publishers to translate their content into any language, via volunteer and paid translators. WWL is designed to be embedded in a wide range of publishing and web service environments. One of the key goals of WWL is to enable collaborative translation for any website that wants to use it. WWL also publishes an API, and they are working on an in situ localization library (SLS : Simple Localization System).

**Transifex**: Transifex (https://fedorahosted.org/transifex/) is a novel system designed to ease the process of contributing translations to projects hosted on various remote and disparate version control systems (VCS). It acts as a proxy between the translator and the project maintainer, making the work of both more efficient. By abstracting the VCS to a common, easy to use interface, it makes the submission process to remote projects easier and straightforward. At the same time, by acting as a translation gateway to remotely hosted resources, developers are enabled to reach out to already established translation communities. In contrast to similar systems, Transifex does not require the source code to be relocated or the translation files to be copied to a downstream VCS. Therefore, translation merging is not needed, and all downstream projects benefit equally. Transifex is already in use by the Fedora Localization Project, counting more than 2000 translators.

**Launchpad Translations:** Launchpad Translations (formerly "Rosetta") is a platform for open source application translation on the internet. It lets anyone help translate their favorite open source application into their favorite spoken language.

**Kartouche**: Kartouche (http://www.dotmon.com/kartouche/) is a web-based translation tool - it allows translations to be submitted via a browser-based interface. Kartouche's sister application, Omnivore, stores the completed translations in a searchable store to which comments and corrections can be added.

**Kyfieithu**: Kyfieithu (http://www.kyfieithu.co.uk/index.php?lg=en&) is a web-based Welsh translation workflow tool. It is based on Kartouche.

**Vertimus**: Vertimus (https://launchpad.net/vertimus) is an open source web tool for managing workflow for translations. Each translation has a status which changes at each step, and users can create an account, book any translation and communicate with a team. An example

of Vertimus in action can be found at http://gnomefr.traduc.org/suivi. The first release provides support for localization between English and French.

**Project Open**: Project Open (http://www.project-open.com/) is a general-purpose project management system with a translation module. The source license is a GPL hybrid, and the translation module is FL (“Free License”), which is “pseudo FOSS”, with extensive limits on redistribution of code.

### Machine Translation

**Apertium**: Apertium is a machine translation platform, initially aimed at related-language pairs, but recently expanded to deal with more divergent language pairs (such as English-Catalan). The platform provides a language-independent machine translation engine, tools to manage the linguistic data necessary to build a machine translation system for a given language pair, and linguistic data for 13 language pairs, with more under development. All the tools are written in C++, with dictionary and rule specification formats in XML. Specification formats are compiled into efficient binary representations. On an average machine the system can process around 4,000 words per second. For document file formats, Apertium supports plain text, HTML, ODT, and SXW.

**Moses**: Moses (http://www.statmt.org/moses/) is a statistical machine translation system that allows one to automatically train translation models from very large bilingual aligned parallel corpora for any language pair.

**OpenLogos**: OpenLogos (http://logos-os.dfki.de/) is the open source version of LOGOS.

**Traduki**: Traduki (http://traduki.sourceforge.net/) is a suite of open source linguistic software. Originally intended to be "just" a machine translation software, Traduki got its author so involved that it eventually grew into a much larger scale project.

### Translation Memory

**OmegaT**: OmegaT (http://www.omegat.org/) is a translation memory application written in Java. It is a tool intended for professional translators. Features include the ability to use multiple translation memories and external glossaries, with rich support for different file formats and Unicode (UTF-8), and compatibility with other translation memory applications (TMX).

**Transolution**: Transolution is a Computer Aided Translation (CAT) suite supporting the XLIFF standard. It provides the open source community with features and concepts that have been used by commercial offerings for years to improve translation efficiency and quality. The suite is modular to make it flexible and provides a XLIFF Editor, translation memory engine and filters to convert different formats to and from XLIFF. The use of XLIFF means that almost any content can be localized as long as there is a filter for it (XML, SGML, PO, RTF, StarOffice/OpenOffice).

**QT Linguist**: Qt Linguist (http://trolltech.com/products/qt/features/tools/linguist) is a tool for adding translations to Qt applications.

**Dictionary, Glossary and Spell-Checking**

**CollaboDict**: CollaboDict (http://www.terminologija.org.mk/) is a web application for collaborative creation of dictionaries. Open or closed groups of users can create a dictionary project and work together in a collaborative and democratic way. An important feature of CollaboDict is that it promotes open content, in the sense that the projects can be created only under a Creative Commons license. This guarantees that the content created by way of CollaboDict will always contribute to the common good, while protecting some rights of the authors.

**dict.org**: Dict.org (http://www.dict.org/bin/Dict) is a hosted dictionary. It is based on the GPL software GCIDE.

**WordNet**: WordNet (http://wordnet.princeton.edu/) is a large lexical database of English, developed under the direction of George A. Miller. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser. WordNet is also freely and publicly available for download. WordNet's structure makes it a useful tool for computational linguistics and natural language processing.

**Wiktionary**: Wiktionary (http://en.wiktionary.org/wiki/Main_Page) is a collaborative project to produce a free-content multilingual dictionary. Designed as the lexical companion to Wikipedia, the encyclopedia project, Wiktionary has grown beyond a standard dictionary and now includes a thesaurus, a rhyme guide, phrase books, language statistics and extensive appendices. The goal is to include not only the definition of a word, but also enough information to really understand it. Thus etymologies, pronunciations, sample quotations, synonyms, antonyms and translations are included.

**Fantasdic**: Fantasdic (http://www.gnome.org/projects/fantasdic/) is a DICT client. It is multi-platform and written in the Ruby programming language. It retrieves definitions from the internet via the DICT protocol instead of from a file on the local hard drive.

**GNOME Dictionary**: GNOME Dictionary (http://live.gnome.org/GnomeUtils) is a DICT client written in C by Emmanuele Bassi and others. It is part of the open-source GNOME desktop software suite, inside the gnome-utils meta-package. It allows users of GNOME to look up words on dictionary sources.

**jDictionary**: jDictionary (http://jdictionary.sourceforge.net/) has an intuitive user interface and is able to upgrade itself, upgrade its plugins, and provide news and information about new plugins.

**GNU Aspell**: GNU Aspell (http://aspell.net/) is a Free and Open Source spell checker

designed to eventually replace Ispell. It can either be used as a library or as an independent spell checker. Its main feature is that it does a superior job of suggesting possible replacements for a misspelled word than just about any other spell checker out there for the English language. Unlike Ispell, Aspell can also easily check documents in UTF-8 without having to use a special dictionary. Aspell will also do its best to respect the current locale setting. Other advantages over Ispell include support for using multiple dictionaries at once and intelligently handling personal dictionaries when more than one Aspell process is open at once.

**Ispell**: Ispell (http://ficus-www.cs.ucla.edu/geoff/ispell.html) is a program that enables users to correct spelling and typographical errors in a file. When presented with a word that is not in the dictionary, Ispell attempts to find near misses that might include the word you meant.

**Hunspell**: Hunspell (http://hunspell.sourceforge.net/) is the default spell checker of OpenOffice.org and Mozilla Firefox 3 & Thunderbird.

**Lingro**: Lingro (http://lingro.com/) is an online multilingual dictionary tool. Lingro's mission is to create an on-line environment that allows anyone learning a language to quickly look up and learn the vocabulary most important to them. Lingro is not open source software, but all content on Lingro is licensed under Creative Commons.

**Wiki Translation**

**Betawiki**: Betawiki (http://translatewiki.net Betawiki) is a wiki dedicated to translating MediaWiki messages, its extensions, FreeCol and other projects. It supports export to .PO for offline translation of most used messages and core messages. Betawiki editors contribute to more than seventy languages every month and this number increases by five to ten languages every month.

**OmegaWiki**: OmegaWiki (http://omegawiki.org) allows people to work on both lexical, terminological and ontological data in multiple languages. The OmegaWiki software is an extension to MediaWiki, adding relational functionality to the core wiki platform. As the data is relational, it is possible to localise the complete user interface and it can be selected in the user preferences.

**Cross Lingual Wiki Engine**: The Cross Lingual Wiki Engine project (http://www.wiki-translation.com/tiki-index.php?page=Cross+Lingual+Wiki+Engine+Project&bl=y) aims to design, develop and test lightweight wiki tools that can be used to translate content in wikis. At present the project is implemented as part of TikiWiki.

**Subtitling**

**GNOME Subtitles**: GNOME Subtitles (http://gnome-subtitles.sourceforge.net/) is a subtitle editor for the GNOME desktop. It supports the most common text-based subtitle formats and allows for subtitle editing, translation and synchronization.

**Subtitle Editor**: Subtitle Editor (http://kitone.free.fr/subtitleeditor/) is a GTK+2 tool to edit subtitles for GNU/Linux/*BSD. It can be used for new subtitles or as a tool to transform, edit, correct and refine existing subtitles. The program also displays sound waves, making it easier to synchronize subtitles to voices. Subtitle Editor is free software released under the GNU General Public License (GPL3).

**dotSUB**: dotSUB (http://dotsub.com/) is a browser-based tool enabling subtitling of videos on the web into and from any language. It is not open source, but supports a rich open data model.

**Code Libraries and Packages**

The **Open Translation Engine** (OTE, http://ote.2meta.com/) is an open source project developing language translation and dictionary tools for the internet community. The prototype system currently supports Dutch to English translations. The OTE is written in PHP and uses a MySQL database.

**Okapi**: The Okapi Framework (http://okapi.sf.net/) is a set of interface specifications, format definitions, components and applications that provides an environment to build interoperable tools for the different steps of the translation and localization process. The goal of the Okapi Framework is to allow tools developers and localizers to build new localization processes or enhance existing ones to best meet their needs, while preserving a level of compatibility and interoperability. It also provides them with a way to share (and re-use) components across different solutions. The project uses and promotes open standards, where they exist. For the aspects where open standards are not defined yet, the framework offers its own. The ultimate goal is to adopt the industry standards when they are defined and usable.

**GNU 'gettext'** (http://www.gnu.org/software/gettext/) offers programmers and translators a well integrated set of tools and documentation that provide a framework to help other GNU packages produce multi-lingual messages. These tools include a set of conventions about how programs should be written to support message catalogs, a directory and file naming organization for the message catalogs themselves, a runtime library supporting the retrieval of translated messages, and a few stand-alone programs to massage in various ways the sets of translatable strings, or already translated strings. A special GNU Emacs mode also helps interested parties in preparing these sets, or bringing them up to date.

**php-gettext** (http://us.php.net/gettext) is a PHP-based emulator for the gettext API.

**xml2po** (http://cvs.gnome.org/viewcvs/gnome-doc-utils/xml2po/) is a simple Python program which extracts translatable content from free-form XML documents and outputs gettext compatible POT files. Translated PO files can be turned into XML output again.

**poxml** (http://packages.debian.org/poxml) is a collection of tools that facilitate translating DocBook XML files using gettext message files (PO-files). Also included are some miscellaneous command-line utilities for manipulating DocBook XML files and PO-files. This package is part of the KDE Software Development Kit.

**intltool** (http://www.linuxfromscratch.org/blfs/view/stable/general/intltool.html) is useful for extracting translatable strings from source files, collecting the extracted strings with messages from traditional source files, and merging the translations into .xml, .desktop and .oaf files.

**Translate Toolkit** also provides library interfaces for other tools wanting to utilize its features.