On Beyond TM: When the Translator Leads the Design of a Translation Support Framework

Reginald L. Hobbs
Army Research Laboratory
2800 Powder Mill Road
Adelphi, MD 20783
hobbs@arl.army.mil

Clare R. Voss
Army Research Laboratory
2800 Powder Mill Road
Adelphi, MD 20783
voss@arl.army.mil

Jamal Laoudi
ARTI,Inc
1555 King Street Suite 400
Alexandria, VA 22314
jlaoudi@arl.army.mil

Abstract

Commercial off-the-shelf machine translation engines and translation support tools, such as translation memory (TM), have been developed primarily for translating grammatically well-formed, edited text. The real-world, foreign language (FL) document collections that our translators work with consist instead of noisy and complex image files. We are currently conducting experiments that involve building and evaluating the effectiveness of different multi-component workflows for the automated processing and translation of these FL images into English text. To support the project's ongoing needs for translations, we are developing a software framework designed *with* and *for* our translator that (i) streamlines users' access to and capability to add-in and modify existing online tools and data resources (ex. MT, TM, dictionaries, morphological analyzers, LM), (ii) builds persistent data objects for later re-use, and (iii) provides users with a *configuration screen page* to select the tools and data resources for their sessions, to set their tools' options and display options for their *translation screen page.* This paper introduces our extreme programming approach to the software engineering of this new, hands-on translator's framework called *TREAT* (Toolkit and Resource Environment to Assist Translation), where the translator---as subject matter expert and experienced software user---participates fully in the software design, evaluation, and iterative modification processes.

1 Introduction

The Army Research Laboratory conducts experiments in enhancing, constructing, and evaluating MT engines and resources. The evaluation side of our experiments requires "ground truth" (GT) character files of the FL text and "reference translations" (RT) of the English text. In observing our team's translator translate FL texts into RTs, we noticed the range of translation-support resources and methods that he applied to the tasks. Given his experience with semi-automated and fully automated MT metrics and his prior involvement in our design of MT user-support and evaluation tools, we decided to leverage his knowledge and involve him directly in the creation of a toolkit that would assist him as well as other translators on our project. While translators may rely on TM and other tools to help with productivity and consistency, they lack a user-centered software framework for assembling different categories of translation tools and customizing their tools and translation screen.

In this paper, we begin with an overview of our software engineering approach and principles for designing a translator's toolkit framework and then present the categories of tools in use by our translator for inclusion in the framework. We then review an example use of existing resources in a translation task and conclude the paper with an introduction to the TREAT framework that we have implemented and continue to refine.

2 Design Approach

The design approach for creating TREAT combines two separate, but complementary software engineering paradigms: *extreme programming* and

user-centered design. This combination allows us to leverage our translator's experience on different translation tasks and in using different translation software. As depicted in **Figure 1**, we distinguish three stakeholder roles in creating such a framework: lead builder (developer), lead designer, and user. In small-scale software development projects, the builder and the designer may be the same person and typically the builder/designer does not have the subject matter expertise of the intended users. As a result, their understanding of the users' tasks and the users' software needs is several steps removed from the user, even when an extensive requirements engineering process is undertaken to document the user's needs.

By choosing an expert translator as lead designer of the toolkit framework, we quickly hear back from the "user" when the implementation of design desiderata by a developer does not support the intended use case. That is, the expert conducts both the *capability evaluation* (validating that the software "does" its share, for example, providing correct data and analyses to support specific tasks) and the *usability evaluation* (determining that the user can make use of the software features to effectively complete specific tasks).

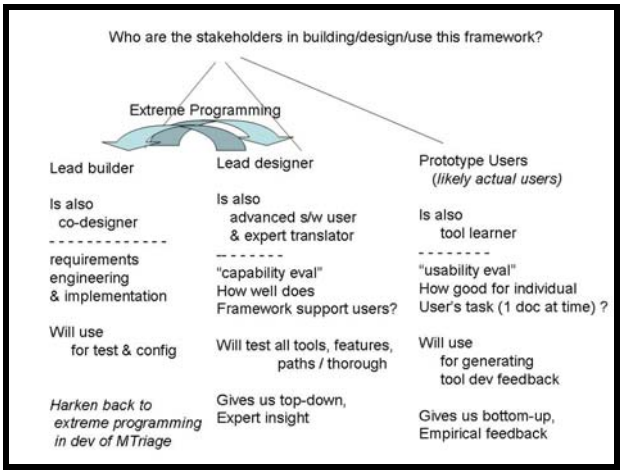

Figure 1 User-centered Design

2.1 Extreme Programming & User-Centered Design

Extreme Programming (XP) is the concept of creating software very quickly through the use of rapid prototyping and incremental development. In *Extreme Programming Explained*, XP is defined as "a light-weight methodology for small- to medium-teams developing software in the face of vague or rapidly changing conditions". (Beck, 1999) The XP software engineering paradigm has been described by six principles: 1) embrace change, 2) keep the design simple, 3) minimize investment upfront, 4) support incremental change, 5) create functional releases of the software and 6) continually re-design based on user feedback.

Extreme programming shortens application development time by compressing the life-cycle and sharing design, coding, and testing between two or more software developers. Our work adheres to these principles both (i) with the choice of an interpreted programming language (as opposed to compiled), such as JavaScript, Java, or Perl, to create rapidly revisable, functional software prototypes with web pages as the user interface and (ii) with user feedback incrementally guiding which features to add, modify, or delete as the GUI for framework software evolves.

User-centered design seeks to involve the end-user in all aspects of the software life-cycle, from initial concept through operational fielding, including the maintenance of a system through user feedback and change requests. Variations of user-centered design, such as *participatory design* and *contextual design*, are all methods for building software that maps to user expectations and needs.

With the lead design/expert translator in the role of the end user, we identified several principles to follow in the development of this framework:

- Keep it simple/short: keep all user-tool interactions as short and intuitive as possible
- Keep it simple/uncluttered: keep all user-views (GUIs) as open and uncovered as possible, ex. make sure tools and resources are non-intrusive: the tools should not get in the way of the translator
- Put the user in charge: enable user to hide or show tools, making them accessible if requested but suppressed if necessary.
- Support the translator-in-training: provide users with easy access to guide and hints.
- Provide extensibility for incorporating new tools: enable user to easily add new tools and expand framework's capabilities
- Support customization by incorporating new resources: enable experienced users to

create their own data repositories and to import data of others

2.2 Progression of User Abilities

Users do not remain at the same level of performance as they interact and gain familiarity with a software application over time. By definition, an experienced user is someone who has performed a task enough times to have internalized the underlying rules or heuristics for effectiveness. We distinguish two dimensions of "experience" among our intended framework users: experience at translation tasks and experience using software tools.

There are translators who are task experts, but have little or no software tool support. There are translators-in-training who are novices at the task, but have been exposed to numerous tools. Our translator is an expert along both dimensions. There are also those translators that would fall in the middle in terms of using software and ability to translate foreign language texts. For example, non-native speakers of English who were educated in a foreign language may be expert translators into their language, but not into English.

The framework currently supports four levels of users: beginning, learning, intermediate, and advanced.

- Beginning user: Translates using default tools and tool settings "As-is".
- Learning user: Makes choices and selects appropriate tool settings for using available tools.
- Intermediate user: Customizes framework by adding their own data resources or settings to unique tool configurations.
- Advanced user: Adds new tools to the framework to solve more complicated tasks.

3 Categories of Tools in the Framework

There are many categories of tools at different levels of maturity and capability that may provide support in translating texts in TREAT. In the following sections, we describe these general categories and provide examples of available commercial or open source tools and data resources that can be included or linked to from within TREAT.

3.1 MT Engines and Multi-MT Tools

MT engines are now available for fully automated and customized translation of digital texts. They vary widely in underlying construction methods (dictionary-based, symbolic rule-based, example-based, phrase-based, syntax-based) and produce output translations that differ significantly in accuracy and fluency. Although MT engines will fail in domains that they were not built or trained to translate when compared against the translation quality of human translators, there are tasks (such as document triage and filtering) that do not require full publication-quality, high-fluency in the target language. Studies now document how MT engines can be used effectively to assist translators on these tasks. (Bonet, 2009)

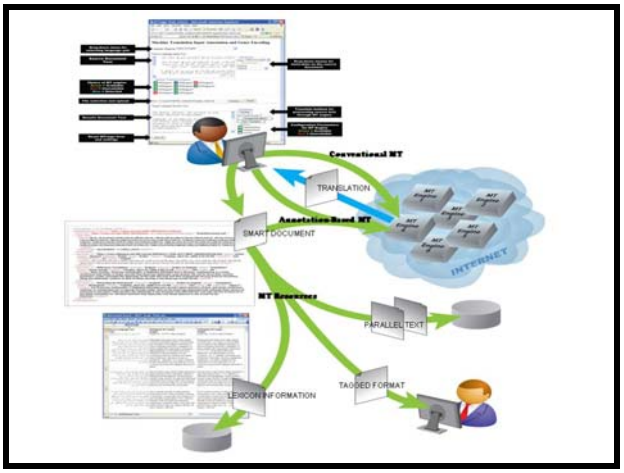

Figure 2 Web client access to multiple MT engines

Our research needs in building and evaluating multiple MT engines provided the catalyst for the development of MTriage, a front-end tool that communicates with several MT engines. (Hobbs et. al., 2008) The web client for MTriage hides the details of configuration settings and transactions with individual MT engines from the users, enabling them to focus on their translation task by eliminating their need to learn the details of a separate GUI for each engine.

Figure 2 depicts the implicit workflows in MTriage available to the translator using the front-end client to access remote MT web services. The design of MTriage further supports the user in a range of research-related tasks---such as, aligning and storing reference translations, generating multiple parallel corpora, deriving bilingual lexicon

entries for updating MT engines---by automatically handling the formatting, organization, and display of the intermediate translation artifacts.

MTriage, as set up within TREAT, can make available one or more its engines (dictionary-based MT engines that use word-for-word lookup, rule-based engines that use syntactic and other linguistic information for translation, statistically-trained engines created using large amounts of parallel corpora) to improve productivity in filtering and gisting tasks.[1]

3.2 Dictionaries

Online dictionaries serve the same purpose as conventional hard-copy dictionaries: allowing the translator to look up words and review the range of their meanings, some times accompanied by exemplar sentences or usage notes. Online dictionaries may also provide conventional and alternate word spellings, syllabification, part-of-speech (POS) and other linguistic information helpful to the translator in choosing the correct words or phrases.

A translator whose native language is the source language (SL) may use monolingual target language (TL) dictionaries to validate their word choice during translation. This situation arises in situations typically when texts in "less-commonly taught languages" (LCTLs) are translated by immigrant speakers and then post-edited in a quality control phase by English native speakers. Similarly, native speakers may rely on both SL-to-TL and TL-to-SL dictionaries to ascertain which familiar SL phrases contain "true" SL words and which are idiosyncratic loan words that require alternate translations.

There are many readily available online dictionaries, such as Merriam-Webster Online and Dictionary.com, which our translators use as needed. Specialized resources, like thesauri that provide synonyms and associated terms, are also available online and as web services. WordNet, the original English lexical database developed and maintained at Princeton, displays synsets (sets of synonyms) for a given word, its context of use in a phrase or sentence, and links up and down the WordNet knowledge structure to hypernyms and hyponyms for ontologically-related terms.

As we have witnessed our current project translator's extensive use of numerous dictionaries (described further in subsection ahead on dictionary "mashups"), we designed TREAT to allow users to configure it with access to all dictionaries available to them, including any they built for their own use.

3.3 Translation Memory (TM)

Translation memory (TM) tools and their larger database-enabled TM systems enable users to import their own or others' TM data, i.e., the natural language translations already produced and encoded in format-standardized structures, text files, or databases, to build their own new TMs, and to modify and store their TMs while translating texts that require high accuracy, publication-level translations. Typically the users are also able to set the values of features via the TM application interface for the type, length, and proportion of matched text that is automatically generated during the user's translation process.

Translators vary in the degree to which they work at their individual tasks with TMs. Our translators' experience suggests that using TMs introduces a significant learning curve when the users are not computer-savvy. Nonetheless, there is general consensus that TMs can augment translators' productivity, consistency, and effectiveness in two types of cases:

- where document collections exhibit a high degree of passage replication, as occurs *within genres* such as technical users' guides, instruction manuals, public announcements, and procedural documentation
- where the document content is specific to a topic or *subject matter domain* and so the text comes from a sublanguage with a controlled vocabulary (set perhaps by a terminology management system), simplified or redacted sentence structures, conventionalized expressions or abbreviations and a high frequency of domain-specific, unambiguous phrases.

When a TM for an industry or for a domain is supplied to the translator by the client to support stan-

[1] In the last year, we have augmented MTriage to incorporate the statistically trained post-MT editors custom-built with the MOSES open-source framework, that is available under GNU General Public License at http://www.statmt.org/moses/. Since MTriage already opens new windows from its right-click option, we do not recreate this extension within TREAT for the separate web-enabled translation of web pages or smaller segments of text from documents. (per Gaspari, 2007)

dardization and consistency across a proprietary document collection, the translators may have access to it directly (such as in TMX file) or indirectly from different software interfaces, such as a web-enabled GUI or a given desktop application.

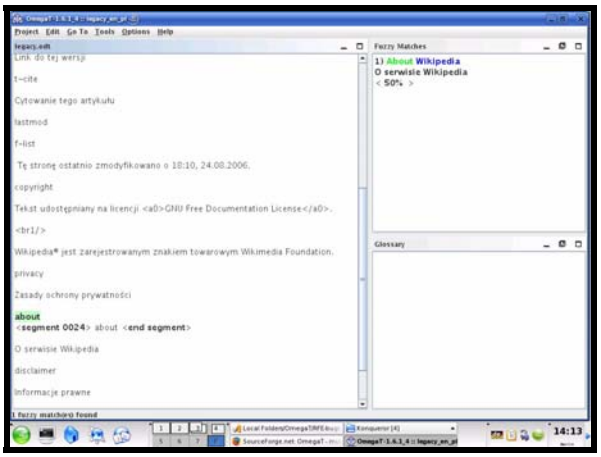

Figure 3 Omega-T Translation Memory Tool

Commercially available TM systems include SDL Trados and LingoTek. [2] Our translators have worked with Omega-T, an open-source, freely available TM tool (see screenshot in **Figure 3**). Omega-T's features include support for fuzzy (partial) mapping of phrases, the ability to handle multiple TMs simultaneously, the use of external glossaries, compatibility with TMX (Translation Memory Exchange) XML data, and UTF-8 encoding for non-Latin character sets.[3]

Given that translators will also create their own stores of their translated phrases so that their work remains internally consistent. (Smith, 2007), we have designed TREAT to allow users to configure it with access to one or more TM tools.

3.4 Transliteration Tools & Named Entity Extraction

Transliterations are mapping patterns that convert characters that are native to one script into another script. MT engines typically give their users the option of applying a built-in transliteration scheme to "not found words" (NFWs, words not found in the MT lexicon) when they cannot read the SL script, and so want to "see" what the NFW is. So for translation of an alphabetic or syllabic script into English, transliteration schemes will "romanize" or map non-Latin characters into Latin letters.

Tools, such as Basis Technology's Transliteration Assistant and Google's ta3reeb online transliteration tool [4] apply standardized transliteration schemes for Arabic-to-English translations. There are several standards for transliteration, including the Intelligence Community (IC) standard, the US Board on Geographic Names (BGN) and the Standard Arabic Technical Transliteration System (SATTS). Ad-hoc (non-standard, and often impromptu and inconsistent) transliterations appear in situations where non-Latin characters are not readily available, such as Arabic-language chatrooms, instant message boards, text messages, or some social networking sites.

Table 1 MT output for alternate name spellings

	Correct Arabic Spelling	*Alternate Arabic Spelling*
	معمر القذافي	معمر قذافي
Statistical MT	Muammar Gaddafi	Muammar Gaddafi
Rule-based MT	Moamar Al-Qadhaffi	long-lived

While transliteration may help MT users sound out and recognize "named entities" in the MT output, i.e. the proper names of persons, locations, or organizations, the existence of multiple transliteration schemes across MT engines yields multiple ways of "spelling out" those names. These variants can cause problems for post-MT software such as Named Entity Extractors (NEE)s that must discern which scheme was used in order to reconcile and standardize the spellings. **Table 1** illustrates the impact on two types of MT of variant spellings and transliterations, which in turn cascades into NEEs.

With variation in the use of diacritics in Arabic, there can be multiple correct spell-outs of Arabic names in English. A further complication arises with MT when a word in a proper name is recognized (not a NFW) and is translated literally by the lexicon, as occurs in the case of *long-lived*, as output by the rule-based MT engine.

The challenge for translators in deciding how to handle named entities is to establish one consistent transliteration scheme that they apply across-the-board in their TL texts, whether that scheme is one that they choose or one that their client has identifies in advance. TREAT supports the user in creating their own scheme or applying an available standard that they upload to the framework.

[2] Garcia (2005) writes that Trados has provided the de facto standard for over 20 years.

[3] Sourceforge application available at http://www.omegat.org/

[4] http://www.google.com/transliterate/arabic/

3.5 Morphological Analyzers

A morphological analyzer typically reads an SL input token and generates one or more analyses of that token's internal structure and content. The range and depth of linguistic information varies widely with each analyzer, from the simplest that output only one-best stemmed forms to the most extensive that output listings of all possible combinations of derivational and inflectional sub-analyses of each token, where each sub-analysis includes linguistic specifics such as lemmas, grammatical feature values, and glosses.

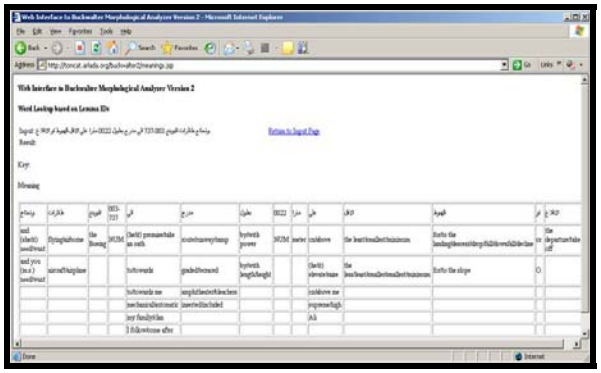

Figure 4. Buckwalter-Based Lookup Tool (BBLT)

One such extensive analyzer that is widely used among MT developers is the Buckwalter Arabic Morphological Analyzer (Buckwalter, 2004). When we discovered the English glosses for lemmas within the lexicon of this analyzer, we realized this would provide us with another way to translate Arabic tokens into English. **Figure 4** is a screenshot of the tool built to look up English glosses for fully analyzed Arabic tokens (Micher et. al, 2008).

Our project translator, a native speaker of Arabic, makes use of BBLT when he wants to see all possible token-for-token translation combinations at once in tabular form for a sentence that he is translating. TREAT has been designed so that users can import and access morphological analyzers (or their own customized versions like BBLT) as desired during translation.

4 Resource Mashup[5]

One challenging subtask in translation can be to isolate the specific sense or meaning of a SL word or phrase as used in context, and then to find its closest TL translation without incurring new or unintended nuances of meaning in the TL sentence. The first part of this subtask, isolating the monolingual word sense, is called w*ord sense disambiguation* (WSD). When carried out however during translation, the challenges of WSD on the SL side are compounded by the need for WSD of each possible TL translations of the isolated SL sense. Since bilingual translators are quite rarely equally proficient in both the SL and TL, they find that must tackle WSD for one or both languages. Furthermore WSD may tap into cultural, colloquial, or domain-specific meanings that are not familiar to the translator.

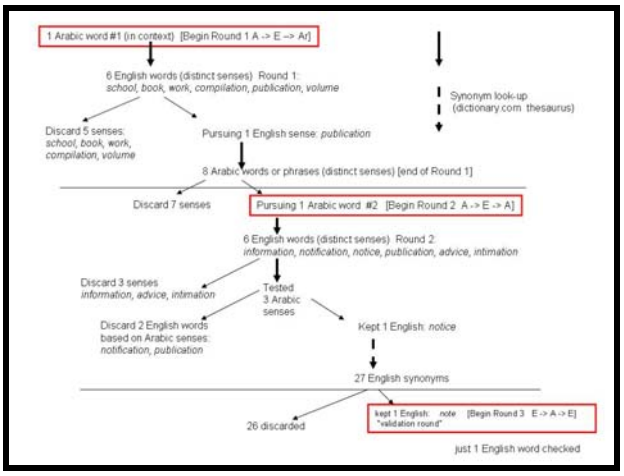

Figure 5. Steps toward Word Sense Disambiguation

The process of selecting the contextually appropriate word choice in both the SL and the TL led our translator to develop an iterative process of looking up over a dozen individual words in bilingual lexicons that initially stunned the non-translators, but was then later easily recognized by an experienced translation teacher as a practice he explicitly taught to new translators. We relate this description of the iterative process here because it led us to realize that we also wanted to provide for mashup tools within TREAT. This mashup is compelling example of the creativity that can arise with easy access to resources and tools.

The following example in **Figure 5** traces two round-trip Arabic-English-Arabic translations and one round-trip English-Arabic-English translation. The Arabic text to be translated was: "كتاب صورة". with an initial word-for-word translation by the human translator yielding "*Copy Book*" in English.

[5] A mashup is an application that combines data or functionality from two or more external sources to create a new service.

This literal translation did not appear to fit the context of the rest of the document. The translator suspected that the second word sense (read from right-to-left) "كتاب" was off due to uncommon usage.

Round 1: The translator used an online Arabic-to-English (A2E) dictionary to yield six possible English translations of the Arabic token: *school*, *book*, *work*, *compilation*, *publication*, and *volume.*

From the context of the original Arabic text, the words "*text*", "*school*", "*book*", "*work*", and "*volume*" are discarded. The word "*publication*" was selected from the original list as the closest in meaning, based on context. This English word was then translated into Arabic using Google's online English-to-Arabic (E2A) dictionary, producing eight tokens or phrases: "كتاب", "إعلام", "كتب", "منشور", "إشهار", "نشرة", "إذاعة", and "الكتب نشر وغيرها". Of these possibilities, the Arabic word "إعلام" seemed the best fit.

Round 2: Translating "إعلام" using the A2E dictionary yielded six translations: *information*, *notification*, *notice*, *publication*, *advice*, and *intimation*.

The word "*notice*" appeared to be closest to the desired context. Translating "*notice*" using the E2A online dictionary generated 14 possibilities, none however similar to the original Arabic token. The translator then used a monolingual English dictionary to find synonyms of the word "*Notice*": *acknowledge*, *advert*, *allude*, *catch*, *clock*, *descry*, *detect*, *dig*, *discern*, *distinguish*, *espy*, *heed*, *make out*, *mark*, *mind*, *note*, *recognize*, *refer*, *regard*, *remark*, *see*, *spot*, *take in*, *flash on*, *get a load of*, *look at*, and *pick up on*. The word "*note*" was selected then as the best candidate of these choices.

Round 3: The word "*Note*" was translated using a different E2A online dictionary[6], returning the Arabic token مُذَكَّرَه. The translator standardized the diacritization of the token to produce a disambiguate version of the Arabic token: مذكرة. Translating this token using A2E dictionary resulted in a list of English words that included the word: "memo." This word fit the original context the best, so the final best translation of the original Arabic phrase was set to "*Memo Copy*".

5 TREAT: The Translator's Framework

Our translator became very adept at switching between tools and using the various formatted outputs as resources for other tasks. Experience with and exposure to the many available tools that assisted our translator became the catalyst for constructing a framework for consolidating access to the tools. The Toolkit and Resource Environment to Assist Translation (TREAT) is the result.

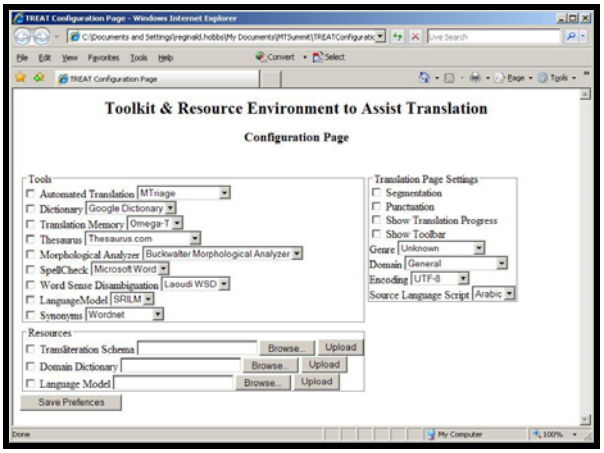

Figure 6 TREAT Configuration Page

TREAT is a front-end to multiple tools and is written as a simple web page user interface. There are two main pages that are viewed by the translator: a configuration page and the translation page.

The screenshot in **Figure 6** shows the *configuration page.* This page is only displayed during the initial start-up of the tool, to allow the translator the ability to set-up the environment prior to the translation tasks. For each category of tool, there is a drop-down menu for the choice (or choices) of tools to be made available. Each of the selected tools will appear on the pop-up context menu available during translation as well as on the toolbar located at the bottom of the translation page.

There is also an area on the configuration page to allow the translator to pre-load custom resources for use during translation. Translators develop their own local transliteration schemes and spelling techniques for translated text. Abjad or consonantary languages, like Arabic, require the speaker to supply the vowels, leading to more than one acceptable spelling of a word. Domain dictionaries are also a useful resource that may be preloaded from the configuration page to be used by tools that have the ability to import external data. Language models are the component within a statistical MT engine that is used to optimize the fluency of the English output.

[6] http://translate.reference.com/

The settings for the translation page are also available on the configuration page. Pre-processing activities for handling the source language data, such as segmentation or punctuation processing, can be selected during configuration. The translator can choose whether to make their progress visible through automatic highlighting and if the toolbar should be shown. Contextual information on the source language document, such as genre, domain, encoding, and language script are denoted during configuration.

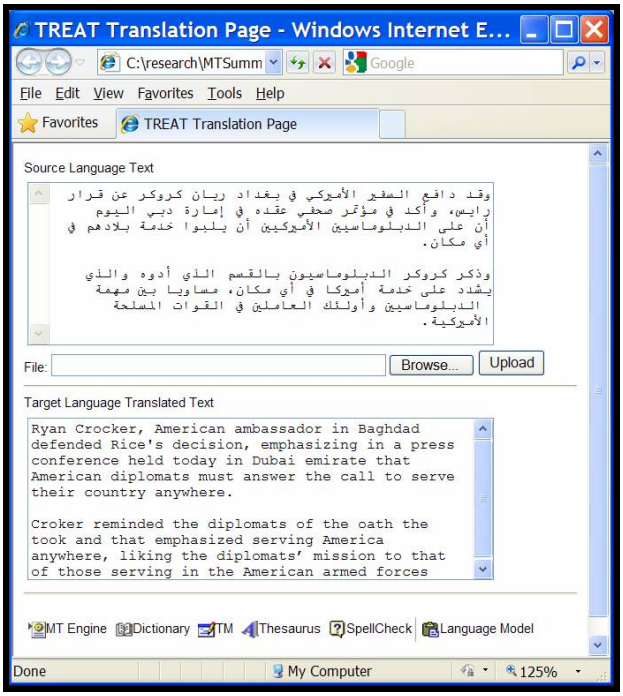

Figure 7 TREAT Translation Page

The *translation page* (**Figure 7**) is designed to be as simple as possible with only two main text areas on the page: the upper text area for the source language text being translated and the lower text area for the target language text translations. The tools selected on the configuration page appear as icons on a toolbar at the bottom of the page. This toolbar can be hidden to maximize the real estate on the screen for translation. The selected tools also appear as entries on the pop-up context menu that can be opened by right-clicking the mouse anywhere on the translation page interface. The TREAT framework can take advantage of user settings and internal coding to automatically modify the page layout, for example, moving the scroll-bar to the right side of the GUI to support right-to-left Arabic scripts.

6 Observations and Ongoing Work

In this paper, we have motivated our design, development, and user-centered approach to TREAT as a software framework, where an expert translator participates on the project as both designer and evaluator. We expect TREAT will evolve as we expose it to new users. In particular, we will track: (i) features that translators-in-training use immediately to establish framework defaults for learners, (ii) resources and tools that more experienced translators actively upload and select to evaluate what new items or categories to include in TREAT, and (iii) high frequency sequences of tool and resource usage for new mash-up ideas.

References

K. Beck. *Extreme Programming Explained: Embrace the Change.* Addison-Wesley. 1999

Josep Bonet. *Is machine translation useful for translators? The technological environment in the European Commission DG Translation.* Translingual Europe 2009, May 13-14, Prague, Czech Republic

Tim Buckwalter. *Issues in Arabic Orthography and Morphology Analysis,* The Workshop on Computational Approaches to Arabic Script-based Languages, COLING 2004. Geneva.

Ignacio Garcia. *Long term memories: Trados and TM turn 20.* Journal of Specialised Translation 4 (July 2005); pp.18-31.

Federico Gaspari & John Hutchins. *Online and free! Ten years of online machine translation: origins, developments, current use and future prospects.* MT Summit XI, 10-14 Sep 2007, Copenhagen, Denmark.

Reginald Hobbs, Jamal Laoudi, & Clare R.Voss. 2008. *MTriage: web-enabled software for the creation, machine translation, and annotation of smart documents*. Proc. of the 6th Language Resources and Evaluation Conference (LREC 2008), Marrakech, Morocco.

Jeffrey Micher and Clare Voss. *Buckwalter-based Lookup Tool as Language Resource for Arabic Language Learners.* Proc. of ACL-08: Software Engineering, Testing, and Quality Assurance for NLP, pgs 66-67, Columbus, June 2008.

Ross Smith. *Your own memory?* The Linguist 47 (1), February-March 2008; pp. 22-23.