

# A Quality Model for the Evaluation of Open Translation Technologies

Silvia Flórez, Amparo Alcina  
Universidad de Antioquia, Colombia; Universitat Jaume I, Spain

## 1 Introduction

In recent years, like many other professions, translation has undergone a series of transformations as a result of the advances made in information and communication technologies. Since the beginning of the nineties the use of computer tools by translators has grown steadily, as has the number and variety of tools available, which range from general programs like text editors or processors to specific tools for translators such as translation memory systems (Alcina 2008). Faced with an ever-increasing array of tools to choose from, the translator is left wondering which of them would best fit his or her needs, often without the parameters required to be able to compare them and make an informed decision.

Now, although the area of technologies applied to translation has undoubtedly received a great deal of attention in the scientific and professional literature, it is also true that free and open source software has been largely neglected without being given the attention it deserves. The software we are dealing with here is characterised by guaranteeing the four fundamental freedoms for users described in the introduction to this volume.

Open software in general has advanced a great deal in recent years and new projects appear every day. Yet, according to the results of a study conducted by García (2008) to determine the situation of the translation technologies market, it would seem that most translators are unaware of and have little interest in the open software specifically designed for translation. Although García's study revealed that a good number of translators use open tools for tasks that are not related to translation, open translation memory systems are only just beginning to be considered as feasible options. In a profession in which the tools that have led the market for years cost hundreds of euros, the predominant popular conception seems to be that something that is free is not likely to be of good quality.

The question then arises as to how to make it easier for translators to identify the open programs that really do meet their needs. To obtain a possible answer to such a question we can resort to the criteria that have been used in the fields of software engineering and information systems, as well as in the specific area of translation technologies.

## 2 Evaluation of Software Quality

To begin with, we find that in software engineering quality is defined as “the extent to which an object (...) (e.g. a process, product or service) satisfies a series of specified attributes or requirements” (Schulmeyer 2006: 6). As regards the definition of the object, there are two different conceptions: one more restricted, known as small *q*, which comprises only the intrinsic product quality, and another more general one, known as big *Q*, which, in addition to taking the product into account, also covers the development process and user satisfaction (Kan 2002).

In practice, in recent decades two main approaches have been followed to understand and study software quality. One of them is diachronic and based on quality management, in which a flexible qualitative standpoint and a corrective methodology (normally used internally within the organisation that develops the software) are adopted. The other one is based on quality models, in which a descriptive methodology is followed with a more rigid perspective from which quality is understood as a quantifiable concept, either in terms of adherence to processes or based on the measurement or appraisal of a series of attributes (Groven et al. 2011).

The ISO 9126 standard (“ISO/IEC 9126. Software engineering. Product quality” 2001), which establishes a software quality model and guidelines for using that model, follows this second approach. This general-purpose quality model is made up of two parts: the first part specifies the characteristics that allow the internal and external quality of the software to be determined, while the second part deals with the concept of quality in use. The internal and external quality of the software as a product refers to the properties of the software itself and, according to the ISO standard, comprises six characteristics: functionality, reliability, usability, efficiency, maintainability and portability (see Figure 1). Quality in use, on the other hand, refers to the extent to which a given user can achieve his or her goals in a specific set of conditions of use. According to the ISO 9126 standard (2001), quality in use can in turn be broken down into four characteristics: effectiveness, productivity, freedom from risk and satisfaction (see Figure 2).

Another standard that also deals with software evaluation is ISO 14598 (“UNE-ISO/IEC 14598. Information Technology. Software Product Evaluation” 1998). This standard provides a general description of the software evaluation process and is therefore normally used in conjunction with the ISO 9126 standard.

In the field of translation technologies, software evaluation has a long history going back to the ALPAC report in 1966 on the status of machine

translation. Yet, given the abundance and diversity of tools and the variety of stakeholders and possible usage scenarios (industry, public administration, researchers, developers, agencies, freelance translators, students, etc.), there is a need for standard evaluation methods that are reliable, acceptable and reproducible (Quah 2006; Rico 2001; Höge 2002).

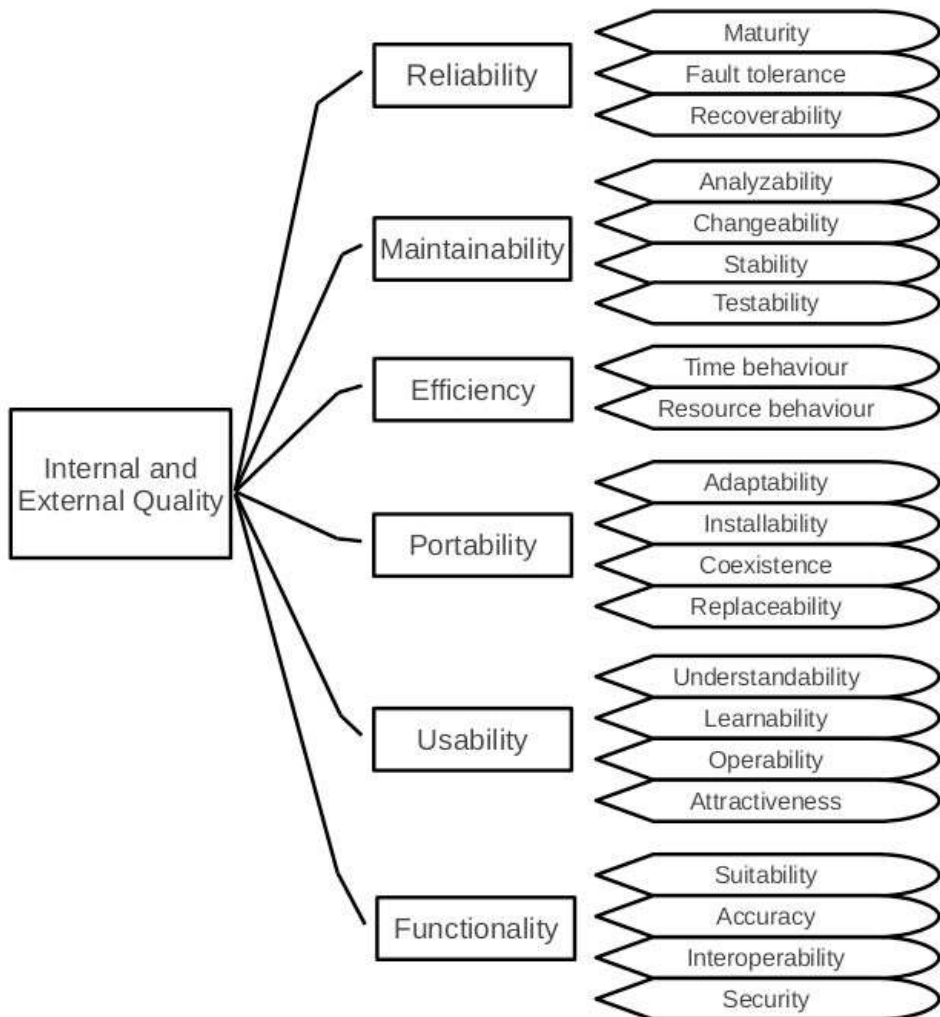
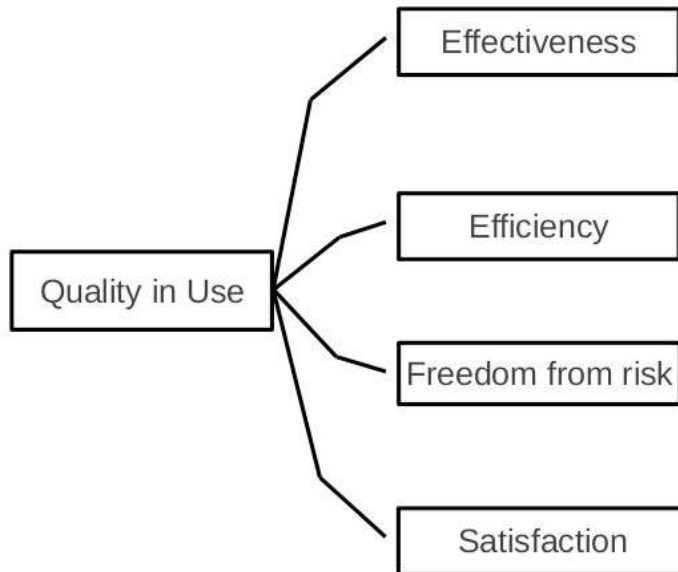


Figure 1: Internal and external software quality according to the ISO 9126 standard (2001).

As highlighted by Quah (2006), in the case of translation memory systems, evaluation is often part of the process of program development and is carried out from the point of view of researchers and developers rather than from that of the final user. Furthermore, in many cases the programs are evaluated by the same companies that develop them and, due to the fierce competition that exists in this field, the results are generally considered to be confidential.



*Figure 2: Quality of use according to the ISO 9126 standard (2001).*

In an effort to find a solution to the problem of the lack of standardised evaluation criteria mentioned above, several attempts have been made to establish a general framework or series of reference guidelines for the evaluation of language technologies (Quah 2006), a category that encompasses translation technologies. The first of these initiatives was undertaken in 1993 by the Expert Advisory Group on Language Engineering Standards (EAGLES), funded by the European Union, and was based on the six quality characteristics proposed by the ISO 9126 standard.

Following the work carried out by EAGLES, in the year 2000 Europe and the United States began a joint project called International Standards for Language Engineering (ISLE). The project had three working groups, one of

which was devoted to the subject of evaluation (Evaluation Working Group, EWG) (Calzolari et al. 2003). The work of this group focused on the area of machine translation, as this is one of the most difficult technologies to evaluate, although the long-term idea was to be able to generalise the results obtained to the evaluation of other language technologies (Calzolari et al. 2003).

The work of this group resulted in the development of the Framework for the Evaluation of Machine Translation in ISLE (FEMTI), which is a structured collection of methods for evaluating machine translation systems (Calzolari et al. 2003; Quah 2006). Another work deriving from the EAGLES initiative was the Test-bed Study of Evaluation Methodologies: Authoring Aids (TEMAA), the main aims of which were to foster thought about the process of evaluating natural language processing tools and to work on the creation of a tool that was capable of carrying out that process automatically (Quah 2006; TEMAA n.d.). Within the framework of the project, case studies were carried out on the evaluation of spelling and grammar checkers, as well as information retrieval tools.

## **2.1 Evaluation of Translation Technologies**

The theoretical model of the ISO 9126 and 14598 standards and the work by the EAGLES group have since given rise to several projects that include some kind of evaluation of translation technologies.

In her doctoral thesis, Höge (2002) presents her thoughts resulting from ten years of work in the field of translation technology evaluation from the user's point of view. Her work applies and complements the theoretical framework of the EAGLES group on the evaluation of different translation memory systems as part of the ESPRIT II project (1987-1992), financed by the European Commission. To apply her methodological proposal, the author evaluates two translation systems: Trados Translator's Workbench and IBM TM/2.

Rico (2001) also puts forward a final user-oriented model of evaluation that is based on the methodology proposed by EAGLES and the quality characteristics defined by the ISO 9126 standard. Her aim was to define a general model that could be re-used and applied in different translation contexts.

Maślanko (2004) conducted a comparative study of the terminological management modules integrated into a number of different translation memory systems (Multiterm iX by Trados, Déjà vu X by Atril and SDLX 2004 by SDL International). Her aim was to create an objective and detailed evaluation methodology that freelance translators and one-person translation businesses could use to select tools in Poland, her country of birth.

In her doctoral thesis, Filatova (2010) proposes adapting a scientific model of evaluation to the practical needs of translators. This project is broader in terms of the types of software evaluated, since it covers not only tools that, according to the author, are specific for translators (multilingual electronic dictionaries, word and character count, corpus analysis, translation memory suites) but also tools that she classifies as office automation software (file compressors, web browsers, e-mail clients, office automation suites, PDF readers and web authoring applications).

Finally, the work by Guillardau (2009) is, according to the author himself and as far as we know, the first study to focus exclusively on the comparative evaluation of free translation memory systems. The author takes the quality criteria proposed by ISO and by the EAGLES group and the doctoral thesis by Lagoudaki (2008) on the functionality of translation memory systems as the basis for a qualitative comparison of two open tools (OmegaT and Anaphraseus) in terms of their functionality, efficiency and usability.

A number of works have addressed the evaluation of translation technologies but have been limited to very specific issues (such as Cerezo 2003; Gow 2003; and Lagoudaki 2007) or to providing simple comparisons of the functionality of the tools (such as, for example, the work by Zerfaß 2002; Bowker and Barlow 2004; Eisele et al. 2009; and Wiechmann and Fuhs 2006).

## 2.2 Evaluation of Free/Open-Source Software

As regards the quality of free software, in recent years the fields of software engineering and information systems have adapted evaluation methodologies that take into account the specific features of this type of software and its development paradigm. In addition to evaluating the software as a product, they also cover aspects related with the communities that support the projects (Samoladas et al. 2008).

The first specific quality models, which appeared between 2003 and 2005, are known as first-generation models and are based on the traditional quality models of proprietary software, but have been adapted and complemented so as to make them applicable to free software (Groven et al. 2011). Some of the more notable first-generation models include the Open Source Maturity Model (OSMM) developed by Capgemini in the year 2003, the OSMM developed by Navica in 2004, one developed by the project Qualification and Selection of Open Source Software (QSOS) (Atos Origin 2006), originally started by Atos Origin in 2004, and the project Business Readiness Rating (BRR) (BRR 2005; Wasserman et al. 2006), which was begun by the Carnegie Mellon West Centre for Open Source Investigation and Intel, among others, in the year 2005 (Groven et al. 2011).

The quality models for free software that have appeared since 2006 are known as second-generation models and are characterised by being based on both the traditional models of proprietary software and on the first-generation models. Moreover, they are focused on the automation of the evaluation process and on providing more advanced metrics and tools for evaluation that are made available as web applications or plug-ins for development environments (Groven et al. 2011). Some of the better-known second-generation quality models include those developed by the projects Quality in Open Source Software (QualOSS) (Deprez 2009), Quality Platform for Open Source Software (QualiPSo) (Wittmann and Nambakam 2010), and Software Quality Observatory for Open Source Software (SQO-OSS) (Samoladas et al. 2008), all of which were funded by the European Community (Deprez and Alexandre 2008).

### **3 Towards a Method of Evaluation for Open Translation Technologies**

In this context, an objective detailed evaluation of the open tools for translators currently available may be a good way to disseminate the concept of free software in our profession and foster its use. The evaluation methods traditionally used for language technologies are focused on sequential or iterative and incremental development cycles and design processes rather than on non-continuous cycles such as those of free software (Gasser, and Scacchi, Ripoche and Penne 2003). Hence, there is a need for an integral evaluation methodology which takes into account not only the software as a final product but also considers aspects related to the development project, such as intellectual property management, forward planning, the dynamics of the user and developer communities, and the technologies supporting them.

In this work we therefore propose a method for evaluating open translation technologies. The method outlined here comprises a quality model and guidelines for its use (the activities, tasks and participants in the evaluation process, and the expected use of the results).

Taking an interdisciplinary perspective that includes technological, sociological and business aspects as our starting point, a qualitative approach was adopted for the evaluation. The reason underlying this decision was that the main interest was to describe the characteristics of the ecosystem of open translation technologies and to explore the feasibility of the programs currently available, rather than to reach generalisations about this type of software. The aim of the proposed method is to help translators when it comes to choosing open tools to integrate within their work environment. The users of

the results of the evaluation are expected to be freelance translators, translation teams, small companies, researchers, and translation students and teachers interested in open translation technologies.

### 3.1 Activities and Steps of the Evaluation

The method of evaluation proposed here comprises three main activities that are in turn divided into a series of steps, as detailed in the following:

- Preparing the evaluation: this consists in defining the type of tests and the quality model (the categories and criteria to be taken into account and the metrics and procedures for consolidating the results) and in designing and implementing the instruments.
- Evaluation: this consists in determining the sample of projects to be evaluated and collecting data by applying the questionnaire, which automatically generates the records with the results.
- Selection: this consists in specifying the user's requirements (existing environment, work formats and functional modules depending on the tasks to be undertaken), comparing programs that meet those requirements and choosing the most suitable.

In this case the first two activities of the process are carried out by the researcher herself, whereas the final or selection phase is to be done directly by the final user. In the following, we will concentrate on detailing the first of these activities, that is to say, on preparing the evaluation. For illustrative purposes, we will present the results of the evaluation of the open translation memory system OmegaT, which was conducted in May 2012.

It should be noted that this work was part of the research carried out by Flórez (2013) for her doctoral thesis, which included the compilation of a catalogue of free/open-source software for translators (see Flórez and Alcina 2011a and 2011b), and the evaluation of eleven development projects working on desktop translation memory systems available under free licences. Both the catalogue of tools and the instruments and full results of the evaluation are available in an online wiki created as part of that project (see Flórez 2012a).

### 3.2 Quality Model

To define the software quality model, the first step was to establish the type of test to be used and the context of evaluation. Bearing in mind that the rationale behind the evaluation of the software in this case was to test the general characteristics of the programs for their possible implementation in the translator's work environment, we decided to use the type of tests called feature inspection, the role of which is only to indicate the presence or



absence of certain features and not to identify bugs in the programs (EAGLES 1996; Höge 2002). This kind of tests was chosen because of its descriptive nature and due to the fact that it is simple, fast and easy to apply, since the data needed can be largely obtained from the documentation of the programs and the websites of the projects.

### **3.2.1 Categories and Criteria**

In the hierarchy for defining the evaluation criteria we started out by drawing a distinction between project and product. The quality model is made up of two parts: the first allows the development projects to be characterised so as to gain a better understanding of the practices and processes involved, as well as the resources and services available to the community of users. The second refers to the quality of the software as a product and makes it possible to determine the features and technical characteristics of the programs.

#### **Project Quality**

With the aim of characterising the free translation technology development projects, based on what was found in the literature and following the recommendation to work from the most general to the most specific, four characteristics were included: strategy, community, maturity and reputation of the project. Project quality is broken down into characteristics and sub-characteristics in Figure 3.

#### **Product Quality**

Taking into account the rationale behind the evaluation and the functional orientation of the programs, three of the six characteristics proposed in the ISO 9126 standard were used as criteria for evaluating the software, namely: functionality, usability and portability. Given the scope of this project, the other three characteristics set out in the ISO 9126 standard (reliability, maintainability and efficiency) were not included in the model. Figure 4 shows the characteristics and sub-characteristics of product quality that were included in the quality model.

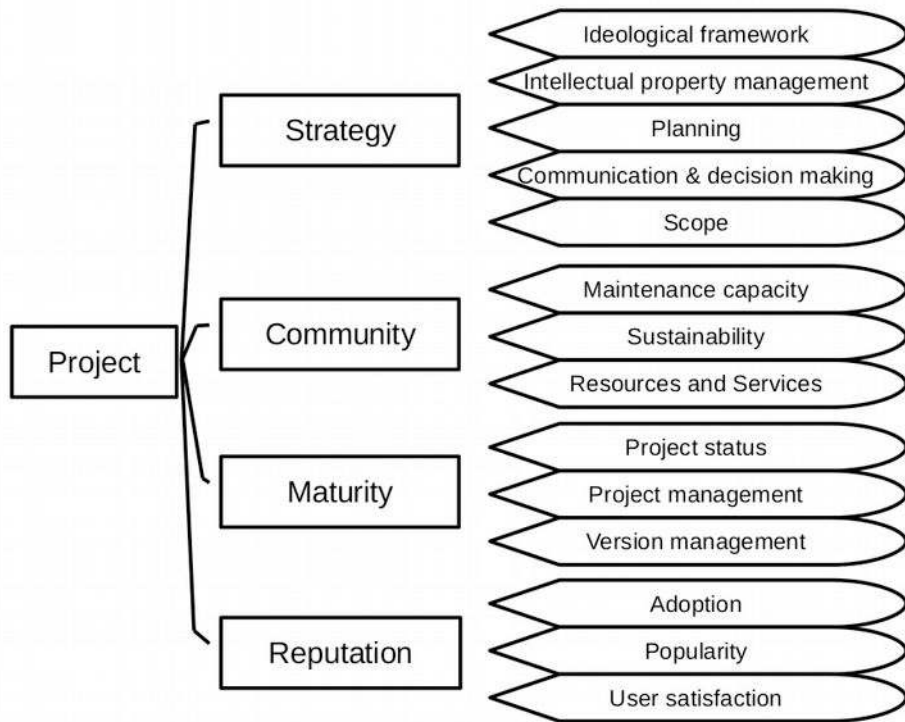


Figure 3: Characteristics and sub-characteristics of the quality of the project.

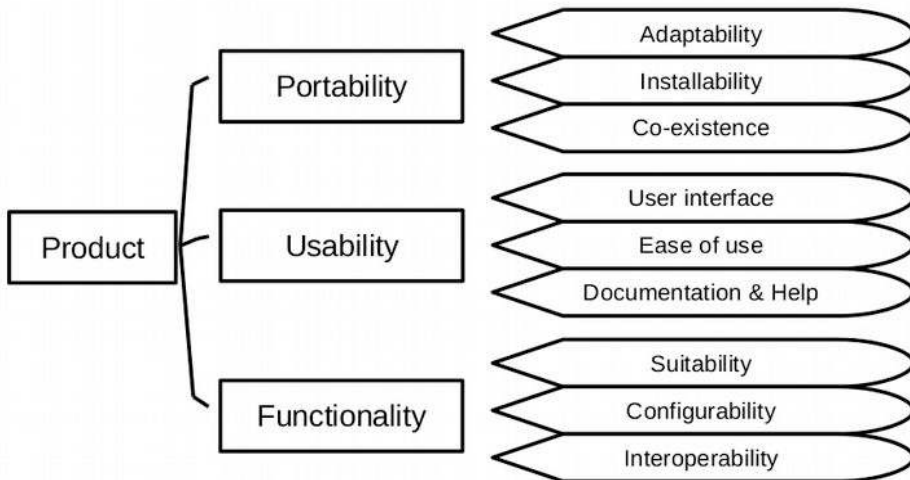


Figure 4: Characteristics and sub-characteristics of the quality of the product.

At this point it is important to note that the attributes corresponding to portability and usability are equally significant for any type of tool. In other words, they are non-functional criteria that can be applied both to a web browser and to an office automation application or to a translation tool. The attributes of the functionality characteristic, in contrast, vary according to the type of tool to be evaluated and the tasks that can be done with it (alignment, translation, proof-reading, invoicing, etc.). It must be made clear that the quality model prepared for this study is limited to analysing the functionality of desktop translation memory systems.

### 3.2.2 Attributes and Metrics

The next step consisted in breaking down each of the quality characteristics and sub-characteristics into one or more attributes. In the case of the project quality characteristics, a qualitative assessment was chosen. This means that for these attributes no quantitative scores were defined; in contrast, the factual information is presented directly on the result sheets so that the users can broaden their knowledge on each project. For the non-functional characteristics of product quality (portability and usability), on the other hand, we defined the corresponding attributes and metrics, that is, the way to obtain the quantitative scores and the scales to be used in each case. Finally, for functionality, a checklist was drawn up where the characteristics that were present could be indicated, but neither scoring was used nor were any appraisals made about the features implemented.

#### Project Quality

The tables below show the attributes defined to evaluate the strategy (Table 1), community (Table 2), maturity (Table 3) and reputation of the projects (Table 4) and the possible answers established for each attribute. As can be appreciated in the tables, some attributes are binary (presence/absence), while others are classificatory and still others are numerical.

Project strategy		
Sub-characteristic	Attribute	Options
Ideological framework of the project	Origin of the project	Independent project Publicly funded project Privately funded project Mixed funding project
	Type of ethics that govern the project	Hacker ethics Hybrid ethics Business ethics

<b>Project strategy</b>		
<b>Sub-characteristic</b>	<b>Attribute</b>	<b>Options</b>
Intellectual property management	General licensing strategy	One free licence Several free licences Dual licensing (free/proprietary) Open core
	Permissiveness of the licence	Without copyleft With weak copyleft With strong copyleft
	Guidelines or transfer of rights agreements for collaborators	Presence Absence
	Ownership of copyright	The owner is a single developer Ownership assigned to a legal body Distributed ownership
Forward planning	Specification of requirements	Presence Absence
	Roadmap	Presence Absence
	Description of new anticipated features	Presence Absence
	Versions planning	Presence Absence
Communication and decision-making structures	Type of process for decision-making	Decentralised Balanced Centralised
	System of governance	Benevolent dictatorship Meritocracy Democracy Anarchy
	Mechanism of representation used by the project to communicate and be identified	Original developer Recognised leaders Foundation Steering committee Sponsoring institution or company

<b>Project strategy</b>		
<b>Sub-characteristic</b>	<b>Attribute</b>	<b>Options</b>
Scope	Integration of code from other free projects	Yes No
	Project derived from another free project	Yes No
	Development of other tools	Yes No

Table 1: Attributes to determine the project strategy.

<b>Community</b>		
<b>Sub-characteristic</b>	<b>Attribute</b>	<b>Options</b>
Maintenance capacity	Type of development community	Independent developer Group of developers Formally organised developers Legal body Commercial body
	Forks or derived tools	Presence Absence
	Institutions linked to the project	Presence Absence
	Number of active developers	Numerical value
	Number of subscribers in the lists of users	Numerical value
Sustainability	Number of users who participated in discussions over the last month	Numerical value
	Average number of messages per month in the users' forum in 2011	Numerical value
	Average response time in the forums (last 5 questions)	Numerical value
Resources and services available	Web portal highlighting significant information about the project	Presence Absence
	Communication spaces actively used in the last year (mailing lists, wikis, blogs, IRC chats, social networks)	Presence Absence

<b>Community</b>		
<b>Sub-characteristic</b>	<b>Attribute</b>	<b>Options</b>
	Personalised technical support	Presence Absence
	Added value subscriptions	Presence Absence
	Training (tutorials, video channel, webinars, etc.)	Presence Absence
	Personalised development	Presence Absence
	Consultancy	Presence Absence
	Software as a service	Presence Absence

Table 2: Attributes for characterising the project community.

<b>Maturity of the project</b>		
<b>Sub-characteristic</b>	<b>Attribute</b>	<b>Options</b>
Project status	Date the project started	Numerical value
	Current development status	Beta Stable Mature Inactive
Project management	Management of the project in one of the main public forges	Presence Absence
	Source code repository with revision tracking system	Presence Absence
	System for managing potential bug reports	Presence Absence
	System for managing new feature requests	Presence Absence
	Existence of documented processes to contribute to the project	Presence Absence
	Platform for managing the localisation of the program and the documentation	Presence Absence

<b>Maturity of the project</b>		
<b>Sub-characteristic</b>	<b>Attribute</b>	<b>Options</b>
	Documented process of eliciting and managing requirements	Presence Absence
Version management	Defined release cycle	Presence Absence
	Versions released in 2011	Numerical value
	Minor updates released in 2011	Numerical value
	Date of last version released	Numerical value

Table 3: Attributes for determining the maturity of the project.

<b>Reputation of the project</b>		
<b>Sub-characteristic</b>	<b>Attribute</b>	<b>Options</b>
Adoption	Books, publications, reviews or entries in blogs about the project	Presence Absence
	Reference implementation/success cases documented on the project website	Presence Absence
	Average number of downloads during the week following the release of the last three versions	Numerical value
Popularity	Number of downloads in the last month	Numerical value
	Discussions in translators' forums (ProZ, LinkedIn, etc.)	Presence Absence
	Packages included in GNU/Linux repositories	Presence Absence
	Project included in software catalogues or directories	Presence Absence
	Profile of the project on Ohloh.net	Presence Absence
User satisfaction	Reviews and scores in the forge used	Presence Absence
	Comments on the project on social networks	Presence Absence

Table 4: Attributes for determining the reputation of the project.

## Product Quality

For the non-functional characteristics (portability and usability) of the software as a product, each sub-characteristic was broken down into a series of attributes, and then a series of possible answers and their associated scores were formulated for each of them. For these two characteristics we decided to use a homogeneous scale ranging from 1 to 3, where 1 means unacceptable, 2 is acceptable and 3 is satisfactory. While drafting the possible answers, efforts were made to consider the situations that are found in real use cases and special attention was paid to avoiding ambiguity, in an attempt to reduce the possibility of different interpretations being made by different evaluators in different contexts. Due to space restrictions, not all the attributes of these two characteristics are detailed here. For illustrative purposes, Table 5 below presents the possible answers for two attributes of portability and Table 6 shows two usability attributes.

Portability		Scoring		
Sub-characteristic	Attribute	1	2	3
Adaptability	Modularity	The design of the tool does not allow for the development of independent components.	The design of the tool allows for the development of independent components that can be integrated within the system, but no documentation is available.	The design of the tool allows for the development of independent components by means of a plug-in architecture or a well-documented public API.
	Scalability	The system is not designed with large-scale implementations in mind and does not include a multi-user mode.	The system can be implemented on a large scale, but it is not designed for multi-user environments or vice versa.	The system can be implemented on a large scale and in multi-user environments.

Table 5: Details of two attributes for evaluating the portability of the product.



Usability		Scoring		
Sub-characteristic	Attribute	1	2	3
User interface	Layout of the user interface	The interface is complex with too much information that is not clearly organised; the manual has to be used.	It takes some time to understand the interface, the information is more or less organised; the manual has to be used from time to time.	The interface is simple and intuitive, the information is well-organised; the manual is practically not needed.
	Availability of the required language	The program and its documentation and help are only available in a language other than the one required.	Localisation is partial (interface in the required language but documentation is not translated or vice versa).	The programme is totally localised into the required language, including both the user interface and the help, as well as other documentation that is included.

Table 6: Details of two attributes for evaluating the usability of the product.

In order to evaluate functionality, the features included, the possible configurations, the capacity to process different input formats and the interoperability were considered. A checklist was established with the main characteristics that one can expect to find in translation memory systems based on the functional descriptions of the principal commercial proprietary systems and on previous knowledge about this kind of tools. Following this same line, the list of features and supported formats can easily be expanded to cover other types of programs.

For each of the functionality attributes the presence or absence of the characteristic in question is indicated, but no scores are calculated and the adequacy of feature implementation is not appraised. In contrast, the full list of characteristics present is included on the result sheet. Table 7 offers details of the attributes that were used to evaluate the functionality of the programs belonging to the type translation memory systems.

Functionality		Scoring
Sub-characteristic	Attribute	Presence or absence
Suitability for purpose	Match between the features included and the expected features according to the type of program	<p>Project options:</p> <ul style="list-style-type: none"> <li>Analysis of originals (wordcount, matches, repetitions)</li> <li>Batch processing</li> <li>Pre-translation of documents</li> <li>Pre-translation prioritising the sources used</li> <li>Pseudotranslation</li> <li>Creation of projects with multiple source documents</li> <li>Possibility of using the memories in both directions</li> <li>Multiple memories per project</li> <li>Multiple glossaries per project</li> <li>Multiple translations for the same original segment</li> <li>Multilingual memories (more than two languages)</li> <li>Simultaneous use of glossaries/memories shared over the web</li> <li>Fuzzy matches</li> <li>Context-based matches</li> <li>Glossary matches</li> <li>Automatic insertion of exact matches</li> <li>Automatic insertion of fuzzy matches</li> <li>Automatic propagation of repeated segments</li> </ul> <p>Editor options:</p> <ul style="list-style-type: none"> <li>Visualisation of metadata of the matches (date, user ID, project, etc.)</li> <li>Segment validation by means of different statuses</li> <li>Option of browsing around the editor by means of filters</li> <li>Possibility of adding comments to the segments</li> <li>Project statistics (number of segments translated/not translated)</li> <li>Global search and replace</li> <li>Search for concordances in original files</li> <li>Search for concordances in reference files</li> <li>On-the-fly auto-complete</li> </ul>

Functionality		Scoring
Sub-characteristic	Attribute	Presence or absence
		<p>On-the-fly spellchecker  On-demand spellchecker  On-the-fly grammar/style checking  On-demand grammar/style checker  Preview of format  Review mode (track changes, comments, export to table)  On-the-fly quality checks  On-demand quality checks</p>
		<p>Integration with external applications:  Integration with local or web-based machine translation engines  Search in external resources (local or via web services)  Integration with voice recognition software (commands and/or dictation)</p>
	File filters implemented	<p>Text and office automation formats: TXT, CSV, TAB, DOC, DOT, RTF, XLS, XLT, PPT, PPS, DOCX, DOTX, XLSX, XLTX, XLSM, PPTX, PPSX, POTX, ODT, ODS, ODP, SRT</p>
		<p>DTP formats: MIF (FrameMaker), XML (FrameMaker), INX (InDesign), IDML (InDesign), tagged TXT (Pagemaker, Ventura), QSC (QuarkXPress), XTG (QuarkXPress), TTG (QuarkXPress), TAG (QuarkXPress), IASCI (Interleaf/QuickSilver), PDF (Acrobat Reader)</p>
		<p>Multimedia formats: PSD (Photoshop), SVG (Photoshop, Illustrator, CorelDraw, generic), DXF (AutoCAD), TXT (AutoCAD)</p>
		<p>Web localisation formats: HTML, XML, ASP, PHP, JSP, INC, NET, RESX, PPSM, XAML, SGM</p>
		<p>Software localisation formats: RC, DLG, EXE, DLL, MO, PO(T), Java Resource Bundles, XML (Android resource), XIB (iOS App resource), TS (Qt Linguist), QPH (Qt Phrase Book), DTD (Mozilla)</p>

Functionality		Scoring
Sub-characteristic	Attribute	Presence or absence
Configurability	Possibility of configuring the system according to different needs	Configurable filters Configurable segmentation rules Possibility of changing segmentation during translation Configurable minimum percentage of matches Customisable spellchecker dictionaries Customisable language corrector rules Searches and replacements based on regular expressions Configurable placeables and localisables (dates, variables, etc.) Configurable quality checks (tags, punctuation, spaces, numbers, terms, etc.) Control of access to the system by means of users and permissions Configurable keyboard shortcuts
Interoperability	Support for data exchange standards	Unicode encoding SRX segmentation rules TMX memories TBX databases Glossaries as delimited text (CSV, TAB or TXT) Pre-translated XLIFF files
	Support for open formats generated by other translation tools	TTX (SDL Trados) TXT (WordFast) TXML (WordFast Pro) NXT (STAR Transit)

Table 7: Attributes for evaluating the functionality of the product.

### 3.2.3 Procedures for Consolidating the Results

Procedures were then defined for summarising the attribute data in global scores per sub-characteristic. Since it was a general exploratory evaluation, all the attributes and characteristics were considered to be of equal importance and we therefore decided not to weight the results because we did not set out from a specific evaluation context that justifies the assignation of particular values. Moreover, the use of different scales (binary, classificatory and ordinal) makes weighted averages unsuitable for the consolidation of results.

As regards the quality of the project, for the characteristics project strategy and reputation we decided not to summarise the results by means of indicators as these aspects were not considered to have a decisive effect on the selection of the tools. In contrast, the information about the project strategy is presented on the result sheets as a descriptive paragraph about the projects, whereas the data found about their reputation is included as reference links for those interested in such information.

The results of the other two characteristics of project quality (community and maturity) were summarised by defining the acceptance criteria shown in Table 8. If the project met the established criteria, a star was given for the corresponding sub-characteristic; the project can thus obtain a maximum of three stars per characteristic. The number of stars obtained is interpreted as follows: 3 stars = satisfactory, 2 stars = acceptable, 1 star = poor, 0 stars = unacceptable. Furthermore, it was decided that for the projects with no stars for the characteristics of community and maturity the software would not be evaluated as a product.

<b>Characteristic</b>	<b>Sub-characteristic</b>	<b>Acceptance criteria</b>
Community	Maintenance capacity	At least one active developer and a users' forum with subscribers.
	Sustainability	Existence of active discussions in the last month and an average of no fewer than four messages per month over the last year.
	Resources and services available	Web portal with relevant information about the project; at least two communication spaces where users can obtain answers to their doubts.
Maturity	Project status	The project must be at least two years old and its current development status must be stable or mature.
	Project management	The code must be managed in a public forge with a change tracking system and bug report management.
	Version management	The project must have released at least one version or update in 2011 and the latest version available must be from 2011 or 2012.

Table 8: Indicators of the quality of the community and maturity of the project.

As to product quality, for the non-functional characteristics (portability and usability) stars are also assigned per sub-characteristic, but in this case the procedure used to obtain the global scores consists in simply adding up the individual scores of the attributes of each sub-characteristic and classifying the results in accordance with Table 9.

Lastly, for functionality, the information is not consolidated but instead, as explained in the previous section, the list of features available and the file formats supported are presented on the result sheet.

<b>Characteristic</b>	<b>Sub-characteristic</b>	<b>Acceptance criteria</b>
Portability	Adaptability	Minimum score equal to or higher than four.
	Ease of installation	Minimum score equal to or higher than six.
	Coexistence	Minimum score equal to or higher than four.
Usability	User interface	Minimum score equal to or higher than eight.
	Documentation	Minimum score equal to or higher than six.
	Ease of use	Minimum score equal to or higher than eight.

Table 9: Quality indicators for portability and usability.

### 3.2.4 Evaluation Instrument

The evaluation instrument was implemented as a complement to the catalog of open-source software for translators available in an on-line wiki created specifically for this purpose (see Flórez 2012a). Thus, we have a repository that makes both the instruments and the evaluation results publicly available. The instrument enabling the evaluator to collect data consists in a series of web forms (one for each quality characteristic, see Figure 5) that are filled in by hand. The data obtained are presented as complementary information on the data sheets in the catalogue.

General Information	Functionality	Project Strategy	Community	Maturity	Reputation
<b>Evaluation date:</b>		<input type="text" value="April"/>	<input type="text" value="2012"/>		
<b>How did the project originate?</b>		<input type="text"/>			
<b>What is the underlying ethics?</b>		<input type="text"/>			
<b>What is the licensing strategy?</b>		<input type="text"/>			
<b>What is the license(s) permissiveness?</b>		<input type="checkbox"/> no copyleft <input type="checkbox"/> weak copyleft <input type="checkbox"/> strong copyleft			
<b>Are there explicit contributor agreements? (URL)</b>		<input type="text"/>			
<b>What is the intellectual rights management policy?</b>		<input type="text"/>			
<b>Is there a product roadmap? (URL)</b>		<input type="text"/>			
<b>Is there a requirements specification? (URL)</b>		<input type="text"/>			
<b>Is there a description of new functions to be implemented? (URL)</b>		<input type="text"/>			
<b>Is there a versioning plan?</b>		<input type="text"/>			
<b>How is the project represented?</b>		<input type="text"/>			
<b>What is the system of governance?</b>		<input type="text"/>			
<b>How can the decision-making process be described?</b>		<input type="text"/>			
<b>Is the project a derivative or fork of another free/open-source project?</b>		<input type="text"/>			
<b>Does the project integrate code from other free/open-source projects?</b>		<input type="text"/>			
<b>Does the project develop other tools?</b>		<input type="text"/>			

Figure 5: Evaluation instrument – Project strategy.

## 4 Results

Below, we present the results of the characterisation of the OmegaT development project and the evaluation of the tool broken down by characteristics.

### 4.1 Characterisation of the Project

In the following subsections we present the results for each of the sub-characteristics of the quality of the OmegaT project, namely strategy, community, maturity and reputation.

#### 4.1.1 Strategy

The OmegaT project began as an initiative by independent developer Keith Godfrey and now has a group of recognised leaders. The work is carried out on a voluntary basis. The software and its features are available under a GNU GPL (strong copyleft) license and ownership is distributed among its developers. According to the philosophy of the project stated on its website it

is a “delegated anarchy”, where anyone is free to contribute to the project and there is a central team of developers who decide what contributions are to be included in the code that is distributed to the community. The project integrates code developed by other free projects (Hunspell, LanguageTool, Lucene Tokenizers and Okapi Framework).

#### **4.1.2 Community**

The project has a website (<http://www.omegat.org>) where relevant information is posted. Development is carried out by a group of developers in a collaborative and informal manner. In March 2012 there were four active developers and the user group had 1720 subscribers, of whom 39 had participated actively in the previous month. Moreover, the project has a general manager, a development manager, a documentation manager and a localisation manager.

In 2011 there were an average of 304 messages per month in the user group and the average response time for the last five questions was 0.3 hours; it is not necessary to be a member of the group to consult the message archive. The project also has a mailing list for developers and another for localisation management. In addition, it has an IRC channel. With regard to the services it offers, it is possible to sponsor the development of new features by getting in touch with the developers directly in order to agree upon the value of the monetary contribution to be paid.

There are several projects derived from OmegaT, some of the more important being: OmegaT+, a fork started by one of the developers following a series of disagreements (at the time of writing there are still disputes between the two projects over the name OmegaT as the trademark registered by the original project); Boltran, a web-based version of OmegaT; and Autshumato ITE, a translation memory system that integrates OmegaT, OpenOffice.org and the machine translation engine Moses (in this case there is some degree of collaboration between the projects).

#### **Appraisal**

In this case the fact that there is a website which is both well organised and offers detailed information about the project is judged positively, as are the number of active collaborators and the existence of derived projects. Furthermore, another positive point is the existence of several communication spaces for members of the project, together with the level of activity and the response time in the users' forum. As regards the professional services on offer, although the possibility of sponsoring the development of new features is valued positively, bearing in mind the characteristics of the project there could be a greater range of professional services on offer.



### **4.1.3 Maturity**

According to the copyright statement, the project began in the year 2000 and was registered on SourceForge on November 28th 2002. The current development status is stable and two main parallel versions are maintained: the so-called standard version, with all the features duly documented, and another called latest, previously known as beta, which the developers claim is equally stable but differs from the first one in that the latest features are not yet documented and the localisation may not be totally up-to-date. In 2011, 2 main versions and 13 updates were released and at the time of the evaluation (May 2012) the most recent standard version (2.5.4) was from May 9th 2012.

The project uses a repository with revision tracking (SVN) for code management and the tools provided by SourceForge for bug management and new feature requests. There is a documented process for contributing with the localisation of the interface and the documentation of the program.

#### **Appraisal**

The age of the project and its current development status are valued positively, as is the use of a public forge and specific tools for code management, bug reports and new feature requests. Furthermore, although there is no predefined release cycle, the regular release of updates and the availability of a recent version are given a positive appraisal.

### **4.1.4 Reputation**

In March 2012 the software was downloaded 5033 times and the average number of downloads carried out during the week following the release of the latest three versions was 1344, a figure which can be used to get an idea of the number of regular users of the tool. A number of publications about OmegaT were found and specific discussions were observed in translators' forums, for example, the support group in ProZ and a group in LinkedIn called OmegaT Translation Professionals. OmegaT is also included in the repositories of several GNU/Linux distros and is listed in several software directories. According to the scores on SourceForge at the time the evaluation was carried out, 88% of users recommend the tool (170 recommendations versus 23 negative ratings). Recent comments were also found on Twitter and the project has an updated profile on Open Hub (previously known as Ohloh), a platform for free software developers and projects where source code repositories of the programs are analysed and summaries of statistics are offered (including lines of code, programming languages and licenses used, level of activity of the projects and their estimated monetary value).

## Appraisal

The existence of publications about the project and the high number of downloads are valued positively. Another positive point was the existence of discussions about the tool in translators' forums and its being included in software directories and GNU/Linux distros. Likewise, the existence of recommendations in the forge and comments on Twitter was valued positively, as was the updated profile on Open Hub.

### 4.2 Evaluation of the Software as a Product

As mentioned earlier, the OmegaT project maintains two parallel versions of the tool: the standard and the latest. The standard version was used for the evaluation of the product as it is the one recommended for users who are beginning to use the tool. At the time of the evaluation (May 2012), the standard version that was available was 2.5.4.

Here we include the results for the functionality of OmegaT (see Table 10). Portability and usability of the tool were also evaluated, but due to space restrictions they are not included here; the detailed results of these two characteristics can be consulted in Flórez (2012b).

Functionality		
Sub-characteristic	Attribute	Characteristics present
Suitability for purpose	Match between the features included and the expected features according to the type of program	Project options: Analysis of originals (wordcount, matches, repetitions) Batch processing Pre-translation of documents Pre-translation prioritising the sources used Pseudotranslation Creation of projects with multiple source documents Fuzzy matches Context-based matches Automatic insertion of exact matches Automatic insertion of fuzzy matches Automatic propagation of repeated segments Glossary matches Multiple glossaries per project Possibility of using the memories in both directions Multiple memories per project

<b>Functionality</b>		
<b>Sub-characteristic</b>	<b>Attribute</b>	<b>Characteristics present</b>
		<p>Multiple translations for the same original segment</p> <p>Multilingual memories (more than two languages)</p> <p>Editor options:</p> <p>Visualisation of metadata of the matches (date, user ID, project, etc.)</p> <p>Option of browsing around the editor by means of filters</p> <p>Possibility of adding comments to the segments</p> <p>Project statistics (number of segments translated/not translated)</p> <p>Search for concordances in original files</p> <p>Search for concordances in reference files</p> <p>On-the-fly spellchecker</p> <p>On-the-fly grammar/style checking</p> <p>On-demand quality checks</p> <p>Integration with external applications:</p> <p>Integration with local or web-based machine translation engines</p>
	File filters implemented	<p>Text and office automation formats: TXT, CSV, TAB, DOC, DOT, RTF, XLS, XLT, PPT, PPS, DOCX, DOTX, XLSX, XLTX, XLSM, PPTX, PPSX, POTX, ODT, ODS, ODP, SRT</p> <p>DTP formats: XML (Infix), IDML (InDesign), XTG (QuarkXPress), TAG (QuarkXPress)</p> <p>Multimedia formats: SVG, XML (Flash export), CAMPROJ (Camstasia Studio)</p> <p>Web localisation formats: HTML, XML, RESX, JSON</p> <p>Software localisation formats: RC, POT, PO, Java Resource Bundles, XML (Android resource), TS (Qt Linguist), DTD (Mozilla), HHC (HTML Help Compiler)</p>
Configurability	Possibility of configuring the system according	<p>Configurable filters</p> <p>Configurable segmentation rules</p> <p>Configurable minimum percentage of matches</p>

Functionality		
Sub-characteristic	Attribute	Characteristics present
	to different needs	Customisable spellchecker dictionaries Customisable language corrector rules Searches based on regular expressions Configurable quality checks Configurable keyboard shortcuts
Interoperability	Support for data exchange standards	Unicode encoding TMX memories TBX databases Glossaries as delimited text (CSV, TAB or TXT) Pre-translated XLIFF files
	Support for open formats generated by other translation tools	TXML (WordFast Pro)

Table 10: Functionality of OmegaT (2012).

Table 10 shows the characteristics offered by OmegaT, version 2.5.4. As can be observed, the list of features included and formats supported is quite extensive and covers the most common requirements for exchanging data in our industry: Unicode, TMX, TBX and XLIFF. It should be noted that some features that were not available at the time of the evaluation (e.g. the search and replace option within the project) have since been implemented in later versions of the tool. Furthermore, the possibility of adding functionality by means of scripts (which were previously available as a plug-in and from version 3.0.3 onwards as a built-in feature) means that OmegaT can be adapted to the specific requirements of the translator's workflow.

### 4.3 General appraisal

The general appraisal is established by combining the appraisals of the characteristics that have been evaluated. The fact sheet of the general appraisal of OmegaT is available in the wiki, as can be seen in the partial screenshot presented in Figure 6. Owing to space restrictions, the list of features and supported formats has been excluded as this information was already shown in Table 10. As can be seen in the figure, according to the data obtained, both the community and maturity of the OmegaT project and the

portability and usability of the tool are considered satisfactory (three stars). The fact sheet also provides information about the strategy of the project, as a descriptive paragraph, and about the reputation of the project, including links to the main resources related to it.






OmegaT		Project Details	
		<p>Launched as an independent initiative, the project is led by a group of recognized project leaders. Code is developed on a volunteer basis. The software and all associated features are available under a single free/open-source license (strong copyleft). Copyright ownership is distributed across the individual developers. The project works on the basis of informal anarchic goal-setting. Decision making is balanced.</p>	
Category:	Translation Tools	Community	
Typology:	Translation environment		
<p>OmegaT® is a free and open source multiplatform Computer Assisted Translation tool with fuzzy matching, translation memory, keyword search, glossaries, and translation leveraging into updated projects.</p>		<p>The software is predominantly developed by several people collaborating in an informal or not industrialized way.</p>	
<a href="http://www.omegat.org/">http://www.omegat.org/</a>		<ul style="list-style-type: none"> <li>✓ Active development.</li> <li>✓ Active user communication venue(s).</li> <li>✓ Professional services available.</li> <li>✓ The project has derivatives.</li> </ul>	
Application Type:	Desktop	Maturity	
Programming language:	Java		
Operating systems:	Windows, GNU/Linux, Mac OS X	<p>The project was started on 2002/11/28 and is registered on a well-known forge.</p>	
Requirements:	Java Runtime Environment 1.7+	<p>Current development status is stable. In 2011, 2 major versions and 13 minor updates were released.</p>	
Latest release:	3.1.9 (2015/03/12)	<ul style="list-style-type: none"> <li>✓ Source code repository.</li> <li>✓ Bug tracking system.</li> <li>✓ Feature request system.</li> <li>✓ Documented contributing procedures.</li> </ul>	
License:	GNU General Public License v. 3	Reputation	
Integrates:	Hunspell, LanguageTool, Lucene Tokenizers, Okapi filters	<p>In March 2012, the software was downloaded 5033 times. Basing on the number of downloads during the week following the last 3 stable releases, the regular user base might be estimated at approximately 1344 users.</p>	
Available Resources		<ul style="list-style-type: none"> <li>✓ Published books/articles/blog posts.</li> <li>✓ Threads on translator-specific forums.</li> <li>✓ Included in GNU/Linux distros.</li> <li>✓ User reviews and ratings.</li> <li>✓ Recent tweets about the project.</li> <li>✓ O!high profile.</li> </ul>	
Download page:	<a href="http://www.omegat.org/en/dl_overview.php">http://www.omegat.org/en/dl_overview.php</a>		
Documentation:	<a href="http://www.omegat.org/en/documentation.html">http://www.omegat.org/en/documentation.html</a>		
User Forum:	<a href="http://groups.yahoo.com/group/omegat/">http://groups.yahoo.com/group/omegat/</a>		
IRC:	<a href="irc://irc.freenode.net/#omegat">irc://irc.freenode.net/#omegat</a>		
Developer Forum:	<a href="http://sourceforge.net/mailarchive/forum.php?forum_name=omegat-development">http://sourceforge.net/mailarchive/forum.php?forum_name=omegat-development</a>		
Software Capabilities			
Portability:			
Usability:			

Figure 6: Partial screenshot of the fact sheet of the general appraisal of OmegaT.

## 5 Discussion

The evaluation instrument was tested with a sample of eleven open-source projects working on desktop translation memory systems; here we present the results for the OmegaT project. In our opinion, the results obtained allow possible users to make inferences about the project evaluated, to compare

them and to select the tool that is best suited to their needs. Additionally, in general terms, the results obtained are considered to reflect the characteristics of the projects evaluated and can help translators to familiarise themselves with the characteristic aspects of the free software that they should take into account when it comes to choosing a tool for their work environment.

Bearing in mind the exploratory approach followed in this work, in general terms the test evaluation has been positive. As a favourable aspect, the instrument can easily be updated to include new features if and when necessary.

During the evaluation process, however, we also detected several possible problems and aspects that could be improved in order to achieve a more rigorous and detailed evaluation. On evaluating the project strategy, for example, for the attributes type of process for decision-making (decentralised, balanced or centralised) and system of governance (benevolent dictatorship, meritocracy or anarchy), the explicit information needed was found on the websites of the projects in only one case. It is therefore clear that these two attributes are more complex than expected and so it would be recommendable to use other techniques to evaluate them, such as a detailed analysis of the archives of the mailing lists or interviews with the developers.

One aspect of the strategy of the projects that was not taken into account and that could help to improve our understanding of the scope of the project is the target users. Some projects, especially in the field of natural language processing, are aimed at users with an advanced knowledge of computers and developers who are used to working on command lines, that is, without graphic interfaces. In other cases the tools are web-based and are not offered as a service, which implies that their installation and maintenance lie beyond the possibilities of users whose technical know-how is limited to the desktop environment. It would therefore be useful to add the attribute target users as part of the sub-characteristic scope of the project, so that these data can be used to filter the tools, according to the technical know-how needed to use them.

With regard to the characterisation of the communities, the breakdown of the sub-characteristic sustainability could be improved. In the method proposed here, three attributes were employed: the number of participants in the user lists in the last month, the average number of messages per month in 2011 and the average response time for the last 5 questions asked in the forums. Nevertheless, the data needed to evaluate this last attribute were found for only two projects.

On evaluating the maturity of the projects, two attributes were considered as part of the sub-characteristic project status: the date the project began and

the current development status. In both cases the data were obtained from the development forges, but in some cases discrepancies were found between the self-classification by the projects themselves and the classification of the forge. Moreover, it is also necessary to take into account that free projects may change development forge, and therefore the date that appears may be at odds with the date the project was initially registered. This information should therefore be confirmed using other sources, such as the information provided by the websites and blogs of the project or the change log that is sometimes included in the downloads.

The evaluation of the reputation of the project is another aspect that could be dealt with in greater depth. This could be achieved using qualitative techniques, such as the analysis of contents posted in translators' forums and social networks, or surveys carried out on users in order to determine their degree of satisfaction with the tools.

As regards the portability of the tools, in order to calculate the time needed to install them, which is covered by the sub-characteristic ease of installation, the instrument could be improved by specifying that this refers to the basic installation of the tool, without including dependencies, plug-ins or add-ons. Furthermore, in order to evaluate the possibility of integrating the tools into the existing workflow, an attribute that corresponds to the sub-characteristic coexistence, the type of test used (feature inspection) may not be sufficient and it would be recommendable to go deeper into the evaluation of this aspect by means of scenario testing within the expected environment of use.

According to our findings, the evaluation of the usability of the tools is perhaps the characteristic that entails the greatest risk of subjectivity. Aspects of the user interface, such as the user-friendliness of its layout or how easy it is to understand the icons and features, largely depend on the evaluator's point of view and perhaps also on his or her degree of familiarity with the type of tools being evaluated. For example, for a translator who is used to working with segments in columns, a horizontal layout may seem less user-friendly and vice versa.

For the sub-characteristic ease of use, on the other hand, although the attributes appraised are of a more objective nature (possibility of browsing and operating with just the keyboard, existence of contextual help and the existence of progress indicators and error messages), more rigorous results could be achieved by using systematic menu-oriented tests, designed to examine all the features offered by a program sequentially.

## 6 Conclusions

In this chapter we present a quality model for the evaluation of open source translation technologies. The model proposed here was implemented in a wiki as a complement to a catalogue of free software and it was tested with eleven free projects working on desktop translation memory systems. Both the evaluation instruments and the results of the eleven projects evaluated are publicly available in a wiki. In our opinion the quality model can be useful, and the results can be of use to translators interested in free software, since the fact sheets that are generated allow them to view the basic information about the project and the tools. We believe that having this kind of information available in a public repository can make it easier for freelance translators to reach a decision when it comes to selecting free tools for their work environment.

**Acknowledgements:** This research is part of the ProjectA research project: Translation projects with statistical machine translation and post-editing (FFI2013-46041-R) funded by the Ministry of Economy and Competitiveness, Spanish Government.

## References

- Alcina, A. (2008) Translation Technologies: Scope, tools and Resources. *Target: International Journal on Translation Studies*, 20(1), 79-102.
- Atos Origin (2006) Method for Qualification and Selection of Open Source Software (QSOS) version 1.6. Available at: <http://www.qsos.org/download/qsos-1.6-en.pdf> [Accessed: 15 March 2013].
- Bowker, L. and Barlow, M. (2004) Bilingual concordancers and translation memories: A comparative evaluation. In *Proceedings of the 20th International Conference of Computational Linguistics COLING-2004*. Presented at the Second International Workshop on Language Resources for Translation Work, Research & Training, Geneva, Switzerland, 70-83.
- BRR (2005) BRR Whitepaper 2005 RFC 1.
- Calzolari, N., McNaught, J., Palmer, M. and Zampolli, A. (2003) ISLE Final Report. ISLE Deliverable D14.2. ISLE. Available at: [http://www.ilc.cnr.it/EAGLES96/isle/ISLE\\_D14.2.zip](http://www.ilc.cnr.it/EAGLES96/isle/ISLE_D14.2.zip) [Accessed: 5 April 2015].
- Cerezo, L. (2003) Hacia la evaluación de dos sistemas comerciales de memorias de traducción. In *Entornos informáticos de la traducción profesional: las memorias de traducción*. Granada: Editorial Atrio, 193-213.
- Deprez, J.-C. (2009) QualOSS Assessment Methodology Version 1.1. QUALOSS Consortium. Available at: [http://www.qualoss.org/deliverables/D4.5\\_StdQualOSSAssessmentMethod-v1.1.tar.bz2](http://www.qualoss.org/deliverables/D4.5_StdQualOSSAssessmentMethod-v1.1.tar.bz2) [Accessed: 15 March 2013].



- Deprez, J.-C. and Alexandre, S. (2008) Comparing Assessment Methodologies for Free/Open Source Software: OpenBRR and QSOS. In A. Jedlitschka and O. Salo (eds.) *Product-Focused Software Process Improvement* (Vol. 5089). Springer Berlin / Heidelberg, 189-203.
- EAGLES. (1996) Evaluation of Translators' Aids. Available at: <http://www.issco.unige.ch/en/research/projects/ewg95/node140.html> [Accessed: 5 April 2015].
- Eisele, A., Federmann, C. and Hodson, J. (2009) Towards an effective toolkit for translators. In *Proceedings of the ASLIB International Conference Translating and the Computer 31*. London: ASLIB. Available at: [http://www.dfki.de/lt/publication\\_show.php?id=4586](http://www.dfki.de/lt/publication_show.php?id=4586) [Accessed: 5 April 2015].
- Filatova, I. (2010) Evaluación de herramientas y recursos informáticos (TAO y ofimática) para la traducción profesional: hacia la configuración de un entorno óptimo de trabajo para el traductor autónomo (doctoral thesis). Universidad de Málaga.
- Flórez, S. (2012a) FOSS4Trans. Available at: <http://traduccionmundolibre.com/wiki> [Accessed: 25 July 2015].
- Flórez, S. (2012b) OmegaT Results. Available at: <http://traduccionmundolibre.com/wiki/OmegaT-Results> [Accessed: 25 July 2015].
- Flórez, S. (2013) Tecnologías libres para la traducción y su evaluación (doctoral thesis). Universitat Jaume I.
- Flórez, S. and Alcina, A. (2011a) Catálogo de software libre para la traducción. *Tradumàtica 9* (Software lliure i traducció), 57-73. Available at: <http://revistes.uab.cat/tradumatica/article/download/5/6> [Accessed: 5 April 2015].
- Flórez, S. and Alcina, A. (2011b) Free/Open-Source Software for the Translation Classroom: A Catalogue of Available Tools. *The Interpreter and Translator Trainer (ITT): Volume 5, Number 2*, 325-57.
- García González, M. (2008) Free software for translators: is the market ready for a change? In Díaz Fouces, O. and García González, M. (eds.) *Traducir (con) software libre*. Granada: Comares, 9-31.
- Gasser, L., Scacchi, W., Ripoché, G. and Penne, B. (2003) Understanding Continuous Design in F/OSS Projects. Presented at the 16th Intern. Conf. Software & Systems Engineering and their Applications, Paris. Available at: <http://www.ics.uci.edu/%7Ewscacchi/Papers/New/ICSSEA03.pdf> [Accessed: 5 April 2015].
- Gow, F. (2003) Metrics for Evaluating Translation Memory Software (master's degree dissertation). University of Ottawa. Available at: <https://www.ruor.uottawa.ca/handle/10393/26375> [Accessed: 5 April 2015].
- Groven, A.-K., Haaland, K., Glott, R., Tannenbergh, A. and Darbousset-Chong, X. (2011) Quality Assessment of FOSS. In *INF5780 H2011: Open Source, Open Collaboration and Innovation*, 73-91. Available at: <http://publications.nr.no/directdownload/publications.nr.no/Compendium-INF5780H11.pdf> [Accessed: 5 April 2015].
- Guillardeau, S. (2009) *Freie Translation Memory Systeme für die Übersetzungspraxis*. Universität Wien. Available at: <http://othes.univie.ac.at/6863/> [Accessed: 5 April 2015].
- Höge, M. (2002) Towards a Framework for the Evaluation of Translators' Aids Systems (doctoral thesis). University of Helsinki, Finland. Available at:

- <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.18.3520&rep=rep1&type=pdf> [Accessed: 5 April 2015].
- ISO/IEC 9126 (2001) Software engineering. Product quality.
- Kan, S. H. (2002) Metrics and Models in Software Quality Engineering (2nd ed.). Reading, Mass.: Addison-Wesley.
- Lagoudaki, E. (2007) Translators evaluate TM systems – a survey. *Multilingual*, (March), 57-59.
- Lagoudaki, E. (2008) Expanding the Possibilities of Translation Memory Systems: From the Translator's Wishlist to the Developers Design (doctoral thesis). Imperial College London.
- Maslanko, K. (2004) A Comparative Study of Terminology Management Tools in Machine-Assisted Human Translation. Available at: [http://www.transsoft.seo.pl/en/translator\\_tools.html](http://www.transsoft.seo.pl/en/translator_tools.html) [Accessed: 5 April 2015].
- Quah, C. K. (2006) Translation and Technology. New York: Palgrave Macmillan Ltd.
- Rico, C. (2001) Reproducible models for CAT tools evaluation: A user-oriented perspective. In *Translating and the Computer 23*. Presented at Aslib, London. Available at: <http://www.mt-archive.info/Aslib-2001-Rico.pdf> [Accessed: 5 April 2015].
- Samoladas, I., Gousios, G., Spinellis, D. and Stamelos, I. (2008) The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation. In E. Damiani and G. Succi (eds.) *Open Source Development, Communities and Quality — OSS 2008: 4th International Conference on Open Source Systems*. Boston: Springer, 237–248. Doi:10.1007/978-0-387-09684-1\_19.
- Schulmeyer, G. G. (ed.) (2006) *Handbook of Software Quality Assurance (4a ed.)*. Boston, London: Artech House.
- TEMAA (n. d.) TEMAA Final Report. Available at: <http://cst.dk/temaa/D16/d16exp-Contents.html> [Accessed: 5 April 2015].
- UNE-ISO/IEC 14598 (1998) Information Technology. Software Product Evaluation.
- Wasserman, A., Murugan, P. and Chan, C. (2006) The Business Readiness Rating Model: an Evaluation Framework for Open Source.
- Wiechmann, D. and Fuhs, S. (2006) Corpus linguistics resources: Concordancing software.
- Wittmann, M. and Nambakam, R. (2010) OMM: CMM-like model for OSS. Qualipso Project.
- Zerfuß, A. (2002) Evaluating Translation Memory Systems. *Language Resources for Translation Work and Research*, 49.