The Evolution of an Evaluation Framework for a Text Mining System

Nancy L. Underwood, Agnes Lisowska

ISSCO/TIM ETI, University of Geneva
40 bd du Pont-d'Arve, 1211 Genève 4, Switzerland
Nancy.Underwood@issco.unige.ch, Agnes.Lisowska@issco.unige.ch

Abstract

The Parmenides project developed a text mining application applied in three different domains exemplified by case studies for the three user partners in the project. During the lifetime of the project (and in parallel with the development of the system itself) an evaluation framework was developed by the authors in conjunction with the users, and was eventually applied to the system. The object of the exercise was two-fold: firstly to develop and perform a complete user-centered evaluation of the system to assess how well it answered the users' requirements and, secondly, to develop a general framework which could be applied in the context of other users' requirements and (with some modification) to similar systems. In this paper we describe not only the framework but the process of building and parameterising the quality model for each case study and, perhaps most interestingly, the way in which the quality model and users' requirements and expectations evolved over time.

1. Introduction

Software quality does not exist in a vacuum. According to the ISO 9126 standards on software quality ISO/IEC (2001 and 2003) the quality of a piece of software must be evaluated in terms of its potential to aid users in achieving their tasks. It is such an ISO-inspired user-centred task-based approach to evaluation which we take here. The work reported on in this article was carried out under the auspices of the Parmenides project[1] which developed a text mining application which was applied in three different domains. During the lifetime of the project (and in parallel with the development of the system itself) an evaluation framework was developed by the authors in conjunction with the users, and was eventually applied to the system. The actual results of the evaluations which took place are confidential, however in this paper we describe the framework developed, the process of building and parameterising the quality model for each user case study and the ways in which the quality model and users' requirements and expectations evolved over time.

The object of the exercise was two-fold: firstly to develop and perform a complete user-centered evaluation of the system to assess how well it answered the users' requirements and, secondly, to develop a general framework which could be applied in the context of other users' requirements and (with some modification) to similar systems.

To set the scene we will first present a brief description of the system which was developed and then the users for whom it was developed. In the following sections we will go on to describe the framework and quality model and how it evolved during the life of the project.

2. The System

The Parmenides project implemented a complex ontology-based text-mining system comprising a number of different components as can be seen in the system architecture diagram in Figure 1.

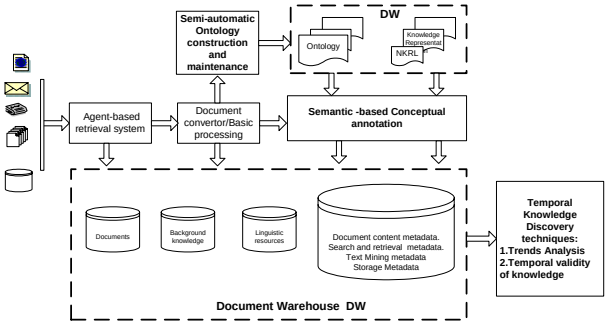

Figure 1: Parmenides System Architecture

The system is intended to support the entire text mining process from gathering documents through information extraction and semantic annotation to the application of data mining techniques. Being ontology-based the system also includes an ontology management system and tools for discovering and extracting new concepts and relations. In addition the tool provides document- and data-warehousing facilities. Although the system can support the entire text mining process, it is also possible for users to employ only a sub-set of the available facilities depending on the task they wish to carry out.

3. The Users

There were three user partners in the project representing widely different sectors and with different needs, but all with an interest in the development of a text mining system: Unilever, a world wide manufacturing organisation involved with foods, home and personal care products; Biovista, a corporate intelligence company in the biotechnology sector, and the research department of the Greek Ministry of Defence, a governmental organisation.

4. The Evaluation Framework

The ISO standard ISO/IEC 9126-1 (2001) and the EAGLES model for evaluation of language technology applications (EAGLES 1996) call for user requirements to

[1] www.crim.co.umist.ac.uk/parmenides

be translated into a quality model in the form of a hierarchy of software quality characteristics which are decomposed into sub-characteristics and eventually metrics which can be directly applied to the software.

ISO distinguishes between three types of software quality: *internal quality, external quality and quality in use*. *Internal quality* takes an internal view of the software product and is evaluated against requirements which are "used to specify properties of interim products". This is the type of evaluation typically done by developers and applied to static and dynamic models, documents and source code. *External quality* takes an external view of the product and is "typically measured and evaluated while testing in a simulated environment with simulated data using external metrics" i.e it can only be done on the system when it is running. Finally, *quality in use* is "the user's view of the quality of the software product when it is used in a specific environment and a specific context of use".

It is the *quality in use* that we were aiming to evaluate. But, *quality in use* characteristics and metrics only apply to deployed systems. In the case of the Parmenides project this was clearly not a feasible approach to take since the system was under development and we also had the aim of developing a general framework which, with some fine-tuning, could be applicable to other systems and users. However ISO 9126 also allows for "Estimated or Predicted Quality in Use" which is based on internal and external quality and this is the approach we have taken here. We concentrated on external quality characteristics and metrics which we hope will function as predictors of the eventual quality in use of the system. Indeed, this is by no means an unusual approach to take as witnessed by the many individual evaluations reported on in the literature (e.g. Rodriguez & Araujo (2002) and the proceedings of other LREC conferences). where the often unspoken assumption is that evaluating, for example, some core functionality of an HLT system will predict its suitability for a particular user or class of users. However, using internal or external metrics to predict quality in use means considering carefully whether or how a particular metric predicts how well the product supports the user in his tasks. It also means taking into consideration a great deal more characteristics of a system than are typically reported in the literature. Having said that, it should be noted that we were not in the business of software testing and bug fixing which was the job of the developers.

4.1 Quality Characteristics

In order to identify the relevant quality characteristics and specify metrics (measures and how to apply them) it is of course also necessary to understand how the system works from a user-oriented (input-output) point of view and how it might fulfill user requirements.

At the outset of the project each of the three users produced case studies exemplifying a problem or activity for which they wanted to use the Parmenides system and a detailed decomposition of their specific requirements on the final system. Since the system was under construction at the beginning of the project we took the developers' detailed system architecture and constructed the top layer of the quality model to consist of the following high level characteristics whic reflect not only the components of the system but also the activities which the users expected to perform:

- **Document Collector and Converter**
  The tool which gathers documents (e.g. from the internet or a file system) and converts them into the .xml format on which the system is based.
- **Semantic Analysis**
  Semantic analysis in Parmenides was performed by the Cafetiere tool[2] which not only annotates texts but also provides an interface to allow the user to amend the annotation.
- **Ontology Construction and Maintenance**
  As well as an ontology management system, tools for the identification and extraction of new concepts and relationships were included in the Parmenides system.
- **Document and Metadata repositories**
  Document repositories are either implemented as part of the file system or in a database, whereas metadata extracted from texts are stored in a database.
- **Temporal Knowledge Discovery**
  Three different data mining applications were provided to enable different types of knowledge discovery.
- **Information Requests**
  This high level characteristic comprises: document and metadata retrieval and the decomposition of information requests into atomic queries to the system. Other ways of interrogating the system are covered under the relevant components.
- **Integrated Demonstrator**
  This characteristic refers to users requirements on the system as a whole.
- **User Interfaces**
  This characteristic in fact cuts across all the other characteristics and its sub-characteristics are intended to be applied to all components.

All these top level quality characteristics are then decomposed into sub-characteristics. Figure 2 shows a simplified example of the sub-characteristics comprising the Semantic Analysis characteristic.

Semantic annotation

Functionality	➢ Suitability	➢ semantic annotation of texts
		➢ annotations vs. gold standard
		➢ metadata revision
Efficiency	➢ efficiency wrt current practice	

Annotation Editor

Functionality	➢ Suitability	➢ metadata correction
Usability	➢ Operability	➢ subjective ease of use

Building and maintaining underlying data

Functionality	➢ Suitability	➢ NLP resource development
		➢ access to NLP resources
		➢ editing NLP resources
		➢ transparency of NLP formalisms

Figure 2: Some sub-characteristics of Semantic Analysis

The ISO 9126 definition of a quality model proposes six top level (internal and external) quality characteristics of

[2] http://nactem.ac.uk/files/phatfile/cafetiere-report.pdf

functionality, *reliability*, *usability*, *efficiency*, *maintainability* and *portability*, but it also admits that "other ways of categorising quality may be more appropriate in particular circumstances" and that it is not feasible to apply all the quality characteristics to a single piece of software (ISO/IEC 2001, page 6). As can be seen in Figure 2 above, we have organised the highest levels of the quality model somewhat differently, introducing the ISO characteristics at the level of sub-components of the system. Not all of the six top level ISO quality characteristics were considered particularly relevant to each component. Although for every component of the system some *functionality* characteristics were applied but because we were designing a user-centred evaluation we concentrated on sub-characteristic of *suitability* (King, 2005).

As mentioned earlier, all the characteristics and sub-characteristics in a quality model bottom out into attributes which can be measured using a metric. The next section describes the definition of metrics for the evaluation of Parmenides.

4.2 Metrics

A metric for a particular system attribute consists of a measure (normally a value from a predefined measurement scale) and a method for applying that measure. By itself a raw score for a particular attribute of the software does not tell us anything about whether that score is to be considered good or bad, nor how satisfactorily a piece of software meets user requirements. So, in order to interpret the measurement which results from applying a metric, a rating scale indicating the positive and negative ends of the measurement scale is included in the description of the metric. In order to be as comprehensive as possible a great many metrics were defined of varying levels of complexity and theoretical interest. Numerically the vast majority of metrics concerned simple tests of functionalities which the users needed in order to be able use the system properly. So, for example, in all those components and activities where results were produced there were metrics for checking whether and how these results could be saved and subsequently retrieved whilst in the case of repositories and the ontology management system metrics were defined to assess searching functionalities. In some cases the application of these metrics led to feedback to the system developers and changes to the system.

Other metrics were more theoretically complex. Some were heavily inspired by standard external metrics such as applying classical recall and precision metrics to the term extraction tool in which the user was required to define a gold standard by hand, identifying potential terms in a set of texts and comparing this with the output of the term finder. Other metrics involved recording the time users took in carrying out particular tasks and then comparing this with the time they normally take when performing the same tasks (without the system). Quite a large number of metrics concerned users' subjective impressions of usability, for example how easy it is to perform certain actions, or understand and learn how to use components or the system as a whole.

Finally, we also developed a number of experimental metrics to evaluate the suitability of the knowledge discovery components (basically data mining: mining for association rules, sequence mining, classification, clustering). The knowledge discovery component of the system in fact comprises three data mining applications: one applies to textual data whilst the two others apply to metadata which has already been extracted from texts.

In the field of data mining there are accepted metrics for evaluating the performance of data mining algorithms from the developers' point of view. However these external metrics do not generally make good predictors of quality in use. This is because, in general, data mining techniques produce results which by their very nature are statistical and indicative rather than factual, and which need to be interpreted by a user in order to be useful. The process of data mining is also usually iterative and interactive, where the user refines parameters in multiple iterations until a useful or interesting result is achieved. The quality of the data which have been mined also has a profound effect on the results. So, a major challenge in designing an evaluation for tools based on data mining is to avoid simply evaluating either the ability of a single user to appreciate the results produced or the quality of the data which has been mined. For a fuller discussion of this question see King and Underwood (2006). Our intention then was to evaluate whether the component is adapted to the expertise of the users and whether it effectively handles the nature of the data being mined (rather than evaluating the software's performance with respect to its specifications, which is the domain of the software developers).

At the very general level, the user could be looking for insights hidden in a particular data set which he will then use to further his investigation or make decisions or recommendations, or alternatively, he could be using the knowledge discovery module to verify certain hypotheses he already has in mind. To achieve either of these objectives, one of two high level data mining tasks may be employed, commonly referred to as *description* and *prediction*. *Description* refers to the task of revealing properties of data (whether this is done by mining association rules, by clustering or even by using classifiers), whilst *prediction* refers to making predictions on the basis of inferences drawn from the data.

The part of the quality model for knowledge discovery then, is largely structured according to these two aspects:

The user's overall objective

- Looking for new insights
- Verification of user's hypotheses

The high level data mining task

- Description
- Prediction

Either *description* or *prediction* can be used to fulfill the user's overall objective and, in principle at least, it may be possible to use any of the data mining techniques to carry out these tasks.

Based on discussions with the users we identified four important quality characteristics (referring to the results produced by the components) to help evaluate the suitability of the system for each user and the specific datasets which they intended to mine:

- Novelty
- Credibility

- Understandability
- Relevance

The metrics associated with each of these quality characteristics are based on users' subjective opinions of the results obtained. In the case of novelty, the user had to decide which discovered patterns were new to him, the final measure being the percentage of new (to the user) patterns which were discovered. For the other three characteristics users were asked to score results on five point scales indicating how credible, easy to understand and relevant they found the results. These metrics sound deceptively simple but this conceals some complex issues of validity which have yet to be resolved. For example if the results are deemed irrelevant by a particular user, this may be because his data source does not in fact contain any relevant knowledge and so it could never be found. Alternatively, if the user is executing a more directed knowledge discovery project, the problem may lie in how he has defined the relevant parameters and how well he understands the system. In these cases it is not clear whether we are evaluating the system, the user, or his data although it does constitute an evaluation of how well this combination of all three aspects supports the user in his task. It is clear to us that further research is necessary to find ways of refining such metrics and developing new ones.

5. Parameterising the Quality Model

The building of the quality model can at the basic level be seen as the provision of a checklist of characteristics and associated metrics which the evaluator must apply to the software under evaluation. However, applying the metrics and recording their results is only part of the evaluation story. Having applied the metrics we then want to see how the raw results obtained indicate the suitability of a system or tool for the user in question. To do this we employ "assessment criteria" for characteristics and metrics. In the final analysis and reporting phase of an evaluation these are converted into weightings on each node of the quality model tree which are then used to interpret and combine the raw results of applying the metrics into an overall evaluation score if that is required.

The three user partners presented very different case studies and consequently their requirements on the system were also different. An acceptable result for one user may not be acceptable for another and certain features or functionalities may be considered indispensable for one user but unimportant for another (especially in such a complex system as Parmenides). In addition, even with system characteristics which the user considers important, they considered some more important than others and in the final analysis of the results such characteristics should be weighted more heavily than less important aspects of the system.

Therefore, once the general quality model for Parmenides was built, users were asked to parameterise it for their own case study by assigning three different types of assessment criterion:

- **Rating Level.** For each metric the users specified the minimum result they would consider acceptable. This was done not only for numerical measures (such as recall in term extraction, or average processing speeds) but also qualitative scales (e.g. *easy to use – quite difficult to use – difficult to use*) and Boolean measures (*yes/no*).
- **Priority.** For each node in the quality model the users assigned an absolute priority on a three-point scale: *Mandatory; Nice to Have;* or *Indifferent.* It might seem strange to have quality characteristics to which users are indifferent, but recall, that this is a general quality model to account for all the users' requirements, so it is possible that some of the characteristics held no interest for particular users.
- **Relative Importance.** With the help of a specially designed tool users assigned relative importance to the sibling nodes in each sub-tree in the quality model. For details of this tool see Lisowska and Underwood (2006).

This resulted in three different specific quality models each tailored to a specific user's needs.

This exercise highlights another important issue to be addressed when building a quality model. In our approach at least, not only the content but the structure of the quality model tree is important and in fact has its own semantics. By definition the quality model tree comprises quality characteristics which are progressively decomposed into their sub-characteristics so that in principle sub-characteristics which are dominated by the same parent node in the tree must be related to one another in a specific way. As well as grouping together related characteristics and their metrics for the sake of conceptual clarity, this structuring allows the user to make the comparisons between characteristics which are necessary when assigning assessment criteria and also allows for the correct combination of the weighted results in order to calculate the value of each parent node, recursively up the tree to the root node. The procedure of having the users apply their assessment criteria to the quality model also provided useful feedback to us about the correct structuring of the model to achieve the aims just described. It is true that when designing a practical evaluation however it may not always be possible to achieve an ideal structure, but we believe that it is a goal to strive for.

6. Evolution of the Quality Model

As the system was developed and implemented, and the users and evaluators gained more experience with the software, the quality model evolved to reflect that experience. Not only did this mean changes to the characteristics and metrics it contained but also to the user's expectations and requirements as embodied in their assessment criteria.

6.1 Changes to metrics

A metric can be seen as a practical recipe for how to evaluate a specific quality characteristic and thus, details of how to apply it are dependent on how the software is actually implemented. The metrics in the original quality model were devised before the system was implemented and so were based on our interpretation as to how the use cases described in the system architecture would be realised in terms of components. The use cases did not describe exactly how something would be implemented and so some discrepancies occurred between our assumptions when designing a particular metric and how

the system actually functioned. This often necessitated slight adjustments to certain metrics.

The original quality model contained a total of 249 metrics but over the lifetime of the project this number was reduced to 182 metrics which were eventually applied to the system. There are various reasons for this. A number of functionalities described in the original specifications were not finally implemented and it was the subject of some discussion with the users as to whether metrics based on those functionalities should remain in the quality model or not. The decision whether to abandon such characteristics and metrics tended to hinge on whether users felt that they were important to their requirements or not.

In other cases it was realised that certain metrics were not relevant after all. For example in the case of the document collector and converter, metrics for assessing the correctness of the XML tags assigned by the converter were originally defined but these were not of interest to users (who are basically concerned with the input-output of the system) since they would not have cause to look at these converted documents and any problems caused by incorrect XML tags would anyway show up at the stage of semantic analysis.

More interestingly though, in experimenting with applying metrics to early versions of the software, some more serious shortcomings were identified which needed to be rectified in order to ensure the validity of the metric. For example in the evaluation of the performance of semantic annotation the generally accepted approach (from the point of view of a developer) is to create a "gold standard" by taking a representative set of texts which the system should be able to treat and marking them up with the semantic annotations the user required from the system. The gold standards thus created are then used as the benchmark against which to compare the results of running the system on those same texts. Although we were developing a user-centred evaluation, we originally chose to adopt such a metric in the belief that it might offer some prediction of how well the system could potentially meet the users' needs.

Originally it had been planned that, over the months preceding the start of the evaluation process, users would build such large-scale gold standards (of at least 100 documents and ideally 1000 documents). However, given the very complex nature of the .xml annotations used in Parmenides, and the highly detailed knowledge of both the NLP algorithms and the common annotation scheme required in order to mark up the documents by hand, it was simply not feasible for users, who are not NLP experts, to mark up the texts with all the required .xml tags necessary to allow a fair evaluation of the semantic annotation module[3]. Another option, of course, would be to create the gold standard by using the system itself which, whilst it might be a reasonable way to proceed for a developer of the system, rather diminishes the objectivity of the experiment when executing a user-centred evaluation since the only annotations on the gold standards would then be those allowed by the system. Applying such a metric would thus fail to identify any cases where the system's existing ontology and NLP rulesets did not allow annotations which the user required.

The aim of carrying out a user-centred evaluation is, in any case, not simply to quantitatively test the accuracy of the software (although this can be an important characteristic) but, even more importantly to try and model how a system might perform if it were deployed. That is, how suitable the system is for the user. It is therefore necessary to look at the whole process of semantic annotation rather than only the accuracy of the algorithm in isolation. The semantic annotation module is designed to be used in semi-automatic mode. That is, for every text that is processed it is possible (in fact advisable) to inspect and, if necessary, amend the semantic annotations before saving the file and the annotations in the system. So it seemed to be most advisable to concentrate on those metrics which evaluate the expected benefits of the system in use rather than try to apply a less than objective metric. Such metrics concerned measuring the amount of revisions which the user did on the automatically annotated texts and comparing the time taken to correctly annotate texts using the system with the average time the user normally took to annotate texts by hand.

6.2 Changes in users' requirements and expectations

The changes in metrics described in the previous section were derived from discussions with users and to that extent clearly reflect changes in their expectations regarding the software capabilities. However, in our framework, there is another way in which users express their expectations, namely through the assessment criteria they attach to each metric and characteristic and in particular through the assignment of priorities.

Over the lifetime of the project users' expectations of the system changed. This was not unexpected. Several factors can lead to such changes including new developments in technologies and increased user understanding of both the potential and the limitations of such technologies. For example, certain characteristics of the system were initially unfamiliar to users and they assigned them a low priority because they did not understand them but later considered them more important as familiarity with the software increased.

To get a clearer picture of how one user's priorities changed, an experiment was carried out which compared two snapshots of a user's prioritisation of system characteristics at different stages in the project. Immediately after the general quality model was developed, all the users assigned their priorities (*mandatory, nice to have* or *indifferent*) to all the characteristics in the quality model. At this point, users assigned priorities strictly on the basis of the system architecture and how they thought that the system reflected their own needs, that is before having any experience with an implementation of the system. Nine months later, after some initial experience with a partially implemented system one user repeated the exercise of assigning priorities to all the metrics and characteristics, without looking back at his original priority assignments.

[3]This can clearly be seen in the following mark-up with just structural tags for a single word which includes, e.g., id numbers, and parts of speech tags: <tok id="t468" pos="DT" lem="the" lookup="NIL" orth="lowercase" zone="body" sepAfter=" ">the</tok>. A user who is really only interested in higher level concepts like entities and events could not be expected to be able to replicate such mark-up.

We then compared the two snapshots and discovered some interesting trends.

The first general trend seems to be an overall decrease in the user's expectations of the system. During the first assignment of priorities the *mandatory* and *nice to have* ratings were dominant. By the second snapshot however, the distribution of priorities was more or less evenly divided between all three priority types with the *nice to have* ratings being only slightly more common than the other two. The most significant decreases in priority levels, occurred with respect to the Document Collector and Converter, Ontology Construction and Maintenance, and Document and Metadata Repositories characteristics where in each case over 50% of the priorities were lower than in the previous exercise. We believe that in the case of the ontology- and repository-related characteristics this may be due to the fact that after experience with the system, the user better understood both the inherent limitations of the technologies involved and that his expectations could not be met within the timespan of the project due to the complexity of the system being built. On the other hand, in the case of the Document Collector and Converter, we believe that the decreased priorities may be due to the realisation that this part of the system was not as important to his work process as he had initially believed.

The priorities assigned rarely showed an increase with respect to the original assignment However, it is interesting to note that the 11% of changed priorities which exhibited an increase from *nice to have* to *mandatory* were all applied to characteristics in the Temporal Knowledge Discovery module. This may be due to the fact that the user only fully realized the impact that this module could have on their work practices after they were able to interact with the system itself.

In general, there appeared to be a preference for robust functionality over aesthetic and usability characteristics in the system. It would be interesting to try to determine in the future if such as trend is specific to systems that are known from the outset to be developmental prototypes, or whether this would also hold for commercial applications. This finding is also borne out in our experiences with all three users when assigning relative importance to characteristics as reported in Lisowska and Underwood (2006).

In a similar vein, rating levels which were assigned at the beginning of the project were typically very high (e.g. 90% recall for term extraction) but, once users had worked with the system as a whole, results which did not achieve such high standards were in some cases nevertheless accepted in the final evaluation. This was most often the case where other system functionalities made up for the perceived shortfall.

7. Conclusions and Future Work

In this paper we have described the development of a large and complex framework for the evaluation of a specific text mining system. We believe however that the approach described here can give inspiration to others designing a user-centred evaluation of similar systems.

User requirements and priorities are central to such evaluations, but as we have seen it is probably unwise to elicit these requirements and priorities once and for all. This was an unusual project in that the original development of the quality model took place in parallel with the implementation of the system and this no doubt had an effect on how the quality model evolved. Nevertheless we believe that, even when evaluating a final market ready application, as users come to understand both the potential and limitations of a system their expectations can also change.

We would not claim that this is the last word in how to evaluate text mining systems from a user-oriented point of view. A great deal more research is needed, particularly in how to define valid external metrics which will predict the quality in use of complex systems which rely crucially on user intervention and the nature of the data to be processed. Indeed work is already underway in further analysing the possibility of defining general models of classes of users and systems which could inform the development of a general framework for user-oriented evaluation of text mining systems.

In addition the authors are currently working on the development of a tool which will automatically calculate weightings and evaluation scores given a quality model containing assessment criteria and raw results of applying metrics.

8. References

ISO/IEC (2001). *9126-1:Software engineering – Product quality – Part 1: Quality Model*. Geneva, International Organization for Standardization and International Electrotechnical Commission.

ISO/IEC (2003a). *9126-2:Software engineering – Product quality – Part 2: External metrics* Geneva, International Organization for Standardization and International Electrotechnical Commission

ISO/IEC (2003b). *9126-3:Software engineering – Product quality – Part 3: Internal metrics.* Geneva, International Organization for Standardization and International Electrotechnical Commission

ISO/IEC (2003c). *9126-4: Software engineering – Product quality – Part 4: Quality in use metrics*. Geneva, International Organization for Standardization and International Electrotechnical Commission

King, M. (2005). Accuracy and Suitability: New Challenges for Evaluation. *Language Resources and Evaluation*, 39, pp.45-64.

King, .M & Underwood, N. (2006), Evaluating Symbiotic Systems: the Challenge. In Proceedings of the Fifth International Conference on Language resources and Evaluation, (LREC 2006). (in press).

Lisowska, A. & Underwood, N. (2006), ROTE: A Tool to Support Users in Defining the Relative Importance of Quality Characteristics. In Proceedings of the Fifth International Conference on Language resources and Evaluation, (LREC 2006). (in press).

Rodriguez, M. G. and Araujo, C. P. S. (2002). Third International Conference on Language Resources and Evaluation (LREC 2002), Canary Islands, Spain,

9. Acknowledgements

The work described here was funded by the Swiss Office Fédéral de l'Education et de la Science (OFES) as part of our participation in the Parmenides project No: IST-2001-39023