Learning Metrics for Natural Language Requirements in an Under-Graduate Course

Giuseppe Lami
Istituto di Scienze e Tecnologie dell'Informazione "A.Faedo", C.N.R. – Pisa (Italy)

Abstract

In this paper the experience of an under-graduate course focused on the requirements elicitation, specification and quantitative analysis is presented. The object (natural language requirements) and methodological approach adopted in the course are discussed to evaluate their suitability to learn metrics.

1. Introduction

Teaching quantitative methods for assessing software is today considered an important issue also for under-graduate university courses, as the Software Engineering Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering [3] witnesses. In fact, in [3] three specific topics (VAV.fnd.4, QUA.pro.1 and QUA.pds.5) directly address software metrics.

Skills related to the Requirement Engineering discipline (as for instance the generation of complete, unambiguous requirement documents, or the requirement change management) are considered crucial for the success of industrial software projects [5, 6]. That determined an increasing awareness of the importance of the specific role of the requirement engineer. Finding effective educational techniques and training methods to teach the requirement engineering skills the market demands is one of the main challenges the experts shall face.

Learning quantitative methods for the analysis of natural language requirements specification is particularly momentous for the software industry. In fact, while the natural language is the most frequently used representation in which to state requirements, in the practice there are few methods and tools supporting their quantitative analysis [7].

In this paper, the experience of an under-graduate course focusing on the elicitation, specification and quality analysis of requirements, is presented. The suitability of such a course to learn metrics is discussed as well.

The paper is structured as follows: in section 2 the under-graduate course where the quantitative analysis of requirements specification has been experienced is described as well the automatic tool used to perform such an analysis: in section 3 a discussion on the effectiveness of the followed approach is provided. Finally, section 4 contains the conclusions.

2. Description of the Course

In this section the principal objectives of the course are discussed along with the consequent course design decisions and trade-offs.

2.1 Course Characteristics

The course, held at the University of Florence (Italy) in the fall period of year 2004, is named "Laboratory of Software Engineering" (LSE).

The background for the LSE course was provided by a separate longer course (teacher: prof. Stefania Gnesi) aimed at teaching the general concepts and notions of Software Engineering.

The LSE course is included at the 3rd year of the Computer Science curriculum of the Mathematics, Physics and Natural Science Faculty. In the following, some information on the LSE course are provided, more details can be found in [1].

The planned duration of the course was 16 hours. The number of attendants was about 20 people. The principal criteria followed in the design of the LSE course were:

1. Orientation to the practice
2. Generality
3. Ability to address actual needs of the today's software engineering community

Due to the time constraints and taking into account the above criteria, the selected topic of the course was the requirements elicitation, specification and assessment.

In fact, the requirements elicitation and requirements analysis are two of the area of the requirements engineering discipline needing to be improved most [5, 6], moreover nowadays the demand for quality and risk identification in software is more and more increasing along with the need to have an approach to the quality more flexible and agile (quality is defined more in terms of "triage", "survival" and "good-enough" than "perfection and exceeding customers expectations" [10]). Risk identification and quality analysis are topics that may effectively address these needs especially if they are

approached avoiding too complex, formal and heavy techniques and methods.
Moreover, the abilities/skills related to the requirements handling (writing and using) the course should aim at providing the students with are general enough to be relevant possibly for every operational context. This addresses the objective 2.
Finally, the activities of the course, in order to satisfy the objective 1, have been approached from a practical point of view, requiring the students to produce requirements documents and perform ambiguity analysis on them.

The following principal results were expected:
- Giving the students the awareness of the need to take care of the quality of the writing style when requirements are documented.
- Giving the students the awareness that the requirements elicitation is an highly interactive phase of the software development where the involvement of the customer (or the other stakeholders of the system to be developed) is necessary to solve possible conflicts and misunderstandings.

The course was divided into 8 modules each of them 2 hours long. The audience was divided into 4 groups. The groups ware requested to participate in the lectures/labs and to perform home works. Details of the modules are provided in the following list:
- Modules 1-2: *Software quality*. The purpose of these two introductory modules was to underline the importance of a culture of quality in the software engineering and to understand the consequences of lack of quality at all the phases of the software development with particular emphasis to the requirements definition phase. Metrics, reviews and process assessments, as means to evaluate the quality, was also presented.
- Module 3: *Criteria and best practices for requirements documentation*. In this module the state-of-the-art techniques for representing requirements (as for instance Use Cases, Graphical notations, ...) have been introduced and compared with the plain natural language. The strengths and weaknesses of each of them have been compared and discussed.
- Module 4-5: *Requirements Elicitation Workshop*: the objective of this part of the course was to clarify the differences among functional, non-functional and quality requirements and to simulate a meeting between customer and supplier for eliciting customer requirements. The workshop was composed of a presentation of the customer needs (the teacher played the role of the customer) and an interactive part where the groups ware encouraged to interview the customer to elicit additional requirements/details, ask for clarifications and solve possible conflicts.
- Module 6: *Techniques and methods for the quality analysis of natural language requirements*. This module aimed at providing an introduction to the matters and problems related to the quality due to the use of natural language requirements. An overview of the existing tools and methods for the quality analysis of natural language requirements was provided with specific focus on the QuARS tool [2] that will be used by the groups in Module 7.
- Module 7: *Use of an automatic tool supporting the quantitative quality analysis of natural language requirements*. In this laboratory session the tool QuARS was run on the requirements documents the students produced. During the laboratory session the students were requested to calculate metrics and report the defects detected according to the outcomes of the tool. At the end of the session the students delivered a corrected version of their requirements documents.
- Module 8: *Analysis and discussion of documents provided by the groups*. This final module aimed at sharing the results of the work made by the different groups. Students were involved and encouraged to provide criticisms to the other groups' work and propose improvements.

Each group was requested to produce, in the week between Module 5 and Module 6, an initial version of a document containing the requirements describing the system functionality of the system presented during the requirements elicitation workshop. Each students group was provided with a requirement document template to be respected by the requirements document they produced in order to assure a common basis for the delivered documents along with a certain degree of completeness and comparability.
The composition of the groups was determined randomly because the education background of the course attendants was essentially homogeneous.
The groups were also requested to deliver a final version of the requirements document after the analysis with the QuARS tool (Module 7).

2.1 Using the QuARS Tool in the LSE Course

The most innovative part of the course was the application of a automatic tool for the linguistic analysis of natural language requirements. The used tool, called QuARS (Quality Analyzer for Requirements Specification), has been developed at the I.S.T.I. with the purpose to provide the practitioner with an easy and light tool able to point out potential linguistic defects in natural language requirement documents as well as calculate quality metrics.
In the following a short description of the QuARS tool is provided; for more details refer to [2].
QuARS performs a lexical and syntactical parsing of a natural language requirements document (in English) taken as input, and provides the following functionalities.

Defective Sentence Identification (Expressiveness Evaluation)
Evaluating the expressiveness of requirements means to identify defects related to the style they are written. Similar to any other evaluation process, the quality evaluation of natural language software requirements has to be conducted against a quality model. The quality model defined for the expressiveness analysis of the requirements allows at achieving a quantitative (i.e. that allows the collection of metrics), corrective (i.e. that could be helpful in the detection and correction of defects) and repeatable (i.e. that provides the same output against the same input in every domain) evaluation of natural language requirements.
The expressiveness quality model consists of three features, to be evaluated by means of indicators. Indicators are linguistic components, directly detectable and measurable in the requirements document, which reveal defects in requirements. Special lexicons (list of words and phrases) contain the indicators that QuARS needs to perform the analysis.
The defect categories addressed with the expressiveness analysis are:

- Non-ambiguity: the capability of having a unique interpretation.
- Understandability: the capability of being fully understood both when used by the software developers and when read by the user.
- Specification Completion: the capability of each requirement to uniquely identify its object or subject.

The results of the lexical and syntactic analysis of the requirements are used to identify those sentences containing defects according to the quality model. The output of the tool is the set of sentences recognized as defective according to the underlying quality model. These sentences are not defective according to the rules of English grammar, rather they are incorrect in terms of the expressiveness characteristics defined above.
A side effect of the use of the QuARS tool is the presence of false positives in the results: false positives are sentences indicated as defective which, after the analysis of the results provided by QuARS, are recognized as correct by the user.
Finally, the metrics QuARS calculates are related to the readability of the requirements document and the rate of defective sentences in the document under analysis for each class of defects (*number of defective sentences / number of sentences in the document*).
Requirements Clustering (View Derivation)
In addition to the expressiveness analysis, QuARS also provides the capability to cluster those requirements that belong to a user-defined lexicon. These clusters are called "Views" in the tool. For example, a list of words related to privacy could be used to identify the location and group all requirements specifying an aspect of privacy. New lexicons can be used for either quality attributes or may be related to a business domain feature-type requirement. In this manner, requirements related by the lexicon can more readily be checked for completeness and consistency.

QuARS is an innovative prototype tool that has been designed for the analysis and detection of defects in requirements, nevertheless it can have a relevant didactic impact because it can enforce the habit to have care of the writing style of requirements documents in natural language by means of the provision of the list of defective sentences.
Moreover, the tool is easy to use (the students had no problems in fully using it just after a 10 minutes demo provided by the teacher at the beginning of the laboratory session (Module 7)) making it suitable for such a educational context.

3. Discussing the Suitability of the LSE Course

The suitability of the object (natural language requirements) as well as the methodological approach adopted for the course are considered in this section to discuss the effectiveness of the LSE course to learn software metrics.
Using natural language requirements as object of study to let the students learn the importance of the quantitative assessment and the consequences of lacks of quality has some advantages:

- assessing natural language requirements is a relevant issue and an actual demand in software engineering [5, 8, 9];
- the natural language requirements assessment is a general practice to be performed independently of the context and the nature of the software project. In fact, requirements are handled (written or used) during the whole software development process and the capability to write unambiguous requirements and discover possible ambiguities (and then risks for the project) is a useful skill for every possible scenario a software engineer will operate in;
- quantitatively assessing and calculating metrics on natural language texts doesn't require specific technical skills and expertise in designing, coding and testing software.

Moreover, the metrics QuARS calculates address real and relevant problems as previous empirical studies witness [2, 4].
The skill of undergraduate students in software quality evaluation and using metrics is expected to be low. For this reason the main didactical objectives should be:

1. to understand the need and advantages of calculating metrics in software engineering and quality assessment;
2. to experience the derivation and calculation of metrics;
3. to understand the scope and validity of the applied metrics (what specific properties of the software can be measured by a metrics and what not).

Objective 1. In the modules 1 and 2 software quality assessment is addressed with the aim to underline the importance of adopting quantitative assessment methods, including metrics. The consequences of lack of quality during the software development process are also discussed in these modules. In addition, the possible effects on the development of the system of a sample of significant errors found in the lab session (Module 7) are discussed in the Module 8.
Objective 2. The orientation to the practice of the course and, in particular, the use of the QuARS tool allow the students to experience the derivation of metrics directly on artefacts they made.
Objective 3. The trust on applied metrics for software shall be combined with the awareness of the real scope and validity of them. This is important mainly for undergraduate course. The student shall understand what is going to be measured and if the metrics used are valid for that. The scope of the metrics for measuring natural language requirements quality applied in the LSE course is well defined, i.e. these metrics address the expressiveness characteristic (writing requirements good) but they cannot address completeness and consistence characteristics (writing good requirements). Considerations on the scope and validity of the used metrics were part of the discussion made in the Module 8. Such a discussion allowed the teacher to address general problems related to the use of metrics for software starting from the results of the group works. In addition, the need of considering the outcomes of the tool to look for false positives, is a stimulus to assess the validity of the metrics themselves (and avoid an excessive, unconditional and uncritical reliance in them) because the value of the calculated metrics may be affected by them.

4. Conclusions

The experience of an undergraduate course addressing the quantitative assessment of software requirements presented in this paper cannot be considered exhaustive to give a satisfactory answer to the need of teaching metrics. This paper should be considered as a contribution in incorporating the study and application of metrics into a software engineering course. In spite of this, the LSE course has some strengths:

- ability to involve the students because they are requested to evaluate the quality of artefacts they made;
- giving the opportunity of applying significant and easy-to-understand metrics;
- the support of an automatic tool;
- modularity, i.e. it can be included in a larger software engineering course as a didactic module;
- scalability, i.e. if the audience becomes larger, the course structure can be maintained and the effort rise is acceptable.

The next year's edition of the LSE course will give the opportunity to improve some aspects of the course itself (as the support to the students to edit the requirements specification document) as well as to enforce its "metrics orientation".

5. References

[1] G. Lami. Teaching Requirements Engineering in the Small: an Under-graduate Course Experience. Proc. of the 1st International Workshop on Requirements Engineering Education and Training. Paris (France), August 30th 2005.

[2] S. Gnesi, G. Lami, G. Trentanni, F. Fabbrini, M. Fusani. An Automatic Tool for the Analysis of Natural Language Requirements. International Journal of Computer Systems Science and Engineering, Special issue on Automated Tools for Requirements Engineering. CRL Publishing Ltd. Leicester, UK. Volume 20 No 1, January 2005.

[3] The Joint Task Force on Computing Curricula. Software Engineering 2004 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. IEEE Computer Society, Association for Cumputing Machinery, August 23, 2004.

[4] G. Lami, R.W. Ferguson, D. Goldenson, F. Fabbrini, M. Fusani, S. Gnesi. Automated Natural Language Analysis of Requirements and Specifications. INCOSE (International Council on System Engineering) International Symposium, Rochester, NY July10-15 2005.

[5] Standish Group. The CHAOS Report, http://www1.standishgroup.com/sample_research/chaos_1994_1.php

[6] Software Engineering Institute, CMMI v1.1 Class A Appraisal Results, August 2004.

[7] L. Mich, M. Franch, P. Novi Inverardi. Market research for requirements analysis using linguistic tools, Requirements Engineering Journal Vol. 9, Num. 1, pages 40 – 56, Springer-Verlag, February 2004.

[8] G. Kotonya, I. Sommerville. Requirements Engineering: Processes and Techniques, Wiley Ed., 1998.

[9] Software Engineering Institute, CMMI v1.1 Class A Appraisal Results, August 2004.

[10] E. Yourdon, "Preparing Software Engineers for the "real world"", ACM Software Engineering Education & Training Conference, Febraury 2002.