icse
SCOTLAND 2004

26th international conference on software engineering

23-28 May 2004, Edinburgh, Scotland

WORKSHOP

W13S

The Second Workshop on Software Quality

Monday 24th May 2004

icse
SCOTLAND 2004

WORKSHOP

W13S

ISBN 0-86341-428-1

supported by

NORTHROP GRUMMAN
Space Technology

NOKIA
CONNECTING PEOPLE

Microsoft®
Research

IBM Research

IBM®

zühlke
DIE DENKFABRIK.

Apple Education

Kodak

Proceedings

of the

Second Workshop on Software Quality

Edinburgh, Scotland, UK

24th May 2004

Editor

Bernard Wong

Table of Contents

Message From the Chairs

Welcome to the Second Workshop on Software Quality. As with the first workshop, the workshop is intended for researchers in the area of software quality and aims to encourage discussion on their research progress and to explore future directions. The goals of the working session are to exchange experience, to discover areas of mutual collaboration and to envision future trends in the field of software quality. To achieve this, the workshop aims to bring together academic, industrial and commercial communities interested in software quality topics to discuss the different technologies being defined and used in the software quality area.

In particular, the workshop will explore whether some consensus definition of "software quality" is achievable that can serve as a basis for reasoning about, measuring, and achieving software quality in sound, consistent, and useful ways. The workshop will also discuss how well, and under what conditions, current and emerging software quality-related standards, methodologies, and techniques enable us to improve the quality of our software projects.

The workshop had 16 papers accepted for presentation and discussion. We are pleased to have gathered a truly international community with papers from Austria, Australia, Belgium, Canada, Denmark, Ireland, Italy, Spain, Thailand, United Kingdom, and the United States. All papers had two reviews.

We would like to thank all the people who have been involved in organizing this workshop, all the authors that submitted papers, and the ICSE organizers for their support.

We hope you will find the workshop productive and memorable.

Barry Boehm
Sunita Chulani
June Verner
Bernard Wong
Workshop Chairs

Empirical Assessment of a Defect Detection Technique for Use Cases *

B. Bernárdez[1], A. Durán[1], M. Genero[2], A. Ruiz–Cortés[1]

[1] Dep. of Computer Languages and Systems
University of Seville
Avda. Reina Mercedes,s/n.
41012 Sevilla (Spain)
{beat,amador,aruiz}@lsi.us.es

[2] ALARCOS Research Group,
Dep. of Computer Science
University of Castilla–La Mancha
Paseo de la Universidad, 4.
13071 Ciudad Real (Spain)
Marcela.Genero@uclm.es

Abstract

In this paper, an empirical assessment of a defects detection technique for use cases is presented. The technique consists of two main elements: a use case quality model and a set of metrics–based heuristics for predicting defects in use cases. The quality model is derived of an exhaustive analysis of existing quality models and of several requirements documents developed by Computing Engineering students at the University of Seville. In this way, we aim to build a detailed taxonomy of defects that were usually detected by the heuristics. The assessment of the heuristics is based on empirical data obtained from requirements documents that were verified by experts in requirements verification using a checklist based in our quality model. This empirical work has not only allowed us to corroborate that our refined quality model improves the capacity of the heuristics in predicting defects in use cases, but also has allowed to review and adjust some of the metrics thresholds values.

1. Introduction

The importance of applying quality assurance (QA) from the first stages of software development is something widely acknowledged within software engineering community. Traditionally, requirements verification and validation (V&V) have been considered as the only requirements quality assurance (RQA) activities in the requirements engineering process. Nevertheless, a wider vision of RQA, like the one we presented in [6], considers not only V&V but also requirements analysis as RQA activities. The reason for this classification of requirements analysis is that its main goal, as defined by Kontoya and Sommerville in [12], is finding problems and conflicts in draft requirements so they could be solved and requirements quality increased.

In the requirements engineering (RE) process, we consider that the goal of requirements verification is to increase the so–called *internal quality* [11] of requirements documents, whereas the goal of requirements validation is to certify that the requirements are consistent with the intentions of customers and users, as defined by Pohl in [15].

In the context of requirements verification, this paper presents an empirical assessment of a defect detection technique for use cases. The technique consists of two main elements: a use case quality model and a set of metrics–based heuristics for predicting defects in use cases.

The quality model is derived of a thorough analysis of existing quality model for requirements and an exhaustive verification of several requirements documents developed by Computing Engineering students at the University of Seville using REM, a free, XML–based requirements management tool developed by the second author of this paper [6]. The goal of this analysis was to identify which quality characterictics are not fulfiled by the use cases which not respect a set of heuristics for predicting defects in use cases presented in [5, 6].

The assessment of the heuristics is based on empirical data from requirements documents that were verified by experts in requirements quality using a checklist based on the quality model proposed by us .

The rest of this paper begins with a brief summary of the defect detection heuristics on which our current research is based. The next section explains how we reached the requirements quality model (RQM) we propose. Section 4 includes the empirical assessment of the RQM and the

*This work has been partially funded by the WedMade project (TIC 2003-02737-C02-01), the CALIPO project (TIC2003-07804-C05-03) and the MESSENGER project (PCC-03-003-1).

defect detection heuristics when applied together. Finally, section 5 describes some conclusions and future work.

2. Heuristics for use case verification

The defect detection heuristics (briefly summarised in this section) are based on our previous work on requirements verification [5, 6]. The intuition behind these heuristics is that there are several use case metrics that can be used as defect-proneness indicators. Every heuristic defined a usual range for a use case metric, outside of which, the probability of use case of being defective increases.

Metric	Description
NOS	Number of steps of the use case (NOS=NOAS+NOSS+NOUS)
NOAS	Number of actor action steps of the use case
NOSS	Number of system action steps of the use case
NOUS	Number of use case action steps of the use case (inclusion or extension)
NOCS	Number of conditional steps of the use case
NOE	Number of exceptions of the use case
NOAS/NOS	Rate of actor action steps of the use case
NOSS/NOS	Rate of system action steps of the use case
NOUS/NOS	Rate of use case action steps of the use case
CC	Cyclomatic complexity of use case (NOCS+NOE+1)

Table 1. Use cases metrics for verification heuristics

The metrics and their original usual ranges were proposed after analyzing 414 non-verified use cases from our students, as described in [5]. Next, for each relevant metric [1] shown in table 1, its mean, standard deviation and usual range are indicated. Moreover, the rationale in which the definition of each heuristic is based on is also provided.

Metric: NOS $\in [0,\infty]$ ($\mu = 5.70$, $\sigma = 2.64$)
Usual range: [3,9]
Rationale: A use case with just a few steps is likely to be incomplete. Too many steps make the use case too complex to be understood.

Metric: NOAS/NOS $\in [0,1]$ ($\mu = 34.52\%$, $\sigma = 17.74\%$)
Usual range: [30%,70%]
Rationale: A use case describes a system-actors interactions, so NOAS/NOS and NOSS/NOS should be around 50%.

Metric: NOSS/NOS $\in [0,1]$ ($\mu = 59.71\%$, $\sigma = 18.80\%$)
Usual range: [40%,80%]
Rationale: Same as above.

[1] Relevant metrics have their names typed in boldface, the rest can be considered as auxiliary.

Metric: NOUS/NOS $\in [0,1]$ ($\mu = 5.77\%$, $\sigma = 10.42\%$)
Usual range: [0%,25%]
Rationale: An abusive use of use case relationships makes use cases difficult to understand.

Metric: CC $\in [1,\infty]$ ($\mu = 2.52\%$, $\sigma = 1.22\%$)
Usual range: [1,4]
Rationale: A high value of CC implies many conditional steps and exceptions, probably making the use case too complex.

These metrics have been defined for the use case model of REM [6]. In this model, a use case is basically seen as a sequence of steps. Actions of these steps can be one of three different classes: actor-action if the action is performed by an actor; system-action if the action is performed by the system, or a use case-action, if the action consists of performing another use case (*i.e.* a *inclusion* if it is not a conditional step, an *extension* otherwise).

In our previous work [2], the main goal was to analyse the validity of our heuristics. In that study, we verified 8 requirements specifications from our students, containing 127 use cases. These requirements specifications were verified searching for defects in requirements according to the list of desiderable properties of requirements proposed by Davis in [4].

In formal words, what we have studied is that, given a use case uc and one of the relevant use case metrics defined m in table 1:

$\exists\, inf, sup : \mathbb{R} \bullet m(uc) \notin [inf, sup] \Rightarrow P_{def}[uc] \gg \overline{P_{def}[uc]}$

where $P_{def}[uc]$ is the probability of the use case uc of containing any defect and $\overline{P_{def}[uc]}$ is the probability of use case uc of not containing any defect at all, i.e. $\overline{P_{def}[uc]} = 1 - P_{def}[uc]$. Notice that in this hypothesis we are only considering the case in which $m(uc) \notin [inf, sup]$, i.e. we are not stating anything about use cases with metric values in $[inf, sup]$. The reason for that is that there could be certain causes of defects not considered in the metric function and therefore not related to the metric value that could make uc to contain defects even when $m(uc)$ is in the usual range. In this work, we want to present how is possible (by means of the quality model definition) to establish a hypothesis that compare the quality of a use case considering whether use cases are inside $[inf, sup]$ or not, i.e. *inf* and *sup* can be used as thresholds metrics values.

As shown in figure 1, the empirical results of the earlier study presented in [2] have corroborated our initial assumption, i.e. most use cases with metrics values outside usual range are defective. After an exhaustive analysis of the empirical data, we have identified the main types of defects in use cases in which $m(uc) \notin [inf, sup]$.

As it can be seen in table 2, these causes are commonly

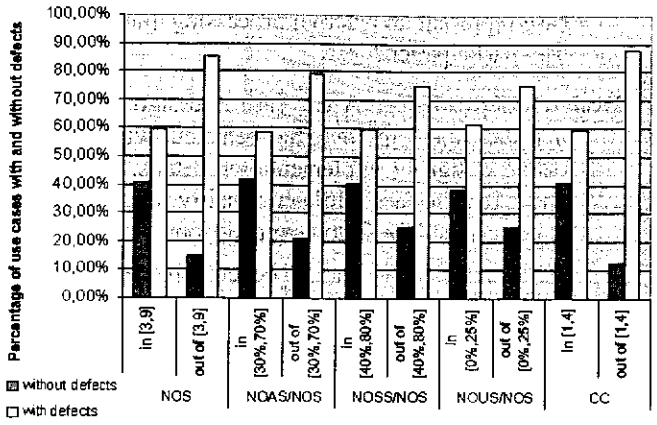

Figure 1. Empirical results of the earlier study

Metric	Value	Main causes
NOS	low (<3)	Incompleteness, too much modularity, triviality
NOS	high (>9)	Ambiguity, too many alternative steps, too much detach of actor and/or system steps
NOAS/NOS	low (<30%)	It does not include all that the system and the actors must accomplish to achieve the goal, too much detach of system steps, it express a batch process, it contains concrete references to user interface, it includes internal action of the system.
NOSS/NOS	high (>80%)	
NOAS/NOS	high (>70%)	It does not include all that the system and the actors must accomplish to achieve the goal, too much detach of actors steps, it include interactions between several actors or between actors and the environment of the system.
NOSS/NOS	low (<40%)	
NOUS/NOS	high (>25%)	Too much modularity, it includes concrete references to elements of the user interface, it express application menus.
CC	high (>4)	Understandability

Table 2. Main defect types in use cases outside usual range

different if the value of the metric is lower than *inf* or higher than *sup*.

On the other hand, since a use case represents an actor–system interaction, the value of the metrics NOSS/NOS and NOAS/NOS should be around 50%. This two metrics are related usually; i.e. a use case with a low value of NOSS/NOS usually has a high value of NOAS/NOS and vice versa. Because of that, the causes that provoke high values in one of these metrics usually coincide with causes that provokes low values in the other one (see table 2).

This analysis, summarised in table 2, has allowed us to define a RQM for use cases whose goal is to focus our defect detection technique on the set of defects types that use cases, which metric value is outside usual range, commonly contain. According to [1], the fact of focusing on specific types of defects should lead to more in–depth analysis of potential errors in the requirements document, although it provides a particular coverage of the document.

3. Use case quality model for verification in REM

The concept of RQM is very important in RQA. A RQM is, according to Gnesi [8], a set of rules against which to evaluate a requirements specification (syntactic, semantic rules, document structure and sentence structure characteristic). This is a list of desiderable properties (also known as *quality characteristics*) that requirements must accomplish, and whose lack is considered a defect.

Several approaches of RQMs have been proposed, some of them described in [10, 16]. As commented by Fabbrini in [7], *"not all the aspects related to evaluation of requirements quality can be addressed in the same way, with the same depth and with the same ease"*. Therefore, a specific requirements quality evaluation technique is defined to evaluate certain aspects of quality in requirements and with a specific requirements notation.

The use case quality model that we present in this paper has been developed using a *bottom–up* approach, i.e. first we have identified which are the defects that commonly occur in use cases pinpointed by the application of the heuristics summarised in section 2. Then, we have study how the presence of these types of defects implies the lack of some of the quality characteristics that are present in some of the following RQM proposals: on one hand the list of desiderable properties collected in [4], that proposes verification of semantic and non–semantic properties of requirements, such unambiguity, understanbility, etc. On the other hand, the review checklist proposed in [13], that checks specific properties of use cases, as the proper use of use cases relationships (*include* or *extends*). Also the style guidelines by Cockburn in [3] have influenced our quality model.

Next, we define the set of quality characteristics for REM use defined in RQM: For each quality characteristic we have identified a set of questions that check if the use case fulfils the corresponding quality characteristic.

Completeness: Adapting the definition of [4] to use cases, a use case is complete if all what the system and the actors must (externally) do to achieve the goal of the use case and all the system responses to abnormal situations are included. In order to know if a use case is complete, the following questions must be checked:

–Do every requests of actor to system obtain system response and vice versa?

–Does use case contain all the set of possible scenarios for achieving a goal?

–Do alternatives of the use case specify all different

results to ordinary sequence?

–Does the use case consider all the possible exceptions to ordinary sequence?

No ambiguity: A use case is considered as unambiguous if it has only one possible interpretation. In order to know if a use case is ambiguous specific questions has not been developed.

Understandability: A use case is understandable if all classes of readers (customer, users, project manager, software developers and testers) can easily comprenhed the meaning of the use case with a minimum of author's explanation [4]. In order to know if a use case is understandable, the following questions must be checked:

–Is it possible to read the use case without excessive re–reading?

–Is difficult to follow the use case sequence because of its *include* or *extends relationships?*

–Is difficult to follow the use case sequence because of its alternative steps?

–Are the actor or system steps too detached?

Conciseness: A use case is considered to be concise if unnecessary information is not included. In order to know if a use case is concise, the following questions must be checked:

–Could its meaning be expressed with less words?

–Are there elements that could be obviated?

–Are unnecessary remarks that make difficult the lecture included?

–Are interactions between actors and elements of the system environment or between primary actors and secondary actors included?

No triviality: A use case is not trivial if its sequence of steps leads the actor to achieve his goal. In REM the level of abstraction of the goal associated to use cases is *user goal level* [3]. Therefore, the realization of the use case must aid the user to achieve a certain goal. In order to know if a use case is not trivial, the following questions must be checked:

–Is the use case named as user goal that the system will support?

–Does the use case reflect a *result of value* to actor, not a set of trivial interactions?

Proper use of use cases technique Use cases technique is proper if the use case represents a functional requirement that requires users interaction [9]. The question associated to this quality characteristic is: Does the use case represent an external processing that require any user participation?

Design independence: A use case is design independent if it describes only external behavour without anticipating design or implementation details. In order to check if a use case fulfils this quality characteristic we have designed the following questions:

–Does the use case contain concrete references to user interface elements?

–Does the use case describe what is to be accomplished by the system but not how?

–Does the use case try anticipating how application menus will be like?

4. Empirical assessment of the defect detection technique

Once RQM for use cases has been defined, we have tried to check if focusing on our quality model defects types we are able to validate a more solide hypothesis that the proposed in section 2, establishing that for a concrete set of defects types, the usual range determine whether the probability of a use case of containing defects is very high, being this probability minimum inside normal range. We assume this because our RQM collected the defects types that ussually occur in use cases which value of any metric is outside usual range proposed by the heuristics. For that purpose, we define a hypothesis based on the subset of defects that commonly occurs in use cases with anomalous values of the relevant metrics shown in table 1.

For each metric m, D_{inf} and D_{sup} are defined as the set of defects types that usually occur when the value of m is lower than *inf* or higher than *sup*, respectively, according to table 2. Therefore, $D = D_{inf} \cup D_{sup}$ stands for the set of defects types related to an anomalous values of a concrete metric.

Formally, once defined usual range $[inf, sup]$, given a use case uc and a metric m_D predictive of defects of types D, the hypothesis is formulated thus:

$$m_D(uc) < inf \Rightarrow P_{D_{inf}}[uc] \gg \overline{P_{D_{inf}}[uc]}$$
$$\wedge$$
$$m_D(uc) \in [inf, sup] \Rightarrow \overline{P_D[uc]} \gg P_D[uc]$$
$$\wedge$$
$$m_D(uc) > sup \Rightarrow P_{D_{sup}}[uc] \gg \overline{P_{D_{sup}}[uc]}$$

where $P_D[uc]$ represents the probability of a use case uc of containing one or more defects included in the set D and $\overline{P_D[uc]}$ stands for the probability of uc of not containing any defect type included in D.

In order to study this hypothesis, we have designed a checklist based on our RQM. Some experts in requirements

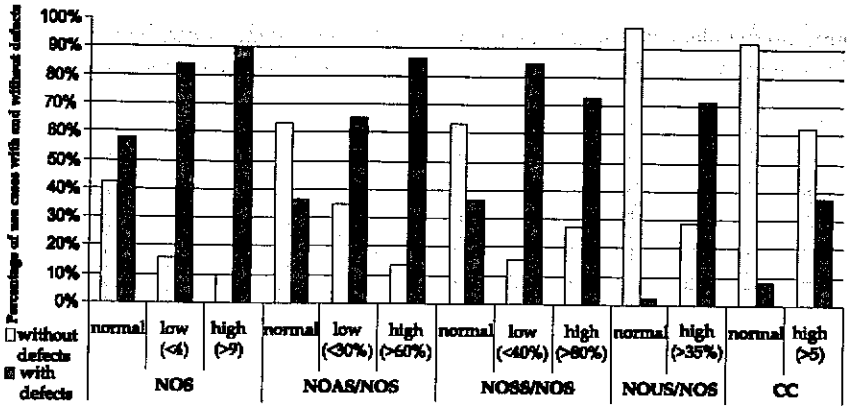

Figure 2. Empirical results when use cases are verified with our quality model

quality have worked as reviewers using this checklist to verify a set of requirements specifications taken by our students (6 requirements document, 131 use cases in total). Then, we have calculated the values of the metrics of these use cases and analysed the data. Figure 2 shows the result of the analysis. In the rest of this section, we summarise the main findings that we have obtained for each heuristic.

Empirical review of the NOS heuristic: The analysis of empirical data confirmed that the probability of use cases outside normal range of containing defects is very high. Nevertheless, the presence of defective use cases with NOS in $[3, 9]$ is significatively high (equal to 61,32%). We have tried to correct this situation by changing the lower limit of the interval. Taking $[4, 9]$ as the usual range we have obtained 57,77% of use cases inside the normal range are defective (as can be seen in figure 2). The result is not too much better, this is because, as we have commented in [2], this heuristic is dramatically affected by the writing style of use cases writers, above all if several actions are grouped in the same step or if they are considered as different steps. In order to avoid this influence, it seems suitable to consider different intervals to NOS metric according to the writing style.

Empirical review of the NOAS/NOS heuristic: The initial usual range for the NOAS/NOS metric was $[30\%, 70\%]$. With this interval, 42,71% of use cases outside normal range were defective. We observed during the analysis of empirical data that some use cases inside this interval but with metric value close to 70% used to present defects. Because of that, we have changed the usual range to $[30\%, 60\%]$. As shown in figure 2, empirical data confirms the assumption of the heuristics because use cases outside normal range present a higher probability of containing defects than the ones inside.

Empirical review of the NOSS/NOS heuristic: As shown in figure 2, empirical data confirms the assumption of the heuristics because use cases outside normal range present a very high probability of containing defects than the ones inside.

Empirical review of the NOUS/NOS heuristic: The analysis of empirical data has made necessary a fitting of the initial range of usual values of NOUS/NOS. Taking $[0\%, 25\%]$, only 25% of use cases outside normal range contained defects. As figure 2 shows, taking $[0\%, 35\%]$, 71,43% of use cases contained defects. This difference is because several use cases with NOUS/NOS inside $[25\%, 35\%]$ have some "identification of actors" steps which are specified in a *abstract* use case to avoid redundant sequence of steps. This *abstract* use case is included in the use case by a *include* relationship increasing the value of the rate NOUS/NOS. This situation does not present any source of defects. Therefore, $[0\%, 35\%]$ will be considered the usual range of the metric NOUS/NOS.

Empirical review of the CC heuristic: The analysis of empirical data has made necessary a fitting of the upper limit of initial usual range. Since the metric is defined as NOCS+NOE+1 it is very easy that its value equals to 4. For example having three exceptions makes CC equals to 4 and this is not necessarly a source of any defect. As shown in figure 2, taking the usual range as $[1, 5]$, 62,50% of use cases outside this interval do not contain any defects of types collected in table 2. A detailed analysis of empirical data has allowed us to check the main causes of this situation:

- The existence of exceptions does not always make more difficult the reading of the use case. Because

of that, we think that it is convenient to redefine the CC metric weighting up the elements of the equation, having NOCS more weight than NOE.

- Some writers of use cases use a conditional step to specify that the system *checks* a condition, i.e. to verify that some business rule are satisfied. Therefore, CC does not always reflect the number of alternative paths of the use case. According to [3], this situation could be avoid by means of replacing sentences like *"If the password is correct the system go ahead"* by *"The system verifies that the password is correct"*. This is a new type of defect not collected in our quality model that we have to consider in further works.

- The use case model used by the REM tool, and on which the heuristics are based on, it is not especially designed for use cases with alternative paths involving more than one step. In these situations, the CC value is artificially increased since it is necessary to repeat the same condition several times). This fact reveals the convenience of changing our original metamodel of use cases by including the possibility of having groups of steps under the same condition.

5 Conclusions and future work

In this paper we have presented a quality model for use cases and an empirical validation of some metrics–based heuristics for verifying use cases. These heuristics rely on the assumption that for each metric there exists an interval outside of which some types of defects usually occur. In order to validate the heuristics, we have established a solide hypothesis that allows us to compare the situations of use cases whether they are inside the interval or not. The analysis of the empirical data has not only allowed us to verify that use cases outside normal range have a higher probability of containing some defects types than the ones inside, but also has allowed to adjust some or their metrics thresholds values and to identify new defects types.

Our future work is focused on confirming this hypothesis on data from "real projects" allowing us to corroborate that these metrics are actual early indicators of use cases defect–proneness. After the confirmation of the hypothesis, it would be interesting to perform a family of controlled experiments in order to know if the usage of the heuristics can really help requirements reviewers to find defects in use cases.

On the other hand, during the development of our quality model we have been able to see that, as in conceptual modelling [14], some quality characteristics are mutually dependent, i.e. the occurrence of one type of defect could provoke the occurence of others, and the elimination of one of them could generate the disappearance of others.

Acknowledgments We want to thanks P. Barrios and L. G. Salgado, members of SQA of Sadiel S.A., for their work as reviewers of the requirements documents.

References

[1] V. R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sørumgård, and M. V. Zelkowitz. The Empirical Insvestigation of Perspective–Based Reading. *Empirical Software Engineering*, 1(2):133–164, 1996.

[2] B. Bernárdez, A. Durán, and M. Genero. An Empirical Review of Use Case Metrics for Requirements Verification. In *1st Software Measurement European Forum*, Rome, 2004.

[3] A. Cockburn. *Writing Effective Use Cases*. Addison–Wesley, 2001.

[4] A. Davis *et al.* Identifying and measuring quality in software requirements specification. In *1st Int'l Software Metrics Symposium*, Los Alamitos, California, 1993. IEEE Computer Society Press.

[5] A. Durán, A. Ruiz, B. Bernárdez, and M. Toro. Verifying Software Requirements with XSLT. *ACM Software Engineering Notes*, 29(1):39–44, 2002.

[6] A. Durán, A. Ruiz-Cortés, R. Corchuelo, and M. Toro. Supporting Requirements Verification using XSLT. In *IEEE Joint International Requirements Engineering Conference (RE'02)*, Essen, Germany, 2002.

[7] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami. The Linguistic Approach to the Natural Requirements Quality: Benefit of the use of an Automatic Tool. In *26th Annual NASA Goddard Software Engineering Workshop*, Greenbelt, Maryland, 2001.

[8] S. Gnesi. Analysis of Software Requirements, 2000. Available in `http://www.iei.pi.cnr.it/ERI/iei/qmslideseri.ppt`.

[9] B. Henderson-Seller, D. Zowghi, T. Klemola, and S. Parasuram. Sizing Use Cases: How to Create a Standard Metrical Approach. In *8th International Conference on Object–Oriented Information Systems*, Springer LNCS 2425, 2002.

[10] IEEE. IEEE Recommended Practice for Software Requirements Specifications. IEEE/ANSI Standard 830–1993, Institute of Electrical and Electronics Engineers, 1993.

[11] ISO/IEC. ISO 9126.1 Software Engineering–Product quality—Quality model. International Standard 9126.1–2001, International Organization for Standarization, 2001.

[12] G. Kontoya and I. Sommerville. Requirements Engineering with Viewpoints. *Software Engineering Journal*, 11(1), Enero 1996.

[13] S. Lilly. Use Case–Based Requirements: Review Checklist. Technical report, SRA International, Inc., 1999.

[14] D. L. Moody, G. G. Shanks, and P. Darke. Improving the quality of entity relationship models - experience in research and practice. In *Conceptual Modeling - ER '98, 17th International Conference on Conceptual Modeling, Singapore, November 16-19, 1998*, Springer LNCS 1507, pages 255–276, 1998.

[15] K. Pohl. Requirements Engineering: An Overview. *Encyclopedia of Computer Science and Technology*, 36, 1997.

[16] G. M. Schneider, J. Martin, and W. T. Tsai. An Experimental Study of Fault Detection In User Requirements Documents. *ACM Transactions on Software Engineering and Methodology*, 1(2):188–204, 1992.