Empirical Evaluation and Review of a Metrics–Based Approach for Use Case Verification*

Beatriz Bernárdez and Amador Durán

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Sevilla
E.T.S. de Ingeniería Informática
Avda. Reina Mercedes s/n, 41012 Sevilla (Spain)
Email: {beat,amador}@lsi.us.es

Marcela Genero

Grupo de investigación ALARCOS
Universidad de Castilla–La Mancha
Escuela Superior de Informática
Paseo de la Universidad 4, 13071 Ciudad Real (Spain)
Email: marcela.genero@uclm.es

In this article, an empirical evaluation and review of some metrics–based verification heuristics for use cases are presented. This evaluation is based on empirical data collected from requirements documents developed by Software Engineering students at the University of Seville using REM, a free XML–based requirements management tool developed by one of the authors. The analysis of the empirical data has not only confirmed the validity of the intuitions that gave rise to the verification heuristics but has also made possible to review and adjust some of their parameters, consequently enhancing their accuracy in predicting defects in use cases. One of the most interesting results derived from the analysis of empirical data is a number of possible enhancements that could be applied to the underlying metamodel of use cases implemented in REM, in which the heuristics are based on, thus providing an important feedback to our current research in Requirements Engineering.

ACM Classification: D.2.1 (Software – Software Engineering – Requirements/Specifications), D.2.8 (Software – Software Engineering – Metrics), D.2.9 (Software – Software Engineering – Management – Software Quality Assurance)

1. INTRODUCTION

It is widely acknowledged within the Software Engineering community that Requirements Engineering (RE) is one of the most critical activities of a quality–driven software development process. Early detection of problems in requirements has a very important impact on the overall quality of the software development process, as shown by Kamsties and Rombach (1997). This early detection is especially important to avoid budget overruns, as confirmed by Boehm (1981).

* This work is partially funded by the following projects: AgilWeb (TIC 2003–02737), Tamansi (PCB–02–001) and Messenger (PCC–03–003–1).

Manuscript received: 15 December 2003
Communicating Editor: Didar Zowghi

Traditionally, requirements verification and validation have been considered as the only requirements quality assurance (RQA) activities in the RE process. Nevertheless, a wider vision of RQA like the one we presented in Durán *et al* (2002b), considers not only verification and validation but also requirements analysis as a RQA activity. The reason for this classification of requirements analysis is that its main goal, as defined by Kontoya and Sommerville (1997), is finding problems and conflicts in the draft requirements so they could be solved and therefore requirements quality increased.

In the UML activity diagram in Figure 1, a RE process model taking into account our wider vision of RQA is shown. In this RE process model, we consider that the goal of requirements verification is to increase the so–called *internal quality* (ISO/IEC, 2001) of the requirements documents according to applicable standards in the developer organization; 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 (1997).

In the context of our RQA model, more specifically in requirements verification, this article presents an empirical evaluation and review of some metrics–based heuristics for use case verification initially presented in Durán *et al* (2002a). The evaluation and later review are based on empirical data collected from requirements documents developed by fourth–year Software Engineering students at the University of Seville using REM (Durán, 2004), a free XML–based requirements management tool.

The rest of this article begins with a brief description of the REM use case metamodel. Some use case metrics which are relevant for the verification heuristics and the verification heuristics themselves are described in Section 3. In Section 4, the analysis of the empirical data collected for the evaluation of the verification heuristics and some of the empirically driven reviews are presented. Some related work is commented in Section 5 and, finally, in Section 6 some conclusions and future work are presented.

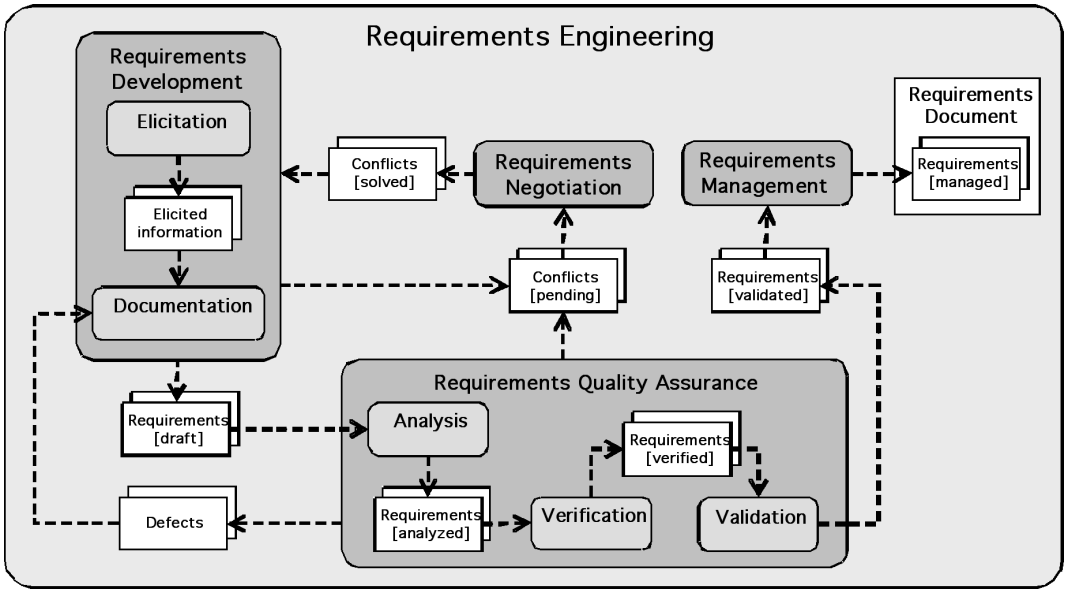

Figure 1: Requirements Engineering process model

2. REM USE CASE METAMODEL

Using REM, a requirements engineer can develop requirements documents containing, amongst other elements, use cases, whose metamodel is shown in the UML class diagram in Figure 2.

REM use cases, whose user interface is shown in Figure 3, are based on Durán's template for use cases, formerly published in Durán *et al* (1999). One of the characteristics of this template is that for a number of its elements some *linguistic patterns* are provided, thus easing use case writing. A simplified example of the use case template is shown in Figure 4, in which some requirements management attributes like version, authors, etc. have been omitted for the sake of brevity. A description of linguistic patterns is out of the scope of this article, but the interested reader can see Durán *et al* (1999) for details. A more extensive work on the use of linguistic patterns for use cases can be found in Ben Achour *et al* (1999), one of the results of the CREWS project.

Apart from inherited requirements attributes, a use case in REM is basically composed of a *triggering event*, a *precondition*, a *postcondition*, and an *ordinary sequence* of steps describing interactions leading to a successful end. In turn, steps are composed of one *action* and may have a condition. Three classes of actions are considered: *system actions* performed by the system, *actor actions* performed by one actor, and *use case actions* in which another use case is performed, i.e. UML

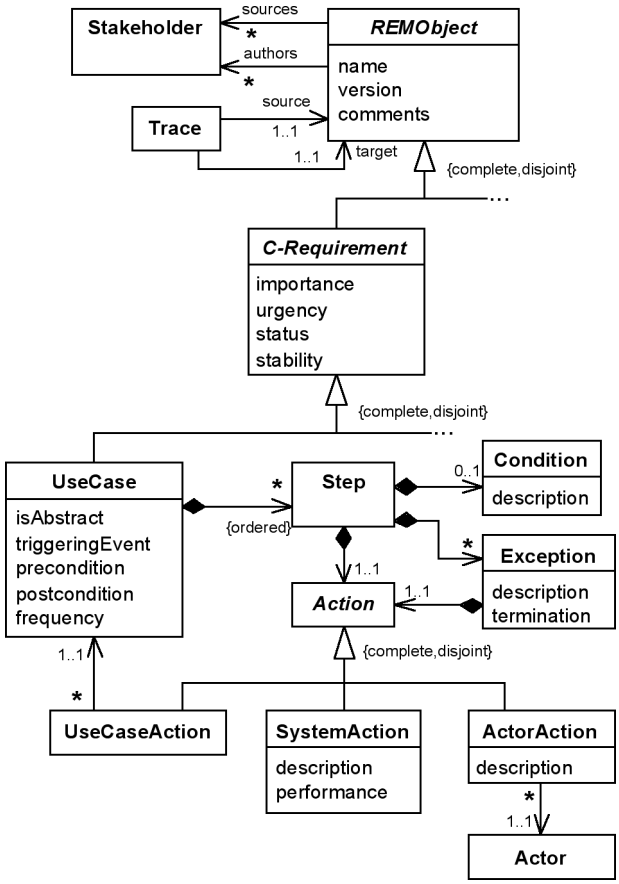

Figure 2: REM use case metamodel

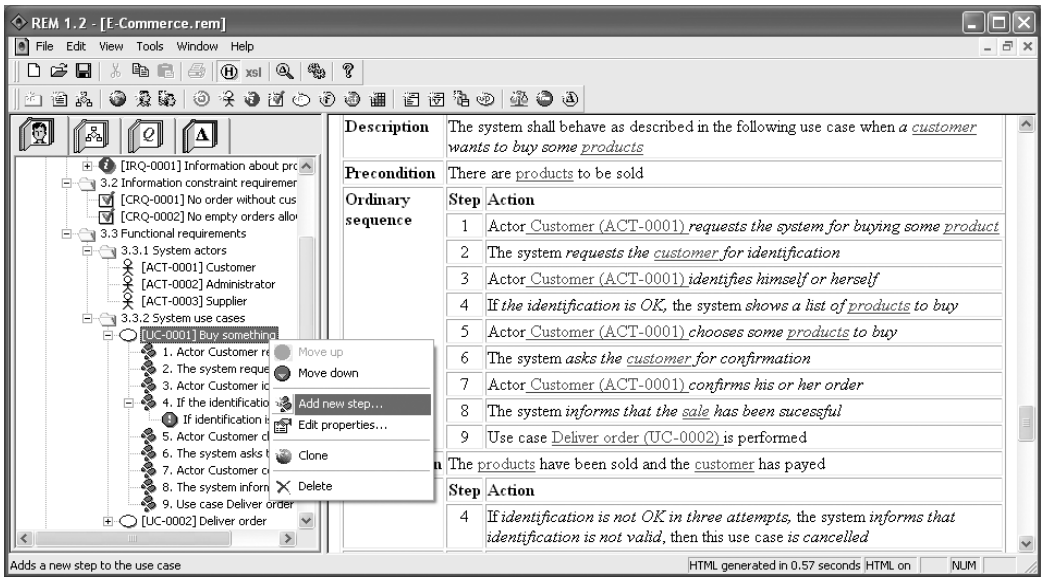

Figure 3: REM user interface for use cases

inclusions or *extensions*, depending on whether the step is conditional or not (OMG, 2003). Steps may also have attached *exceptions*, which are composed of an exception condition (modelled by the *description* attribute, see Figure 2), an action (of the same class than step actions) describing the exception treatment, and a *termination* attribute indicating whether the use case is resumed or canceled after the performance of the indicated action.

3. VERIFICATION HEURISTICS

As described in Durán *et al* (2002b), the experience of some of the authors in the verification of use cases developed by students using REM led to the definition of several metrics–based defect detection heuristics. These heuristics are based on a simple idea: there are some use case metrics which are indicators of defect–proneness and for which a range of *usual* values can be defined; if for a given use case, its metric value is out of its corresponding usual range, then the use case is considered as potentially defective and it should therefore be checked.

Notice that the heuristics make no hypothesis about use cases with metrics value in usual ranges. The reason for that is that there could be certain causes of defects not considered in the metrics definitions, and therefore not related to the metrics values, that could make use cases to contain defects even when there seems to be no problem from the heuristic point of view.

In Table 1, the definition of the metrics used in the verification heuristics is shown. As described in Durán *et al* (2002b), these metrics can be easily computed in XSLT and the heuristics can be applied automatically in REM using the XML representation of use cases.

The heuristics whose evaluation and review are presented in this article are described below. The usual metrics ranges were chosen after a statistical analysis of 414 use cases developed by fourth–year Software Engineering students using REM. For the used metrics, their initial usual ranges were defined as *[lower quartile, upper quartile]*, i.e. the box in a box plot. See Figure 5 for the box plot corresponding to NOAS_RATE, NOSS_RATE and NOUS_RATE metrics. For details on other metrics–based verification heuristics whose evaluation is not included in this article, see Durán *et al* (2002b).

UC-0015	**Register Book Loan**	
Dependencies	• OBJ–0001 *To manage book loans* (objective) • OBJ–0005 *To know library users' preferences* (objective) • CRQ–0003 *Maximum number of simultaneous loans* (business rule) • CRQ–0014 *Return date for a loan* (business rule)	
Description	The system shall behave as described in the following use case when *a library user requests a loan of one or more books*.	
Precondition	*The library user has been identified by means of his or her identity card, has picked up the books to loan from the shelves, has not reached the maximum number of simultaneous loans and has no penalty.*	
Ordinary Sequence	**Step**	**Action**
	1	Actor *librarian requests the system for starting the book loan registering process*.
	2	The system *requests for the identification of the library user requesting a loan*.
	3	Actor *librarian provides identification data of the library user to the system*.
	4	The system *requests for the identification of the books to be loaned*.
	5	Actor *librarian provides identification data of the books to be loan to the system*.
	6	The system *displays the return date for each of the books to be loan and requests loan confirmation for each of them*.
	7	Actor *library user confirms the librarian which books he or she wants to loan after knowing return dates*.
	8	Actor *librarian re–confirms the book loans confirmed by the library user to the system*.
	9	The system *informs that the book loans have been successfully registered*.
Postcondition	*The library user can take the loaned books away and the system has registered the book loans*.	
Exceptions	**Step**	**Action**
	3	If *the library user has already reached the maximum number of simultaneous loans or has a penalty*, the system *informs of the situation*, then this use case *is cancelled*.
Comments	*The maximum number of simultaneous book loans and the loan period depend on the library policy and can change in the future. See business rules CRQ–0003 y CRQ– 0014*.	

Figure 4: Use case example using a simplified version of Durán's template

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 (*inclusions* or *extensions*)
NOCS	Number of conditional steps of the use case
NOE	Number of exceptions of the use case
NOAS_RATE	Rate of actor action steps of the use case (NOAS/NOS)
NOSS_RATE	Rate of system action steps of the use case (NOSS/NOS)
NOUS_RATE	Rate of use case action steps of the use case (NOUS/NOS)
CC	Cyclomatic complexity of the use case (NOCS+NOE+1)

Table 1: Use case metrics description

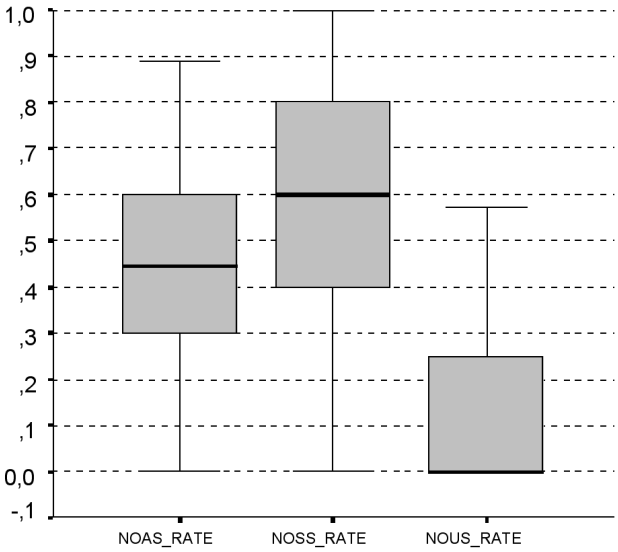

Figure 5: Box plot of NOAS_RATE, NOSS_RATE and NOUS_RATE metrics

3.1 Verification heuristics summary

Heuristic (A): NOS should be in [3, 9].
Rationale: A use case with just a few steps is likely to be incomplete. Too many steps usually indicate too low level of detail and make the use case too complex to be understood and defect–prone.

Heuristic (B): NOAS_RATE should be in [30%, 60%].
Heuristic (C): NOSS_RATE should be in [40%, 80%].
Rationale: A use case describes system–actor interactions, so the rate of actor and system steps should be around 50%, considering also steps whose action is either an inclusion of or an extension from another use case.

Heuristic (D): NOUS_RATE should be in [0%, 25%].
Rationale: An abusive use of use case relationships makes use cases difficult to understand – customers and users are not familiar with procedure call semantics. Use them to avoid repetition of common steps only.

Heuristic (E): CC should be in [1, 4].
Rationale: The basic idea behind this heuristic is that use cases with many alternative paths are usually difficult to understand and sometimes illegible. The number of alternative paths of a use case can be defined as the cyclomatic complexity (CC) of a use case in the same sense that McCabe (1976) defined CC for source code. As McCabe's CC, use cases CC can be defined as the number of decision points plus one, i.e. the number of conditional steps plus the number of exceptions plus one (see Table 1).

4. EMPIRICAL EVALUATION AND REVIEW

The empirical evaluation and analysis of the verification heuristics were carried out by manually verifying 127 use cases in eight requirements documents developed using REM by fourth–year students of Software Engineering at the University of Seville. The students had a basic knowledge on requirements engineering and use case concepts, especially on Duran's template (see Figure 4); they were able to use REM, but were not aware of the verification heuristics. None of them had previous experience on writing neither requirements specifications in general nor use cases in particular. The general problem domain of all requirements documents were information systems, although students were free to chose a specific problem domain. As a result, the specific problem domains of the eight requirements documents were quite different from one another.

The manual verification of the requirements documents was performed by the first author of this article using the lists of desirable properties by Davis *et al* (1993) and Lilly (1999). After manual verification, the verification results were compared with metrics values automatically computed by REM. The general results are depicted in Figure 6. The specific results for each assessed heuristic are described below.

4.1 Evaluation of Heuristic A (NOS metric)

This heuristic, which follows Cockburn's (2001) recommendations on use case length, was widely validated by empirical data: 85% of the use cases out of the usual range of the NOS metric were identified as defective (see Figure 6). A subsequent analysis of the detected defects revealed that

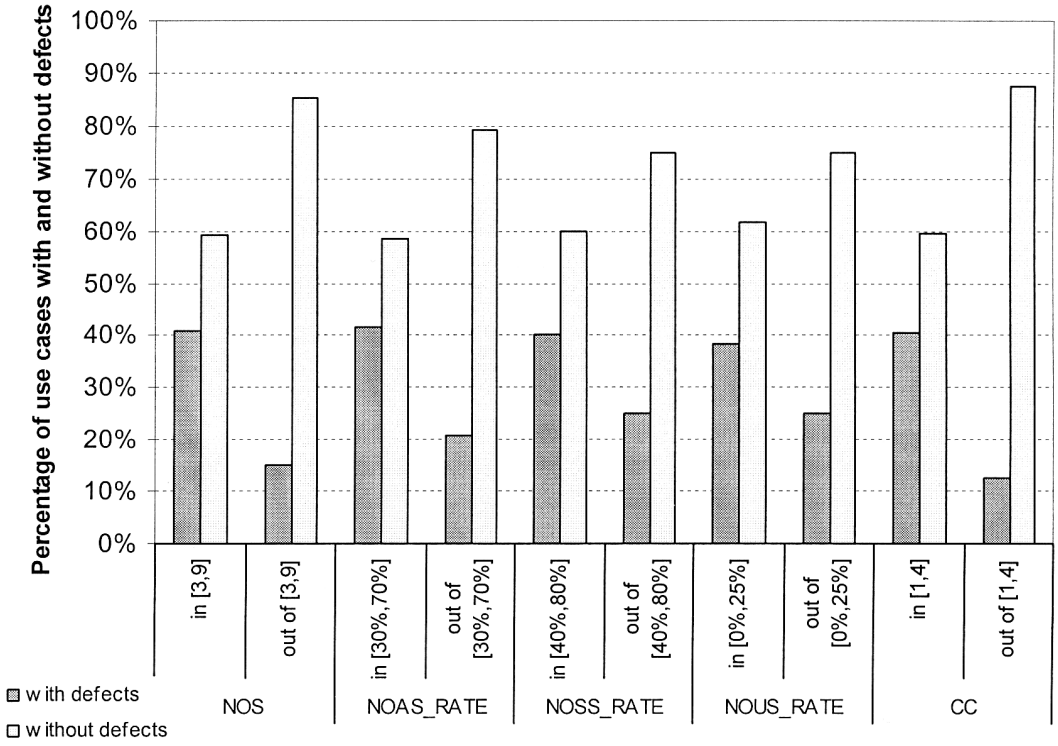

Figure 6: Empirical evaluation of verification heuristics: general results

whereas those use cases with too low NOS were usually either incomplete or trivial or described no interaction at all, use cases with too high NOS were usually at a too low level of detail, as commented by Lilly (1999), i.e. *non concise* according to Davis *et al* (1993) quality model.

On the other hand, for the most part of the 15% of the use cases that were wrongly identified as potentially defective, NOS was high because of the *writing style* or because of the presence of *actor–to–actor* interactions.

4.1.1 Writing Style

The writing style has been identified as a very important factor for the accuracy of heuristic A. Whereas some students used only one step for specifying a sequence of consecutive actions carried out either by the system or by a given actor, others used one step for each action, thus increasing NOS (see Cockburn, 2001) for a comparison of both writing styles). This heuristic was designed with the former writing style in mind, but as commented in Bernárdez, Durán and Genero (2004a), a further analysis for identifying another usual range for the latter writing style is currently being carried out.

4.1.2 Actor–to–Actor Interactions

The inclusion of actor–to–actor interactions cannot be in anyway considered as a defect in use cases, but it dramatically affects the accuracy of heuristic A by increasing NOS without making use cases defective (heuristics B and C are also affected, see below).

We consider that actor–to–actor interactions add interesting information to use cases, but the results reveal the convenience of modifying the REM use case metamodel shown in Figure 2 by specializing class ActorAction in two new disjoint classes: ActorActorAction for actor–to–actor actions, and ActorSystemAction, for actor–to–system actions. In this way, we could redefine the metric NOAS as counting only actor–to–system steps.

4.2 Evaluation of Heuristics B (NOAS_RATE) and C (NOSS_RATE)

Heuristics B and C were also confirmed by empirical data with 80% and 70% respectively of defective use cases out of usual ranges. Both heuristics are tightly coupled because a high value of one of them implies a low value of the other.

Use cases with high NOAS_RATE (and therefore low NOSS_RATE) are usually incomplete use cases in which system behavior has been omitted or non–defective use cases in which a lot of actor–to–actor interactions have been considered (usually *business use cases*), as commented above. Notice also that the writing style in which consecutive actions of an actor are specified in several consecutive steps makes use cases present a high NOAS_RATE value but does not necessarily imply the presence of defects.

On the other hand, use cases with high NOSS_RATE (and therefore low NOAS_RATE) are usually defective use cases describing *batch* processes or internal system actions only without considering actor participation. The problem in these situations is that the use case is not actually a use case and it should have been expressed as a functional requirement in plain text.

Abstract use cases (Jacobson, Griss and Jonsson, 1997), i.e. use cases containing common steps that have been extracted from other concrete use cases in order to avoid redundancy, are an exception to heuristics B and C. Sometimes, non–defective abstract use cases present very low values of either NOSS_RATE or NOAS_RATE because they are simply a fragment of another use cases. As an incomplete use case, an abstract use case does not present the usual rates of different types of steps than concrete use cases do. This fact has made us review both heuristics so they should be applied to concrete use cases only.

4.3 Evaluation of Heuristic D (NOUS_RATE)

Heuristic D, confirmed with 75% of use cases out of usual range being defective, usually detects use cases with a high number of extensions due to a *menu–like* structure, i.e. use cases without a clear goal in which, depending on an actor choice, a number of different use cases are performed. Another source of defects detected by this heuristic is the *programmer–like* way of structuring use cases. Some students developed a very complex model of use cases with lots of inclusions and extensions that were not strictly necessary but that, in their own words, made use case model more *modular*. Obviously, that kind of *modularity* makes use cases almost impossible to understand, especially for customers and users.

Nevertheless, most of the 25% of non–defective use cases out of usual range were use cases in which the impossibility of the REM metamodel of representing *conditional blocks* of steps, i.e. a group of steps with the same condition, forced students to create extending use cases in order to avoid the repetition of the same condition along several consecutive steps. The same happened when the treatment of an exceptional situation required more than one single action to be performed (the metamodel in Figure 2 only allows one action to be associated to an exception). In this case, students were also forced to create an extending use case that was performed when the exception occurred.

4.4 Evaluation of Heuristic E (CC)

This heuristic was confirmed by empirical data with 87% of use cases out of usual range being defective. The usual cause of defect was the abusive use of conditional steps of the form "*if <condition>, the system goes to step X*", making use cases almost impossible to understand. As commented above for heuristic D, the lack of conditional blocks was the usual case for abnormally high values of CC in non–defective use cases when students decided not to create an extending use case but repeating the same condition along several steps, thus artificially increasing CC value.

In our experience with some local software development companies, we have found that sometimes when use cases have a high CC value, analysts add an UML activity diagram semantically equivalent to the use case for the sake of comprehensibility. From our point of view, this solution has some disadvantages like repeating the same task twice, the need of verifying consistency between both representations of the use case and the added confusion that can cause to customers and users.

Nevertheless, the evaluation of the heuristics has made evident that the REM use case metamodel (see Figure 2), as well as Leite *et al* (2000) scenarios, are not especially designed for use cases with alternative paths involving more than one step and must therefore be enhanced by including the possibility of having blocks of steps under the same condition. Another possibility, as it is commented by Henderson-Sellers *et al* (2002) and Lilly (1999), is to consider a specific use case section for the alternative paths instead of mixing alternative steps within the ordinary sequence. Something that, in our opinion, fragments use cases and makes them more difficult to understand.

5. RELATED WORK

As summarized in Bernárdez, Durán and Genero (2004b), several studies on use case metrics can be found in the literature. Most of them propose to define metrics such as the number of use cases, the number of actors or the number of steps to measure size or complexity of requirements specification and use those measures as predictors of some attributes of the system to be built, such as effort or size (Feldt, 2000; Henderson-Sellers *et al*, 2002; Marchesi, 1998; Schneider and Winters, 1998). Other proposals, (Alexander, 2001; Kim and Boldyreff, 2002) focused on the RE process, define use case metrics related to requirements importance, status or progress in order to highlight RE process areas that need attention.

Nevertheless, our main goal is to automate as much as possible the requirements verification process. In this area, there are some proposals based on Natural Language Processing (NLP) like Fabbrini *et al* (1998). The usual goals of NLP when applied to RE are to know the used vocabulary, the writing style, detecting ambiguity (degree of syntactic and semantic uncertainty of the sentence), the information conveyed by requirements, discovering underspecifications, missing information and unconnected statements.

NLP is also adaptable to use case verification. For example, Fantechi *et al* (2002) present a set of metrics to quality evaluation of use cases. For Fantechi *et al* (2002), a use case is seen as a set of sentences (with an underlying template) for which metrics such as the average number of words per sentence or the average of sentences containing acronyms not explicitly and completely explained can be collected in order to perform a quantitative evaluation of a requirements document, especially related to expressiveness.

In Zowghi, Gervasi and McRae (2001) an approach to discover inconsistencies in requirements by means of an automatic processing is provided. The process consists of two steps, both supported by a tool. In the first step, the natural language sentences are translated into predicate logic. Then, the logic specification is instantiated and the behavior of the system under certain conditions is simulated. The result of the simulation is used by the requirements engineers or by the users for analyzing whether the simulated behavior was the desired or not, and consequently if the set of requirements involved in the simulation can be considered as acceptable.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have presented an evaluation and review of some of the metrics– based verification heuristics for use cases proposed in Durán *et al* (2002a) and Durán *et al* (2002b). These heuristics are based on the intuition that some use cases metrics are predictors of defects and that there is a usual range of values for each defect– predictive metric. If the metric value for a given use case is not in the usual range, the probability of the use case of being defective is higher than on the contrary. The analysis of the empirical data has not only allowed us to verify that use cases out of usual range have a high probability of containing defects but also to identify new defects types and reveal the convenience of modifying some aspects of the original REM use case metamodel.

Our future work will be focused on defining a suitable requirements quality model containing a detailed taxonomy of defects usually detected by the heuristics. Using that quality model, we want to make the heuristics more specific so they could become actual predictors of the presence, or absence, of specific types of defects.

Furthermore, it is convenient to evaluate the evolved heuristics using data from real projects. After the evaluation, it would be interesting to perform a family of controlled experiments in order to know if the usage of the heuristics can really help RQA people to find defects in use cases.

REFERENCES

ALEXANDER, I. (2001): Visualising requirements in UML. Accessed March, 2004. http://easyweb.easynet.co.uk/~iany/consultancy/reqts_in_uml/reqts_in_uml.htm

BEN ACHOUR, C., ROLLAND, C., MAIDEN, N. A. M. and SOUVEYET, C. (1999): Guiding use case authoring: Results of an empirical study, *Proceedings of the 4th IEEE International Symposium on Requirements Engineering*, Limerick.

BERNÁRDEZ, B., DURÁN, A. and GENERO, M. (2004a): An empirical review of use case metrics for requirements verification, *Proceedings of the 1st Software Measurement European Forum* (SMEF), Rome, Italy.

BERNÁRDEZ, B., DURÁN, A. and GENERO, M. (2004b): *Metrics for Software Conceptual Models*, Imperial College, chapter Metrics for Use Cases: A Survey of Current Proposals. In press.

BOEHM, B. W. (1981): *Software Engineering Economics*, Prentice–Hall.

COCKBURN, A. (2001): *Writing Effective Use Cases*, Addison–Wesley.

DAVIS, A., OVERMYER, S., JORDAN, K., CARUSO, J., DANDASHI, F., DINH, A., KINCAID, G., LEDEBOER, G., REYNOLS, P., SITARAN, P., TA, A. and THEOFANOS, M. (1993): Identifying and measuring quality in software requirements specifications, *Proceedings of the 1st International Software Metrics Symposium*, IEEE Computer Society Press, Los Alamitos, California.

DURÁN, A. (2004): REM web site. Accessed March, 2004. http://rem.lsi.us.es/REM

DURÁN, A., BERNÁRDEZ, B., RUIZ, A. and TORO, M. (1999): A requirements elicitation approach based in templates and patterns, *Proceedings of the Workshop in Requirements* (WER), Buenos Aires.

DURÁN, A., RUIZ, A., BERNÁRDEZ, B. and TORO, M. (2002a): Verifying software requirements with XSLT, *ACM Software Engineering Notes* 27(1): 39–44.

DURÁN, A., RUIZ-CORTÉS, A., CORCHUELO, R. and TORO, M. (2002b): Supporting requirements verificarion using XSLT, *Proceedings of the IEEE Joint International Requirements Engineering Conference (RE)*, IEEE Computer Society Press, Essen, Germany, 141–152.

FABBRINI, F., FUSANI, M., GERVASI, V., GNESI, S. and RUGGIERI, S. (1998): Achieving quality in natural language requirements, *Proceedings of the 11th International Software Quality Week*, San Francisco.

FANTECHI, A., GNESI, S., LAMI, G. and MACARI, A. (2002): Application of linguistic techniques for use case analysis, *Proceedings of the IEEE Joint International Requirements Engineering Conference (RE)*, IEEE Computer Society Press, Essen, Germany.

FELDT, P. (2000): Requirements metrics based on use cases, *Master thesis*, Department of Communication Systems, Lund Institute of Technology, Lund University, Sweden.

HENDERSON-SELLERS, B., ZOWGHI, D., KLEMOLA, T. and PARASURAM, S. (2002): Sizing use cases: How to create a standard metrical approach, *Proceedings of the 8th International Conference on Object–Oriented Information Systems*, Vol. 2425 of Lecture Notes in Computer Science, Springer Verlag.

ISO/IEC (2001): ISO 9126.1 Software engineering–product quality—quality model, International Standard 9126.1–2001, International Organization for Standarization.

JACOBSON, I., GRISS, M. and JONSSON, P. (1997): *Software Reuse: Architecture, Process and Organization for Business Success*, Addison–Wesley.

KAMSTIES, E. and ROMBACH, H. D. (1997): A framework for evaluating system and software requirements specification approaches, *Lecture Notes in Computer Science* 1526.

KIM, H. and BOLDYREFF, C. (2002): Developing software metrics applicable to UML Models, *Proceedings of the 6th International Workshop on Quantitative Approaches in Object–Oriented Software Engineering*, Málaga, Spain.

KONTOYA, G. and SOMMERVILLE, I. (1997): *Requirements Engineering: Processes and Techniques*, Wiley.

LEITE, J. C. S. P., HADAD, H., DOORN, J. and KAPLAN, G. (2000): A scenario construction process, *Requirements Engineering Journal* 5(1).

LILLY, S. (1999). Use case–based requirements. Review checklist, *Project report*, SRA International, Inc.

MARCHESI, M. (1998): OOA Metrics for the unified modelling language, *Proceedings of the 2nd EUROMICRO Conference on Software Manteinance and Reengineering*.

McCABE, T. J. (1976): A complexity measure, *IEEE Transactions on Software Engineering* SE–2(4): 308–320.

OMG (2003). Unified modelling language specification, v1.5, The Object Management Group, Inc.

POHL, K. (1997): Requirements engineering: An overview, *Encyclopedia of Computer Science and Technology* 36.

SCHNEIDER, G. and WINTERS, J. P. (1998): *Applying Use Cases: a Practical Guide*, Addison–Wesley.

ZOWGHI, D., GERVASI, V. and McRAE, A. (2001): Using default reasoning to discover inconsistencies in natural language requirements, *Proceedings of the 8th Asia-Pacific Software Engineering Conference*, Macau, China.

BIOGRAPHICAL NOTES

Beatriz Bernárdez is assistant professor at the Department of Computer Science of the University of Seville, Spain. She received her MS degree in computer science in the Department of Computer Science of the University of Seville in 1998. Her research interests are requirements engineering, quality requirements, software measurement and software process improvement. She has published several papers in international conferences.

Beatriz Bernárdez

Amador Durán is associate professor at the Department of Computer Science of the University of Seville, Spain. He received his MS degree in computer science in the Department of Computer Science of the University of Seville in 1993 and his PhD at the same university in 2000. His research interests are requirements engineering, software product lines, conceptual modelling, software engineering for web applications and empirical software engineering. He has published several articles in international journals, several papers in conferences like IEEE Requirements Engineering, and edited a book about requirements engineering.

Amador Durán

Marcela Genero is assistant professor at the Department of Computer Science at the University of Castilla-La Mancha, Ciudad Real, Spain. She received her MSc degree in Computer Science in the Department of Computer Science of the University of South, Argentine in 1989 and her PhD at the University of Castilla-La Mancha, Ciudad Real, Spain. Her research interests are advanced database design, software metrics, conceptual data models quality, database quality. She has published several papers at prestigious conferences and in journals such as CAISE, E/R, OOIS, METRICS, ISESE, SEKE, Journal of Systems and Software, International Journal of Software Engineering and Knowledge Engineering, Information and Software Technology, Software Quality Journal, etc. She has co-edited the book "Information and database quality", 2002, Kluwer Academic Publishers, USA.

Marcela Genero