Evaluating the Effectiveness of a Goal-Oriented Requirements Engineering Method

Huzam S. F. Al-Subaie
King's College London
Department of Computer Science
UK
huzam.al-subaie@kcl.ac.uk

Tom S. E. Maibaum
McMaster University
Department of Computing and Software
Canada
tom@maibaum.org

Abstract

As an attempt to answer the need for methods and tools in requirements engineering (RE) which are domain specific and can address the main RE objectives (REOs), and the growing interest in the goal oriented requirements engineering (GORE) approach that overcomes the inadequacy of the traditional systems analysis approaches, we systematically evaluate the KAOS method, and the Objectiver tool, using the major REOs widely accepted as being important attributes of requirements specifications. In addition, we examine whether KAOS and Objectiver meet their own self-defined objectives. We use two target problems as a basis for the evaluation. The result of the target problems is raw data consisting of error reports and observations that support the evaluator's judgment. The evaluation itself is qualitative, not a statistical experimental evaluation. Its result will help to answer the research questions: (i) How well do KAOS and Objectiver meet the criteria established in the discipline of RE; and (ii) How well do KAOS and Objectiver achieve their own self-defined objectives.

1. Introduction

There has been strong evidence that RE needs proper engineering methods and tools, which are domain specific and comprehensive, in supporting major REOs, and that require very detailed tools associated with them, to produce high quality requirements, to save time and the effort of rework on requirements, and to reduce resources, such as the size of RE teams [1]. Moreover, there has been growing interest in GORE approach that is based on the identification of system goals and the transformation of these goals into requirements; it addresses concerns of why a certain goal is required, how it can be achieved and who is responsible for it in the system and/or the environment [2] [3] [22]. To address these issues, the research we are undertaking proposes to evaluate KAOS, a GORE method, and Objectiver, as associated tool for KAOS, for the following reasons: (i) the role of goals in RE is fundamental while they are a main part of use-cases in object-oriented and goal-oriented approaches; moreover, the goal notion is increasingly being used on current RE methods and techniques because there is a perceived inadequacy of the traditional systems analysis approaches when they are applied to complex software systems [2] [3] [5]; (ii) the KAOS method is the only method of the GORE family that pays special attention to the use of formal proof for model analysis; (iii) the Objectiver tool fully supports the KAOS methodology and is a commercially supported tool that can be relied on to help in the evaluation, and has been applied to a number of industrial problems and case studies already at Cediti- Belgium; (iv) the RE community is eager not only to understand KAOS and Objectiver, but also to be aware of how to improve them to be more supportive for the RE area and to extend the range of their success in solving most common RE complications; and (v) this research is believed to be the first research that related method-detected errors to REOs and objectives of the method itself, in order to give some processes or techniques as a guide to detect the diversion away from the REOs or their own objectives and get the development back on track.

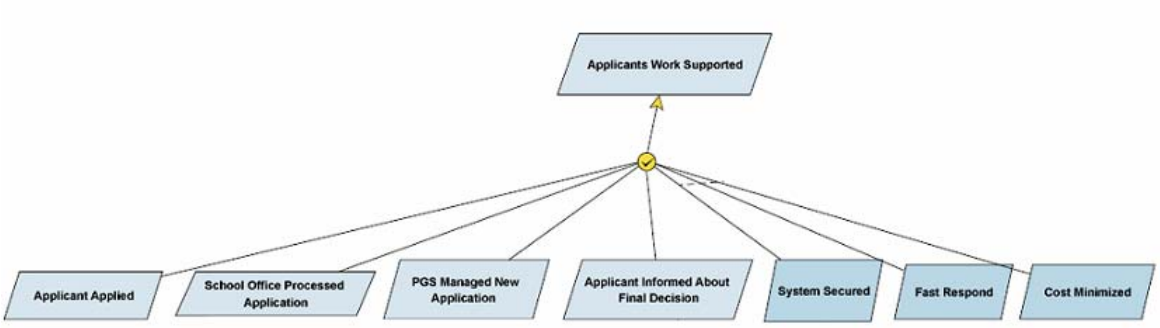

Fig. 1: The strategic goals model of the target problem

Our research is a systematic study based on the following:

1. The major REOs accepted by RE researchers and practitioners as being important attributes of requirements specifications; these include: completeness, correctness, non-ambiguity, pertinence, consistency, and traceability [11][15].
2. The objectives of KAOS itself, which includes providing constructive assistance during requirements engineering activities, such as: (a) create problem descriptions by using predefined concepts; (b) analyse the problem through a systematic technique for eliciting, discovering, and structuring goals; (c) identify roles and responsibilities of the stakeholders; (d) provide formal definition of the requirements of the most critical parts of the system; and (e) establish efficient stakeholders communication [6] [16].
3. The objectives of Objectiver itself which includes: (i) supporting the KAOS method semi-formally to identify, model and write requirements; (ii) enabling a systematic derivation of requirements documents from requirements models; and (iii) improving the validation process, the quality of requirements documents and stakeholder communication [7].

The evaluation result will help to answer the research questions: (i) How well do KAOS and Objectiver meet the criteria established in the discipline of RE; and (ii) How well do KAOS and Objectiver achieve their own self-defined objectives. This paper describes our research, but, due to the page limitations, it presents only the part of the research that takes into account the major REOs using the first target problem and some of the primary results of the evaluation.

2. The KAOS method

The KAOS (Knowledge Acquisition in autOmated Specification) methodology has been designed at the informatics department at UCL (Louvain-La-Neuve) in the early 1990s, and continues to be extended and refined [9][23][4]. It provides a multi-view graphical language for system modelling, a small formalism for model specification, an optional real-time temporal logic for model analysis, a systematic method for model elaboration, and techniques for goal refinement and operationalizations, conflict management, agent responsibility assignment, and obstacle management [30]. The methodology is supported by various tools such as Objectiver, which is a semi-formal tool and fully supports the KAOS methodology; it is a tool specifically designed to engineer business and technical requirements [29]. Target problem background

3. Target problem background

This target problem is based on local requirements, in the Department of Computer Science (DCS) at King's College London. The DCS has a requirement for dealing with an administrative system related to the work of the Postgraduate Secretary (PGS) in the departmental office. The problem includes completing the requirements engineering phase of developing new software, and to effectively support the PGS in his\her work of keeping track of postgraduate applicants, from the time they first apply, to the time they receive the department's decision**.** The procedure is: an applicant obtains an application form, fills it in, hopefully gets all necessary documents together, and sends them to the School Office. At the School Office, they enter the basic information about the applicant into the college database and generate an identification

number for the application. They then bring it to the PGS at the departmental office. The PGS puts the basic information in his Spreadsheet system and sends it to the PhD tutor. The PhD tutor has a look to make sure the application form and the academic documents are complete, and the applicant has met the academic standards of King's College. The PhD tutor then either passes the application form to the member of staff that the applicant has highlighted to be his/her potential supervisor or advertises it to all members of staff. The PhD tutor makes his decision about whether the application has been rejected or accepted and who the supervisor is, depending on the availability of an appropriate supervisor. After that, the application form comes back to the PGS to log the PhD tutor's decision in the system. The PGS then takes the application back to the School Office, they log it in their DB and send out a letter to inform the applicant whether he/she has been offered a place or rejected. Figure 1 represents the strategic goals model of the target problem.

4. Evaluation steps

The evaluation is performed in eleven steps, explained as follows:

1. Selecting the candidate method and tool to be evaluated: the KAOS method and the Objectiver tool have been chosen for the reasons discussed in section 1.
2. Identifying the purpose of the evaluation: The purpose of the study was to investigate the level of success of the KAOS method in solving the most common RE difficulties. In other words, the purpose is to judge KAOS and Objectiver regarding the level of support they actually provide for REOs and their own self defined objectives.
3. Selecting the type of the evaluation: The evaluation type in this research is qualitative. Qualitative evaluation is based on the knowledge base of the evaluator to assess the extent to which the method or the tool provides the expected objectives in a usable and effective way [18]. The evaluation is not an experimental study in the sense that it produces a dataset that can be analysed statistically. It is a subjective evaluation, which provides qualitative information about KAOS and Objectiver based on the evaluator's observations.
4. Deciding the scope of the evaluation: It is difficult and very time consuming for an evaluator to measure and analyse all aspects of KAOS and Objectiver, in addition to the complex nature of REOs themselves, chosen as the criteria for the evaluation. Therefore, it is worthwhile to focus evaluation on assessing the area of KAOS requiring most understanding and improvement. The evaluation in this research was solely concerned with the issues of the method and the tool that contribute to understanding and improving their support for the major objectives of RE. Management issues, such as price and marketing, were not included, nor were teamwork issues, because the target problems were implemented and evaluated by one user only.
5. Identifying a list of the main objectives to be used as criteria: This involves identifying the REOs, in addition to KAOS's and Objectiver's own specific objectives, and applying KAOS and Objectiver to a real administrative system. The main REOs, which are the important objectives in the context of RE and GORE, along with their definitions, are explained in section 8. The specific objectives of KAOS and Objectiver themselves, along with their definitions, are briefly explained in section 1.
6. Outlining a measurement system: This concerns the construction of a measurement system, providing a subjective scale that can be applied to each criterion, as explained in section 7.
7. Prioritising the criteria: explained in section 9.
8. Selecting evaluation methodology: explained in section 5.
9. Analysing the study to produce raw data: explained in section 6.
10. Producing a set of scores for all the criteria: explained in section 7.
11. Combining the evaluation results: Once KAOS and Objectiver have been evaluated and "scores" for each criterion are obtained, using an ordinal scale, the results of the two target problems have to be combined to present an overall measurement. Classes can be combined in some way to give the same sort of indication as the measurement.

5. The evaluation methodology

This study was not planned as a case study, in the sense defined by [47]. Rather the method emerged naturally as an empirical method, suitable for the work performed. The purpose of the study was to

investigate the range of success of the KAOS method in solving the most common RE difficulties. This involves using KAOS and Objectiver to address a target problem and the experience of that is used to evaluate the effectiveness of the method. The raw data was collected in three forms: (i) problem reports generated by KAOS and Objectiver about their ability to cover REOs and to achieve their own objectives; (ii) a Log book was used to record challenges that occurred while dealing with KAOS, and the actions taken by an engineer to overcome any methodological or tool difficulties; and (iii) questions about what data to keep and record, motivated by our understanding of general requirements criteria and the observations of KAOS and Objectiver, such as aspects, which were difficult or impossible to achieve using KAOS and/or Objectiver, and how the engineer overcomes any methodological or tool difficulties. These data are required to evaluate the strengths and weakness of KAOS and to explain why the KAOS method and the Objectiver tool are/are not appropriate for helping to solve the difficulties of RE.

6. Study analysis

The most important part of the study analysis is to produce the raw data to support the evaluator's judgment about KAOS and Objectiver, in addition to developing logical and persuasive arguments to aid the method and the tool improvement. These raw data can be categorised into the following sets: (i) the set that contains the problems regarding traceability, correctness, understandability, or pertinence that are detected by Objectiver, e.g., results of using query functions; (ii) the set that contains the problems regarding traceability, correctness, understandability, or pertinence that are detected by KAOS, which were not caught by Objectiver, e.g., problems detected by using the temporal logic; (iii) the set that contains the problems regarding traceability, correctness, understandability, or pertinence that are detected by the stakeholders during the validation activity and which were not caught by KAOS/Objectiver, e.g., assigning a requirement to an incorrect agent; and (iv) the set that contains the developer's immediate observations based on the general criteria while using KAOS and Objectiver on the target problems, e.g., the absence of information in the requirements on the Head of the Department because of delegating her/his duties. A single user worked full time for 24 weeks on the evaluation and the development of the requirements of the problem. The user was a novice when he started and some of the problems encountered were to do with being a novice, but there are still substantial problems, which are inherent in the methods and the tool. We used what we think is a reasonable and systematic way of eliminating problems, which are irrelevant or double counting. This way is to take errors reports of the first target problem and trace it to any error reports, which are generated on the second problem. Then, ones, which do not have corresponding at the end are presumably not really counted problems because they are problems were to do with being a novice.

7. Measurements

Measuring the degree of coverage of KAOS in relation to the RE objectives is too difficult because of the complexity of the nature of RE. It can therefore only be done by developing a measurement system that helps the evaluator to outline how to measure the degree to which these criteria have been met. In this research, ordinal scale has been chosen because it is the suitable scale to measure qualitatively the degree of support offered by methods/tools [18]. These scales represent ascending levels of achievement which can be associated with the KAOS method meeting the general RE objectives. The alphabetic ranking represents ranking only and the higher the level the greater the degree of achievement applied in testing whether the KAOS method has met the criterion. For example, an objective that is well covered by KAOS or Objectiver is "fully achieved" or score A on the scale for it, and that the objective that is not addressed at all, is "fail to be achieved" or score E on the scale for it. Table 1 presents the measurements system used in this evaluation.

8. RE objectives (REOs)

The REOs are defined as those attributes that need to be achieved to produce complete, valid, correct, pertinent, consistent, traceable, unambiguous and understandable requirements [11] [15].

These objectives have to be accomplished in order to avoid producing vague, incomplete or wrong requirements that lead to the development of a poor quality system which costs a great deal of time, effort and money to correct.

We produced the hierarchy in Figure 2 corresponding to the major four objectives of RE: pertinence, correctness, traceability, and understandability. There is general agreement that

these are the appropriate RE properties to be addressed, see [11] [15]. It has been justified in terms of how these REOs are defined in the RE literature. Validity, completeness, and consistency need not be considered at the top level because they will flow from correctness as well as ambiguity, alternatives, and conflicts which in turn flow from consistency. It is part of this research to use these major REOs as criteria for the evaluation of the KAOS method.

Traceability is a major REO; generally it involves the relationships between requirements and their sources to clearly identify the origin of the requirements in order to manage requirements easily [34]. In [13] it is the ability to know the origin and the development of a requirement life, in a forward and backward direction.

In general, whatever is the definition of traceability is, depends on what the stakeholder is trying to trace requirements to. Therefore traceability might be interpreted into three notions [36]:

- Requirements to sources traceability, which links the requirement to the pre-existing informal requirements (information describing the system from people or documents).

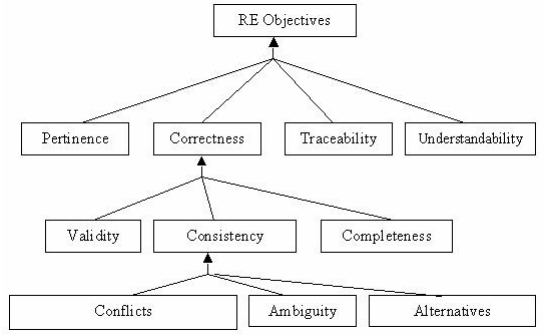

Fig 2: REO hierarchy

- Requirements to design traceability, which links requirements to the software engineering process, used to implement the requirement.
- Requirements to requirements traceability, which links part of the requirements with other parts of requirements, which are, in some way, dependent on them

Correctness is a major REO and always an important one. Correctness might be defined as the extent to which the requirements specification fulfils stakeholders' needs without any error or loss.

Zowghi and Gervasi have concluded that correctness can be formally defined as the combination of consistency and completeness [41]. Furthermore we think correctness involves rather more than the combination of consistency and completeness; Correctness is not just about what is dealt with but it is also about whether the right thing is done. For requirements to be correct they need to be complete, in some sense, with no inconsistency and they should be right; because developers can have complete and consistent requirements without having what the stakeholders want. For example, in a given situation, stakeholders want (x) to happen but the requirements say (y) happens. Now this is not a problem of completeness or consistency but of having the right requirements. So the components of correctness should be completeness, consistency and there is another component, which is validity. Completeness means that every necessary situation is included and defined in the requirements, but does not say that in every situation the right requirements happened. Completeness is a sort of coverage principle that all situations, which are relevant, are considered in the requirements specification. Consistency is more straightforward, that there is no internal contradiction. A completeness check guarantees that all the necessary information is included in the requirements while a validity check guarantees that all true information is presented in the requirements. From all of the above, there is sufficient understanding that a set of requirements can only be considered to be 'correct' when it is complete, valid and consistent, in explaining the system to be with its environment. This would be a satisfactory definition.

Validity is a major REO. Validity is defined as the extent to which the requirements specification fulfils stakeholders' needs without any error. It is very important because if developers do not capture the right requirements, they will produce the wrong system. It includes the review and negotiation processes with stakeholders to ensure the system provides the requirements to meet their real needs, in addition to reaching agreement about valid choices, which are important issues to be considered during the requirements elaboration process. Completeness is to do with not missing cases, thus covering all necessary requirements, which is not necessarily the same as matching the customer's wishes.

Completeness is a major RE objective. It is defined as the extent to which the requirements specification fulfils stakeholders' needs without any loss. It implies that requirements include everything needed to develop the software that satisfies the

stakeholders [35]. Completeness is considered to be the most difficult of the requirements objectives to define and incompleteness of requirements the most difficult to detect [12]. This is because there is no simple systematic process for determining when the stakeholders have told the developers everything that they need to know about the system [19]. In addition, it is difficult to establish and measure because it is a relative measure of quality, rather than an absolute one, and must be examined with respect to some external reference point [42]. Furthermore, according to [26], completeness might be divided into two notions: (i) internal completeness checks to discover any overlooked gap in the requirements relating to what is already (partially) there. For example, if we have a requirement to say what to do if an applicant has applied, the requirements certainly need to say what to do when the application is not complete; and (ii) external completeness checks to discover any completely missing information from the requirements, because of the stakeholders having forgotten to mention it. For example, if the requirements talk about accepting the application only, the internal completeness check may be able to discover the details of it (by talking about accepting the application only). But if there is no mention of rejecting the application, the requirements are not externally complete since all requirements related to rejecting the application are completely forgotten.

Consistency is another major RE objective; it refers to situations where a requirement contains no internal contradictions. Requirements consistency is strongly related to removing ambiguity, resolving conflicts, and evaluating alternatives. In [28], a variety of possible causes of inconsistencies arising between stakeholders could be mapped to the differences in views they hold, languages they speak, development strategies they deploy, stages of development they address, or objectives they want to achieve.

Ambiguity as defined by the Wikipedia encyclopaedia: "A word, phrase, or sentence is ambiguous if it has more than one meaning" [38]. Ambiguity is strongly related to misinterpretation of information that cause the developer and other stakeholders to have different opinions about its meaning [43].The resolution of ambiguity issues is left to the developers to do during the validation activity. It is to make sure that different stakeholders, including developers, understand the same requirement in the same way clearly, without any multiple beliefs or expectations that could lead to incorrect or inconsistent conclusions.

A. Fully achieved:
   1. The objective completely supported by:
      - Providing step-by-step assistance to the developer.
      - Providing explicitly detailed information in the literature.
   2. All aspects of the objective are covered:
      - No number of error reports mapped to this particular objective.
      - No number of negative observations mapped to the objective.
B. Strongly achieved:
   1. The objective largely supported by:
      - Providing step-by-step assistance to the developer.
      - Providing explicitly detailed information in the literature.
   2. All aspects of the objective are covered but the full achievement of the objective depends on the expertise and the talent of the developer:
      - Some error reports from the first target problem mapped to this particular objective but not from the second one.
      - Some negative observations to the objective from the first target problem mapped to this particular objective but not from the second one.
C. Partially achieved:
   1. The objective partially supported by:
      - Providing some assistance to the developer.
      - Providing some information in the literature.
   2. Part of aspects of the objective are covered but the full achievement of objective depends on the help of another tool or method:
      - Lightweight error reports mapped to this particular objective.
      - Lightweight negative observations mapped to this particular objective.
D. Slightly achieved:
   1. The objective supported in very limited degree by:
      - Providing Limited assistance to the developer.
      - The literature does not cater for the objective.
   2. Very limited achievement of objective:
      - Developer needs to extend the method to overcome its limitation.
      - Heavyweight error reports mapped to this particular objective.
      - Heavyweight negative observations mapped to this particular objective.
E. Fail to be achieved:
   1. The objective unrealised
      - The objective is not addressed.
   2. The objective totally left out.
      - The objective is not referred to in the literature.

Table 1: Measurements system.

Resolving conflicts is a major RE objective that should be satisfied in order to achieve requirements consistency. A conflict typically occurs when one party makes changes that obstruct the other party's development [41].

Evaluating alternative models: is a major RE objective that should be satisfied in order to achieve requirements consistency. An agreement for valid models is an important issue to be considered during the RE process. Evaluating alternative models is related to consistency in the sense that each alternative model must be internally consistent, but there are may not be full consistency between different models. However, the relation of evaluating alternative models to validity is regarding a single model, but not regarding comparison between models. So the idea of an alternative model is not related to validity because by definition each of these models should be correct. It is true that the alternatives have some elements of correctness, which has to do with the utility that refers to the worth of an alternative to its stakeholders [37]. The utility concept is outside the scope of this research.

Pertinence is a major RE objective and it refers to the avoidance of redundant requirements.

Understandability is a major RE objective in all life cycle of the requirements, starting from the elicitation activity, through the validation activity where developers desire to discuss requirements with stakeholders and finally at the production of the final specifications document. Understandability is relatively in reverse to complexity [21]. It involves interpreting and understanding stakeholder terminology, concepts, viewpoints and goals [27]. It then transforms this information into specified clear requirements that can be easily understood by stakeholders including system designers and maintenance personnel. There are three parts of requirements understandability: notation, organisation, and level of abstraction [39]. Notation should be straightforward, information should be well organised, and the level of abstraction should suppress irrelevant details and focuses on the crucial parts of the requirements [39].

From the above, understandability can be defined as the degree to which all stakeholders clearly recognize the meaning of requirements along with their definitions.

9. Prioritising the criteria

The aim of prioritising the REOs is to know the importance of the error reports and observations that are mapped to the REOs. For example, an error that is related to completeness is probably more important than the one related to understandability. However, no studies in the literature have really tried to make concrete the relative weight of REOs in RE. It was difficult to estimate what the importance of REOs should be, but the literature helped to define the weight based on what is considered more or less critical for RE. Weighting the REOs needs special attention to the interpretation of scores if they are combined into single numbers. Giving simple numerical figures to quantify the "importance" of such properties maybe misleading, because of the nature of the ordinal scales and subjective measurements that are used in this evaluation [18]. For instance, a score of 4 is not always twice as good as a score of 2. (This is a common mistake made in interpreting ordinal scales.)

From experience, a good method or tool is one that supports the essential REOs for RE and achieves its own essential objectives. In this study, the importance of a REO is decided by considering whether it is an essential objective or just an orthogonal one. A REO that is not essential is, by definition, orthogonal. An objective is essential if the requirements are not acceptable unless this objective is satisfied [46]. Orthogonal objective means it would enhance the requirements, but the requirements are not unacceptable if this objective is absent.

Correctness, as explained in section 8, is an essential objective because if the requirements are not correct, the design of the system will be incorrect too and programmers will produce the wrong system.

Traceability, as explained in the previous section, is an important objective, as, when it is used, it can offer considerable benefits for requirements management. However, the requirements could be complete, valid, consistent, pertinent, and understandable even if they are not traceable. They maybe take extra time and effort to modify and maintain.

Understandability is an important objective, but, if the requirements are hard to understand, they can still be correct; however, it may be difficult to check that they are correct. Imagine requirements engineers having a requirements method, which supports requirements completeness very well, even though requirement specifications are not that easy to understand. For instance, if there is a formal specification language which will be difficult for

stakeholders to understand the logical notation, but a checker is available for it, so developers could formulate queries and ask them to the stakeholders and get answers back and decide whether they are correct or not. So understandability seems to us to be an orthogonal issue.

Pertinence is an important objective but seems to be an orthogonal issue too because the requirements engineer can have correct requirements that have redundancy, and also she/he can have pertinent requirements, but they are not correct or more difficult to use.

It can be concluded that the evaluation of the KAOS method can be based on a correctness objective but traceability, pertinence and understandability seem to be orthogonal issues. Therefore, correctness is the essential objective of the REOs and needs to have more attention and concentrations. Consequently, the errors or limitations that map to correctness have more importance that affects the requirements' success. Traceability, understandability, and pertinence are orthogonal objectives, which make requirements easier to manage and to check correctness. Also, it can be concluded that traceability, pertinence and understandability have equally important value.

10. Primary results

The following are our primary reports and observations as a result of using KAOS and Objectiver on the target problem, which were produced from the raw data (see section 5):

1. Primary reports and observations mapped to validity.

Problem 1.1: There is uncertainty about the correctness of the decomposition process that transitions the system goal into subgoals.

The implications related to the problem: Operationalisation of the leaf goals are not enough for the developer to ensure validity, completeness, and understandability.

Nature of the implications: The developer's immediate implications based on the general criteria while using KAOS and Objectiver on the target problems.

Implications: The validity objective in GORE means that all goals and their definitions in the model are correct and relevant to the system [20]. Therefore, KAOS and Objectiver deals with validity issues by using stakeholders' own strategic goals and involving reasoning techniques for each step of building the goal model supported by formalisation, besides the iterative validation process with stakeholders. However, KAOS and Objectiver advise developers to operationalise the leaf goals only and that does not support the developer in formulating requirements precisely, as well as the relationship between subgoals. So we had to develop our own way to extend the method to operationalise most of the goals in the model, to increase the completeness, validity and understandability of the system.

2. Primary reports and implications mapped to completeness.

Problem 1.2: There are difficulties with internal completeness check.

The implications related to the problem: There are difficulties with internal completeness using temporal logic in KAOS.

Nature of the implications: The developer's immediate implications based on the general criteria while using KAOS and Objectiver on the target problems.

Implications: KAOS offers a formal temporal logic when formalisation is necessary to prove that a goal refinement is correct and complete, and to detect conflicts [17] [10]. However, temporal logic cannot be applied at top level or to soft goals, and this is a problem in the sense that developers can only talk about completeness when they have goals, which are formalisable. Temporal logic becomes applicable only when developers have a goal which can eventually be operationalised. Other soft goals, even if they could be formulated somehow, would not mean anything particularly useful. Top-level goals, including functional ones, may not be reliable in this sense, but they become reliable when developers make them specific enough that they can use them as functional goals of the system, including environmental agents. Therefore, the completeness notion can only be relevant to the top-level reliable goals. It seems a crucial problem for the KAOS method and its vendors need to provide some means of overcoming this difficulty.

Problem 2.2*:* There are difficulties with external completeness check (see section 8).

The implications related to the problem: KAOS and Objectiver do not say much about requirements developers do not have in the goal model.

Nature of the implications: The developer's immediate implications based on the general criteria while using KAOS and Objectiver on the target problems.

***Implications*:** There are difficulties with external completeness check in KAOS related to the absence of information on the head of the department because of delegating her/his duties. KAOS does not help in capturing the chain of responsibilities. For example: in the target problems, it is noticeable that the head of the department is not an agent in the model, because all his/her departmental duties are delegated to various people like the PhD Tutor. Therefore s/he does not appear directly. Similarly, the school office personnel are not the people who are technically responsible for achieving the goals assigned to them. It is the school administrator and the head of the school who are responsible, but again, they delegate what they do to the school personnel. KAOS does not deal with issues like roles and delegation, and this is another shortcoming of KAOS. It is worth mentioning as part of our experience. To solve this matter, the idea of roles and the delegation of responsibilities, which is very difficult to model, should be introduced to KAOS, but for the purpose of these target problems, it will complicate the study and the evaluation as well.

3. Primary reports and implications mapped to pertinence.

Problem 3.1: There are no problems with detecting some of the irrelative requirements using KAOS and Objectiver.

The implications related to the problem: KAOS and Objectiver alert the developer about the irrelative requirements and the redundant concepts.

Nature of the implications: Problems regarding pertinence that are detected by KAOS and Objectiver.

Implications: In GORE, a requirement is pertinent if its specification is used in the proof of at least one goal in the problem domain [40]; requirements are pertinent if they contribute to the strategic top-level goals, so the semantic net of a goal model in KAOS connects relevant requirements to these goals and helps with the validation activity to eliminate irrelevant requirements and unnecessary concepts that may have been elicited at the beginning. Query checks in Objectiver support pertinence regarding concepts; for example: it helps to detect concepts never appearing in a document, concepts never appearing in a diagram, and concepts without definition; Objectiver also does not allow the developer to duplicate the concept in one page [44].

4. Primary reports and implications mapped to understandability.

Problem 4.1: There are no problems with detecting the understandability difficulties using KAOS and Objectiver.

The implications related to the problem: KAOS and Objectiver alert the developer about understandability problems.

Nature of the implications: Problems regarding understandability that are detected by KAOS and Objectiver.

Implications: In GORE, goal models and goal definitions may facilitate the understanding of requirements. Goal models supports big image views by structuring complex requirements in a natural way for high-level readability and understandability [23]. KAOS used informal goal definitions in textual form alongside the graph model to increase understandability. Also Objectiver adds different colours and shapes to different concepts, which enables greater understandability of the requirements.

5. Primary reports and implications mapped to ambiguity.

Problem 5.1: There are no problems with detecting the difficulties related to ambiguity using KAOS and Objectiver.

The implications related to the problem: KAOS and Objectiver alert the developer about ambiguity problems.

Nature of the implications: Problems regarding ambiguity that are detected by KAOS and Objectiver.

Implications: In KAOS, ambiguity is removed by using informal goal definitions in a textual form alongside the graph model and by using formal specification, which helps to minimise ambiguity and clarify developer understanding to provide a basis for consistency. KAOS also helps remove ambiguity regarding who is really responsible for executing the final goal, by decomposing the goal into requirements under the responsibility of a single agent. As a GORE method, KAOS helps to reduce the ambiguity issues by separating high strategic goals from requirements, and allows

stakeholders to answer 'how' and 'why' questions about the goals.

6. Primary reports and implications mapped to the traceability objectives.

Problem 6.1: There are no problems with requirements to requirements traceability using KAOS and Objectiver.

The implications related to the problem: Objectiver and KAOS link the requirements to other requirements in the model (see section 8).

Nature of the implications: The developer's implications based on the general criteria while using KAOS and Objectiver on the target problems.

Implications: The notion of requirements to requirements traceability with KAOS are fully achieved; because in GORE generally, a requirement appears because of a final goal that provides a requirement base [8]; thus, a goal model connects high-level strategic objectives and low-level technical requirements through semantic net navigation. Based on KAOS definition of traceability, it is ensured by the goal model to link requirements to goals and the operation model to link operations to requirements [29]. However, Objectiver cannot deal with large indexes and this causes a traceability problem for Objectiver but not for KAOS and may also affect the validation and understandability of the model.

Problem 6.2: The requirements to sources traceability is not supported by KAOS and Objectiver.

The implications related to the problem: Objectiver and KAOS do not link the requirements to their sources.

Nature of the implications: The developer's implications based on the general criteria while using KAOS and Objectiver on the target problems.

Implications: We find that KAOS does not provide any support for the requirements to sources traceability (see section 8). So KAOS and Objectiver are failed to achieve this notion of traceability. Requirements to design traceability would be outside the scope of this research to test this particular notion of traceability.

7. Other primary reports and implications.

Problem 7.1: KAOS does not guarantee that the developers choose the right goal.

The implications related to the problem: Developers may choose a goal and work on it for a long time, and at the validation phase realise that the goal they have chosen was wrong or merely a single possible solution. After that, they need to remodel the whole part again and this requires a lot of time and effort [31] [45].

Nature of the implications: The developer's immediate implications based on the general criteria while using KAOS and Objectiver on the target problems.

Implications: Whatever the method is, when the developers have to build the requirements model, they could face the situation that they make a sub-optimal initial choice and proceed on that basis only to find later on that they made, perhaps not an incorrect choice, but not a sensible choice. So they have to revise the model in some way and redo a lot of their work. In some sense this is unavoidable because developers are not seeing everything right from the beginning. From this point of view and the result of the target problems, KAOS cannot be criticised for not helping developers choose the right goal; it has been criticised, though, for not providing enough support for revising a big model once the developer has made the discovery that something has been left out.

Problem 7.2: KAOS gives the freedom to use any of the information gathering techniques to build up initial domain knowledge about the problem.

The implications related to the problem: KAOS starts helping developers when they find some initial goals to start building the goal model, but not before that [33].

Nature of the implications: The developer's immediate implications based on the general criteria while using KAOS and Objectiver on the target problems.

Implications: it is well known that in any engineering method, the more freedom the user has, the less effective the method is. Therefore, since KAOS gives the developer no guidance about how to deal with building up initial domain knowledge, it will then be harder to follow the method. It was considered by the developer as a shortcoming of KAOS because it requires information to be structured in a certain way in order to be more usable, and if it is not structured in such away, then KAOS is not going to provide the necessary support. However, we have changed our ideas after the second target problem, because we realised that maybe a knowledgeable user of KAOS would eventually learn how to get the

information in a way which is helpful, while as an initial users we do not have any experience to rely upon.

Problem 7.3: KAOS and Objectiver do not support libraries of goal models.

The implications related to the problem: Developer cannot reuse fragments he has built before.

Nature of the implications: The developer's immediate implications based on the general criteria while using KAOS and Objectiver on the target problems.

Implications: We noticed that knowledgeable users of KAOS, who build requirements several times for similar application, may want to reuse fragments they have built before. For example: if developers build the goal model for the first target problem, and during the second one they found that there is something in common, they want to reuse part of the first target problem in the second one. Can they borrow that part from the model in the first target problem? And does KAOS or Objectiver then support reuse? The answer is no, KAOS does not provide support for libraries of goal models, but Objectiver has this ability as an export/import function that works on packages. It works well for single concepts but it is not an effective function at all for a medium/big part of the model because of the dangling pointers to link and reference the exported model to its concepts defined in the other package. Concept inclusion and exclusion takes a great deal of time to be performed, but it is worse for the medium/big part of the model; the software may hang and need to be reset.

11. Conclusions

This paper presents our methodology of evaluation, its rationale and some initial observations and conclusions. To complete our evaluation of KAOS and its associated tool, Objectiver, we must complete the second target problem and then evaluate the 'observations' and the implications generated over the two target problems we addressed. For this, it is crucial for the purposes of evaluation to decide on relative weights to identify the level of importance the requirements community assigns the REOs, along with the objectives of KAOS and Objectiver, since these objectives will have direct influence on the outcome of the evaluation. Observations and error reports as a result of the target problems will be mapped to these objectives. As things stand, we have some initial conclusions about the effectiveness of KAOS and Objectiver in meeting requirements objectives. These suggest that there is much room for improvement.

12. References

[1] The 10th Anniversary of international IEEE RE conferences and symposium, http://www.re02.org/, Germany 2002.

[2] E. Kavakli, "Goal-Oriented Requirements Engineering: A Unifying Framework". Requir. Eng. 6(4): 237-251 (2002).

[3] J. Mylopoulos and E. Yu, "Why Goal-Oriented Requirements Engineering", 4th REFSQ'98, Italy, 1998.

[4] E. Letier, A. Lamsweerde, "Requirements analysis: Deriving operational software specifications from system goals", Proceedings of the 10th ACM SIGSOFT, ACM Press, South Carolina, 2002.

[5] A. Lapouchnian, "An Overview of the Current Research", Department of Computer Science, University Of Toronto, 2005.

[6] R. Darimont, E Delor, "Software Quality Starts with the Modelling of Goal-Oriented Requirements", 6th International Conference "Software & Systems Engineering and their Applications, Paris - 2003.

[7] R Darimont, "Requirements Engineering with Objectiver from Goal Analysis to Automatically Derived Requirements Documents", RE'03, California, USA,2003.

[8] A. Dardenne, S. Fickas and A. van Lamsweerde, "Goal-Directed Concept Acquisition in Requirements Elicitation", Proc. IWSSD-6, Como, 1991.

[9] A. Dardenne, A. van Lamsweerde and S. Fickas, "Goal Directed Requirements Acquisition," Science of Computer Programming, vol. 20, 1993.

[10] R. Darimont, "Process Support for Requirements Elaboration", PhD Thesis, Universite catholique de Louvain, Dept. Ingenierie Informatique, Louvain-la-Neuve, Belgium, 1995.

[11] A. Davis, "Software Requirements: Objects, Functions and States", Prentice Hall, 1993.

[12] A. Davis, Software Requirements: Analysis and Specification, 2nd Edition, Prentice Hall, 1993.

[13] O. Gotel, A. Finkelstein, "An Analysis of the Requirements Traceability Problem", Proc. of 1st International Conference on RE, pages 94-101,1994.

[14] IEEE, "A Compilation of IEEE Standard Computer Glossaries", New York, 1990.

[15] IEEE Recommended Practice for Software Requirements Specifications, IEEE Std. 830-1998.

[16] I. Jureta, "Engineering Requirements for Information Systems using KAOS and Request frameworks", FUNDP - Institution - Belgium .2005.

[17] E. Letier, "Reasoning about Agents in Goal-Oriented Requirements Engineering", PhD thesis, Université catholique de Louvain, Belgium, April 2001.

[18] B. Kitchenham, "DESMET: A method for evaluating Software Engineering methods and tools", TR96-09, Dep. of Computer Science, University of Keele, U.K., 1996.
[19] I. Sommerville and G. Kotonya, "Requirements engineering with viewpoints", Software Engineering Journal, Jan. 1996.
[20] Krogstie, J. "Using Quality Function Deployment In Software Requirements Specification", 5th REFSQ'99, Heidelberg, Germany 1999.
[21] K Laitinen, "Estimating understandability of software documents", ACM SIGSOFT, Software Engineering Notes, Vol. 21, No. 4, July, 1996.
[22] A. Lamsweerde. "Requirements engineering in the year 00: A research perspective", the 22nd ICSE 2000.
[23] A. van Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour", IEEE Transactions of Requirement Engineering, 2001.
[24] A. van Lamsweerde, R. Darimont and E. Letier, "Managing Conflicts in Goal-Driven Requirements Engineering", IEEE Trans. on Software Engineering, 1998.
[25] A KAOS tutorial, www.objectiver.com , , 2003.
[26] B. Boehm, "Verifying and validating software requirements and design specifications", IEEE Software, 1984.
[27] B. Nuseibeh, S.Easterbrook, "Requirements engineering: a roadmap", the future of Software engineering, Limerick, Ireland , 2000.
[28] B. Nuseibeh, "To be and not to be: on managing inconsistency in software development", Proceedings of the 8th IWSSD'96, IEEE Computer Society Press, 1996.
[29] Web site of Objectiver, www.objectiver.com, 2003.
[30] Department of Computing Science and Engineering, Université catholique de Louvain, Belgium, http://www.info.ucl.ac.be/Research/Areas/SoftwareE/GoalOrientedRE.php.
[31] G. REGEV, "A Systemic Paradigm for Early IT System Requirements Based on Regulation Principles: The Lightswitch Approach", Ph.D thesis no. 2810, EPFL 2003.
[32] W.N., Robinson, "Integrating Multiple Specifications Using Domain Goals", Proc. IWSSD-5, IEEE, 1989.
[33] C. Rolland, N. Prakash: "From conceptual modelling to requirements engineering", Ann. Software Eng. 10: 151-176, 2000.
[34] I. Sommerville, "Software Engineering", 6th edition, Addison- Wesley, 2000.
[35] I. Sommerville and G. Kotonya, "Viewpoints for requirements definition", Software Engineering Journal 7,1992.
[36] I. Sommerville and G. Kotonya, "Requirements Engineering Processes and Techniques", John Wiley & Sons, 1998.
[37] National Institute Of Standards And Technology, Guidelines, Information Quality Standards, and Administrative Mechanism, www.nist.gov/director/quality_standards.htm, 2002.
[38] Wikipedia encyclopedia, Version 1.2 http://en.wikipedia.org/wiki/Ambiguity, 2002.
[39] Williams, G. Lloyd, Assessment of safety-critical specifications. IEEE Software, 11(1):51-60,1994.
[40] K. Yue, "What does it mean to say that a specification is complete?", Proceedings of IWSSD'87, 1987.
[41] D. Zowghi, V. Gervasi, "On the Interplay Between Consistency, Completeness, and Correctness in Requirements Evolution", the Journal of Information and Software Technology, Volume 45, Issue 14, 2003.
[42] R Carson, "Requirements Completeness: A Deterministic Approach," 8th Annual International Symposium on Systems Engineering, Seattle, 1998.
[43] F Cioch, "Measuring software misinterpretation. Journal of Systems and Software", 14(2): 85-95,1991.
[44] R. Darimont, "Requirements Engineering with Objectiver: from Goal Analysis to Automatically Derived Requirements Documents", RE02, 2000.
[45] J. You, "Goal-Oriented Requirements Engineering: Promising or Falling", Department of Computer Science, University of Toronto, 2004.
[46] K. Wiegers, "First Things First: Prioritizing Requirements" Software Development, Vol. 7, No. 10, Oct. 1999.
[47] R. Yin, "Case study research: Design and methods", 2nd ed., Beverly Hills, CA: Sage Publishing (1994).