ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2006, Vol. 35, No. 3 A

COMPARING GOAL-MODELLING TOOLS WITH THE RE-TOOL EVALUATION APPROACH*

Raimundas Matulevičius[1], Patrick Heymans[1], and Guttorm Sindre[2]

[1] *Computer Science Department, University of Namur, Belgium*

[2] *Dept. of Computer and Info. Science, Norwegian Univ. of Science and Technology*

Abstract. Goal modelling usually takes place during the early information systems development phase known as requirements engineering (RE). RE is a key factor for project success where a good tool support is necessary. Several goal-modelling tools exist and several approaches can be used to evaluate them. In this paper, we report on an experiment to evaluate two goal-modelling tools - KAOS/Objectiv*er* and *i**/OME. We use an RE-tool evaluation approach (R-TEA) in order to determine which of the tools is better at supporting the creation of goal models. It turns out that KAOS/Objectiv*er* apparently offers better model creation support but the quality of the resulting models is more dependent on situational language characteristics such as the focus on early (vs late) requirements.

1. Introduction

With the increasing complexity of today's businesses, the development of their supporting Information Systems (IS) becomes an ever more difficult and risky endeavour. A clear and complete understanding of the requirements is a necessary prerequisite for successful IS development and maintenance. Requirements Engineering (RE) is defined as "a set of activities concerned with identifying and communicating the purpose of a software-intensive system, and the contexts in which it will be used. Hence, RE acts as the bridge between the real world needs of users, customers, and other constituencies affected by a software system, and the capabilities and opportunities afforded by software-intensive technologies" [5].

For such a complex activity as RE, powerful tool support is paramount. However, it has been observed [16] that mainstream RE practice relies more on office (text editors and spreadsheets) and drawing (paint, Visio, DIA) tools, rather than on dedicated RE tools. This situation inevitably leads to under-exploiting RE-specific tool functionalities.

Currently, *goal modelling* has become a major technique to support early IS development stages and RE in particular. Goal models express the scope and rationale for designing an IS, they help to resolve conflicts early in the lifecycle and ensure that the voices of the various stakeholders are heard by the development team. Tools have been proposed to support goal-modelling languages [1, 3, 4, 6, 17, 18, 19]. However, to date, we do not know of any systematic evaluation of their capabilities. We believe that such comparisons are likely to have an impact on the quality of the proposed tools. Furthermore, by having independent sources to inform the potential adopters about the pros and cons of tools, we hope to allow them to make the best choice for a tool that actually suits their needs.

This paper reports on an experiment where the RE-tool evaluation approach (R-TEA) [16] is applied to evaluate goal-modelling tools. The main research question it addresses is:

RQ.1: Which tool provides better support to create goal models?

The research question focuses on the *creation* of goal models. Thus it avoids investigating how goal models are maintained by the tools, or how the tools might help to reason about goal models. However, we also considered the correlation between the quality of tools and that of their supported languages, as well as between the quality of tools and that of the goal models that they helped to produce. These later two issues are separate research questions in their own right. In this work we hint at them (see section 5) but mainly focus on *tool evaluation*.

The paper is structured as follows: Section 2 introduces R-TEA and how it is applied to test the tools. In Section 3 we describe our research method. Section 4 presents the results of an experiment which are further discussed in Section 5. Finally, Section 6 concludes the paper and suggests directions for future work.

* This work is partially supported by the Commission of the European Communities under the sixth framework programme (InterOP Network of Excellence, Contract 508011), URL: *http://www.interop-noe.org/*.

2. The RE-Tool Evaluation Approach

R-TEA [16] suggests a set of guidelines (see Figure 1) following which a detailed tool evaluation is performed. It considers two groups of stakeholders: the evaluation team and the tool users. The evaluation team plans, organises and executes the tool evaluation process, and coordinates the evaluation steps. The tool users evaluate and compare the suitability of tools among which they may have the intention to select one for future use.

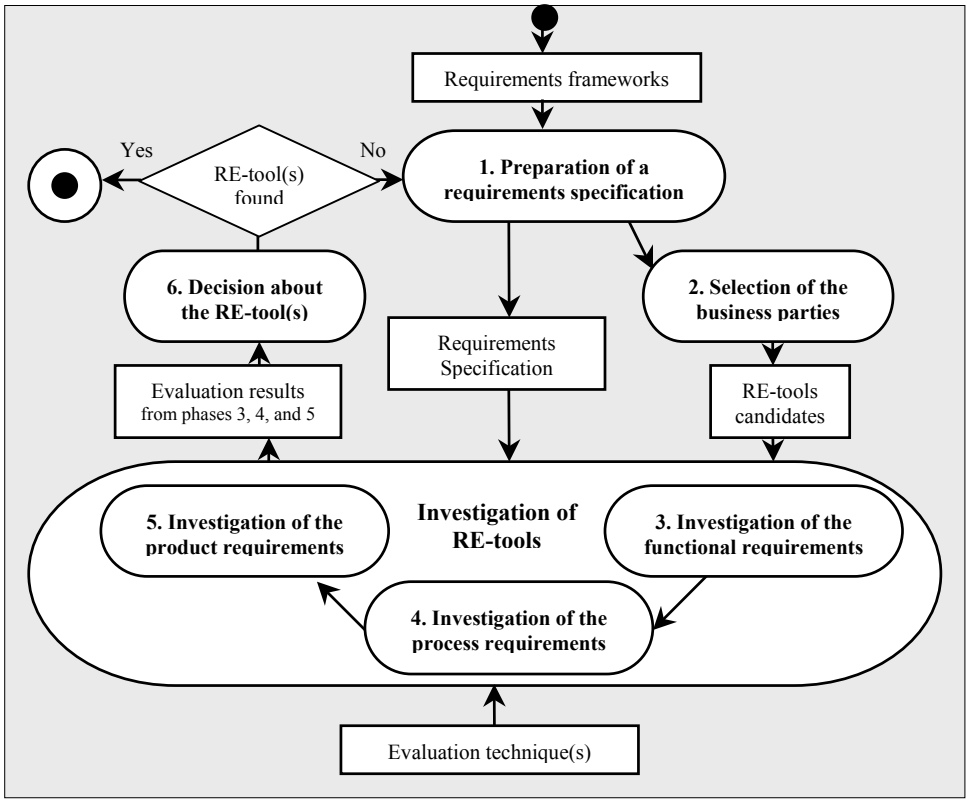

Figure 1. RE-tool evaluation approach

R-TEA consists of six basic phases (see Figure 1):

1. Preparation of a requirements specification for the tool selection. It consists of analysing the requirements for the RE-tools. In the literature we could find several lists of requirements [14, 23] as well as systematic evaluation approaches and frameworks [7, 8, 10, 16]. Wiegers [23] suggests 16 requirements and applies them to assess RE-tools. Elsewhere [14], Lang and Duggan characterise a requirement management, communication and cooperative work system by 12 requirements. However, in both cases the requirements are fairly basic and at a high level of abstraction; they are thus not appropriate for a detailed tool evaluation.

The INCOSE framework [10] classifies 52 requirements into 14 categories. However, the terminology used in the framework is not defined, so the evaluations are hard to compare when the requirements are interpreted differently by different evaluators. The priority-based evaluation framework [7] created in consultations with practitioners classifies 53 requirements according to three priority levels - *high* (essential), *medium* (useful) and *low* (desirable, but not essential). But organisations are not homogeneous environments, so priorities depend on various objective and subjective circumstances. The framework has no guidelines for how to analyse the RE-tool if user priorities vary in different environments. The role-based framework [8] suggests 93 requirements, which are grouped according to roles: *developer*, *project administrator* and *RE-tool administrator*. However, requirements cannot be entirely partitioned according to roles. Furthermore, the role-based framework does not consider guidelines for the context-specific application. The authors also do not provide empirical evidence of the framework's validity.

The R-TEA method [16] introduces two requirements frameworks (R-TEA frameworks). The framework for *functional* RE-tool requirements consists of three requirements dimensions [21]. The *representation* dimension denotes the degree of formality at which the tool allows to represent requirements. The *agreement* dimension indicates whether the tool supports improving the level of agreement among participants e.g. by means of collaboration or rationale maintenance techniques. The *specification* dimension deals with improving the understanding and completeness of the requirements. The framework for *non-functional* RE-tool requirements separates process, product and external requirements. Process requirements characterise constraints placed upon the user's work practice that influence the tool. Product requirements specify the desired qualitative characteristics of RE-tools.

External requirements are separated to organisational requirements and requirements to business parties.

Once a framework is chosen, the evaluation team adapts the requirements to their specific context by prioritising them. The result of this phase is thus a prioritised requirements specification.

2. Selection of business parties involves the investigation of the RE-tool market. The evaluation team requests trial and demonstration RE-tool versions from the business parties, briefly investigates them and makes a short list that will undergo further evaluation.

3. Investigation of the functional requirements delivers a functionality evaluation of the tool candidates. Several evaluation techniques (e.g. test based on tutorial or small case study) help to get familiar with the tools' functionalities.

4. Investigation of the process requirements is performed in correspondence to functional analysis. The phase tries to spot (in)adequacies between the user activities and the support offered by the tool.

5. Investigation of the product requirements involves assessing usability, performance and reliability. The evaluation team should also investigate which portion of maintenance could be fulfilled by the tool users internally, and which should be redirected to the RE-tool vendors.

6. A decision about the RE-tool selection is made after summarising the results from phases 3, 4 and 5. One of three decisions should be made: (i) the users adopt the "best-evaluated" RE-tool without changing the RE process; (ii) the users start using the "best-evaluated" RE-tool, but they have to reconsider the RE process; or (iii) the "best-evaluated" tool is not suitable for the users and they need to repeat the RE-tool evaluation (reconsider requirements, and/or search for other tool candidates).

3. Research Method

In this section we start performing activities of the R-TEA method. Section 3.1 presents how the selection of the tool requirements and preparation of the requirements specification was performed (step 1). In section 3.2, the pre-study and selection of the goal-modelling tools are described (step 2). Section 3.3 introduces the evaluation technique used to investigate the tools (steps 3-5).

3.1. Preparation of Requirements Specification

The preparation of the requirements specification for the tool evaluation involves the definition of requirements according to which tool compatibility will be assessed. We have chosen to use two most recent evaluation frameworks: the role-based framework [8] and the R-TEA frameworks [16]. In each framework, we identified 50 requirements deemed relevant for goal-modelling tools (see appendix Table 9 and 10). As a simplifying assumption, we considered that each of the requirements was of the same importance.

Instead of preparing a requirements specification for tool selection, we adapted each of two frameworks to evaluation forms. In these forms, it is asked that the compatibility of the tools wrt each requirement be evaluated on a scale from 1 to 5 (1 – compatibility is poor; 5 – compatibility is excellent). Other assessment values included 0 (the tool fulfils the requirement but this capability was not used when creating a goal model) and -1 (the requirement was not understood).

3.2. Selection of Tools (Pre-study)

The second step of the R-TEA method is the selection of the tools to be tested. After screening the research literature and web resources on goal modelling, we identified 7 tools (Table 1) and performed their pre-evaluation. All the tools are research prototypes except KAOS/Objectiv*er* which is a commercial tool. We have been using the R-TEA functional framework to make a first assessment of the tools. In addition, we paid attention to some non-functional requirements like tool reliability (absence of tool malfunctions) and user-friendliness (how easy is to perform the basic functions). The pre-study was performed by "executing" tool tutorials and/or other material found of the tool's website. Based on the results of the pre-study[1] we have selected *i**/OME and KAOS/Objectiv*er* to be used in the experiment. Although TROPOS/ST-Tool is evaluated higher than other tools and seems to be the most mature academic project in the field, for the purpose of the experiment we decided to use the best evaluated *i** tool (*i**/OME) because it supports a more mainstream version of the *i** language.

3.3. Investigation of Tools. Evaluation technique

The experiment was carried on at the University of Namur (UoN) with 19 Computer Science graduate students in their 2nd year[2]. The experiment was a part of the mandatory assignments of the Requirements Engineering course. The students were divided into four groups of 4-5 persons (see Table 2). The treatment involved the course material and theoretical lectures given to the students. Attending the lectures was not compulsory but participants actively participated (minimum 17 participants per lecture). So, all the participants received the same treatment and information about the experiment.

The experiment consisted of three steps (Figure 2): interviewing, creating goal models and evaluating tools.

[1] In Table 1, the pre-study results are shown as the sums of tool functional compatibility values.

[2] That is, the 4th year of the whole Computer Science curriculum.

Table 1. Goal-modelling tools

Supported language/Tool	Project type	Pre-study result	Selected
***i**/OME** [18]	Research project at University of Toronto	88	Yes
***i**/OpenOME** [19]	Research project at University of Toronto	63	No
***i**/DIA plug-in** [4]	Open source project	11	No
***i**/TAOM4E** [1]	Research project at ITC-IRST, Trento	58	No
KAOS/Objectiver [17]	Commercial tool by Respect-IT	172	Yes
KAOS/DIA plug-in [4]	Open source project	7	No
TROPOS/ST-Tool [3]	Research project at University of Trento	114	No

Table 2. Instruments used in the experiment

	Group 1	Group 2	Group 3	Group 4
Group size	5	5	5	4
Tools	*i**/OME	KAOS/Objectiver	*i**/OME	KAOS/Objectiver
Framework	R-TEA frameworks	Role-based framework	R-TEA frameworks	Role-based framework

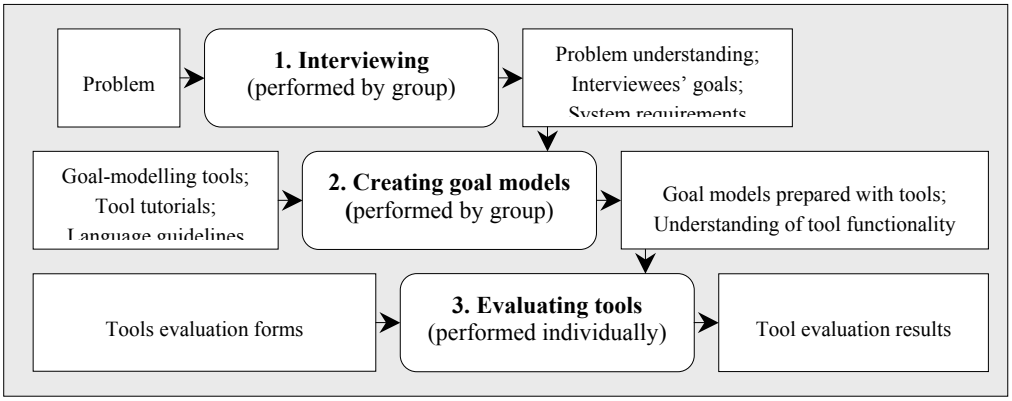

Figure 2. Experiment design

Interviewing. The experiment was initiated by the presentation of its settings to the participants. The problem for which the participants had to create goal models was stated in one sentence: "What are the major goals and requirements for an information system to be used by academics and researchers at our university for reporting on scientific activities." All the details – goals and requirements – had to be discovered by the students by interviewing two users and one developer of the existing system. The interviewees were involved neither in the experiment nor in its treatment. The participants had followed lectures on requirements elicitation and were simply asked to use the techniques that they found the most appropriate. They all chose face-to-face interviews with open-ended questions. Each interview session lasted 30 minutes (1 hour 30 minutes on total for one participant group). The interviews provided the participants with an understanding of the problem domain, interviewees' goals and system requirements.

Creating goal models. Each group was then randomly assigned a goal modelling tool (Table 2). In addition, the groups were provided with the tool tutorials and guideline documents on how to use the modelling language supported by the tool. The groups worked for two weeks independently; but they could always ask questions to the teaching staff. Besides delivering goal models, the participants also acquired knowledge and understanding on the functionality of the tool.

Evaluating tools. In the last step, each student, individually, had to fill in the tool evaluation form introduced in Section 3.1 and prepared according to the assigned framework (see Table 2). When filling the evaluation form, the participants were advised to run the tool in order to be sure about the evaluation. The outcomes of this step were the tool evaluation results: 10 filled evaluation forms based on the R-TEA frameworks, and 9 filled evaluation forms based on the role-based framework.

4. Results

In this section we use hypothesis testing to answer our research question. We also analyse how the results received in the pre-study and experiment correlate between each other. Finally we discuss the threats to the validity of our experiment.

4.1. Analysis Method

Based on the research question formulated in the introduction we define the null hypothesis that tools give equal support (H_{10}), and an alternative hypothesis

that KAOS/Objectiv*er* is better (H_{11}**)**. The motivation for the latter is that KAOS/Objectiv*er* is a commercial tool, while *i**/OME is a research prototype.

H_{10}: Both, *i**/OME and KAOS/Objectiv*er* provide the same level of support to create goal-models.

H_{11}: KAOS/Objectiv*er* provides better support to create goal models than *i**/OME.

The subsequent hypotheses address the validity of the tool evaluation. Again, for each, we have a null and an alternative hypothesis:

H_{20}: Tool assessment results received in the pre-study and in the experiment are the same.

H_{21}: Tool assessment results received in the pre-study and in the experiment are not the same.

H_{30}: Agreement (between the participants) on the tool evaluation is the same both in the pre-study and in the experiment.

H_{31}: Agreement on the tool evaluation is not the same in the pre-study and in the experiment.

The second hypothesis (H_2) tests whether the tool evaluation performed by participants corresponds to the pre-study results (for *i**/OME and KAOS/Objectiv*er*) described in section 3.2. The third hypothesis (H_3) investigates the variation of the tool evaluation among the participants. Even if the tools are evaluated differently in the pre-study and in the experiment, the difference might be in the different interpretation of the requirements' support levels.

In order to address the first and the second null hypothesis we will use the *t-test*, *paired two sample for means*. The third null hypothesis will be analysed using the *t-test*, *two-sample assuming equal variances*.

4.2. Tool Comparison

The tool evaluation results are shown in the appendix. Since we were using two evaluation frameworks [8, 16], we need to apply the paired t-test twice. Table 3 shows the summary of the t-test when it was applied to the results received using the role-based framework (see Table 9 in appendix) and the R-TEA frameworks (see Table 10 in appendix) respectively. The t-test indicates that we can reject H_{10} and accept the alternative hypothesis H_{11}. The result indicates that KAOS/Objectiv*er* does provide better means to create a goal model.

4.3. Validation of Tool Evaluation

In the pre-study we used the R-TEA functional framework. We now compare the pre-study results and the experiment results received when using the R-TEA frameworks. Table 4 shows the summary of the *t-test* (*paired two sample for means*) for the tool evaluation in both cases. The result shows that we can reject the null hypothesis H_{20} for evaluation of *i**/OME. This means that in the experiment *i**/OME was evaluated better than in the pre-study. But we cannot reject H_{20} for KAOS/Objectiv*er*, because the t-test is lower than critical t value. In both cases this tool was assessed on the same level.

Table 3. *t-test* of the tool evaluation means using different evaluation frameworks ($t_{crit\ two\ tail}$ = 2,009575; df=49; α=0,05)

Framework	Tools	Mean of means	Standard deviation	p-value	t-test
Role-based framework	***i**/OME**	2,095	2,30758	0,001586	**3,3448**
	KAOS/Objectiv*er*	2,811	1,68002		
R-TEA frameworks	***i**/OME**	1,988	1,4994	0,0000902	**4,2674**
	KAOS/Objectiv*er*	2,801	1,9798		

Table 4. *t-test* of the tool evaluation ($t_{crit\ two\ tail}$ = 2,030107915; df=35; α=0,05)

Tools	Evaluation	Mean of evaluations	Standard deviation	p-value	t-test
***i**/OME**	Pre-study	1,208	1,9196	0,0226	**2,3858**
	Experiment	1,594	0,8948		
KAOS/Objectiv*er*	Pre-study	1,972	3,4135	0,1454	**1,4892**
	Experiment	2,505	2,1752		

Table 5. *t-test* of the agreement about tool evaluation ($t_{crit\ two\ tail}$ = 1,994437; df=70; α=0,05)

Tools	Evaluation	Mean of evaluations	Standard deviation	p-value	t-test
***i**/OME**	Pre-study	1,208	1,9196	0,1717	**1,3809**
	Experiment	1,594	0,8948		
KAOS/Objectiv*er*	Pre-study	1,972	3,4135	0,1809	**1,3515**
	Experiment	2,505	2,1752		

4.4. Agreement about Tool Evaluation

Agreement on the tool evaluation in the pre-study and the experiment could be compared using the variance measure. Table 5 shows the summary of the *t-test* (*two-sample assuming equal variances*) for the agreement of tool evaluation in both cases.

Table 5 shows that we cannot reject the null hypothesis H_{30}. So even if *i**/OME is evaluated differently (see section 4.3), the difference exists only in the (subjective) choice of the evaluation range.

4.5. Threats to Validity

We will analyse in turn the threats to conclusion, internal, construct and external validity. Threats to *conclusion validity* concern issues that affect the ability to draw the correct conclusion about the relationship between the experiment's treatment and its outcome [24]. Conclusion validity deals with the *power of the statistical test*. If the power is low, there is a high risk that conclusions are incorrect. Another threat is the *experiment treatment*. Participants were given treatment related to RE in general, but not in particular to the experiment settings. Validity might also depend on the *formulation of the compatibility requirements* and the *design of the evaluation form*. In order to mitigate the latter threats, the evaluation form was reviewed by two independent researchers not involved in the experiment.

Threats to *internal validity* analyse issues that might indicate a causal relationship [24]. We were not influencing the formation of the *participant groups*. However the participants might have formed their group according to the known skills of their colleagues. Two other threats to internal validity are related to the settings of the pre-study and the experiment. First, the *experiment design* was prepared by the same researcher who performed the pre-study. Second, the *experiment treatment* was given by the same person who designed the experiment. Both cases might influence the internal validity, however, in order to decrease these threats the experiment itself was conducted as a self-controlled exercise by the participants.

Threats to *construct validity* concern the extent to which the experiment settings reflect the construct under study [24]. In the experiment there is a threat of *misunderstanding* or *misinterpretation* of the requirements according to which the tools were evaluated. In the evaluation form participants identified 11,79% (evaluation set to "-1") of requirements that were not understood. But the real threat appears when participants interpret a certain requirement in an unintended way. In order to mitigate this threat we provided a self study material describing both evaluation frameworks. Also, having the same person in charge of both the treatment and the experiment design (see above) has the advantage that the terminology is more consistent between the courses and the forms.

Threats to *external validity* refer to the ability to generalise experiment results outside the experiment settings [24]. A threat to external validity might be that the participants had no *real ambition* to select a tool; hence the motivation for performing an elaborate evaluation may have been smaller than in a real case. Furthermore, the experiment involved *students* rather than practitioners. Hence, the participants had basic knowledge but limited experience in RE practice. But they all were following the same study program for 3,5 to 4 years, i.e., they were quite homogeneous regarding age and background. The use of students is a common experimentation approach in software engineering (e. g., [11, 20]). Since the participants were in their 4th year and had only 1 study year left, their knowledge were quite close at least to junior practitioners.

5. Discussion

In this section we discuss the low and high evaluated tool requirements. We then highlight the correspondences between tools used in the experiment and the language that these tools support, as well as the quality of the goal models created with the tools. We end the section with a survey of related work.

5.1. Tool evaluation

In the appendix we provide the average capability values of tool compatibilities for each requirement/tool pair. Based on them we define requirements compatibility to tool as *low* (if mean < 2), *medium* (if 2 ≤ mean ≤ 3,5), and *high* (3,5 < mean).

Common low and high requirements selected from both frameworks are summarised in Table 6. We did not calculate correlations between different requirements, but certain tendencies could be easily noticed. For example, limitation to define user accounts results in no support for collaborative work activities, and also influences agreement, negotiation and discussion facilities. In general both tools have high evaluated requirements for graphical representation of the goal model. But formal representation is evaluated as low although there are possibilities to define formal goal models in both tools. Most likely, this is because (i) it was not required by the experiment settings and (ii) formal definition is performed by typewriting and not edited through an editor of a tool. Table 6 also shows the importance to evaluate non-functional requirements (such as ease of use, ease of learning, efficiency to solve problem) requirements of the tools. Non-functional requirements might be an important factor for tool acceptance.

Some of the requirements (see Table 9 and 10 in appendix) are not evaluated by the participants. The reason might be threefold: (i) the tool does not support the requirement, (ii) the participants were not using the particular feature of the tool to solve the problem, or (iii) the participants were not able to identify the

requirement in the tool although the tool supports it. Finally, it is also not surprising that KAOS/Objectiver, a commercial tool, was evaluated better then *i**/OME, a research project. However, the experiment on tool evaluation showed the features of both tools that should be improved in later developments both for supporting goal model creation and as well as its maintenance in later RE activities.

Table 6. Weak and high evaluated requirements for goal-modelling tools

Low evaluated requirements (requirement ID number from the framework)	***i**/OME**	**KAOS/Objectiver**
Definition of user accounts and groups (RQ39, RQ40 and FEF2.1.1, FEF2.1.2)	X	X
Agreement, negotiation, discussion means (RQ15 and FEF2.1.4)	X	
Collaborative work (RQ37 and FEF2.3.2; FEF2.3.3)	X	X
History (versions) of requirements model (RQ16-21 and FEF2.1.5)	X	
Interfaces with other tools (RQ13, RQ30-32 and FEF1.5, NF1.4)	X	X
Extensibility (RQ44-46 and NF2.6)	X	
Traceability between different kind of information (RQ25 and FEF1.4.5)	X	
Reporting (RQ33-34, RQ36 and FEF3.2.1-3)	X	
Structured information import (RQ29, RQ32 and FEF1.1.3)	X	X
Formal requirements definition (FEF1.3.)	X	X
Requirements specification (RQ5 and NF1.1)	X	X
High evaluated requirements (requirement ID number from the framework)		
Unique identification (RQ.1 and FEF1.1.2)		X
Graphical definition of requirements model (RQ4, RQ9 and FEF1.2)	X	X
Basic formatting (RQ11)	X	X
Reuse (RQ3 and FEF3.1.1-2)		X
Number of element entries not fix (RQ41)		X
Views (RQ8, RQ10 and FEF2.2.1, FEF3.2.2)		X
Number of database entries (RQ41)	X	X
Two directional traceability links (RQ22, RQ23, RQ26 and FEF1.4.5)		X
Easy to learn (NF2.1.1)	X	X
Efficiency to solve problem (NF2.1.2)	X	X
Easy to remember (NF2.1.3)	X	X
Easy to understand (NF2.1.5)	X	

5.2. Language, Goal Model and Tool

Beside the comparison of the goal-modelling tools we have also investigated the correlation between the tools and (i) the quality of the language they support as well as (ii) the quality of the models they helped to produce. Both the languages and the created goal models were evaluated by the participants using the semiotic quality framework (SEQUAL) [13], a well accepted model and language quality assessment framework. SEQUAL considers various dimensions of quality: physical, empirical, semantic, syntactic, pragmatic and social quality. Because of the space limits we will not discuss all the observed dependencies, but just provide the basic ones in Table 7. The results indicate that even if we have a tool and a language that are of high quality, they do not guarantee that the product (the created goal model) will be of high quality too. The reason for this might be found in the literature [12]. In this paper, the authors survey most goal-oriented languages in RE and classify them according to their suitability to support RE activities. There, KAOS is characterised as more suited to support late RE activities performed when specifying requirements. On the other hand, *i** is judged more suitable for eliciting early requirements. As a consequence, KAOS might appear more complete in terms of constructs. It makes the language richer. But when one has to make a first, high level, goal model, the language may appear too rich.

Table 7. Comparison of modelling instruments

Comparison		**Results**
Tools support for creation of goal models	*i**/OME	KAOS/Objectiver
	KAOS/Objectiver	
Languages quality	*i**	KAOS
	KAOS	
Goal model quality	*i** model	*i** model
	KAOS model	

5.3. Framework comparison

In this work we adapted two frameworks – role-based [8] and R-TEA [16] – to assess the goal-modelling tools. In addition, the participants were asked to evaluate the performance of the two frameworks according to the criteria listed in Table 8. We formulated the hypothesis that both frameworks provide equal support for tool evaluation. But the *t-test* showed that we need to reject the null hypothesis and accept the alternative saying that performance of the R-TEA frameworks is better than the role-based framework. Those results will be detailed in further publications.

Table 8. Framework evaluation and t-test of their comparison ($t_{crit\ two\ tail}$ =2,5706; df=5; α=0,05)

Criteria	Role-based framework	R-TEA framework
	Mean of 9 evaluations for each criteria	Mean of 10 evaluations for each criteria
Help you to evaluate tools	3,11	3,2
Easy to understand framework requirements	2,56	2,9
Sufficient material to learn the framework	2,67	3,2
Framework contributes to understanding of the tools	2,22	2,8
Framework contributes to understanding of how tool should look like	3,44	3,8
Use of the evaluation framework for a real life problem?	2,67	3,3
Mean of means	2,78	**3,2**
Variance	0,1861	0,124
p-value	0,0036	
t-test	**5,1555**	

5.4. Related work

There is a wide variety of approaches to assess commercial-off-the shelf (COTS) products. The most extensive surveys are provided in [22] and [16]. The surveys report that approaches essentially focus on the process of defining the compatibility requirements. Furthermore, we found only three reported cases [2, 15, 16] where RE-tools were evaluated. For instance, the procurement-oriented requirements engineering method [15] introduces a case that starts with 30 RE-tools (later narrowed to 6) and results with 2 tools suggested for use. The work also draws important lessons for the selection of the COTS products such as "structure requirements", "stakeholder representatives should be present", "record information", and others.

Elsewhere [2], a quality-based approach is used jointly with the COSTUME (composite software system quality model development) to assess 5 RE-tools. The case study contributes with the description of the RE-tool domain and provides guidelines how to construct quality models based on the ISO/IEC 9126-1 standard.

Separate constraints of the R-TEA method are tested in several case studies in the academic environment [16]. The most comprehensive one includes the assessment of 4 commercial RE-tools by ten different participant groups. The study shows the usefulness of R-TEA, and points out the importance of different requirements (functional *vs* non-functional) categories.

The most closely related work performed with the goal-modelling tools is the *i**-wiki project [9]. The project proposes a questionnaire to assess the tool maturity level, extensibility and some interoperability issues. An interesting category is "*i** suitability" that might serve as an extension to the R-TEA frameworks when analysing the (*i**-based) goal-modelling perspective in tools. However the questionnaire facilitates only tool description, but not in-depth analysis. 4 goal-modelling tools (OME [18], OpenOME [19], TAOM4E [1] and REDEPEND_REACT-BCN [6]) are overviewed; however the survey might be subjective in the sense that it was performed by tool developers or promoters.

We have found no case studies considering *goal-modelling tools*. In this work, we screened 7 goal-modelling tools in the pre-study and compared 2 of them in details during the experiment. We also investigated dependencies between (i) the tools and languages they support and between (ii) the tools and the goal models produced with these tools.

6. Conclusions and Lessons Learnt

This paper reports on an experiment where the RE-tool evaluation approach (R-TEA) is used to compare goal-modelling tools *i**/OME and KAOS/Objectiv*er*. The experiment shows that KAOS/Objectiv*er*, a commercial goal-modelling tool, provides better support to create goal models. However, we observe that the quality of a goal model does not necessarily depend on the general quality of the means (language and supporting tool) used to create it, but rather on particular characteristics of the modelling language with respect to a given context [12].

As most of the current goal-modelling tools are prototypes [1, 3, 6, 18, 19], the experiment highlights improvements required to adequately support RE activities. The basic ones include means to negotiate and agree about goal/requirements model. It is also important to improve traceability between a goal

model and its informal and formal representations. Finally, in order for the goal-modelling tools to become more mature, they should be able to prepare and maintain not only the goal models, but also the requirements specifications which are the output of RE.

Possible future works include the repetition of similar experiments in order to validate and enhance the current findings. We also plan to evaluate tools that support other goal-modelling languages than *i** and KAOS. In addition to tools, the evaluation of goal-modelling languages is also in our scope.

Acknowledgement. We wish to thank Robert Darimont for supporting and providing us the tool tested in the experiment.

References

[1] **D. Bertolini, L. Delpero, J. Mylopoulos, A. Novikau, A. Orler, A. Perini, A. Susi, B. Tomasi.** A Tropos Model-Driven Development Environment. *CAiSE 2006 Forum proceedings*, 2006, 56-59.

[2] **J.P.Carvallo, X. Franch, C. Quer.** A Quality Model for Requirements Management Tools. *Requirements Engineering for Sociotechnical Systems*, *Idea Group Publishing*, 2004.

[3] **P. Giorgini, F. Massacci, J. Mylopoulos, N. Zannone.** ST-Tool: A CASE Tool for Security Requirements Engineering, *Proceedings of the* 13*th IEEE International Conference on Requirements Engineering* (*RE'05*), 2005, 451-452.

[4] **DIA**, URL: *http://www.gnome.org/projects/dia/*.

[5] **S. Easterbrook.** Requirements Engineering, *Lecture Notes*, *University of Toronto*, 2006.

[6] **G. Grau, X. Franch, N. Maiden.** REDEPEND-REACT: an Architecture Analysis Tool. *Proceedings of the* 13*th IEEE International Conference on Requirements Engineering* (*RE'*05), 2005, 455-456.

[7] **E. Haywood, P. Dart.** Towards Requirements for Requirements Modelling Tools. *Proceedings of the* 2*nd Australian Workshop on Requirements Engineering* (*AWRE'*97), 1997, 61-69.

[8] **M. Hoffmann, N. Kuhn, M. Weber, M. Bittner.** Requirements for Requirements Management Tools. *Proceedings of the IEEE Joint International Conference on Requirements Engineering* (*RE'*04), 2004, 301-308.

[9] ***i**-wiki project**. *URL: http://istar.rwth-aachen.de/tiki-index.php*.

[10] **INCOSE**. *INCOSE: Tools Survey: Requirements Management (RM) Tools by International Council on Systems Engineering (INCOSE) URL: http://www.incose.org/*, 2002.

[11] **L. Karlsson, P. Berander, B. Regnell, C. Wohlin.** Requirements Prioritisation: An Experiment on Exhaustive Pair-Wise Comparison versus Planning Game Partitioning. *Proceedings of the Empirical Assessment in Software Engineering*, 2004.

[12] **E. Kavakli, P. Loucopoulos.** Goal Modeling in Requirements Engineering: Analysis and Critique of Current Methods. *Krogstie J., Halpin T., Siau K. (eds), Information Modeling Methods and Methodologies*, *Idea Group Publishing*, 2005, 102–124.

[13] **J. Krogstie.** A Semiotic Approach to Quality in Requirements Specifications. *Proceedings IFIP* 8.1 *Working Conference on Organisational Semiotics*, 2001.

[14] **M. Lang, J. Duggan.** A Tool to Support Collaborative Software Requirements Management. *Requirement Engineering*, 6 (3), 2001, 161-172.

[15] **N.A. Maiden, C. Ncube.** Acquiring COTS Software Selection Requirements. *IEEE Software*, 1998, 46-56.

[16] **R. Matulevičius.** Process Support for Requirements Engineering: A Requirements Engineering Tool Evaluation Approach. *PhD theses*. *NTNU*, 2005, 309.

[17] **Objectiver**. *URL: http://www.objectiver.com/*.

[18] **OME**. *URL: http://www.cs.toronto.edu/km/ome/*.

[19] **OpenOME**,*URL: http://www.cs.toronto.edu/km/openome/*

[20] **P. Shoval, A. Yampolsky, M. Last.** Class Diagrams and Use Cases – Experimental Examination of the Preferred Order of Modeling. *Proceedings of CAiSE 2006 workshop on Exploring Modeling Methods for System Analysis and Design* (*EMMSAD* 2006), 2006, 453-472.

[21] **K. Pohl.** The Three Dimensions of Requirements Engineering: a Framework and its Applications. *Information systems*, 19(3), 1994, 243-258.

[22] **G. Ruhe.** Intelligent Support for Selection of COTS Products. *Proceedings of the NODe* 2002 *Web and Database-Related Workshops on Web, Web-Services, and Database Systems*, 2003, 34 - 45.

[23] **K. Wiegers.** Automating Requirements Management. *Software Development*, 7 (7), 1999.

[24] **C. Wohlin, P. Runeson, M. Høst, M.C. Ohlsson, B. Regnell, A. Wesslen.** *Experimentation in Software Engineering*, *Kluwer Academic Publishers*, 2002.

Received August 2006.

Appendix. Evaluation of Goal-modelling Tools

Table 9. Evaluation of goal-modelling tools using the role-based framework

Category	Requirements	*i**/OME		KAOS/Objectiver	
		Number of evaluations	Ave-rage	Number of evaluations	Ave-rage
1	2	3	4	5	6
Information model	**RQ1.** Every object must be **uniquely identifiable** over its lifetime.	5	2,60	3	4,00
	RQ2. The model must be **changeable** during the project.	5	3,20	4	5,00
	RQ3. Reuse should be available for all classes, types and attributes.	4	2,75	3	3,67
	RQ4. It could be possible to define the model **graphically**.	5	4,80	4	4,50
	RQ5. The tool could support models that are needed when using **standard RE templates** (e.g., Volere or IEEE 830-1998).	1	3,00	0	
Views	**RQ6.** The tool must allow views to be defined in a **user-specific manner**.	5	3,00	4	2,25
	RQ7. The views must be freely **configurable**, including complex filters on objects, relations, and attributes.	5	2,20	4	2,50
	RQ8. The objects must be **changeable** in the current view.	5	3,20	4	5,00
	RQ9. Graphical views of the requirements should be available.	5	4,40	4	4,50
	RQ10. The tool should allow views to be **predefined for user roles**.	5	1,20	2	4,00
Formatting, Multimedia and External files	**RQ11.** The tool should support the **basic formatting**.	5	4,20	4	4,50
	RQ12. Non-text objects should be saved.	5	2,60	2	4,50
	RQ13. External objects must be viewed either through a pre-viewer inside the tool or in the native application if called directly from the tool's user interface.	5	1,00	1	3,00
Change Management and Comments	**RQ14.** The **change requests** should have public status information like **pending**.	0	0	1	1,00
	RQ15. There could be a **comments** or **discussion function** tightly linked to the requirements	5	1,00	3	4,67
Documen-tation of the history	**RQ16.** All **changes** to the requirements must be **tracked**.	5	1,00	4	2,25
	RQ17. The **object** in the tool must be **versioned**.	5	1,00	3	2,00
	RQ18. Changes must be **tracked** down to the smallest unit of data structures.	5	1,00	4	2,25
	RQ19. A tool must allow a requirement to be **changed back** to any previous state anytime.	5	1,00	4	2,75
	RQ20. The tool must generate freely configurable **change reports**.	5	1,00	3	2,00
	RQ21. A **comment** should be saved with a **change** to enable it to **be understood** later on.	5	1,00	3	3,00
Traceability	**RQ22.** Link must be directed and an **object** must be a **source and target** at the same time.	5	5,00	2	3,50
	RQ23. It must be possible to follow **link directly in both directions**.	5	4,00	3	4,33
	RQ24. It must be possible to give the **link attributes**.	5	2,60	4	3,00
	RQ25. It must be possible to create roles for governing **what kinds of requirements** must **have links** to what other **kind of requirements**.	5	1,00	4	3,25
	RQ26. Links must **connect any objects**, not only in the same subset.	5	4,80	4	3,75
	RQ27. The tool must feature a **practical**, **user friendly** and **concise** graphical presentation and navigation of the traces.	5	3,00	3	3,00
Baselining	**RQ28.** The tool must support **baselines**.	0	0	1	4,00
Analysis function	**RQ29.** The tool can **scan** the description texts of the requirements **for** patterns like **unsuitable/inexact language** or **wrongly used terminology**.	5	5,00	2	1,00
Tool integration	**RQ30.** The tool must have open **interfaces to other tools** used in the development process and make information stored in them visible and linkable.	5	1,00	1	1
Import	**RQ31.** The tool must **recognise** text marks, formatting, line ends, grammatical structure or keywords to **interpret** them as the beginning or end of requirement texts.	5	1,20	3	2,67
	RQ32. The tool should support a **semiautomatic import** of requirements from existing documents.	5	1,20	0	0
Document generation	**RQ33.** The subset of data to be included in the document must be **flexibly configurable**, **comparable to views**.	1	1,00	1	3,00
	RQ34. The document generation must be able to include **all information available in the tool**.	5	1,60	4	3,25
	RQ35. The document generator must be able to create document in a **certain standard formats**.	5	3,80	4	3,00
	RQ36. The document generator must be **extensible** via a programming interface provided by a tool.	5	1,00	2	2,50
Collaborative working	**RQ37.** It must be possible for many user to **work on the same data at the same time**.	5	1,00	3	2,67
Web access	**RQ38.** The tool should have a **web interface** or another browser based client.	5	1,00	2	1,00
User roles and rights	**RQ39.** The administrator must be able to manage **user accounts** and **group** and **role** assignments.	5	1,00	2	1,00
	RQ40. Users must be defined **centrally** for all projects.	1	1,00	2	1,00
Size restriction	**RQ41.** The number of element entries must not be of a **fixed size**.	2	5,00	3	4,67
Workflow management	**RQ42.** The tool could support system development via an administrable, organized and **structured process**.	0	0	1	2,00
	RQ43. The process must not simple restrict the **users**, but **guide** them through the process.	4	1,00	2	3,00
Extensibility	**RQ44.** The tool must provide an open and well-documented model and API which makes all data and functions accessible to **extensions**.	5	1,00	2	2,00
	RQ45. The object model and the API must be **stable** across versions of the tool.	4	1,00	2	2,50
	RQ46. The **user interface** of the tool must be customizable and extensible with a standard script language.	5	1,00	2	2,00
Database	**RQ47.** The tool must be **reliable** (Did you experience any tool malfunction, crashes?).	5	3,00	4	2,75
	RQ48. The data must have a **consistency analysis** and **data integrity check**.	5	5,00	3	3,33
	RQ49. It must be possible to **export all project data** and to **import** them again at a different time or places from/with different tool.	5	1,40	4	2,50
Encryption	**RQ50.** The information stored in the database of the tool must **not** be **readable to intruders**.	1	1,00	2	2,50

Table 10. Evaluation of goal-modelling tools using the R-TEA frameworks

Category	Requirements	*i**/OME		KAOS/Objectiver	
		Number of evaluations	Ave-rage	Number of evaluations	Ave-rage
1	2	3	4	5	6
FEF1.1. Specify **uniquely identifiable** description using **informal language**.	**FEF 1.1.1** provide **natural language** description.	3	1,33	4	3,25
	FEF 1.1.2 allow specifying **unique identification** (ID) for each separate requirements (goal/requirement/actor/etc).	5	1,00	3	4,33
	FEF 1.1.3 allow **importing of requirements** (goals/actors/etc.) and their description from text document.	5	1,00	2	1,00
FEF1.2. Specify requirements using **semi- formal** language(s).	**FEF 1.2.1** provide tools for **semiformal** (graphical) language description.	5	4,20	5	5,00
	FEF 1.2.2 provide **forward/ backward traceability between semiformal**, and informal, formal descriptions.	5	1,40	0	0
FEF 1.3. Specify requirements using **formal** language(s).	**FEF 1.3.1** provide tools for **formal** language description.	4	1,00	5	1,80
	FEF 1.3.2 provide **forward/ backward traceability between formal** and informal, semiformal descriptions.	5	1,20	3	1,00
FEF 1.4. Define **traceable associations** between requirements and the different elements of requirements specification.	**FEF 1.4.1** provide functions for testing traceability **between informal, semiformal and formal** requirement description.	5	1,00	4	1,25
	FEF 1.4.2 create **parent-child** traceable relations between requirements.	5	3,60	5	4,40
	FEF 1.4.3 maintain **peer-to-peer** traceable relations between requirements.	5	3,80	5	3,20
	FEF 1.4.4 maintain traceable relation between **various related information**.	3	2,67	2	4,00
	FEF 1.4.5 maintain forward/ backward traceability between a **source of requirements** (goals/actors/etc.), **the requirements** (goals/actors/etc.) and **design**.	4	1,00	1	4,00
FEF 1.5. Connect seamlessly with **other tools** and systems, by supporting interoperable protocols and standards.	**FEF 1.5.1** allow **importing/exporting** requirements (goals/actors/etc.) description from/to **text documents**.	5	3,00	3	2,00
	FEF 1.5.2 allow **importing/exporting** requirements (goals/actors/etc.) description from/to **graphical** documents.	5	3,00	3	1,67
FEF 2.1. Maintain an audit trail of **changes**, archive **baseline** versions; and engage a mechanism to **authenticate** and **approve change** requests.	**FEF 2.1.1** maintain **user authentication** to the system (i.e. user name, password).	5	1,00	5	1,00
	FEF 2.1.2 allow **grouping users** into different groups.	5	1,00	5	1,00
	FEF 2.1.3 allow creating **different views** for **different groups of stakeholders**.	5	2,20	4	2,00
	FEF 2.1.4 register **agreement/ rationale/ discussion/ negotiation/ changes/ history** of requirements and by how it was achieved.	5	1,00	5	1,00
	FEF 2.1.5 call the **earlier requirement** (goals/actors/etc.) **description/ versions** and register them into **history** context.	5	1,00	4	1,00
FEF 2.2. Classify requirements into logical user- defined **groupings**.	**FEF 2.2.1** allow **specifying attributes/ properties** of the requirement.	5	1,40	4	4,25
	FEF 2.2.2 provide **sorting** according to different attributes/ properties.	5	1,20	4	2,75
	FEF 2.2.3 provide **filtering** according to different attributes/ properties.	5	1,00	2	1,00
FEF 2.3. Support secure, concurrent **cooperative work** between members of a multidisciplinary team, which may be geographically distributed.	**FEF 2.3.1** provide **platform independent** interface for geographically **distributed users**.	5	2,40	3	1,00
	FEF 2.3.2 allow making a copy for **modification of an already approved version** of requirements description in different abstract levels (document, requirement).	4	1,50	4	4,00
	FEF 2.3.3 provide a change approval cycle for multiple change negotiation and approval before posting into common repository.	5	1,00	2	1,00
FEF 2.4. Maintain a **data dictionary** of all project components and requirements in a shared repository.	**FEF 2.4.1** provide the single repository or data and **concept dictionary**.	5	1,00	1	1,00
	FEF 2.4.2 provide separate data dictionaries for **non-technical** and **technical** users.	5	1,00	5	1,40
	FEF 2.4.3 provide the **help system** to the users.	5	1,00	5	5,00
FEF 3.1. Collect and store a **common** system's and a product family's domain **requirements** (goals/actors/ etc.).	**FEF 3.1.1** enable **selection** and **extraction** of common domain requirements and requirements which differentiate systems in product line.	1	1,00	2	4,00
	FEF 3.1.2 incorporate requirements to a concrete project.	4	1,50	4	3,75
	FEF 3.1.3 adapt/ spread **changes in domain requirements** to concrete projects within domain.	5	1,00	4	3,25
	FEF 3.1.4 provide comparison of domain requirements **feasibility**.	5	1,00	3	2,67
FEF 3.2. Generate **reports**, **documents** that comply with standard industrial templates, with support for presentation-quality output and in-built **document quality controls**.	**FEF 3.2.1** provide **wizards** for report generation.	5	1,00	3	2,00
	FEF 3.2.2 provide possibility to print report according to **views and sorting**.	5	1,00	4	4,50
	FEF 3.2.3 provide possibility to print results of **agreement**.	5	1,00	4	1,50
	FEF 3.2.4 provide techniques for **error checking**.	5	3,00	5	4,20
Non-functional process requirements	**NF1.1.** Support the requirements specification **standards** (e.g. IEEE830-1998, Volere).	3	1,00	4	3,25
	NF1.2. Support the selected modelling perspectives (e.g. **goal-oriented modelling**).	5	5,00	5	4,00
	NF1.3. Support the software development models (Could the RE-tool be used when applying **waterfall, spiral, transformational**, etc, development).	5	2,60	2	2,00
	NF1.4. Support the interfaces with the text, **editing**, **modelling** and **implementation** tools (please identify [if any] the observed tools in the **comments**).	1	1,00	5	2,80
Non-functional product requirements	**NF2.1.1** How **easy is** it **to learn** the functionality of the tool.	5	4,20	5	4,20
	NF2.1.2 Is the tool **efficient** enough to solve the problem specified in the exercise?	5	3,80	5	4,20
	NF2.1.3 How **easy is** it **to remember** the functionality of the tool**?**	5	4,20	5	4,20
	NF2.1.4 Are you **satisfied** with the tool usage?	5	2,60	5	4,20
	NF2.1.5 How easy is it to **understand** the functionality of the tool?	5	4,20	5	3,40
	NF2.2. Provide a satisfying **reliability** of the system.	5	3,00	5	4,40
	NF2.3. Provide a satisfying **performance** of the system.	5	4,00	4	4,25
	NF2.5. Does the tool have sufficient functionality for ensuring the **safety** of the information (safety from unauthorised use of data)	5	1,60	5	2,80
	NF2.6. How **easy** is it **to extend** the tool with additional functionality?	5	1,60	3	2,00
	NF3.1 Does the tool have **sufficient material** for understanding it? (tutorials, illustrative examples, information on the Web, etc.)	5	3,20	5	4,20