Requirements Eng (2008) 13:87–115
DOI 10.1007/s00766-007-0056-y

ORIGINAL ARTICLE

Clear justification of modeling decisions for goal-oriented requirements engineering

Ivan J. Jureta · Stéphane Faulkner · Pierre-Yves Schobbens

Received: 17 December 2006 / Accepted: 30 August 2007 / Published online: 9 October 2007

Abstract Representation and reasoning about goals of an information system unavoidably involve the transformation of unclear stakeholder requirements into an instance of a goal model. If the requirements engineer does not justify why one clear form of requirements is chosen over others, the subsequent modeling decisions cannot be justified either. If arguments for clarification and modeling decisions are instead explicit, justifiably appropriate instances of goal models can be constructed and additional analyses applied to discover richer sets of requirements. The paper proposes the "Goal Argumentation Method (GAM)" to fulfil three roles: (i) GAM guides argumentation and justification of modeling choices during the construction or critique of goal model instances; (ii) it enables the detection of deficient argumentation within goal model instances; and (iii) it provides practical techniques for the engineer to ensure that requirements appearing both in arguments and in model instance elements are clear.

A preliminary version of this paper appears in Proceedings of the 14th International Requirements Engineering Conference (RE'06).

I. J. Jureta (✉) · S. Faulkner
Information Management Research Unit (IMRU), University of Namur, 8, rempart de la vierge, 5000 Namur, Belgium
e-mail: iju@info.fundp.ac.be

S. Faulkner
e-mail: stephane.faulkner@fundp.ac.be

P.-Y. Schobbens
Institut d'Informatique, University of Namur, 21, Rue Grandgagnage, 5000 Namur, Belgium
e-mail: pys@info.fundp.ac.be

Keywords Goal modeling · Argumentation · Clarification · Goal-oriented requirements engineering

1 Introduction

Requirements engineering (RE) is a structured approach to the assessment of the role that a future information system (IS) is to have within a relatively well-delimited human and/or automated environment. It involves the identification of goals to be achieved by the IS, their operationalization into implementable IS services and constraints, the identification of resources required to perform those services and the assignment of responsibilities for the resulting requirements to agents, such as humans, devices, and software.

A usual starting point in RE is the elicitation of goals that the future IS will need to achieve once developed and deployed [57]. Goal modeling can be defined as the activity of representing and reasoning about IS goals using models, in which goals are related through relationships with other goals and/or other model elements, such as, e.g., actions that system agents are expected to execute, resources that they can use, or roles that they can occupy. With a number of currently established RE methods relying on goal models in the early stages of requirements analysis (e.g., [8, 12, 17, 19, 39]; see, [32, 57] for overviews), there seems to be a consensus that goal models are useful in RE.

When an instance of a goal model (henceforth, "goal diagram") is constructed by few stakeholders having similar backgrounds, during a very limited amount of time, and consequently for a relatively simple system, there is generally no need to record the details of the decision process that has led to the final goal diagram. However, as the system under scrutiny gains in complexity—which tends to occur for most but toy systems, and is certainly true for IS employed in automating, e.g., government and healthcare services, air–traffic management, industrial production processes—the inherent inability of individual human stakeholders to grasp the full extent of interactions and interdependencies between system components, to predict with detail and/or certainty the future conditions in which the system is expected to operate and the influence of such conditions on its functioning, makes the construction of the goal diagram an intricate task, critical for the success of the subsequent IS development activities. Moreover, stakeholders' preferences are rarely absolute, relevant, stable, consistent, precise, or exogenous (e.g., [42] and later), thus making it difficult to choose among alternative requirements.

A prominent consequence of system complexity, environment unpredictability, and preference variability is that the requirements provided by the stakeholders will be, among others, ambiguous, overgeneral, and vague (overall: "unclear"), thus unavoidably leading the requirements engineer to transform such information into a clear form written in a goal diagram. This transformation consists essentially of the engineer *interpreting* the information provided by a stakeholder, relating the interpretation to the semantics of the goal model, and finally, establishing how to represent it using the syntax of the model. In doing so, the engineer encounters two significant issues:

- With stakeholders' tendency to express expectations in natural language prone to misinterpretation, it is difficult for the engineer to be assured that her understanding corresponds to what the stakeholders intended to communicate.
- Although it is reasonable to assume that problem and solution knowledge relevant for the engineering of the future IS rests mostly with the stakeholders, system complexity and environment unpredictability make it necessary for the engineer to take an active role in shaping requirements, namely by questioning the rationale behind the expectations expressed by the stakeholders.

Failing to address the above in a structured manner has apparent consequences on the success of the RE phase in system development:

1. Traceability between expectations and their representation in a goal diagram is undoubtedly weak: why a particular natural language requirement is represented in a particular way and not another remains unknown. It is to expect then that stakeholders reviewing the diagram will not know why another stakeholder or the requirements engineer made the given modeling decision. The result may be an unnecessary review of the diagram, changes, or additional explaining. These activities require time and resources that could have been employed in a more productive manner.
2. A stakeholder cannot recall the reasons for making a modeling decision. While goal modeling is an iterative process, it is not uncommon to review the prior decisions because of imperfect recall of reasons leading to them in the first place. Future iterations could be better informed if rationale for prior ones is explicit.
3. The ideas, arguments, and assumptions underlying a decision remain implicit and/or are lost over time. Alternative ideas and confronting views that could have led to different, possibily more adequate modeling choices are lost as well. Both favor a poor understanding of the problem and solution domains. Empirical data suggest that this is an important cause of RE project failure [16].
4. There is no guarantee whatsoever that expectations are not misinterpreted by the engineer, as intuitive detection of ambiguities, vagueness, and so on in stakeholders' statements by itself gives no serious grounds for claiming proper understanding. If there are no checks for clarity of requirements statements, it is difficult for a stakeholder or the requirements engineer to establish whether a statement is unclear and how it can be clarified. As illustrated in the remainder, some such clarity checks are obvious, but many are not at all trivial. Leaving the clarity checking implicit is likely to lead to the application of trivial and intuitive checks, while non-trivial ones will be disregarded for simplicity.
5. Inconsistencies that could have been identified by clarifying initial requirements in a structured manner would remain hidden until later steps of the RE process. Inconsistencies identified early on help in eliciting additional requirements that would otherwise have been missed.

1.1 Contributions

One possible approach to reducing the issues 1–5 is to (i) externalize and document arguments that led the stakeholders to express some requirements as well as the arguments that led the requirements engineer to transform these requirements into goal diagram content; and, (ii) to analyze the clarity of these arguments and of the information given within the goal diagram, revising it if proves necessary. To facilitate tasks (i) and (ii), the present paper proposes the so-called "goal argumentation method (GAM)", which integrates a decision process, an argumentation model, and techniques combined to enable the analysis of argument structure, justification of modeling choices, and clarification of information appearing in arguments and elsewhere in the the goal modeling decision process. Drawing on design rationale literature (see, [40] for an overview), the decision process suggests an intuitively acceptable organization of the goal modeling task. The argumentation model, inspired by work in artificial intelligence argument models (see, [11] for an overview), is introduced in the evaluative step of the decision process, allowing various degrees of structure and rigor in the provision of arguments for modeling choices. The analysis techniques serve for the justification of modeling choices, the study of argument interaction (e.g., defeat and counterargumentation), the detection of deficient argumentation, and the checking for unclear information and subsequent clarification.

An important characteristic of GAM is that it does not integrate a particular goal model; instead, it is independent of specific goal model syntax and semantics, allowing its combined use with any available RE framework which employs goal models. In conjunction with any available goal-oriented RE framework, GAM fulfils three roles:

1. GAM guides argumentation and justification of modeling choices during the construction or revision/critique of IS goal diagrams.
2. It enables the detection of deficient argumentation within available goal diagrams (which need not have been built using GAM).
3. It provides practical techniques for the requirements engineer to ensure (to a reasonable extent) that information appearing both in arguments and in diagram elements is clear (i.e., is not ambiguous, overgeneral, vague, among others).

The principal contributions of this paper to the RE field are the introduction of generic argumentation and clarification conceptualizations and techniques in goal modeling for RE. Hopefully, the present paper highlights that the activities of argumentation and clarification are method-independent concerns and therefore significant for RE at large. Following the initial presentation of GAM at the 14th International Requirements Engineering Conference (RE’06), the present paper extends the method considerably, to include clarification of arguments and stakeholders’ statements of requirements.

1.2 Case studies

While the reader may assume from the above that GAM addresses issues arising only when the complexity of the future system surpasses some considerable threshold, it turns out that such a threshold is, in actual application, unexpectedly low. Having exemplified GAM initially using the classical case study involving the engineering of requirements for a meeting scheduler system [58], GAM was shown to apply to a wide range of settings, including systems that are less complex than many to which GAM has initially been oriented. To ensure that the text is readable and that the main features of the method are salient, the same case study is maintained herein. Since the presentation of preliminary results at the RE’06 conference, GAM was applied to a realistic case study of considerable complexity, involving the clarification and justification of requirements for an air–traffic management (ATM) IS. Requirements are based on extensive documentation providing records of numerous meetings of the stakeholders involved in the European Organisation for the Safety of Air Navigation [20]. Part of the results obtained in the ATM case study are employed herein to illustrate the applicability of some specific features of GAM to the engineering of requirements for complex systems.

In the present paper GAM is used with the Tropos RE method [8, 23]. Tropos’ goal model is based on the well-known *i** modeling framework (see, [60] and related). The *i** notation features a simple yet expressive notation, which is briefly overviewed below (Sect. 2). An advantage of adopting Tropos is that the modeling primitives of *i** are usually present in similar form within most goal-oriented RE frameworks (see, e.g., [57]), thus maintaining the discussion generic.

1.3 Organization

Problems of justification and clarification are first exemplified within the two case studies (Sect. 2). Examples are then used to illustrate the features and use of GAM, that is, the decision process (Sect. 4), the clarification techniques (Sect. 5), and the argumentation model and associated analysis techniques (Sect. 6). After reviewing related work (Sect. 7) and discussing limitations of GAM (Sect. 8), conclusions are drawn and directions for future work identified (Sect. 9). Each section presenting parts of the method provides and discusses a set of definitions of concepts, techniques employing these concepts, and examples illustrating the use of techniques in the case studies. The suggested techniques are derived from our experience in using GAM and the literature associated to concepts employed in GAM but introduced in related research.

2 Illustration of the problem

Consider a system for scheduling meetings, similar to that described in [58, 61]. The meeting scheduler should try to select a convenient date and location, such that most potential participants participate effectively. Each meeting participant should provide acceptable and unacceptable meeting dates based on his/her agenda. The scheduler will suggest a meeting date that falls in as many sets of acceptable dates as possible, and is not in unacceptable date sets. The potential participants will agree on a meeting date once an acceptable date is suggested by the scheduler.

A goal diagram for such a system would be represented in Tropos as an instance of the *i** Strategic Rationale (SR) model. The *i** framework comprises, in addition to the SR model, the so-called "strategic dependency (SD)" model, which features a subset of the modeling primitives of the SR—the latter is therefore taken as the reference Tropos model in the remainder. An example SR diagram for the scheduler, taken as-is from [61], is reprinted in Fig. 1. It shows actors such as *Meeting Scheduler* and *Meeting Participant*, their interdependencies in the achievement of goals, the execution of tasks, and the use of resources, and their internal rationale when participating in the given IS. For example, the *Meeting Be Scheduled* goal of the *Meeting Initiator* can be achieved (represented via a means-ends link) by scheduling meetings in a certain way, consisting of (represented via task-decomposition links): obtaining availability dates from participants, finding a suitable date (and time) slot, proposing a meeting date, and obtaining agreement from the participants. Cloud-shaped elements designate softgoals which differ from goals in that there are no crisp criteria for their satisfaction. Softgoals are commonly used to represent nonfunctional requirements in a goal diagram.

During the RE process, *the appropriateness of this goal diagram can be evaluated on the basis of arguments that support its content*. The appropriateness is defined here as the likelihood of the IS produced from the given diagram to satisfy stakeholder needs. If arguments are missing, alternative diagrams that could be produced by the stakeholders or requirements engineers for the same IS could be considered appropriate, provided that no errors are made when using the syntax and semantics of the relevant goal model. While the SR in Fig. 1 serves as a valuable example to illustrate the syntax and semantics of the *i** framework in [61], it is difficult to accept without justification that diagram as more appropriate than another one in a RE project. Lack of arguments to support the diagram in Fig. 1 lead a stakeholder to challenge it by pointing, e.g., to unnecessary extensiveness or to incompleteness, as in the following questions:

- How does the initiator inform participants that a meeting is being organized?
- Would it not be user friendly for the meeting initiator to inform participants about the meeting using the meeting scheduler?
- Would it not be user friendly if the scheduler looked available dates up in participants' electronic agendas?
- Does the scheduler remind participants of the meeting date? If yes, how/when does it do so? If no, why not?

If arguments were given explicitly for the diagram in Fig. 1, stakeholders might know that, e.g., the initiator prefers to inform participants verbally, that different formats of electronic agendas make it costly to develop a scheduler that can communicate with each participant's software, and so on. Even if such questions are not asked, making it unnecessary for the requirements engineer to address them, ideas and assumptions that could have surfaced and led to additional requirements as a result of the questions would remain hidden. It would be easier to

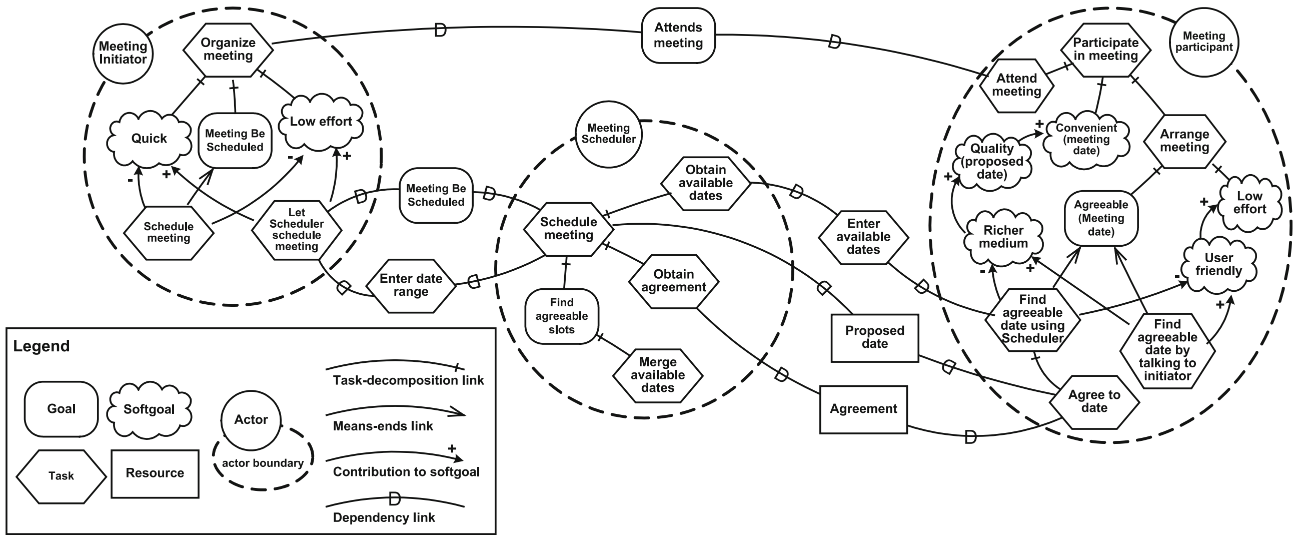


Fig. 1 An *i** Strategic rationale diagram from [61]

consider the goal diagram in Fig. 1 appropriate if modeling decisions leading to it are justified.

Adding arguments alone certainly seems helpful in answering questions such as above, though it is a different issue altogether to ensure answers given in arguments are meaningful, and this in the same (or, at least very similar) way to all relevant stakeholders—it would otherwise be particularly difficult to agree on the relevance and acceptability of arguments given to support modeling decisions; in practice, this translates in extensive counter-argumentation due primarily to lack of clarity. The same applies to information already placed within the goal diagram—consider, e.g., the nonfunctional requirement *Quality of the proposed date* represented as a softgoal in Fig. 1. The stakeholder could ask the following questions:

- What does "quality" of a meeting date mean?
- Who decides when the quality of the date is high/low?
- Under what conditions is the quality of the proposed date considered high or low?
- What if my criteria for meeting date quality are different from those assumed in the diagram?

The questions point to stakeholders' unsatisfactory understanding of the goal diagram, due to a lack of *clarity* of the information in the diagram. For a more elaborate illustration of clarity issues in statements of requirements, consider an actual requirements document [20] which is a result of consultations that took place from 1994 to 1998 of a number of stakeholders selected by and involved in the European Organization for the Safety of Air Navigation. The document compiles information about the needs the given parties expressed regarding the air traffic management strategy and information systems that would support it for the period after the year 2000. The following is an example of a requirement from the cited document (more such examples are given in Sect. 5):

> "For short haul and regional operations there is a need for ATM to make them wait on the ground rather than in the air in case of weather contingencies and to be treated fairly with respect to arriving long haul aircraft in the air while they have not taken off yet. Once given the takeoff clearance they should be able to fly directly to the destination" [20] (p.18).

A number of questions arise, some answerable through a specialized glossary (e.g., What are the conditions for a flight to qualify as short haul?) while other are more intricate (e.g., Under what conditions can be said that there is a weather contingency? What is a "fair" treatment? When can the fair treatment be overrun by human operators?). Provided that the system involved in satisfying the above requirement is partly automated and as argued earlier, the latter questions involve interpretation by a requirements engineer involved in the specification of the automated parts of such a system. There is seemingly no pressing issue here if the requirements engineer is also an expert in air traffic management, for this person would already know the precise answers to these questions and would know how to appropriately specify them in a requirements diagram. However, while air traffic is sufficiently safety-critical for society to warrant the existence of bodies that specialize in managing requirements acquisition, analysis, maintenance for ATM systems, in most situations the RE expertise lies not with the same people as expertise in the problem that the engineered system is expected to resolve. Empirically, the said separation of effort is observed to weigh on the success of RE projects [16]. Separation entails that the requirements engineer ought to take particular care when interpreting requirments stated by the stakeholders who are likely to be more knowledgable in the specific problem at hand. As observed in Sect. 7, it is surprisingly rare for available RE frameworks to suggest ways to raise the awareness of the requirements engineer to unclear information and the problems of interpretation of requirements. GAM addresses this issue through the concept of unclear information, introduced and discussed in Sect. 5.

The preceding discussion highlights essentially two issues:

1. It is desirable to argument and justify modeling decisions, as the notion of appropriateness of a goal diagram would lose all meaning—i.e., any diagram would be appropriate provided that no technical errors are made in modeling.
2. Information used and produced during RE can be unclear, thus increasing the probability that it will be misinterpreted by the requirements engineer and that the resulting diagram will be inappropriate.

The position adopted in the remainder is that the two problems are related: the arguments leading to specific modeling choices can be unclear *and* the information contained in a goal diagram can be unclear—both therefore require clarification. Hence the analysis of information clarity (Sect. 5). But as clarification of a piece of information can result in choosing among alternative clear forms of the given information (in case it is, e.g., ambiguous), and the apparent interest in arguing for some choices over other, understanding how to proceed to argumentation is of relevance as well.

Although it may be interesting to further claim that the clarity problem may arise in part from restrictive (or, e.g., ambiguous and incomplete) syntax and semantics of a goal model that is being employed, the present discussion does not go so far: the existing literature is followed in assuming the value of the chosen Tropos goal model for RE activities (e.g., [8, 23, 61]). Additional information, aimed at clarifying and arguing for modeling choices, is therefore given *with* (instead of *in*) diagrams to avoid changing the original model or the associated, wider RE framework.

A useful complement to a goal diagram contains arguments that justify or challenge decisions for clarifying and then modeling requirements in a particular way. Although argumentation and clarification can be informal and without a particular structure, the following benefits can be gained by using a structured method:

- Of relevance to traceability, the use of a decision process explicitly adapted for argumentation allows each element introduced in a goal diagram to be related to a set of arguments that justify or criticize the modeling decision leading to the given representation of that element.
- When new information becomes available, the change of the diagram that it may require can be easier to understand if arguments for prior decisions are explicit. Otherwise, the engineer may not be capable of understanding the relationship between the new elements added to the diagram and the existing ones, leading the engineer to overlook additional changes for ensuring the consistency of information in the diagram.
- If arguments are explicit and structured according to a few simple rules Sect. 6, the justification process can be analyzed for inconsistency and preference over arguments can be established. If arguments are further formalized, the justification process for a goal diagram can be at least partly automated.
- In addition to qualifying some information as unclear, it is possible to determine the kind of information that clarifies it. Namely, it will be illustrated that information can be unclear in many *different* ways, so that no single clarification technique always applies. Accordingly, it is useful for the requirements engineer to determine in which way the information is unclear so as to apply specialized techniques.

3 Brief overview of GAM

To address the problem outlined in Sect. 2, GAM combines three components:

- A *decision process* (Sect. 4) which highlights the key steps to take when creating a new or modifying an existing goal diagram, and gives an overall organization to the argumentation and clarification activities in relation to goal modeling.
- A set of *clarification techniques* (Sect. 5) to check the information used and produced during decision making for clarity, and clarify it in case it is judged unclear.
- An *argumentation model* (Sect. 6) which imposes restrictions on how pieces of information are composed into arguments, and how the latter combine to arrive at a justified decision.

While the decision process and the clarification techniques are essentially informal, the argumentation model is suggested in both an informal and a formal variant. The formalisms facilitate and make precise the presentation of the definitions of argument and dialectical trees. It should be noted that the formalization of arguments is not required for the remarks and suggestions made below to apply; however, avoiding a structured language (be it first-order or a propositional one) is not helpful when automated translation between goal diagrams and arguments is attempted (see, Sect. 6.1). The three following sections present and discuss respectively the decision process, the clarification techniques, and the argumentation model.

4 GAM decision process

GAM employs a decision making process to organize the activities involved in clarification and argumentation, and whose outcome is a change in the content of a goal diagram. Although systematic decision procedures are generally adapted for specific application domains, most involve a number of common steps shown in Fig. 2. A decision maker generally proceeds by setting the problem to be resolved by application of the process. The problem statement provides initial directions for the identification of alternative solutions, which are in turn evaluated, by making qualitative or quantitative arguments to support or defeat each alternative. The chosen alternative takes the form of a solution statement. A new problem can then be tackled and the decision process repeated until the stakeholders involved in decision making consider that the relevant issues have been addressed.

The GAM decision process, given in Fig. 2 is purposefully generic, for its principal purpose is to highlight the steps at which the clarification and argumentation activities are to be executed within *any* decision process suggested in the engineer's RE framework of choice. The activities and deliverables appearing in Fig. 2 are based on recomendations of design rationale research (e.g., [14, 40, 52]), which is concerned with assisting humans when reasoning about the rationale behind decisions that lead to the production of an artifact. A design rationale expresses elements of the reasoning which have been invested behind the design of the artifact [52]. The various design rationale approaches that have been suggested in the software engineering literature give a set of concepts and suggest ways in which these can be manipulated during a design

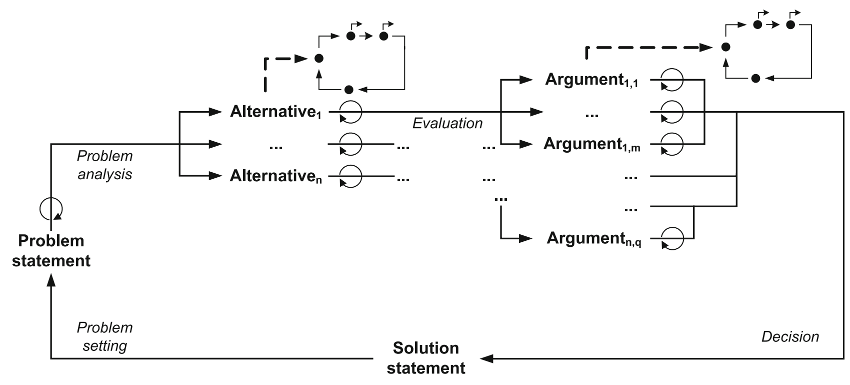


Fig. 2 Outline of the GAM decision process

activity (for an overview, see [40]). For example, the IBIS [14] approach consists of relating *issues* that need to be deliberated to *positions* that resolve issues, and *arguments*, that support or object to positions. More recently, [40] suggested the *reasoning loop model* (RLM), which integrates common characteristics of established design rationale approaches. It starts from a description of a problem which generates *goals* that characterize potential solutions. Then, *hypotheses* about potential solutions that satisfy goals are generated through problem analysis. Evaluation of alternative hypotheses leads to a *justification* of a selected alternative, which in turn leads to *deciding an action*. The result of an action is likely to lead to new goals, thus restarting the reasoning loop. It is not difficult to establish immediate links between the concepts given in Fig. 2 and the ones common in design rationale research:

- The *problem statement* designates any objective to be reached, demand to be satisfied, problem to be solved, issue to be discussed, in general anything one would like to achieve through problem resolution. It corresponds to the concepts of *issue* (in IBIS [14]), *goal* (RLM [40]), *requirement* (REMAP [47]), *decision problem* (DRL [35]), *question* or *criterion* (QOC [41]).
- The *alternatives* are potential solutions to the stated problem, and correspond to *positions* (IBIS, REMAP), *hypotheses* (RLM), *alternatives* (DRL), and *options* (QOC).
- The *argument* is a piece of information (e.g., a statement) that either provides support for, or is provided against choosing an *alternative*. It is conceptually close to *argument* (IBIS, REMAP, QOC), *justification* (RLM), and *claim* (DRL).
- The *solution statement* encompasses both the alternative chosen as the result of evaluation and the action taken to conform to the prescription given in the chosen alternative. As such, it is similar to *decision* (REMAP) and *design action* (RLM).

Technique 1. (Applying the GAM decision process) The decision process is applied as follows. Starting from an empty or already elaborate goal and/or decision diagram (the latter is an output of the decision process), stakeholders proceed by setting the problem, which consists of finding gaps between the desired and the observed in both diagrams. To describe the result of the problem setting activity, a problem statement is devised in natural language by stakeholders who consider the given diagrams' content inadequate. The *problem statement* should not be mistaken for a goal of the IS for which the requirements are being engineered: a problem statement may result in adding an IS goal to a goal diagram, but it may also lead to adding any other modeling element or changing any of the two diagrams in some other way. Stakeholders then suggest ideas and make proposals about the resolution of the stated problem. They generate a set of *alternatives* (shown as $Alternative_1,\ldots,Alternative_n$ in Fig. 2). Alternatives are evaluated by providing one or more *arguments* to support or contest each alternative. When only one alternative remains justified and all others defeated, a decision is reached, and the diagram is changed according to the information contained in the justified alternative. The completion of a cycle in the reasoning process leads to the initiation of a new loop, until stakeholders agree that no further reasoning about a diagram is required. A loop must be closed, i.e., all activities executed—otherwise, the stated problem is not considered to be treated adequately.

Technique 2. (Initiating additional decison processes) The dashed arrows in Fig. 2 indicate that additional decision process loops may be initiated from some of the specific decision deliverables. It is possible to identify new problem statements from alternatives (e.g., an alternative may highlight, be it is selected or not, the need for existing diagram elements to change in a way not anticipated in the problem statement) and arguments (e.g., some of the arguments provided for alternatives may be based on information already in the goal diagram, so that additional arguments that contest the former may point to problems in the goal diagram). There are no dashed arrows from the decision and problem statements, as any issue identified as a result of the decision statement generates a problem statement, which itself initiates a new instance of the decision process.

Technique 3. (Unstructured clarification in the decision process) The small circular arrows in Fig. 2 indicate that there may be a need for clarification of information produced in the activities of the decision process. Information that requires clarification is written in a problem statement of a new reasoning loop, in which alternatives can be, e.g., different clear forms of the unclear statement. Until Sect. 5, clarification is left to the intuition of the engineer and the participating stakeholders.

Example 1. (Applying the GAM decision process) To illustrate the application of the decision process in GAM, part of an *i** SR diagram for the meeting scheduler is constructed starting from an empty diagram. The output of the decision process, i.e., the *decision diagram*, is on the left-hand side of Fig. 3. Next to it is an incomplete SR derived from part of the information contained in the decision diagram. To relate SR diagram elements to those of the decision diagram, each SR element is annotated with the reference of the reasoning diagram element from which it is derived, and the expression from the decision diagram that is translated into the SR diagram is underlined. For example, the goal *Schedule Meeting* is marked with *PS1* indicating the decision element (here, the problem statement at the root of the reasoning diagram) that led to the introduction of the goal in the goal diagram. As a convention, arguments for an alternative are marked with *Ax* + /–, where *x* is the number of the argument in the list, and + (plus) and – (minus) symbols are used, respectively, to indicate that the argument supports or contests the alternative. Clarification is the first activity that has been realized: the *PS1* problem statement has been considered unclear, in that the meanings of user friendly and effortless, as well as the potential relationship between the two are unclear. Two alternatives were suggested and a decision has been taken to adopt the second alternative (*1.Dec*) based on arguments given for the alternative. Some of the alternatives led to additional problem statements that were in turn subjected to the decision process. Each element of the decision diagram is labeled with the name of the stakeholder that suggested it, and the time of writing. Several additional observations can be made:

- There is information in the decision diagram that is not in the goal diagram (e.g., the reasoning behind the construction of the goal diagram), and there is information in the goal diagram that is not in the decision diagram (e.g., that some expression in an argument is interpreted as a task or a goal). The two diagrams are complementary, allowing a stakeholder to discover why a goal diagram has been constructed in a particular way by reading it in conjunction with the decision diagram.
- Because the construction of the (incomplete) diagrams in Fig. 3 started from an empty sheet, the information in both of them is still unclear. For example, one could ask if the task marked 3.*Alt*2.*A*3 + is a decomposition of another goal diagram element, and if so, which one; or if the goal *Schedule Meeting* could be made more precise and how, etc. Clarification will lead to additional decision loops (as was initially the case for *PS1*). As illustrated above, the result can be increasingly precise decision and goal diagrams.
- GAM can be used to document the decision making behind the use of specific goal analysis techniques,

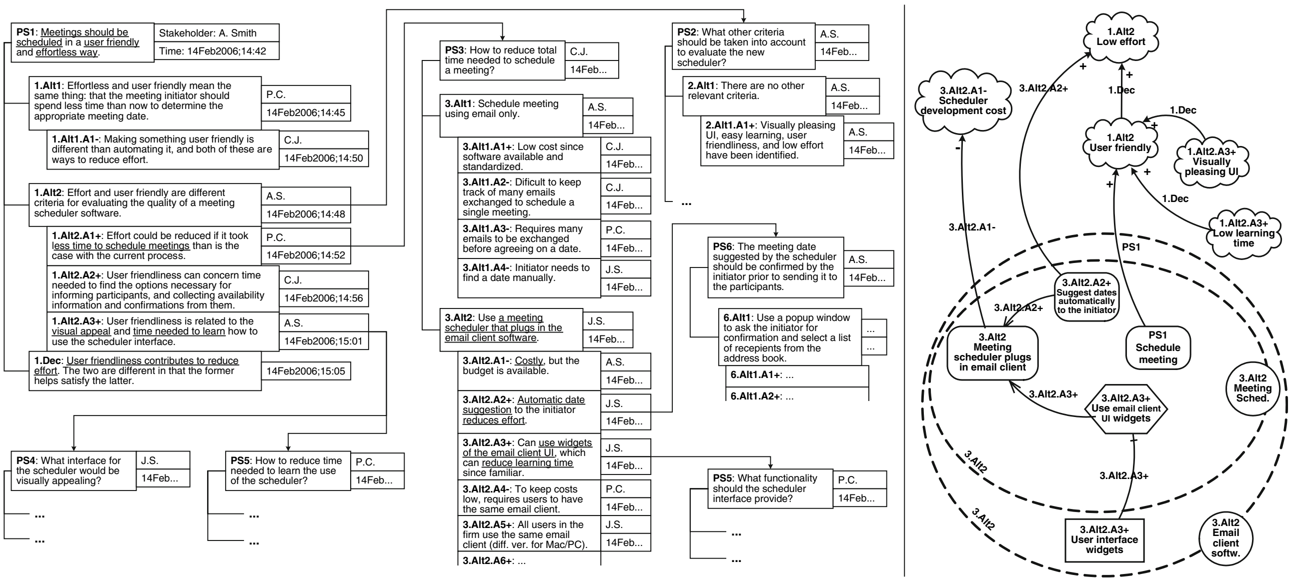

PS1: Meetings should be scheduled in a user friendly and effortless way.
Stakeholder: A. Smith
Time: 14Feb2006;14:42
1.Alt1: Effortless and user friendly mean the same thing: that the meeting initiator should spend less time than now to determine the appropriate meeting date.
P.C.
14Feb2006;14:45
1.Alt1.A1-: Making something user friendly is different than automating it, and both of these are ways to reduce effort.
C.J.
14Feb2006;14:50
1.Alt2: Effort and user friendly are different criteria for evaluating the quality of a meeting scheduler software.
A.S.
14Feb2006;14:48
1.Alt2.A1+: Effort could be reduced if it took less time to schedule meetings than is the case with the current process.
P.C.
14Feb2006;14:52
1.Alt2.A2+: User friendliness can concern time needed to find the options necessary for informing participants, and collecting availability information and confirmations from them.
C.J.
14Feb2006;14:56
1.Alt2.A3+: User friendliness is related to the visual appeal and time needed to learn how to use the scheduler interface.
A.S.
14Feb2006;15:01
1.Dec: User friendliness contributes to reduce effort. The two are different in that the former helps satisfy the latter.
14Feb2006;15:05
PS4: What interface for the scheduler would be visually appealing?
J.S.
14Feb...
...
...
PS5: How to reduce time needed to learn the use of the scheduler?
P.C.
14Feb...
...
...
PS3: How to reduce total time needed to schedule a meeting?
C.J.
14Feb...
3.Alt1: Schedule meeting using email only.
A.S.
14Feb...
3.Alt1.A1+: Low cost since software available and standardized.
C.J.
14Feb...
3.Alt1.A2-: Dificult to keep track of many emails exchanged to schedule a single meeting.
C.J.
14Feb...
3.Alt1.A3-: Requires many emails to be exchanged before agreeing on a date.
P.C.
14Feb...
3.Alt1.A4-: Initiator needs to find a date manually.
J.S.
14Feb...
3.Alt2: Use a meeting scheduler that plugs in the email client software.
J.S.
14Feb...
3.Alt2.A1-: Costly, but the budget is available.
A.S.
14Feb...
3.Alt2.A2+: Automatic date suggestion to the initiator reduces effort.
J.S.
14Feb...
3.Alt2.A3+: Can use widgets of the email client UI, which can reduce learning time since familiar.
J.S.
14Feb...
3.Alt2.A4-: To keep costs low, requires users to have the same email client.
P.C.
14Feb...
3.Alt2.A5+: All users in the firm use the same email client (diff. ver. for Mac/PC).
J.S.
14Feb...
3.Alt2.A6+: ...
PS2: What other criteria should be taken into account to evaluate the new scheduler?
A.S.
14Feb...
2.Alt1: There are no other relevant criteria.
A.S.
14Feb...
2.Alt1.A1+: Visually pleasing UI, easy learning, user friendliness, and low effort have been identified.
A.S.
14Feb...
...
PS6: The meeting date suggested by the scheduler should be confirmed by the initiator prior to sending it to the participants.
A.S.
14Feb...
6.Alt1: Use a popup window to ask the initiator for confirmation and select a list of recepients from the address book.
...
...
6.Alt1.A1+: ...
6.Alt1.A2+: ...
PS5: What functionality should the scheduler interface provide?
P.C.
14Feb...
...
...
1.Alt2 Low effort
3.Alt2.A1- Scheduler development cost
3.Alt2.A2+
+
+
1.Dec
1.Dec
1.Alt2 User friendly
+
1.Alt2.A3+ Visually pleasing UI
+
+
1.Dec
-
3.Alt2.A1-
1.Alt2.A3+ Low learning time
PS1
3.Alt2.A2+ Suggest dates automatically to the initiator
3.Alt2.A2+
3.Alt2 Meeting scheduler plugs in email client
PS1 Schedule meeting
3.Alt2 Meeting Sched.
3.Alt2.A3+
3.Alt2.A3+ Use email client UI widgets
3.Alt2
3.Alt2.A3+
3.Alt2
3.Alt2.A3+ User interface widgets
3.Alt2 Email client softw.

Fig. 3 Using the GAM decision process to build an *i** diagram for the meeting scheduler

already established in RE (such as, e.g., goal refinement and operationalization [57]). For example, the modeler can document reasons leading to, e.g., a particular refinement or decomposition of the *Schedule Meeting* goal in Fig. 3. This way, the decision process in GAM helps to fill a gap between abstract suggestions on how the goal modeling activity should be organized (e.g., elicit goals from available documentation and look further by asking why and how questions [57]) and very precise, formal techniques already available in RE methodologies (such as formal refinement, abstraction, operationalization [36, 57]), without requiring them to change to fit the design rationale approach.

The reader may question whether the GAM decision process should further incorporate concepts that tend to commonly appear in RE decision making: for instance, the REMAP [47] decision rationale approach features additional concepts such as "constraint" and "design object", as well as a number of specialized links to relate the different concepts, including, e.g., "generalizes", "specializes", "responds to", etc. It would then appear interesting to introduce the RE-specific concepts such as, "goal", "task", "actor" within the decision process model. It has, however, been observed that particularly close integration of design rationale with the specific artifacts (here, goal models) may result in the loss of the original vision of argumentative decision making [52]. Since GAM is intended to assist more extensive RE methods that already rely on specific techniques to organize the RE process as well as specific ontologies, it was important not to use an approach that requires adaptation of the established methods. Therefore, the direction of design rationale approaches that aim to be minimally prescriptive, lightweight, informal, non-intrusive on the design activity they complement, and place no restrictions on the artifact being produced are favored, so as to facilitate the practical applicability of GAM. It can also be observed that the concepts appearing in the GAM decision process purposefully leave much to interpretation, leaving emphasis on argumentation and clarification in choosing among alternative solutions. GAM is in this respect an argumentative approach to goal modeling: by involving and integrating argument and argumentation of alternatives as, respectively, a key concept and activity, the proposal follows the noted relevance of argumentation in dealing with problems that lack a precise, agreed-upon formulation or available plans of action (e.g., [14, 50]).

The construction of the example given in Fig. 3 uses neither the specific clarification techniques nor the argumentation model available in GAM. Arguments are given without verifying their clarity and without a particular structure. Moreover, no precise criteria have been applied to determine whether arguments are inconsistent and if some arguments defeat others. Clarification remained informal, grounded in the intuition of the engineer and the stakeholders while structured around the decision process, in which the unclear information is considered the problem, and alternatives its various clear forms. Clarification techniques are proposed in the following section to avoid ambiguity, overgenerality, synonymy, and vagueness of arguments and of information in the goal diagram.

5 Clarification

Even in safety-critical systems such as those used in air-traffic management, unclear information abounds, as the example below illustrates.

Example 2. (Unclear information in ATM documentation) Consider the following excerpt on stakeholders expectations about the ATM system [20]:

> "All users want total visibility on costs and charging (cost recovery) mechanisms regarding ATM system development and operation expenditures. Constant monitoring of ATM services costs and quality of service ('benchmarking') delivered to the client is needed to ensure the cost effective provision of such services" [20] (p.12).

The above seems ambiguous (does "total visibility" involve control of expenditures in addition to their transparency?), overgeneral (for it refers in a much too general way to the entities it concerns: e.g., Who are the "users"? What are the "cost and charging mechanisms"?), carries synonyms (is there a difference between "cost" and "expenditure"?), and vague (since "cost effective" relies on a gradeable adjective, admits borderline cases of application, and the Sorites paradox—see Sect. 1). Similar issues are found in the excerpt below:

> "... also applies to pre-tactical (day minus one) and tactical (same day) flight planning phases" [20] (p.17).

Where "day minus one" admits alternative and contradictory interpretations: day minus one can be measured in different ways, while "day" can mean 24 h or a working day, thus requiring disambiguation.

> "... the flexibility of ATM to cope with unforeseen short term changes in demand or partial failures whilst ensuring that the repercussions for all airspace users remain acceptable" [20] (p.20).

Above, "acceptable" is vague, as is "excessive communication" below:

"... eliminating excessive routine voice communications should apply to all flight phases in the ground/ ground and air/ground communications... The pilot wants as few frequency changes as possible..." [20] (p.17).

Stakeholders may have a number of reasons to communicate statements of requirements or give arguments that are unclear in one way or another: their knowledge of problem and solution domains may be restricted; they learn as the project unfolds, thus establishing and/or changing their preferences over alternative (parts of) problem and solution statements; for ease, they are likely to avoid a rigorous representation language, expressing their needs in natural language. It can then be reasonably assumed that the production of a specification obliges the requirements engineer to clarify the stakeholders' statements either in a passive or an active way. A passive approach would be to consider a statement sufficiently clear if there are no stakeholders that explicitly question it. This may be unhelpful because: (i) a stakeholder may not question a statement since its understanding is based on implicit domain knowledge shared with other stakeholders; (ii) there may be few motives/rewards for a stakeholder to participate actively in the RE process; (iii) a stakeholder may consider either being incompetent to question the statement, or unaffected by the part of the system that the statement concerns. In case of (i), domain knowledge of the engineer is likely to remain limited, while important requirements are also likely to be overlooked if (ii) and/or (iii). An active approach, in which the requirements engineer questions stakeholders' statements openly by using specialized clarification techniques therefore appears desirable.

Unclear information is problematic because the engineer has difficulties in giving it a unique interpretation and consequently translating it into elements of a goal diagram or using it as useful arguments in justifying modeling decisions. Although the engineer may perceive some information as unclear, to act in order to clarify it, the engineer ought to know how to detect a lack of clarity and to identify directions for the enchancement of unclear information. Moreover, as ambiguity differs from vagueness, synonymy differs from each of the latter, and so on, there can be various distinct techniques for detecting lack of clarity and subsequent clarification: i.e., clarity is a multi-facetted construct. In practical terms, in addition to perceiving a piece of information as unclear, the method must enable the engineer to determine along which facets it is unclear, and clarify accordingly.

From there on, an active clarification process is conceptualized as a successive application of a set of basic *clarification techniques*, each being a transformation of information initially considered unclear into that perceived as clear by the stakeholder(s). The aim of the requirements engineer is to move on each dimension towards a direction assumed desirable: e.g., moving from "more" to "less" ambiguity, from more to less vagueness, etc. In addition to clarification techniques, clarity checking techniques are required to detect if some information is unclear along a particular facet. It follows that a way of helping the engineer in an active approach is to provide a rich catalog of clarity facets, to define each facet for easier identification, and to suggest clarity checking and clarification techniques to be applied when a facet is identified.

The catalog of four facets—ambiguity (Sect. 5.2), overgenerality (Sect. 5.3), synonymy (Sect. 5.4), and vagueness(Sect. 5.5)—introduced herein is not meant to be complete and its extension is encouraged: it is impossible, knowing the extent of the literature on linguistic phenomena such as ambiguity, to provide a full account herein. Practicality has therefore been the focus, with discussion and careful reuse of established results in linguistics (e.g., [24, 33, 48]), philosophy (e.g., [2, 25, 55, 59]), and artificial intelligence (e.g., [4, 26]). The proposed facet classification, along with the clarity checking and clarification techniques make no attempt at settling debates on the essence of concepts such as vagueness or ambiguity. Instead, the proposal draws on various literatures, taking as given some of the existing philosophical and AI results while introducing techniques specialized for the problem at hand.

5.1 Introduction to clarity checking and clarification

As argued above, the aim of the requirements engineer is to know whether a piece of information is unclear along a particular facet, and if so, to know how to clarify it. Therefore, each facet is associated with one or more domain-independent clarity checking and clarification techniques. Both are in essence informal, for no solid conceptual and formal foundations exist for the various concepts (e.g., vagueness, ambiguity) that underlie the facets. In the present paper at least, and in absence of fully automated requirements acquisition frameworks, the informal treatment of the present issue is deemed sufficient. The problem of providing a formal acquisition language which would allow explicit representation and reasoning about all of the four facets is not to be underestimated.

Because lack of clarity can appear in any fragment of information in a goal diagram or argument or elsewhere (i.e., information used as a source for goal modeling and argumentation), clarification need be applicable to any part of thereof. The basic approach consists of a labeling technique outlined below, involving the checking of information for lack of clarity, and subsequent application of clarification techniques associated to ambiguity, overgenerality, synonymy, and vagueness. All information obtained through clarification is written in a thesaurus.

Technique 4. (Clarification by labeling) Clarification techniques associated to the cited four facets share a common approach to the labeling of unclear information, albeit employing different techniques for the identification of elements to mark and their subsequent clarification. Labeling proceeds by the following steps:

1. Choose a word or expression and test it for lack of clarity along a particular clarity facet. Choice is not arbitrary, but guided by stakeholders' or engineer's questions about what a word or expression is intended to mean.
2. If unclear along one or more clarity facet (see, Sect. 5.2–5.5), place brackets around it and label it accordingly (below, assume that the fragment of interest in a sentence is of the form: "... word(s)..."):
   (a) If ambiguous, then "$\ldots{}^{A_i}[\text{word(s)}]^{A_i}\ldots$", where A is to refer to the ambiguity facet, and i is a number to ensure a unique reference to the given label.
   (b) If overgeneral, then "$\ldots{}^{G_i}[\text{word(s)}]^{G_i}\ldots$ ".
   (c) If synonymous, then "$\ldots{}^{S_i}[\text{word(s)}]^{S_i}\ldots$ " with the same label (i.e., S_i) applied to all words synonymous with "word(s)".
   (d) If vague, then "$\ldots{}^{V_i}[\text{word(s)}]^{V_i}\ldots$".
3. Clarify the element in brackets it by applying a technique suggested for the given facet.
4. Transfer the result of the clarification into the thesaurus, and enforce the result of clarification over other artifacts so that the agreed meaning is maintained accross the project.

Example 3. (Clarification by labeling) Returning to the excerpt used in Example 2, labeling leads to the following:

> "All ${}^{G_1}[\text{users}]^{G_1}$ want ${}^{A_1}[\text{total visibility}]^{A_1}$ on ${}^{G_2}[{}^{S_1}[\text{costs}]^{S_1}$ and charging (cost recovery) mechanisms$]^{G_2}$ regarding ATM system development and operation ${}^{S_1}[\text{expenditures}]^{S_1}$. Constant monitoring of ATM services ${}^{S_1}[\text{costs}]^{S_1}$ and quality of service ('benchmarking') delivered to the ${}^{G_3}[\text{client}]^{G_3}$ is needed to ensure the ${}^{V_1}[\text{cost effective}]^{V_1}$ provision of such services" [20] (p.12).

Labels—A for ambiguity, G overgenerality, S synonymy, and V for vagueness—are thus introduced for each of the four facets, applied to unclear information according to results of checks for clarity, and are then ready for analysis using clarification techniques outlined in the following subsections.

Dealing with these four facets requires the understanding of linguistic phenomena that ambiguity, overgenerality, synonymy, and vagueness refer to, as discussed below.

5.2 Ambiguity

An encyclopedic entry [2] suggests that a word or an expression (i.e., several related words) is ambiguous if it has more than one meaning. Examples include words, such as "light" (which can mean not heavy or not dark), or phrases, such as "user's agendas provide their availability" (Whose availability is provided?). While people seem capable in many cases to intuitively detect the occurrence of ambiguity, a useful criterion for doing so seems elusive (e.g., [24, 49]). The aim at present is to suggest a practical criterion for ambiguity that is acceptable in many cases. In addition, it is required that ambiguity is distinguished from the other three facets. For instance, separating ambiguity from overgenerality can be difficult if a word designating a class is considered ambiguous if the designated class is divisible onto subclasses. If this is accepted as a necessary condition for ambiguity, and knowing that a subclass contains instances of the subdivided class (see, Sect. 5.3), all ambiguous words would be considered as general. However, this is not appropriate: Hospers [28] has argued that a word is ambiguous neither because (i) the class of objects it refers to can be broken down into smaller classes or subclasses, nor (ii) when it may have many instances of use. Moreover, it is now accepted that ambiguity does not require generality [24].

Ambiguity is multiplicity of meaning of an expression, regardless of whether it originates from polysemy [48] of individual words or from multiplicity of structural analyses [24]. Polysemy occurs when a word, taken out of context, admits multiple meanings [48]—e.g., "run" as a verb has 29 distinct meanings and 125 sub-meanings according to the Webster's dictionary. Context of use tends to resolve problems of polysemy in communication [48], in that a word which is taken with other words in an expression loses most of its alternative senses. It is, however, not necessary to have a polysemous word in an expression for it to be ambiguous, as the expression "in airports, the police cannot shoot suspects with guns" illustrates. In this latter case, it is not polysemy that generates ambiguity, but the fact that the proposition admits different structural analyses [24]—each structural analysis leads roughly to one alternative reading of this proposition. The reader should note that matter is more elaborate than the above indicates: for instance, negation with "not" in English is often ambiguous for it leaves open whether it is the truth or the assertability of a proposition that is negated.

It is therefore difficult to propose a unique and general clarity check for ambiguity. For instance, it is useful to check if there is a state of affairs in which the ambiguous expression can both be affirmed and denied. For instance, in a state in which "armed police cannot shoot unarmed suspects on airport grounds" and "armed police can shoot armed suspects on airport grounds" both hold, the expression "in airports, the police cannot shoot suspects with guns" can both be affirmed true and false: in the last expression, the reader cannot know who carries guns (and is thus armed)—the police or the suspects, or both. A true reading of the given expression is, e.g., "in airports, the armed police cannot shoot unarmed suspects", while a false one is "in airports, the police cannot shoot armed suspects". Tautologies and contradictions cannot, however, be addressed by the given clarity check. Moreover, this check does not point to the source of ambiguity—finding it requires additional knowledge about the language being used, and thus varies accross languages (e.g., [24]).

Although imperfect, structural analysis and word polisemy can be first used to detect potential for ambiguity. Once detected, the above clarity check is applied to verify if there is a state in which some alternative readings can be affirmed and others denied.

Definition 1. An expression e is **ambiguous** if there are at least two of its alternative readings e_j and e_k such that one can be affirmed and the other denied within the body of knowledge (denoted $\mathcal{A}$, and referred to in the remainder as *domain knowledge*) contained in the project artifacts (i.e., goal diagram, arguments, thesaurus, stakeholders' requirements documentation) in which the expression is employed: $ambiguous(e) \not\models \bot$ iff:

1. There is a non-empty set E which contains alternative readings of e.
2. There are two readings e_j and e_k in E such that one gives rise to inconsistency given $\mathcal{A}$, while the other does not: $\exists e_j, e_k \in E$ s.t. $(\mathcal{A}, e_i) \models \bot$ and $(\mathcal{A}, e_j) \not\models \bot$.[1]

Technique 5. (Brute force ambiguity check) Identify as many alternative readings as feasible, and search for information relevant to the given body of knowledge which when combined with the readings is consistent with some but inconsistent with other. While it may seem cumbersome, this form of clarity checking is often feasible, for many alternative readings can be intuitively eliminated, whereas the remaining few can be subjected to scrutiny to relevant stakeholders through informal discussion.

[1] No restrictions are placed on the way domain knowledge is represented—both informal and formal representations of stakeholders' knowledge about the domain are allowed in $\mathcal{A}$.

Technique 6. (Ambiguity resolution in the thesaurus) The ambiguous element is labeled and introduced in the thesaurus with a list of alternative readings elicited by the engineer. Resolving ambiguity amounts to choosing one of the available readings and enforcing it throughout other fragments of the requirements specification through a thesaurus $\mathcal{D}$, where the the ambiguous word is carried over and accompanied with its chosen reading.

Example 4. Consider the following excerpt from a stakeholder expectation about the ATM:

> "... also applies to the pre-tactical (^{A_i}[day minus one]^{A_i}... " [20] (p.17).

Speaking of duration in terms of days engenders ambiguity by polysemy, as different and contradictory interpretations are available for "day" in "day minus one"[2]—among others: 24 h before takeoff, and working day before takeoff. Because this seems to be a case of ambiguity from polysemy, the Technique 5 above applies. The identified alternative readings are written down in the thesaurus:

(Ambiguity, A_i) **Day minus one:**	1. 24h before takeoff; ●
	2. One working day before takeoff;

The ambiguous expression "day minus one" has been labeled according to Technique 4. The expression, along with the alternative readings and the decision (i.e., the choice of a reading—decorated with "●") is transferred to the thesaurus. Choosing randomly one of the interpretations is inappropriate, for the probable impact an inadequate choice would have—e.g., scheduling problems, which are bound to result in lack of efficiency in airport operations.

Technique 7. (Structural analysis ambiguity check) In English, knowing that a "noun phrase can have complementary propositional phrases" and a "verb phrase can contain just a verb and propositional phrase" [24] allows showing that "in airports, the police cannot shoot suspects with guns" admits two structural analyses: (1) in airports, the police [[cannot shoot]$^{\text{verb}}$[suspects [with guns]$^{\text{prop. phrase}}$]$^{\text{noun phrase}}$]$^{\text{verb phrase}}$; and (2) in airports, the police [cannot shoot [suspects [with guns]$^{\text{prop. phrase}}$]$^{\text{noun phrase}}$]$^{\text{verb phrase}}$.

Example 5. For the meeting scheduler, the following is ambiguous:

> "^{A_j}[User's agendas provide their availability.]^{A_j} "

[2] Similar can be said for "day" in "same day" but the example focuses on "day minus one" only.

as the following two readings can be identified using structural analysis:

(Ambiguity, A_j) **User's agendas provide their availability:**
1. Agendas that belong to the users provide the availability of the users; •
2. Agendas that belong to the users provide information on whether they (the agendas) are available for fulfilling a request;

5.3 Overgenerality

Overgenerality is introduced to characterize a relationship present between expressions such as "provide awareness of traffic in the areas involving aircraft movement" and "provide awareness of traffic in the areas involving helicopter movement", where the first is general with regards to the second if helicopter is understood to designate a subclass of aircraft.

Definition 2. A word g is **overgeneral** if domain knowledge $\mathcal{A}$ contains an expression about an instance or specialization of g, say p, whereby the expression with p contradicts with an expression containing g. As a convention, let $F(a) \in \mathcal{A}$ in the definition indicate that an expression $F(a)$ containing a word of interest a is in domain knowledge $\mathcal{A}$. $F(a) \in \mathcal{A}$, that is, there is an affirmable expression $F(a)$ in domain knowledge, where a is a word that occurs once in $F(a)$ and can be replaced with the particular word p or the general word g. Then, by convention, $F(a)/g$ denotes the affirmable expression obtained by replacing the word a with the word g. We have $overgeneral(g) \not\models \perp$ iff:

1. The expressions $F(a)/g$ and $F(a)/p$ are inconsistent given the domain knowledge in $\mathcal{A} : \mathcal{A}, (F(a)/g), (F(a)/p) \models \perp$.
2. *general*(p,g) holds, i.e., it must be verified that g is a general of p. A word g is general with respect to a word p iff for an affirmable expression $F(a)$, the expression $\neg(F(a)/g) \wedge (F(a)/p)$ is a contradiction.

The given clarity check for generality was suggested in [43] and defended against alternative checks in [24]—it is taken as appropriate for the purpose herein. In the ATM case study for instance, g can stand for the word "aircraft" and p for "helicopter", see, Example 6.

Technique 8. (Criteria-based resolution of overgenerality) Define criteria for distinguishing entities that fall in, from those that do not fall in the set designated by the expression $F(a)$ in which substitution for g and p gives inconsistency. The entry in the thesaurus explicates the criteria used to discriminate what particulars can be used as substitutes for a in $F(a)$.

Example 6. If the general piece of information is "aircraft landing is allowed on strip Z", if helicopter is considered a particular of aircraft, and if the particular appears in an expression, e.g., "helicopter landing is not permitted on strip Z", then the two expressions are inconsistent, making the expression "aircraft landing is allowed on strip Z" overgeneral. A criterion that may be used to clarify over the overgenerality facet in this case would express that helicopters do not land on airstrips.

Example 7. The following fragment expresses stakeholders expectations about aircraft movement:

"The main goal of an A-SMGCS is to provide all 'control authorities'... with positive situation awareness of traffic evolving in the areas used for ^{G_i}[aircraft movement]^{G_i}, the runway strip..." [20], (p.22).

It is expressed elsewhere in the documentation that aircraft movement inside hangars and repair areas is not to be communicated to all control authorities. Applying the Technique 8 results in adding the expression labeled above as an entry in the thesaurus and providing a criterion which accounts for the exception:

(Overgenerality, G_i) **Aircraft movement:** excludes aircraft movement inside hangars and repair areas.

Example 8. Simplistic expressions of expectations tend to contain overgeneral information, e.g.,:

"^{G_j}[Any user of the meeting scheduler can initiate a meeting]^{G_j}. "

(Overgenerality, G_j) **Any user of the meeting scheduler can initiate a meeting:** Full-time employees that have at least the managerial status in the research department can initiate meetings.

5.4 Synonymy

Synonymy treats the use of common terminology in an inconsistent manner—it highlights the use of syntactically different terms, which are given the same semantics.

Definition 3. Words w_1 and w_2 are synonymous within $\mathcal{A}$ if they are can be used interchangeably in $\mathcal{A}$: *synonymous*(w_1, w_2) holds iff:

1. Both words appear in $\mathcal{A} : w_1, w_2 \in \mathcal{A}$;
2. w_1 appears in expression $F_1(w_1)$ and w_2 appears in expression $F_2(w_2)$, and interchanging the two words within the expressions maintains the meaning of the new expressions equivalent with original ones. That is,

$F_1(w_1)/w_2$ has the same meaning as $F_1(w_1)$, and $F_2(w_2)/w_1$ has the same meaning as $F_2(w_2)$.

Technique 9. (Synonymy by interchangeability check) Intuitively, words used within similar expressions are candidates for synonymy. Clarity checking for synonymy proceeds by replacing a word in an expression with another word within one or more expressions in which the first appears. For instance, if $F(a)$ and $G(b)$ are two expressions appearing in domain knowledge, and the requirements engineer believes that both expressions refer to the same properties or behaviors of the IS, then if the engineer understands $F(a)/b$ (i.e., the expression $F(a)$ in which each occurrence of the word a is replaced with the word b) in the same way as $F(a)$, and $G(b)/a$ in the same way as $G(b)$, then a and b are synonymous for the given expressions.

Example 9. Consider the following potential synonymy:

> "For ${}^{S_i}[\text{incident}]^{S_i}$ and ${}^{S_i}[\text{accident}]^{S_i}$ investigation purposes the ATM network must provide mechanisms to record and make available any data..." [20] (p.23).

By applying Technique 9 over the above and other requirements fragments, it has been established that the two expressions "incident investigation" and "accident investigation" are not synonymous, leading to:

(Synonymy, S_i) **Accident investigation $\neq$ Incident investigation:** distinct processes, which is executed depends on the gravity of the event.

Example 10. The excerpt below illustrates how synonymy can be involved in inconsistency—resolving it expectedly resolves the inconsistency:

> "${}^{S_j}[\text{A meeting participant}]^{S_j}$ is anyone who has confirmed attendance at the meeting . . . ${}^{S_j}[\text{A participant}]^{S_j}$ is any the person who attends the meeting, except the initiator."

(Synonymy, S_j) **Meeting participant = participant:** A participant is any the person who has confirmed attendance and attends the meeting, except the meeting initiator.

Technique 10. (Resolving synonymy by equivalence or criteria in case synonymy is absent) In case two words are checked for synonymy, resulting in the conclusion that they are nor synonymous, the criteria for their distinction are provided in the corresponding thesaurus entry to avoid any further questioning. Otherwise, if words are considered synonymous, their equivalence is written down in the thesaurus as well.

5.5 Vagueness

According to [3], "any grammatical element whose contribution to truth conditions requires perception, categorization, or judgement of gradient contingent facts—including tense, aspect, and plurality suffers from an incurable susceptibility to vague uncertainty". Consider the expression:

> "All users want total visibility on costs and charging (cost recovery) mechanisms regarding ATM system development and operation expenditures. Constant monitoring of ATM services costs and quality of service ('benchmarking') delivered to the client is needed to ensure the cost effective provision of such services" [20] (p.12).

The above is vague in that what exactly means to count as "cost effective" is indeterminate. According to linguists (e.g., [33] and related), such expressions have three distinguishing characteristics:

1. *Truth conditional variability:* The truth valuation of the expression depends on the context in which it is used.
2. *Existence of borderline cases:* Whatever the context of use, there will generally be sets of entities to which "is cost effective" either clearly applies or not, but there will also be entities for which it is difficult or impossible to determine whether the said predicate applies or not.
3. *The Sorites Paradox:* When employed within a particular form of argument, the predicate will give rise to the Sorites paradox: e.g., if a $2 million project is cost effective, and any project that costs 1 cent less than a cost effective one is still cost effective, then any project is cost effective. In such a line of reasoning, the argument appears valid, premises true, yet the conclusion false.

Admitting vagueness in a requirements specification leaves room for questioning the degree to which the goals are satisfied *after* the corresponding software has been built. Leaving vague expectations thus leaves room for misunderstanding. An immediate consequence thereof is the difficulty with translating expectations into development or evaluation decisions, that is, difficulty to operationalize the requirements specification. This is precisely because calling a part of the software, e.g., "highly usable" will depend on the stakeholder, or, more importantly, will depend on what the stakeholder knows about usability and about the system in question: it will depend on the conditions in which the evaluation is to be made. The more the engineer and the stakeholders agree on the content of this set of information, the more likely that vagueness will be reduced.

The locus of vagueness in many vague expressions, just as the one above, is the presence of a predicate headed by a *gradeable adjective*, (above: "effective"). Such predicate designates a property of having a degree of cost that is at least as great as some standard of comparison of cost, that itself is not part of the meaning of "effective" but is determined by the context in which the said adjective is used. From there on, truth assignment can change as the standard changes (and as context changes).

This matter is, however, more intricate, as setting a standard of comparison seems to eliminate borderline cases altogether (and subsequently the Sorites paradox). While attractive, this seems removed from reality: assuming e.g., that \$2.5 million is a mean cost for similar projects (and, say, the standard for comparison), then some stakeholders may still refuse to accept that the given project of \$2 million is cost effective, for they have witnessed similar projects in which cost was significantly lower (i.e., the variance in the sample from which the mean was computed can be considered high). A solution to this is suggested in [25], where for a borderline case to be described truthfully with the given vague predicate, it is necessary for it not to exceed the standard without exceeding it by a *significant* amount—in practice, two very similar cases along the scale of measurement associated to the vague predicate will be taken same (i.e., will carry the same truth valuation) if the cost of discriminating between them outweighs the benefits of doing so. They will count as the same for the given purposes [25]. Unfortunately, it seems that how much significant it is, is itself vague—the only realistic solution then remains seeking stakeholders' agreement on the standard and its enforcement throughout a chosen context.

These established positions on gradable adjectives (e.g., [25, 33][3]) already provide relevant practical indications for the problem at hand.

Definition 4. An adjective *e* is gradeable and can be assumed giving rise to **vagueness** within the expression in which it appears, if the following conditions are met (it is said then that *vague*(*e*) holds):

1. First assumption on the gradeable adjective: the adjective maps its arguments onto abstract representations of measurement, or *degrees*.
2. Second assumption on the gradeable adjective: the set of degrees totally ordered with respect to some *dimension* (e.g., cost, size, etc.) constitute a *scale*.

   The first and second condition together give an ontology for gradeable adjectives which provides indications on what adjectives to consider when clarity checking for vagueness.
3. Presence of context-dependent standard of comparison: the adjective itself does not entail a standard for comparison, so that such a standard varies with context.
4. Presence of borderline cases: it should be possible to identify borderline cases of application of the given adjective.
5. Truth conditional variability: the truth of the expression in which the adjective appears should vary with the change of standard which is accepted to distinguish when the adjective applies from when it does not.
6. The Sorites Paradox: the predicate generated by the adjective can be used in lines of reasoning that follow the one taken in the Sorites paradox (see above).

Technique 11. (Vagueness check) See the conditions for the presence of vagueness in Definition 4.

Technique 12. (Resolving vagueness from gradeable adjectives) For gradeable adjectives that generate vagueness, the desired solution is to specify the standard of comparison, and to treat borderline cases individually. Within the specification process, the identified source of vagueness is placed between brackets $[...]^V$, transferred to the thesaurus, and associated with a sharp criterion.

Example 11. Returning to the example cite above, consider the following fragment:

> "... quality of service ('benchmarking') delivered to the client is needed to ensure the ^{V_i}[cost effective]^{V_i} provision of such services." [20] (p.12).

"(Cost) effective" is a gradeable adjective—applying Technique 12 results in the definition of a criterion or standard of comparison, as follows:

(Vagueness, V_i) **Cost effective:** Not cost effective if above the budget permitted at the outset of the development project.

If feasible, the above thesaurus entry can be extended to include explicit indications on the scale to employ when measuring the degree of cost effectiveness—e.g., if a global indicator of stakeholder satisfaction is available, it can be combined to the difference between actual project cost and the budget.

Example 12. In the following statement about the meeting scheduler system, the word "few" generates vagueness:

> "The scheduler should send a reminder ^{V_j}[a few days before the meeting date]^{V_j}. "

[3] The reader is reminded that the present work is not one focused on linguistics, so that no specific references will be given beyond overview and extensive discussions from the aforementioned field. For instance, no minority positions are mentioned herein. For details, the reader will refer to the works cited within the given references.

(Vagueness, V_j) **A few days before the meeting date:** The meeting initiator should have a choice of automatically informing the participants 1, 2, ..., 7 days before the meeting date.

Following the above discussion, it appears that many softgoals identified early on in the Tropos RE process can be qualified as vague. In this respect, softgoals can be dealt with at the very early stages using GAM in a relatively novel manner, without harming the applicability of established RE approaches that employ the softgoal concept. For instance, instead of leaving a softgoal "1.Alt2 User friendly" of the meeting scheduler as is and then establishing contribution links between this and other goal diagram elements, taking this softgoal as vague in GAM would require the application of clarification techniques, which would likely result in more detail as to what is actually meant and agreed upon to be the meaning of the given softgoal. The information obtained by clarification and written in the thesaurus can subsequently serve as the source for finding ways to operationalize the clarified softgoal.

5.6 Clarification and argumentation

As suggested above, the given techniques can be applied to clarify information used and produced when using the GAM decision process, being therefore useful both to clarify the information to be placed in a goal diagram and the information employed when justifying the modeling choices. In both cases, arguments can be compared over specificity (as usual in the argumentation modeling literature—see, Sect. 6), but also according to their relative clarity: the following section introduces preferences orderings over arguments based on the presence or absence of clarity therein. For instance, an argument can be rejected in favor of another one if the former is unclear and the latter clear along a clarity facet. Clarification thus provides additional informal criteria for the comparison and discrimination between alternative arguments.

6 Argumentation and justification

With clarification, the information used and produced when applying the GAM decision process can be checked for clarity and clarified to avoid misunderstanding. Returning to the evaluation activity of the decision process, argumentation and justification are introduced to provide a structured approach to the discussion, documentation, and revision of rationale behind modeling decisions—in the evaluation step, techniques presented below serve as an integrative approach to qualitative selection among alternative modeling choices.

Argumentation modeling literature [11] in the artificial intelligence field focuses on formalizing commonsense reasoning in the aim of automation. An *argumentation model* [56] is a static representation of an *argumentation process*, which can be seen as a search for arguments, where an argument consists of a set of rules chained to reach a conclusion. Each rule can be rebutted by another rule based on new information. To formalize such defeasible reasoning, elaborate syntax and semantics have been developed (e.g., [6, 11, 53]) commonly involving a logic to formally represent the argumentation process and reason about argument interaction.

A structured argumentation system (i.e., a model and processes employing the model) is introduced in GAM for a number of reasons: (i) it is needed for a rigorous justification process in the *Evaluation* step of the GAM decision process; (ii) content of the decision diagram can be more closely related to the content of the goal diagram than was illustrated in the previous section; and, (iii) the structured approach is a basis for automating some of the argumentation-specific analyses in GAM.

Example 13. To further illustrate the need for an argumentation system in GAM, consider the example in Fig. 3. In the problem statement 3 (*PS3*), assume that a decision has been made to adopt alternative *3.Alt2* and not *3.Alt1*. Such a decision can be considered as justified only because the clearly unsupportive argument in *3.Alt2* (i.e., *3.Alt2.A4-*) is rebutted by *3.Alt2.A5+*, while the other negative argument *3.Alt2.A1-* is written in such way that its second part provides support against its first part. The requirements engineer could overlook the ambiguity in *3.Alt2.A1-* and conclude that there are no arguments that interfere with *3.Alt2*, accepting it then as justified. In contrast, because there are no arguments that interfere with the negative ones, which in turn interfere with *3.Alt1* (i.e., *3.Alt1.A2-* to *3.Alt1.A4-*), choosing *3.Alt1* is not justified. In presence of an argumentation system, it is required that the arguments for each alternative be more precise in terms of their structure, and their relationships explicit for more rigor in justification.

To arrive at a structured argumentation system, the concept of argument is first defined below, followed by a set of argument relationships, and the justification process.

Definition 5. An *argument* is defined recursively as follows:

1. Any information of the form $P \therefore c$ is an argument, where c is called "conclusion" and is a speech act of any type (i.e., assertive, directive, commissive, expressive, or declarative [51]), $\therefore$ is a conclusion indicator

(read it "therefore"), and P is a set of assertive propositions suggested to support the conclusion (such a proposition is called "premise").

2. If for A and B such that $A = P_A \therefore c_A$ and $B = P_B \therefore c_B$, $P_A \subseteq P_B$ then A is a subargument of B.
3. An argument cannot have its conclusion as a premise for its conclusion: if $A = P_A \therefore c_A$ and $c_A \in P_A$ then A is not an argument.
4. There can be no inconsistent propositions in P.
5. Nothing is an argument unless it obeys the rules above.

The suggested definition is common in philosophy (see, e.g., [27] for a discussion) and AI (for an overview, see [11]). It has the desirable properties of avoiding inconsistent information as support for a conclusion, allows complex arguments, in which a premise can be a conclusion of another argument, and bans cyclical argumentation.

The above definition can be formalized as follows. Assuming a first-order language $\mathcal{L}$ defined as usual, let $\mathcal{K}$ represent a consistent set of formulae (i.e., $\mathcal{K} \nvdash \perp$), each being a piece of information about the universe of discourse, and let $\mathcal{K} \equiv \mathcal{K}_N \cup \mathcal{K}_C$. Members of the set $\mathcal{K}_N$, called *necessary knowledge*, represent facts about the universe of discourse and are taken to be formulae which contain variables, thus stating relationships. Necessary knowledge is taken at face value (i.e., is assumed unquestionable). The set $\mathcal{K}_C$, called *contingent knowledge* are information that can be put in question or argued for. It is then said that the knowledge an agent a (here, a stakeholder of the RE project) can use in argumentation is given by the pair (K_a, Δ_a), where K_a is a consistent subset of $\mathcal{K}$ (i.e., $K \subset \mathcal{K}$ and $K \nvdash \perp$), and Δ_a is a finite set of *defeasible rules*. A defeasible rule has the form $\alpha \rightsquigarrow \beta$. The relation $\rightsquigarrow$ between formulae α and β is understood to express that "reasons to believe in the antecedent α provide reasons to believe in the consequent β". In short, $\alpha \rightsquigarrow \beta$ reads "α is reason for β".

Definition 6. Let A a set of agents (e.g., stakeholders), $K \equiv \bigcup_{a \in A} K_a$, and $\Delta \equiv \bigcup_{a \in A} \Delta_a$ Given (K_a, Δ_a) and $P \subset \Delta_a^{\downarrow}$, where $\Delta_a^{\downarrow}$ is a set of formulae from Δ_a instantiated over constants of the formal language (i.e., variables appearing in these formulae are replaced with specific values), P is an **argument** for $c \in \mathcal{K}_C$, denoted $\langle P, c \rangle_K$, if and only if:

1. $K \cup P \mid\!\sim c$ (K and P derive c)
2. $K \cup P \nvdash \perp$ (K and P are consistent)
3. $\nexists P' \subset P, K \cup P' \mid\!\sim c$ (P is minimal for K)

where "$\mid\!\sim$" is called the *defeasible consequence* [53] and is defined as follows. Define $\Phi = \{\phi_1,..., \phi_n\}$ such that for any $\phi_i \in \Phi$, $\phi_i \in K \cup \Delta^{\downarrow}$. A formula ϕ is a defeasible consequence of Φ (i.e., $\Phi \mid\!\sim \phi$) if and only if there exists a sequence $B_1,..., B_m$ such that $\phi = B_m$, and, for each $B_i \in \{B_1,..., B_m\}$, either B_i is an axiom of $\mathcal{L}$, or B_i is in Φ, or B_i is a direct consequence of the preceding members of the sequence using modus ponens or instantiation of a universally quantified sentence.

The formal argument definition given above is well-understood and established in the AI literature. Looking at alternative approaches (for extensive overviews, see, [11, 46]), the principal difference is in the choice of "$\mid\!\sim$", that is, of the assumptions made on how the premises formally relate to the conclusion. For instance, [6] situate their discussion within classical propositional logic and take classical logical consequence "$\vdash$" instead. Choosing one over other formal approaches to argumentation (here, [53]) does not impose significant restrictions on the present discussion—the intuition behind the definitions proposed here are well known and do not differ within the AI argumentation literature.[4] Observe that the formal and the intuitive definition of argument fit—for an argument $P \therefore c$, the formal definition writes $\langle P, c \rangle_K$[5] so that the $\therefore$ is replaced with binary relationships (being either "$\rightarrow$" in case of necessary knowledge or the "$\rightsquigarrow$" for defeasible knowledge) between premises in P; also, definitions of subargument do not differ, minimality ensures that there is no circular argumentation, and consistency of premises is ensured in both definitions.

While an argument can be constructed by combining explicitly expressed knowledge (e.g., from a knowledge base, as is often the case in AI argumentation modeling literature), the aim with GAM is to start from a conclusion and build arguments that support it from the knowledge that stakeholders provide, and that can be related to the conclusion. An important observation about the above definitions is that an argument $P \therefore c$ or, equivalently, $\langle P, c \rangle$ can be seen as a tree in which the root is the conclusion c and the leaves are members of P. A subargument is then a subtree in the tree of the argument. The evaluation activity of the GAM decision process amounts to combining two techniques: the argumentation of each alternative, which consists of constructing an *argument tree* AT for each alternative, whereby the root of the tree is

[4] It is, however, significant to note that the choice of derivation *is* critical if the aim is to build arguments automatically from a knowledge base: in case, e.g., arguments for requirements are to be obtained automatically from a knowledge base, the derived arguments will differ depending on the chosen derivation. It is obvious that following the above suggestions would require a knowledge base which contains defeasible rules.

[5] In the remainder, the subscript K will be omitted, since no knowledge base other than K (which is taken here to contain any knowledge that the stakeholders can provide) will be used.

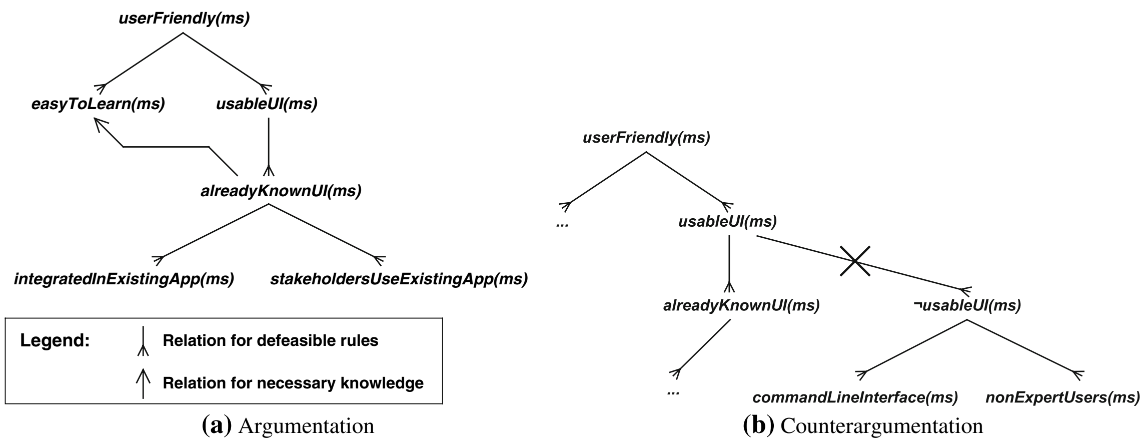


Fig. 4 In **a** an argument tree with premises supporting the conclusion userFriendlyUI(ms). In **b** argument concluding ¬ usableUI(ms) counterargues the argument concluding userFriendlyUI(ms), at usableUI(ms)

the alternative, and the nodes are premises proposed to support the given alternative; and the comparison of alternatives, which is the identification of arguments that counterargue the arguments which appear in argument trees.

Technique 13. (Argumentation of an alternative) Supporting an alternative *Alt* consists of recursively defining an argument tree AT_{Alt} as follows:

1. Define *Alt* as the root of the tree AT_{Alt} and set $c = Alt$.
2. Let $\langle P, c\rangle$. Identify $p_1,..., p_n$ such that $\{p_1,..., p_n\} = P$, $P \subseteq K \cup \Delta^{\downarrow}$.
3. Define a node for each premise $p_i \in P$ and define an edge from that node to c. Draw the edge "$\longrightarrow$" if $p \in K$, or as shown in Fig. 4(a) in case $p \in \Delta^{\downarrow}$ (i.e., notice that the symbol for defeasible rule relation "$\rightsquigarrow$" is changed in the argumentation tree in Fig. 4).

Each premise can be in turn argued for, that is, steps 1–3 above can be repeated for each premise $p_i \in P$ by letting p_i be a conclusion of a new argument, e.g., $\langle P_i, p_i\rangle$. The tree built for $\langle P_i, p_i\rangle$ is a subtree of the tree built for $\langle P, c\rangle$. By building subtrees subsequently for premises of $\langle P_i, p_i\rangle$ and this for some or all $p_i \in P$, the argumentation can proceed until the stakeholders and/or the requirements engineer judge that the argument tree for *Alt* has been constructed to a satisfactory extent.

Example 14. The following example illustrates how the definitions above are used to build an argument in GAM. Consider the suggestion (see, Fig. 3): "*1.Alt2.A3+: (The Meeting Scheduler is) user friendly if well integrated in the user interface of the email client*." For a meeting scheduler *ms*, a formula *c* for which we wish to argue or interfere with can be written: *userFriendly(ms)*. Stakeholders may then suggest a set of defeasible rules:

$$\begin{aligned}\Delta = \{&easyToLearn(x) \wedge usableUI(x) \rightsquigarrow userFriendly(x),\\ &standardizedUI(x) \vee alreadyKnownUI(x)\\ &squigarrowusableUI(x), integratedInExistingApp(x)\\ &\wedge stakeholdersUseExistingApp(x) \rightsquigarrow alreadyKnownUI(x)\}\end{aligned}$$

And the following necessary knowledge, where "→" is the standard implication operator:

$$K = \{alreadyKnownUI(x) \rightarrow easyToLearn(x)\}$$

An argument tree that supports *userFriendly(ms)* can be constructed using the given knowledge and defeasible rules, and represented using a tree-like structure shown in Fig. 4a. The root of the tree is the sentence for which the argument structure provides support. The defeasible and necessary knowledge have been instantiated over the meeting scheduler *ms*. The argument in Fig. 4 can be written:[6]

$$\begin{aligned}&\langle\{integratedInExistingApp(ms) \rightsquigarrow alreadyKnownUI(ms),\\ &stakeholdersUseExistingApp(ms) \rightsquigarrow alreadyKnownUI(ms),\\ &alreadyKnownUI(ms) \rightsquigarrow usableUI(ms), usableUI(ms)\\ &\rightsquigarrow userFriendly(ms), easyToLearn(ms) \rightsquigarrow userFriendly(ms)\},\\ &userFriendly(ms)\rangle\end{aligned}$$

where

$$\begin{aligned}&\langle\{integratedInExistingApp(ms) \rightsquigarrow alreadyKnownUI(ms),\\ &stakeholdersUseExistingApp(ms) \rightsquigarrow alreadyKnownUI(ms)\},\\ &alreadyKnownUI(ms)\rangle\end{aligned}$$

[6] Note that the argument is extracted from the tree by ensuring minimality, according to Definition 6, so that some branches of the tree need not be maintained.

is one of the subarguments of the argument shown in Fig. 4a.

The argument tree (or, simply argument) shown in Fig. 4a contains only information that supports the conclusion that the meeting scheduler is user friendly. Albeit being useful as it is, a particular interest in argumentation is in confronting arguments and rejecting some conclusion in favor of other. It is therefore necessary to define relationships between arguments.

Definition 7. Two arguments $\langle P_1, c_1\rangle$ and $\langle P_2, c_2\rangle$ **disagree**, denoted by:

$$\langle P_1, c_1\rangle \bowtie_{\mathcal{K}} \langle P_2, c_2\rangle$$

if and only if $\mathcal{K} \cup \{c_1, c_2\} \vdash \bot$.

Definition 8. Instead of seeking contradiction of conclusions, the **counterargument** relation looks at the incompatibility of an argument's conclusion with the conclusion of a subargument of another argument. An argument $\langle P_1, c_1\rangle$ *counterargues at c* the argument $\langle P_2, c_2\rangle$, denoted by:

$$\langle P_1, c_1\rangle \not\rightsquigarrow^{c} \langle P_2, c_2\rangle$$

if and only if there is a subargument $\langle P, c\rangle$ of $\langle P_2, c_2\rangle$ such that $\langle P_1, c_1\rangle \bowtie_{\mathcal{K}} \langle P, c\rangle$.

Example 15. The following argument counterargues at *usableUI(ms)* the argument in Fig. 4a:

$$\langle\{commandLineInterface(ms) \wedge nonExpertUsers(ms) \rightsquigarrow \neg usableUI(ms)\}, \neg usableUI(ms)\rangle$$

Counterargumentation is represented by a crossed line, as in Fig. 4b, directed from the root (i.e., conclusion) of the argument that counterargues to the node which is countered.

Definition 9. In case two arguments are such that one counterargues the other, it is necessary to determine which of the two is to be maintained. An argument $\langle P_1,\ c_1\rangle$ **defeats at c** an argument $\langle P_2,\ c_2\rangle$, denoted by:

$$\langle P_1, c_1\rangle \gg^{c} \langle P_2, c_2\rangle$$

if and only if:

1. $\langle P_1, c_1\rangle \not\rightsquigarrow^{c} \langle P_2, c_2\rangle$; that is, $\langle P_1, c_1\rangle$ counterargues $\langle P_2, c_2\rangle$ at c;
2. there is a subargument $\langle P, c\rangle$ of $\langle P_2, c_2\rangle$ such that $\langle P_1, c_1\rangle \succ^{spec} \langle P, c\rangle$; i.e., $\langle P_1,\ c_1\rangle$ is more *specific* than $\langle P,\ c\rangle$.

Definition 10. The specificity relation "$\succ^{spec}$" is an order relation over arguments, defined so that arguments containing more information, i.e., which are more specific, are preferred over other. An argument $\langle P_1, c_1\rangle$ is *strictly more specific than* $\langle P_2, c_2\rangle$, denoted by:

$$\langle P_1, c_1\rangle \succ^{spec} \langle P_2, c_2\rangle$$

if and only if:

1. $\forall e \in \mathcal{K}_C$ such that $\mathcal{K}_N \cup \{e\} \cup P_1 \mathrel{|\!\sim} c_1$ and $\mathcal{K}_N \cup \{e\} \not\mathrel{|\!\sim} c_1$, also $\mathcal{K}_N \cup \{e\} \cup P_2 \mathrel{|\!\sim} c_2$, and
2. $\exists e \in \mathcal{K}_C$ such that:
   (a) $\mathcal{K}_N \cup \{e\} \cup P_2 \mathrel{|\!\sim} c_2$
   (b) $\mathcal{K}_N \cup \{e\} \cup P_1 \not\mathrel{|\!\sim} c_1$
   (c) $\mathcal{K}_N \cup \{e\} \nvdash c_2$

Example 16. The argument:

$$\langle\{easyToLearn(ms) \wedge usableUI(ms) \rightsquigarrow userFriendly(ms)\}, userFriendly(ms)\rangle$$

is more specific than:

$$\langle\{easyToLearn(ms) \rightsquigarrow \neg userFriendly(ms)\}, \neg userFriendly(ms)\rangle$$

because although *easyToLearn(ms)* alone is taken as sufficient for arguing the conclusion $\neg userFriendly(ms)$, it cannot by itself be sufficient to argue *userFriendly(ms)*. If *easyToLearn(ms)* ^ *usableUI(ms)* alone is used to argue *userFriendly(ms)*, the argument that supports ¬ *userFriendly(ms)* can also be concluded. In other words, the former argument contains at least the same premises as the latter—the additional information makes it more specific.

Technique 14. (Comparison of arguments over clarity) Arguments employed in the examples above are constructed of premises and conclusions made of primitive propositions, which in practice are references to parts of the specification or initial documentation, or are replaced with sentences of natural language which express in a rich manner the given premise or conclusion. There is therefore no particular guarantee that the content of premises or conclusions carries clear content. Consequently, premises and conclusions can be clarified over ambiguity, overgenerality, synonymy, and vaguenes using the techniques presented earlier. In addition then to comparing arguments using the ordering relation defined by specificity, the following order relations can be defined:

- Arguments containing non-ambiguous information are preferred to those containing ambiguous information:

$$\langle P_1, c_1\rangle \succ^{A} \langle P_2, c_2\rangle$$

iff $\exists e \in P_2 \cup \{c_2\}$ s.t. $ambiguous(e) \nvdash \perp$ and $\forall e' \in P_1 \cup \{c_1\}, ambiguous(e') \vdash \perp$.

- Arguments containing non-overgeneral information are preferred to those containing overgeneral information:

$$\langle P_1, c_1 \rangle \succ^G \langle P_2, c_2 \rangle$$

iff $\exists e \in P_2 \cup \{c_2\}$ s.t. $overgeneral(e) \nvdash \perp$ and $\forall e' \in P_1 \cup \{c_1\}, overgeneral(e') \vdash \perp$.

- Arguments containing non-synonymous information are preferred to those containing synonymous information:

$$\langle P_1, c_1 \rangle \succ^S \langle P_2, c_2 \rangle$$

iff $\exists e_i, e_j \in P_2 \cup \{c_2\}$ s.t. $synonymous(e_i, e_j) \nvdash \perp$ and $\forall e_k, e_l \in P_1 \cup \{c_1\}, synonymous(e_k, e_l) \vdash \perp$.

- Arguments containing non-vague information are preferred to those containing vague information:

$$\langle P_1, c_1 \rangle \succ^V \langle P_2, c_2 \rangle$$

iff $\exists e \in P_2 \cup \{c_2\}$ s.t. $vague(e) \nvdash \perp$ and $\forall e' \in P_1 \cup \{c_1\}$, $vague(e') \vdash \perp$.

Technique 15. (Justification) The *justification process* consists of recursively defining and labeling a *dialectical tree* $\mathcal{T}\langle P, c \rangle$ as follows:

1. A single node containing the argument $\langle P, c \rangle$ with no defeaters is by itself a dialectical tree for $\langle P, c \rangle$. This node is also the root of the tree.
2. Suppose that $\langle P_1, c_1 \rangle, \ldots, \langle P_n, c_n \rangle$ each defeats $\langle P, c \rangle$. Then the dialectical tree $\mathcal{T}\langle P, c \rangle$ for $\langle P, c \rangle$ is built by placing $\langle P, c \rangle$ at the root of the tree and by making this node the parent node of roots of dialectical trees rooted respectively in $\langle P_1, c_1 \rangle, \ldots, \langle P_n, c_n \rangle$. One way of finding arguments $\langle P_1, c_1 \rangle, \ldots, \langle P_n, c_n \rangle$ that defeat $\langle P, c \rangle$ is to look for arguments that support the negation of a premise in P or the negation of the conclusion c.
3. When the tree has been constructed to a satisfactory extent by recursive application of steps (1) and (2) above, label the leaves of the tree *undefeated* (U). For any inner node, label it undefeated if and only if every child of that node is a *defeated* (D) node. An inner node will be a defeated node if and only if it has at least one U node as a child. Do step (4) below after the entire dialectical tree is labeled.
4. $\langle P, c \rangle$ is a *justification* (or, P justifies c) if and only if the node $\langle P, c \rangle$ is labelled U.

Technique 16. (Justification in the GAM decision process) The GAM decision process is combined with the justification process in the following way. First, when *problem analysis* leads to the identification of a set of alternatives, a dialectical tree is built for each alternative, whereby the root of the dialectical tree is the argument tree of the given alternative. The *evaluation* decision activity consists of labeling each dialectical tree, and accepting the one justified alternative, this being the alternative whose dialectical tree is such that the root node is labeled *undefeated U. At most one alternative must remain justified*—in case more than one alternative appear justified, additional arguments need to be added as leaf nodes to each alternative's dialectical tree until only one alternative remains justified. The *decision* decision activity amounts to choosing the justified alternative, and acting upon it in terms of changing the associated goal diagram.

Example 17. For illustration, Fig. 5 shows the dialectical tree for *3.Alt1 "Schedule meeting using email only"*. To simplify the representation, the dialectical tree is built with formulae. The justification process showed that the alternative is unjustified and therefore cannot be accepted. Argument trees for *scheduleByEmail(mi)* and *complicatedScheduling(ms)* are shown. The argument $\langle \{\ldots\}, complicatedScheduling(ms) \rangle$ defeats $\langle \{\ldots\}, scheduleByEmail(mi) \rangle$ at $\neg$ *userFriendly(ms)* with to the subargument:

$$\langle \{manualScheduling(ms) \wedge manyEmailsToRead(mi)$$
$$\rightsquigarrow \neg userFriendly(ms)\}, \neg userFriendly(ms) \rangle of \langle \{\ldots\},$$
$$complicatedScheduling(ms) \rangle.$$

Observe that the given dialectical tree suggests alternative 3.Alt1 should to be rejected as it does not satisfy user friendliness. It appears, however, that the same alternative satisfies low implementation cost, for there are no arguments that defeat *lowImplementationCost(ms)*. Rejecting the alternative assumes that user friendliness is preferred by the decision makers to low implementation cost—preferences orderings ought to be specified outside GAM, i.e., within the goal diagram constructed using the RE framework that GAM complements. Integrating preferences in justification is a matter that requires a separate treatment (for an introduction, see, e.g., [29]).

6.1 Analyzing arguments in existing goal diagrams

The definitions given earlier can be applied to analyze goal diagrams constructed without using the GAM decision process, thus allowing GAM to be applied in a relatively straightforward manner to study the arguments implicit behind modeling decisions of which only the solution statements are available to the requirements engineer.

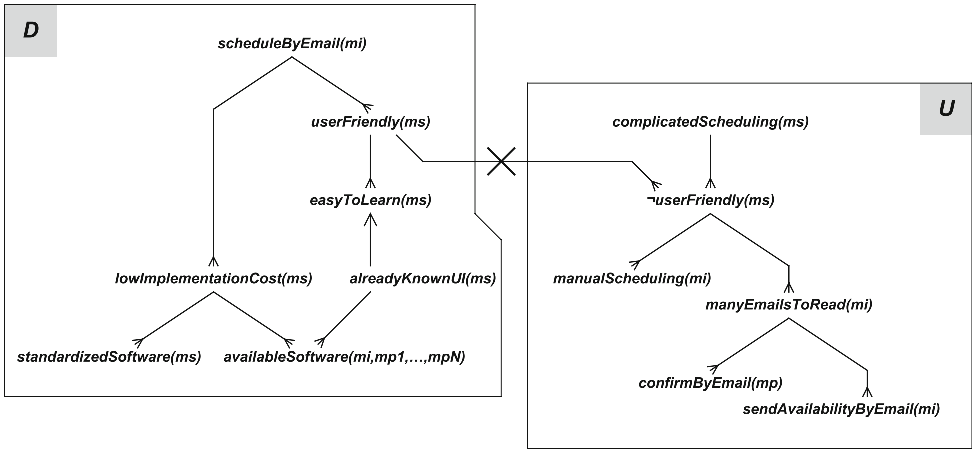


Fig. 5 A dialectical tree for 3.Alt1 "Schedule meeting using email only"

This is realized by defining a mapping for translation between a goal diagram and a dialectical tree. In the present paper, only the rules for translating between a Tropos goal diagram (TGD) and a dialectical tree (DT) are suggested. Similar rules can be defined when using GAM with other goal-oriented RE frameworks. The DT can thus be submitted to Tropos-specific analyses, provided it is translated into a TGD; and a TGD can be subjected to the analysis of the justification behind the modeling decisions that led to it.

Mapping is defined using an intermediary language and a set of keywords illustrated with the meeting scheduler case study examples in Fig. 9. The translation rules are bidirectional. When translating from a TGD to a DT (moving from left to right in Fig. 9) the intermediary language is used to write down th structure of the TGD. The obtained TGD specification is then translated into formulae labeled with a restricted set of keywords. The rules used for translating between the intermediary language and the DT are referred to as the "GAM/Tropos translation rules" and are formalized as follows. The operator "$\overset{\approx \text{label}}{\longleftrightarrow}$" denotes translation rules, with *label* indicating the name of the rule being used in the translation (*f* stands for "formula")—definitions are given in Figs. 6 and 7.

Observe that the relationships in the goal model are interpreted as defeasible rules. Intuitively, this seems adequate: e.g., if a task is decomposed into a resource in a TGD, the need to provide a resource can be interpreted to exist because that resource is used when executing the given task; in a negative contribution, the link in the TGD is directed from an element that contributes negatively to the target softgoal, whereas in a DT, a negative contribution exists between a defeater argument and the argument it defeats. The dependency relationship is interpreted as a chain of two defeasible rules (i.e., $\alpha \rightsquigarrow \beta \wedge \beta \rightsquigarrow \chi$). In the first, β marks the dependency between actors and α is the dependum of the dependency (because, e.g., in a goal dependency, the dependum goal is the reason for the dependency to exist: the depender cannot achieve the goal

goal(Name)	$\overset{\approx \text{goal}}{\longleftrightarrow}$	*achieve(f)* | *maintain(f)* | *achieve&maintain(f)* | *avoid(f)*
task(Name)	$\overset{\approx \text{task}}{\longleftrightarrow}$	*do(f)*
resource(Name)	$\overset{\approx \text{resource}}{\longleftrightarrow}$	*use(var)* | *provide(var)*
softgoal(Name)	$\overset{\approx \text{softgoal}}{\longleftrightarrow}$	*optimize(f)*
actor(Name)	$\overset{\approx \text{actor}}{\longleftrightarrow}$	*var*
cgmodel	$\overset{def}{\equiv}$	*goal(Name)* | *task(Name)* | *resource(Name)* | *softgoal(Name)* | *actor(Name)*
cgmtype	$\overset{def}{\equiv}$	*goal* | *task* | *resource* | *softgoal*
ckeyword	$\overset{def}{\equiv}$	*achieve(f)* | *maintain(f)* | *achieve&maintain(f)* | *avoid(f)* | *do(f)* | *use(var)* | *provide(var)* | *optimize(f)*

Fig. 6 GAM/Tropos translation rules

without the dependee). In addition, α can contain one or more *ckeyword* to indicate why the depender alone is unable to obtain the dependum (this is required if a Tropos SR is being translated, whereas it is often unknown in a Tropos SD). In $\beta \rightsquigarrow \chi$, χ expresses goals / tasks/... softgoals that the dependee is expected to, respectively achieve /... / optimize in order to assist the depender in obtaining the dependum (χ is often unknown in a Tropos SD, whereas it is available in a Tropos SR). The use of the tranisition rule for dependencies is illustrated in Fig. 8.

To translate a DT to a TGD, the formulae appearing in the DT are transformed into *labeled* formulae. Labels are used to derive a goal diagram element from a formula (i.e., a label maps to one or more concepts in the ontology of the goal model). For formulae that are to be translated into goals in a TGD, the Tropos goal taxonomy [23] is employed, giving four labels: *achieve(f)*, *maintain(f)*, *achieve&maintain(f)*, and *avoid(f)*. For formulae that will result in resources, the label *provide(f)* is used when a resource is being provided by an actor, whereas the label *use(f)* is applied when the resource is to be used by an actor. Figs. 6, 7 and 9 give other labels along with their corresponding TGD representation.

Example 18. The arguments in Fig. 10 have been obtained by applying the GAM/Tropos translation rules to the Tropos

Fig. 7 Continued from Fig. 6: GAM/Tropos translation rules

$$
\begin{aligned}
&\textit{contribution[+](cgmodel}_1\textit{, cgmodel}_2\textit{)} \xleftrightarrow{\approx \text{ contribute[+]}} && (\textit{cgmodel}_1 \xleftrightarrow{\approx \text{ cgmtype}} \textit{ckeyword}_1\\
&&& \wedge \textit{cgmodel}_2 \xleftrightarrow{\approx \text{ cgmtype}} \textit{ckeyword}_2\\
&&& \wedge \textit{ckeyword}_2 \rightsquigarrow \textit{ckeyword}_1)\\
&\textit{contribution[}-\textit{](cgmodel}_1\textit{, cgmodel}_2\textit{)} \xleftrightarrow{\approx \text{ contribute[-]}} && (\textit{cgmodel}_1 \xleftrightarrow{\approx \text{ cgmtype}} \textit{ckeyword}_1\\
&&& \wedge \textit{cgmodel}_2 \xleftrightarrow{\approx \text{ cgmtype}} \textit{ckeyword}_2\\
&&& \wedge \exists \langle P_1, c_1\rangle, \langle P_2, c_2\rangle \text{ such that } c_1 \text{ is the formula in } \textit{ckeyword}_1\\
&&& \text{and } c_2 \text{ is the formula in } \textit{ckeyword}_2\\
&&& \text{and } \langle P_2, c_2\rangle \gg^c \langle P_1, c_1\rangle)\\
&\textit{task-decomposition(task(Name), cgmodel)} \xleftrightarrow{\approx \text{ task-decomp}} && (\textit{cgmodel} \xleftrightarrow{\approx \text{ cgmtype}} \textit{ckeyword} \wedge \textit{task(Name)} \xleftrightarrow{\approx \text{ task}} \textit{do(f)}\\
&&& \wedge \textit{do(f)} \rightsquigarrow \textit{ckeyword})\\
&\textit{means-ends(goal(Name), cgmodel)} \xleftrightarrow{\approx \text{ means-ends}} && (\textit{cgmodel} \xleftrightarrow{\approx \text{ cgmtype}} \textit{ckeyword}\\
&&& \wedge(\textit{goal(Name)} \xleftrightarrow{\approx \text{ goal}} \textit{achieve(f)} \mid \ldots \mid \textit{avoid(f)})\\
&&& \wedge(\textit{achieve(f)} \mid \ldots \mid \textit{avoid(f)} \rightsquigarrow \textit{ckeyword}))\\
&\textit{dependency(mel}_1\textit{, cgmodel, mel}_2\textit{)} \xleftrightarrow{\approx \text{ dependency}} && (\textit{cgmodel} \xleftrightarrow{\approx \text{ cgmtype}} \textit{ckeyword}\\
&&& \wedge(\forall i = 1, 2, \textit{mel}_i = (\textit{cgmodel}_{i,1}, \ldots, \textit{cgmodel}_{i,r})\\
&&& \wedge \forall 1 \le k \le r, r > 0, \textit{cgmodel}_{i,k} \xleftrightarrow{\approx \text{ cgmtype}} \textit{ckeyword}_{i,k}\\
&&& \text{with } \textit{cgmodel}_{i,1} \xleftrightarrow{\approx \text{ actor}} \textit{var}_i\\
&&& \wedge \forall i, \textit{dep}_i = \bigwedge_m \textit{ckeyword}_{i,m}, 2 \le m \le r)\\
&&& (\textit{ckeyword} \wedge \textit{dep}_1) \rightsquigarrow \textit{depend(var}_1\textit{, ckeyword, var}_2\textit{)}\\
&&& \wedge \textit{depend(var}_1\textit{, ckeyword, var}_2\textit{)} \rightsquigarrow \textit{dep}_2)
\end{aligned}
$$

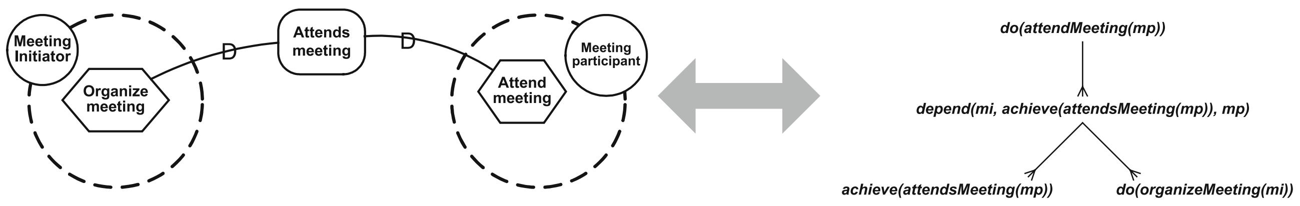


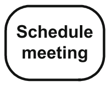
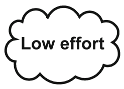
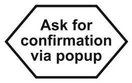
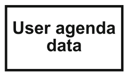
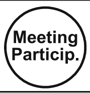
Fig. 8 Example illustrating the use of the GAM/Tropos transition rule for dependencies

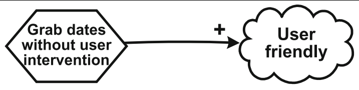
SR in Fig. 1. Using the obtained dialectical tree (shown in its developed form, that is, with detail of its component arguments, in Fig. 10), the stakeholders can, e.g., question modeling choices by providing new arguments that defeat existing ones and lead to changes in the tree. For example, it can be observed that there is cyclical argumentation in two cases: *do(agreeToDate(mp))* $\rightsquigarrow$ *depend(mp, provide(proposedDate(ms)), ms)*, and *do(scheduleMeeting(ms))* $\rightsquigarrow$ *depend(ms, do(enterDateRange(mi)), mi)*. As this is undersirable, the stakeholders may consider adding defeasible rules so that either the antecedent or the consequent is defeated or replaced. The same applies to arguments that some stakeholders may consider inappropriate—in response, they could add defeating arguments in order to request a change in the goal diagram.

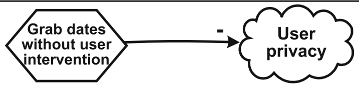
While the discussion and the translation rules presented in this subsection remain Tropos-specific, the intention is not to provide within this paper a library of translation rules between GAM and various RE frameworks' goal models. Instead, the above illustrates how mappings can be defined and used, and therefore constitute exemplified guidelines for requirements engineers aiming to use GAM with their favorite framework. Moreover, while the translation rules suggested in Fig. 6 seem to fit intuition, we cannot suggest that the given rules are definite and cannot be adjusted as the requirements engineer deems appropriate.

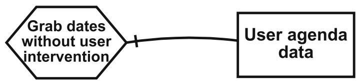
7 Related work

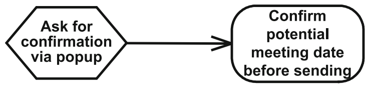
The need for justifications of modeling choices has not been overlooked in various RE methods that employ goal models. In particular, high-level goals in a diagram can be understood as reasons for representing lower-level goals (i.e., the need for the latter is justified by the presence of the former) and any other element in a goal diagram. Informal and formal AND/OR refinement and decomposition techniques, widespread in RE (for an overview, see [57]), can therefore be seen as incorporating argumentation and justification, in that sub-goals could be understood as arguments supporting parent goals. A refinement alternative would then be justified if there are no conflicts

Element in the Tropos goal diagram	*Intermediary language*	*Labeled well-formed formula in a dialectical tree*
	*goal(Schedule meeting)*	*achieve(schedule_meeting(ms))*
	*softgoal(Low effort)*	*optimize(schedule_meeting(ms))*
	*task(Ask for confirmation via popup)*	*do(ask_for_confirmation(ms))*
	*resource(User agenda data)*	*provide(user_data_agenda(email_client))*
	*actor(Meeting Participant)*	*meeting_participant* or **ms** (a variable, not a wff)

	*contribution[+](task(Grab dates withour user intervention), softgoal(user friendly))*	*do(grab_dates_automatically(ms))*
		 optimize(user_friendly(ms))

	*contribution[-](task(Grab dates withour user intervention), softgoal(user privacy))*	*do(grab_dates_automatically(ms))* *¬optimize(user_privacy(ms))*

	*task-decomposition(task(Grab dates withour user intervention), resource(User agenda data))*	*provide(user_data_agenda(email_client))*
		 do(grab_dates_automatically(ms))

 Ask for confirmation via popup
Confirm potential meeting date before sending
	means-ends(task(Ask for confirmation via popup), goal(Confirm potential meeting date before sending))	*do(ask_confirmation_via_popup(ms))*
		 achieve(confirm_before_sending(mi))

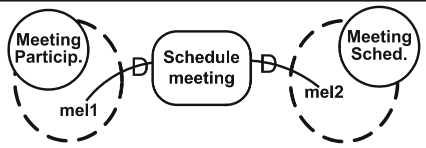
 Meeting Particip.
mel1
D
Schedule meeting
D
mel2
Meeting Sched.
	dependency(mel1, goal(schedule_meeting), mel2)
	 (see GAM/Tropos translation rules for meaning of **mel1** and **mel2**)
		(see transl. rules)
		 depend(mp, achieve(schedule_meeting(ms)), ms)
		 achieve(schedule_meeting(ms)) (see transl. rules)

Fig. 9 Exemplified GAM/Tropos translation rules

between sub-goals (i.e., it is consistent), as few obstacles as possible harm sub-goal achievement, there are no superfluous sub-goals (the refinement is minimal), and the achievement of sub-goals can be verified to lead to achieving the parent goal (if refinement is formal [18]). While this interpretation may seem satisfactory, argumentation and justification processes differ from and are complementary to refinement in several respects:

1. Regardless of clarification, argumentation anterior to modeling uses and produces richer information than is contained in a refinement recorded in a goal diagram, due to the relatively strict syntax and semantics of the underlying goal models. To leave such information unrecorded is likely to lead to issues 1–3 highlighted in Sect. 1 and not analyzing it would lead to 4–5. This has been argued and illustrated throughout the paper.
2. Historically, weak refinement links have been proposed in the NFR goal model [44], which has been inspired in part by work in informal design rationale approaches that involve argumentation (e.g., [34]). One result has been the integration of a class of argumentation goals to justify modeling choices. Any argumentation goal in NFR is of the sort “claim” (with subsorts “FormalClaim” and “InformalClaim”) and represents “evidence or counter-evidence for other goals or goal refinements” [44]: taking the example from [44], the following is an argumentation goal *InformalClaim*[“Rigorous examination is recommended for publication by employees”]

which supports this argumentation goal:

$$FormalClaim[\exists e : ValidatedBy[e, attributes(Employee)] \\ \wedge EmpStatus(e, SecI)]$$

The formalism above states that class I secretaries (*SecI*) review employee data. Notice that an argumentation goal is not by itself an argument, since there is no conclusion indicator and only one piece of information is given

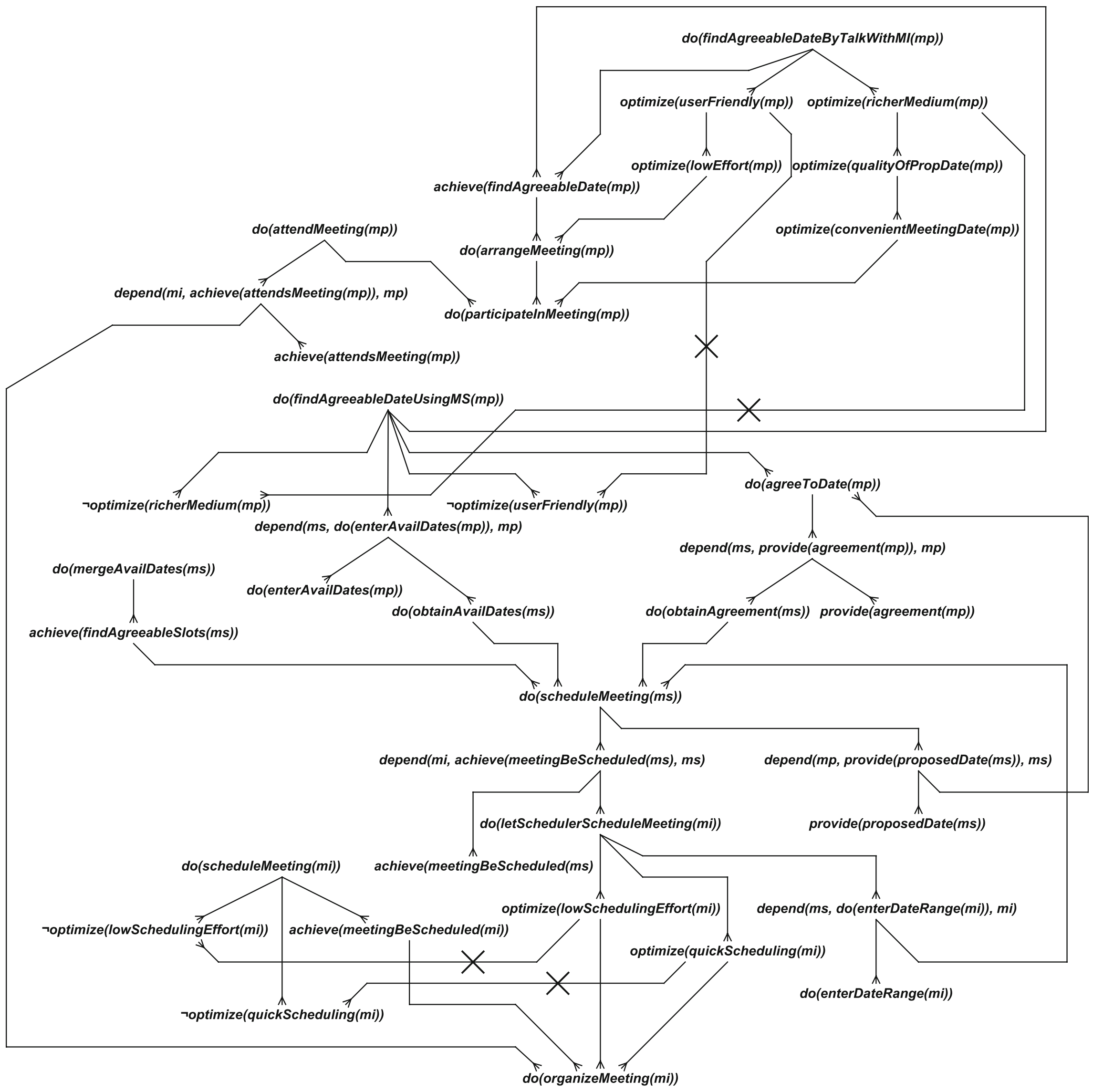

do(findAgreeableDateByTalkWithMI(mp))
optimize(userFriendly(mp))
optimize(richerMedium(mp))
optimize(lowEffort(mp))
optimize(qualityOfPropDate(mp))
achieve(findAgreeableDate(mp))
do(attendMeeting(mp))
do(arrangeMeeting(mp))
optimize(convenientMeetingDate(mp))
depend(mi, achieve(attendsMeeting(mp)), mp)
do(participateInMeeting(mp))
achieve(attendsMeeting(mp))
do(findAgreeableDateUsingMS(mp))
do(agreeToDate(mp))
¬optimize(richerMedium(mp))
¬optimize(userFriendly(mp))
depend(ms, do(enterAvailDates(mp)), mp)
depend(ms, provide(agreement(mp)), mp)
do(mergeAvailDates(ms))
do(enterAvailDates(mp))
do(obtainAvailDates(ms))
do(obtainAgreement(ms))
provide(agreement(mp))
achieve(findAgreeableSlots(ms))
do(scheduleMeeting(ms))
depend(mi, achieve(meetingBeScheduled(ms), ms)
depend(mp, provide(proposedDate(ms)), ms)
do(letSchedulerScheduleMeeting(mi))
provide(proposedDate(ms))
do(scheduleMeeting(mi))
achieve(meetingBeScheduled(ms)
optimize(lowSchedulingEffort(mi))
depend(ms, do(enterDateRange(mi)), mi)
¬optimize(lowSchedulingEffort(mi))
achieve(meetingBeScheduled(mi))
optimize(quickScheduling(mi))
do(enterDateRange(mi))
¬optimize(quickScheduling(mi))
do(organizeMeeting(mi))

Fig. 10 Translation of the Tropos SR diagram in Fig. 1 using GAM/Tropos translation rules

(i.e., one does not know if the content of the argumentation goal is a premise or a conclusion). In GAM, the above two argumentation goals are written as the following argument:

$$r\langle\{[\text{``}Rigorousexaminationisrecommended\\ forpublicationbyemployees''] \rightsquigarrow [\exists e :\\ rValidatedBy[e, attributes(Employee)]\\ \wedge EmpStatus(e, SecI)]\}, [\exists e :\\ ValidatedBy[e, attributes(Employee)]\\ \wedge EmpStatus(e, SecI)]\rangle$$

Argumentation goals produced using NFR can thus be reused in GAM. As a rigorous argumentation and justification process employing argumentation goals has not been proposed, GAM can be usefully combined with NFR to provide a rigorous argumentation and justification process. Another difficulty with weak refinements may be that semantics of weak refinement links (called contribution links) between goals remain, according to [37] “too vague for deep, accurate understanding of the [softgoal] model”, leading to recent critiques in [37] on their meaning and subsequent applicability when rigorous qualitative analysis is required.

3. Formal goal refinement has clear, but limited semantics: It remains difficult to compare alternative refinements in a rigorous qualitative way (thorough quantitative comparison is possible using probabilistic measures of goal satisfaction [37], although it is applicable to already clear requirements). It has been suggested to use a temporal logic [36] to compare

alternative refinements for e.g., minimality, conflict, and obstacles. Such techniques are RE method-specific and applicable when goals are already clear, leading [57] to argue that systematic approaches for alternative comparison are still not available.

A different approach, explored in this paper, is to allow qualitative and quantitative information by constructing and analyzing arguments. One novelty of GAM is thus to combine a design rationale approach to organize commonsense reasoning with an argumentation model to document the arguments leading to a modeling decision and allow structured justification to take place, while allowing informal and formal use with any goal model. As arguments accept both qualitative and quantitative information, the importance of quantitative evidence can be acknowledged, along with qualitative, subjective, and defeasible information.

Regarding design rationale and argumentation modeling literature, GAM adopts and adapts existing approaches, adjusting them for combined application in RE activities. In this respect, an important characteristic of this work is its integrative aim and its appropriateness to RE. As argued above, the GAM decision process is not a novelty in itself, and maps nicely to existing design rationale approaches; however, it is adapted to the purpose to which it has been put in the present paper.

Apart from the contribution of the integrative approach to justification in relation to goal models, another contribution is the identification of the clarity facets. While a considerable amount of work on, e.g., vagueness and ambiguity exists in various fields (as noted in Sect. 1), its interpretation in the context of RE has not been proposed, and an extensive treatment has not been realized. Although the issues raised in Sect. 1 are pressing for RE, in that the reader will undoubtedly agree that initial requirements tend to be, among others, ambiguous and vague, the research in RE comparable to the one here is limited. Ambiguities arising from the presence of coordination conjunctions *and*, *or*, and *and/or* in natural language requirements have been studied [10]. Text appearing in instances of modeling primitives in goal models normally does not contain coordination conjunctions since these are avoided with goal modeling constructs such as AND/OR refinement or decomposition (see, e.g., [8, 17, 61]). The approach suggested in [10] can be combined with GAM: while GAM focuses on goal diagram content, the said approach can be applied before goal modeling, on information represented in textual documentation from which the engineer is expected to derive goal diagram elements. Ambiguity of natural language requirements has been discussed and techniques suggested for detecting ambiguity without discussing clarification in [30]. Their approach does not consider ambiguity detectable by structural analysis. Others [5] have concentrated on ambiguity that may arise from the use of plural in natural language requirements. Fuzzy logic has been suggested for dealing with vague requirements (e.g., [38] among others—a more elaborate discussion is given in. However, Sect. 1 clearly illustrates that vagueness is merely one among many facets. Somewhat close to our effort here is [45], where three dimensions along which any RE project or framework can be described, are identified. It is suggested there that the overall aim in any RE project is to move from informal to formal representations of requirements (along the *representation* dimension), from individual to shared views (the *agreement* dimension), and from an opaque understanding of the system to its complete specification (the *specification* dimension). It may be interesting in such a framework to perceive clarity facets as refining the three dimensions with a number of dimensions, and in particular, enriching the specification dimension by disaggregating it onto many dimensions that we called clarity facets.

8 Discussion

In real world RE projects, GAM can be applied to guide the construction, questioning, and critique of (fragments of) a goal diagram in three ways:

1. When only the GAM decision process (Sect. 4) is put to use, clarification and argumentation are unstructured and left to intuition of the requirements engineer and of the stakeholders. Although the discussions of clarification (Sect. 5) and argumentation (Sect. 6) illustrate that neither of these is trivial, using the GAM decision process alone helps in organizing the goal modeling activity. In addition to being grounded in established results in design rationale research, the decision process is intuitive: adding a new fragment or revising an existing part of the goal diagram is normally due to particular reasons (described in the problem statement); there are often alternative ways of changing the diagram to solve the stated problem (the alternatives); each alternative has its merits and drawbacks (i.e., arguments for and against each alternative), whereby merits and drawbacks need to be compared to select one alternative over others (the evaluation activity); finally, a decision is reached (decision statement) and the goal diagram is modified accordingly. As highlighted at the outset of the paper, this or a very similar approach is usually implicitly performed by the engineer. It is when the system to engineer is of non-trivial complexity and when stakeholders participate actively in modeling that it becomes useful to have a structured

and shared approach to goal modeling. Having a sequence of clearly identifiable steps allows each participant to know how others take part in the modeling activity and how to intervene to question others' modeling decisions.

Applying a rationale approach to structure the design process (regardless of whether the rationale is recorded or not) is not trivial. Fitting the thinking involved in the design activity to a framework can be perceived as unnatural and thus involves further effort from the participants [14, 52]. In GAM, this issue is tempered by using a small set of imposed concepts and activities, thus facilitating the learning of the decision process.

Whether the information produced in the various activities of the GAM decision process is recorded is a project-specific choice. Recording gives rise to the following difficulties:

(a) As already observed in design rationale research (e.g., [40]), recording rationale requires additional effort from the engineer and the stakeholders participating in the design activity.
(b) It is noted in literature on design rationale that the artifacts containing records of design rationale tend to become rapidly complex and thus difficult to use.
(c) Allowing stakeholders to modify records of design rationale requires training.

Issue (1a) above is tempered in GAM by using a small set of intuitively acceptable concepts and techniques, as in the reasoning loop model [40] on which the GAM decision process draws. In addition, the visual syntax of the decision diagram remains flexible so that various available tools can be used (e.g., gIBIS [14], Euclid [54], *CM*/1 [13], Belvedere [9], Compendium [15], Hermes [31]). Annotations of the decision and goal diagrams also allow the use of simpler tools, such as common word processors to record the decision process. When applying the GAM decision process to guide and record the details of the rationale invested in building a goal diagram, elaborate decision diagrams (see, issue (1b) above) are obtained even for goal diagrams of limited size (see, e.g., Fig. 3). Again, annotations provide a simple means to relate fragments of decision and goal diagrams to focus questioning and revision on such fragments only. Regarding issue (1c), training is necessary both if meetings are organized to collaboratively apply GAM and if distributed acquisition of information for the decision process is allowed. Distributed acquisition supported by simple tools was sufficient for the case studies: for instance, a web log (i.e., blog) platform (in which each post and comment is annotated as in the decision and goal diagrams) is not to underestimate for recording and making available to participants the content of the decision diagram at all times, with the engineer updating the goal diagram using a standard diagramming solution and posting the updated versions on the blog. Keeping the participants posted of new additions to the blog by automatically generated email messages proved a simple and useful solution, avoiding the need for learning a new tool. This approach, however, encountered difficulties in meetings (more suitable for brainstorming than distributed acquisition) as the engineer is expected to act as the facilitator and record changes to the goal diagram.

2. A second way to apply GAM is to combine the GAM decision process (Sect. 4) with the clarification (Sect. 5) and argumentation (Sect. 6) techniques.

The engineer applies clarification techniques on information recorded in the design rationale. Candidate information for clarification has one or more of the following characteristics: (i) the engineer does not understand the information; (ii) a stakeholder indicates a difficulty in understanding the information; and (iii) the information gave rise to arguments and counterarguments in which it appears that the confrontation results from lack of shared understanding of the given information. Clarification requires the writing and updating of the thesaurus in order to enforce clarification choices throughout the goal diagram and accompanying documentation. In the case studies, a cross-referenced document was created and maintained with a word processor, while markup and cross-referencing was performed with Atlas.ti [1], which allows markup and cross-referencing of both text and graphical (including diagrams) artifacts. A more advanced tool would automatically verify internal thesaurus consistency and automate the enforcement of clarification decisions in associated artifacts (i.e., decision and goal diagram). It remains for future work. In practice, clarification techniques proved particularly useful in raising awareness of clarity issues in stakeholders' statements and various documentation used in the case studies. This lead the participants to rephrase pieces of information in the goal and decision diagrams and the thesaurus so as to check whether they understand them appropriately. When rephrased information appeared different than the original, clarification techniques were applied to check what clarity issue is present and subsequently resolve it. Clarification is time consuming: one potential direction for improvement lies in linking the thesaurus to lexicons such as, e.g., WordNet [21] in order to automatically generate lists of potential synonyms and facilitate the writing of definitions, which is useful in dealing with overgenerality.

Using the argumentation techniques does not necessarily require the formalization of argument or goal diagram content (see point (5) below). Applying techniques such as the argumentation of an alternative and justification already render more rigorous the Evaluation activity in the GAM decision process. Such rigor is needed to avoid issues which stem from unstructured argumentation and justification, and described in Example 13 (e.g., the limited justification of alternatives). It has been empirically observed that nonmonotonic reasoning is hard for humans [22]. Effort involved in finding arguments in GAM is considerable, which is in line the cited empirical result. Some techniques derived from theory are particularly hard to apply in practice: for instance, comparing arguments for specificity appeared counterintuitive and was thus seldom used. The suggested informal orderings based on clarity of arguments appeared, however, practical for rejecting unclear arguments in favor of clear ones. Because these orderings are derived from clarification techniques, clarifying arguments was combined to argument comparison over clarity. To help stakeholders' in their search for arguments, it is useful to question why they believe in the information they provide (thus expecting answers in which they give reasons for what they suggest). Prior experience and resources about the debated domain are relevant sources of arguments, so that referring to these is suggested. Although the difficulties are considerable when applying argumentation and justification, a significant benefit is that these techniques lead to the externalization of information usually left implicit in goal modeling. Abstract entities such as goals of the goal diagram are thus associated with more precise information that led the participants to introduce the given goal in the first place, and that can subsequently be used when operationalizing goals. Moreover, the information made explicit is available to a number of stakeholders who can, through argumentation and justification, question and revise the goal modeling decisions. Lessons can be learned from past modeling problems as sources of these issues (such as, e.g., fallacious argumentation) can be identified by going back to the recorded design rationale and the arguments therein.

3. A third, most rigorous and resource intensive way of applying GAM is to formalize the content of arguments. If the engineer intends to arrive at a formal specification of the goal diagram, it is useful to formalize premises and conclusions using the formal logic of the chosen goal modeling framework: for instance, in Tropos, a temporal first order logic is used, so that the premises and conclusions would be written in that logic, while any automated detection of argument relationships would be performed using the theory of the argumentation framework (Sect. 6). However, as there is normally much more information describing the design rationale than is in the product of the design activity, formalizing the former because the latter needs to be formalized may be difficult to justify in terms of cost and benefits. When GAM is applied to an already elaborate goal diagram (not necessarily constructed with GAM) which is accompanied by a formal specification, argument formalization is facilitated by reusing fragments of the specification. The choice of formalizing depends also on the characteristics of the system: formalizing arguments and justifications of requirements for a safety-critical system can prove relevant, as it is likely that using formal instead of natural language would give rise to less clarity issues. The choice of formalizing or not remains a difficult one (some general suggestions can be found in [7]) in particular since current support for automated analysis of formalized arguments is limited. This, however, does not limit the benefit of introducing the argumentation and justification activities in GAM by using a formal notation, as it simplified the discussions and helped make the presentation precise.

9 Conclusions and future work

Goal modeling unavoidably involves the transformation of, often unclear requirements expressed by stakeholders, inexperienced in RE goal modeling techniques, into more precise information written often in an instance of a goal model. Difficulty further arises when many stakeholders with different backgrounds participate in the engineering of requirements.

The suggested Goal Argumentation Method (GAM) addresses these issues by advancing the available research in the following ways:

1. Techniques for identifying and resolving problems of clarity in expectations expressed by the stakeholders are suggested. The issue of clarity is identified and treated in GAM as one of central problems that appear when dealing with initial statements of requirements. Lack of clarity is treated in a systematic manner, relying on precise and operational definitions of the clarity facets, along with specialized techniques for clarifying information to arrive at more elaborate understanding of stakeholders' expectations.
2. Argumentation of modeling choices is considered critical in obtaining justifiably appropriate instances of requirements models. Argumentation and justification allow straightforward, yet rigorous discussion, revision, and settling on the goal modeling choices.
3. Combination of clarity facets and argumentation allowed the definition of novel ordering relations based on the analysis of argument content to prefer

non-ambiguous, -overgeneral, -synonymous, and -vague arguments to other.

The proposed framework advances the state of the art both on the argumentation and clarification issues. While GAM is method-independent, aiming for complementarity to established and potentially future RE methods, its use has been illustrated in combination with the Tropos goal model. The meeting scheduler and air–traffic management case studies served for illustration.

9.1 Future work

Both argumentation and clarification can be refined and studied further than has been presented in this paper. For instance, justification could be partly automated when GAM is employed with formal RE methods, such as KAOS [17, 36], where arguments could be constructed automatically from already formalized requirements. In this respect, automated translation between goal diagrams and argument trees would allow novel automated checking of goal diagrams.

One important direction of future effort is the extension of GAM beyond goal models, and the use of argumentation and clarification with standardized modeling languages, such as the UML. While GAM has been initially developed and presented herein with goal modeling in mind, it is difficult to accept that UML diagrams are necessarily well argued for or always contain clear information.

Finally, clarification can be further studied. In its current form, a formalism for requirements clarification is unavailable. As mentioned above, arriving at such a formalism is difficult, as it requires the integration of already well discussed, but also still debated topics in logics and AI, such as the formalization of vagueness and generality. Any specification language that would claim to allow the formal representation and reasoning about clarity facets would need to be based on a well-structured formal system, while remaining usable. Arriving at such a formal system would be a significant contribution, for it would take RE a step closer to increased automation of requirements acquisition, whereby such automated requirements assistants would interactively clarify initial statements of requirements. Following the discussion in Sect. 1, such a formalism would allow novel precise and structured representation and reasoning about quality and nonfunctional requirements which tend to involve vague and ambiguous information, an open and pressing issue in requirements and software engineering during the last 4 decades.

Acknowledgments The first author acknowledges funding from the Belgian ICM/CIM Doctoral Fellowship Program.

References

1. ATLAS.ti Scientific Software Development GmbH. ATLAS.ti—The Knowledge Workbench. Available online at http://www.atlasti.com
2. Bach K (1998) Ambiguity. In: Craig E (ed) Routledge encyclopedia of philosophy online. Routledge, London
3. Barker C (2006) Vagueness. In: Brown K (ed) Encyclopedia of language and linguistics, 2nd edn. Elsevier, Amsterdam
4. Bennett B (1998) Modal semantics for knowledge dealing with Vague concepts. In: Proceedings of international conference on principles of knowledge representation and reasoning
5. Berry D, Kamsties E (2005) The syntactically dangerous all and plural in specifications. IEEE Softw 22(1):55–57
6. Besnard P, Hunter A (2001) A logic-based theory of deductive arguments. Artif Intell 128(1–2):203–235
7. Bowen JP, Hinchey MG (1999) High-integrity system specification and design. Springer FACIT Series, London
8. Castro J, Kolp M, Mylopoulos J (2002) Towards requirements-driven information systems engineering: the Tropos project. Info Sys 27(6):365–389
9. Cavalli-Sforza V, Suthers DD (1994) Belvedere: an environment for practicing scientific argumentation. Worksh Comput Dialectics
10. Chantree F, Nuseibeh B, de Roeck A, Willis A (2006) Identifying nocuous ambiguities in natural language requirements. In: Proceedings of international conference on requirement engineering
11. Chesñevar CI, Maguitman AG, Loui RP (2000) Logical models of argument. ACM Comput Surv 32(4):337–383
12. Chung L, Nixon BA, Yu E, Mylopoulos J (2000) Non-functional requirements in software engineering. Kluwer, Dordrecht
13. CMSI (1992) *CM*/1 Product description. Corporate Memory Systems, Inc., 8920 Business Park Dr., Austin, Texas
14. Conklin J, Begeman ML (1988) gIBIS: a hypertext tool for exploratory policy discussion. ACM Trans Info Syst 6(4):303–331
15. Conklin J, Selvin A, Buckingham Shum S, Sierhuis M (2001) Facilitated hypertext for collective sensemaking: 15 years on from gIBIS. In: Proceedings of ACM conference on hypertext and hypermedia
16. Curtis B, Krasner H, Iscoe N (1988) A field study of the software design process for large systems. Commun ACM 31(11):1268–1287
17. Dardenne A, van Lamsweerde A, Fickas S (1993) Goal-directed requirements acquisition. Sci Comput Program 20:3–50
18. Darimont R, van Lamsweerde A (1996) Formal refinement patterns for goal-driven requirements elaboration. In: Proceedings of ACM SIGSOFT symposium foundations of software engineering
19. Donzelli P (2004) A goal-driven and agent-based requirements engineering framework. Req Eng 9(1):16–39
20. Eurocontrol (1999) ATM user requirements document volume 1 and Volume 2. European Air Traffic Control Harmonisation and Integration Programme, Ref FCO.ET1.ST04.DEL01, European Organisation for the Safety of Air Navigation
21. Fellbaum C (1998) WordNet: a lexical reference system and its application. MIT Press, Cambridge
22. Ford M, Billington D (2000) Strategies in human nonmonotonic reasoning. Comput Intell 16(3):446–468
23. Fuxman A, Liu L, Mylopoulos J, Pistore M, Roveri M, Traverso P (2004) Specifying and analyzing early requirements in Tropos. Req Eng 9(2):132–150
24. Gillon BS (1990) Ambiguity, generality, and indeterminacy: tests and definitions. Synthese 85:391–416
25. Graff D (2000) Shifting sands: an interest-relative theory of vagueness. Philos Top 20:45–81

26. Halpern JY (2004) Intransitivity and vagueness. In: Proceedings of international conference on princip of knowledge representation and reasoning
27. Hitchcock D (2007) The concept of argument, and informal logic. In: Gabbay D, Thagard P, Woods J (eds) Philosophy of logic, handbook of the philosophy of science 5. Elsevier, Dordrecht (in press)
28. Hospers J (1953) An introduction to philosophical analysis. Prentice-Hall, Englewood Cliffs
29. Kaci S, van der Torre L (2007) Preference-based argumentation: arguments supporting multiple values. Int J Approx Reason (in press)
30. Kamsties E, Berry D, Peach B (2001) Detecting ambiguities in requirements documents using inspections. In: Proceedings workshop inspection in software engineering
31. Karacapilidis N, Papadias D (2001) Computer supported argumentation and collaborative decision making: the HERMES system. Info Syst 26:259–277
32. Kavakli P, Loucopoulos P (2005) Goal modeling in requirements engineering: analysis and critique of current methods. In: Krogstie J, Halpin T, Siau K (eds) Information modeling methods and methodologies (Advanced Topics of Database Research). IDEA Group
33. Kennedy C (2007) Vagueness and grammar: the semantics of relative and absolute gradable predicates. Linguist Philos (in press)
34. Lee J (1991) Extending the Potts and Bruns model for recording design rationale. In: Proceedings of international conference on software engineering
35. Lee J, Lai K-Y (1991) What's in the design rationale? Hum Comput Interact 6(3–4):251–280
36. Letier E (2001) Reasoning about agents in goal-oriented requirements engineering. PhD Thesis, Département d'ingénierie informatique, Université catholique de Louvain
37. Letier E, van Lamsweerde A (2004) Reasoning about partial goal satisfaction for requirements and design engineering. ACM Sigsoft Softw Eng Notes 29(6):53–62
38. Liu XF, Yen J (1996) An analytic framework for specifying and analyzing imprecise requirements. In: Proceedings of international conference on software engineering
39. Liu L, Yu E (2004) Designing information systems in social context: a goal and scenario modeling approach. Info Syst 29:187–203
40. Louridas P, Loucopoulos P (2000) A generic model for reflective design. ACM Trans Softw Eng Meth 9(2):199–237
41. Maclean A, Young RM, Belotti VME, Moran TP (1991) Questions, options, and criteria: elements of design space analysis. Hum Comput Interact 6(3–4):201–250
42. March JG (1978) Bounded rationality, ambiguity, and the engineering of choice. Bell J Econ 9(2):587–608
43. Margalit A (1983) A Review of Scheffler (1979). J Philos 80:129–137
44. Mylopoulos J, Chung L, Nixon B (1992) Representing and using nonfunctional requirements: a process-oriented approach. IEEE Trans Softw Eng 18(6):483–497
45. Pohl K (1993) The three dimensions of requirements engineering. In: Proceedings of international conference of advanced infomation system engineering
46. Prakken H, Vreeswijk G (2002) Logical systems for defeasible argumentation. In: Gabbay D, Guenther F (eds) Handbook of philosophical logic, 2nd edn. Kluwer, Dordrecht, pp 219–318
47. Ramesh B, Dhar V (1992) Supporting systems development by capturing deliberations during requirements engineering. IEEE Trans Softw Eng 18(6):498–510
48. Ravin Y, Leacock C (eds) (2000) Polysemy: theoretical and computational approaches. Oxford University Press, New York
49. Richman RJ (1959) Ambiguity and Intuition. Mind, New Series 68 269:87–92
50. Rittel HWJ, Webber MM (1973) Dilemmas in a general theory of planning. Policy Sci 4:155–169
51. Searle JR (1969) Speech acts: an essay in the philosophy of language. Cambridge University Press, London
52. Shum BS, Hammond N (1994) Argumentation-based design rationale: what use at what cost? Int J Hum Comput Stud 40(4):603–652
53. Simari GR, Loui RP (1992) A mathematical treatment of defeasible reasoning and its implementation. Artif Intell 53:125–157
54. Smolensky P, Fox B, King R, Lewis C (1987) Computer-aided reasoned discourse, or how to argue with a computer. In: Guindon R (ed) Cognitive science and its applications for human computer interaction. Erlbaum, Hillsdale, pp 109–162
55. Sorensen R (2003) Vagueness. In: Zalta EN (ed) The Stanford encyclopedia of philosophy. Stanford University, Stanford
56. Toulmin S (1958) The uses of arguments. Cambridge University Press, London
57. van Lamsweerde A (2001) Goal-oriented requirements engineering: a guided tour. In: Proceedings of interantional conference on requirement engineering
58. van Lamsweerde A, Darimont R, Massonet Ph (1992) The Meeting Scheduler Problem: Preliminary Definition. Département d'ingénierie informatique, Université catholique de Louvain
59. Williamson T (1994) Vagueness. Routledge, London
60. Yu E (1994) Modelling strategic relationships for process reengineering. PhD Thesis, (also Tech. Report DKBS-TR-94-6) Dept. of Computer Science, University of Toronto
61. Yu E (1997) Towards modelling and reasoning support for early-phase requirements engineering. In: Proceedings of international symposium on requirement engineering
62. Zave P, Jackson M (1997) Four dark corners of requirements engineering. ACM Trans Softw Eng Meth 6(1):1–30