Reasoning About Alternative Requirements Options*

Axel van Lamsweerde

Département d'Ingénierie Informatique, Université catholique de Louvain,
Place Sainte Barbe 2, B-1348 Louvain-la-Neuve (Belgium)
avl@info.ucl.ac.be

Abstract. This paper elaborates on some of the fundamental contributions made by John Mylopoulos in the area of Requirements Engineering. We specifically focus on the use of goal models and their soft goals for reasoning about alternative options arising in the requirements engineering process. A personal account of John's qualitative reasoning technique for comparing alternatives is provided first. A quantitative but lightweight technique for evaluating alternative options is then presented. This technique builds on mechanisms introduced by the qualitative scheme while overcoming some problems raised by it. A meeting scheduling system is used as a running example to illustrate the main ideas.

1 Introduction

Poor requirements were recurrently recognized to be the major cause of software problems such as cost overruns, delivery delays, failure to meet expectations, or degradations in the environment controlled by the software. The early awareness of the so-called requirements problem [2], [3], [6] raised preliminary efforts to develop modeling languages for requirements definition and analysis [1], [40], [18]. With the increasing complexity of software-intensive systems, the research challenges raised by the requirements problem were so significant that an active community emerged in the nineties with dedicated conferences, workshops, working groups, networks, and journals. The term "*requirements engineering*" (RE) was introduced to refer to the process of eliciting, evaluating, specifying, consolidating, and changing the objectives, functionalities, qualities, and constraints to be achieved by a software-intensive system.

John Mylopoulos was involved in requirements engineering research since the early days. His ICSE'82 paper brought early RE research efforts significantly further [16]. The *SADT* graphical language [40] allowed analysts to model two dual system views, the data view and the operation view, together with rudimentary forms of events, triggering operations, and performing agents. As in *Structured Analysis* [13], stepwise model refinement was supported. The *RML* language introduced in [16] provided richer structuring mechanisms such as generalization, aggregation and classification. In that sense it was a precursor to object-oriented analysis techniques. These structuring mechanisms were applicable to three kinds of conceptual units:

* Sections 4 and 5 of this paper are slight adaptations of Sections 16.3.1 and 16.3.2 in Chapter 16 of A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*, Wiley, 2009.

A.T. Borgida et al. (Eds.): Mylopoulos Festschrift, LNCS 5600, pp. 380–397, 2009.

entities, operations, and constraints. Constraints were expressed in a formal assertion language providing, in particular, built-in constructs for temporal referencing. *RML* was also the first requirements modeling language to have a formal semantics, defined in terms of mappings to first-order predicate logic [17]. The *RML* breakthrough was made possible, I believe, thanks to John's unique position at the intersection of three different areas: database modeling [41], [33], knowledge representation [5], and formal specification [4].

In the early nineties, John's group and mine converged on two observations (without any interaction at that time):

- Models for RE needed richer and higher-level abstractions than those provided by design modeling languages and software specification techniques. (It took more than a decade for this observation to gain wide acceptance [32], [36], [22], [24]).
- Some well-established AI techniques were relevant to the RE challenges we wanted to address, in particular in the area of general problem solving, knowledge representation, and knowledge acquisition [21], [34].

Goal-Oriented Requirements Engineering (GORE) emerged from these premises. Two GORE frameworks appeared independently: *KAOS* [10], [11] and *NFR/i** [34], [43]. While both frameworks addressed common targets such as modeling goals and their responsible agents, there were differences and complementarities in focus.

- By and large, the emphasis in NFR/i* was more on qualitative reasoning about soft goals for: analysis of goal contributions [8]; evaluation of alternative goal refinements [35]; reasoning about dependencies among organizational agents [43] and vulnerabilities resulting from such dependencies [31]; customization of user-software interactions [19]; and transition to agent-oriented programming [7].
- In KAOS, the emphasis was more on semi-formal and formal reasoning about behavioral goals for: derivation of goal refinements [12], goal operationalizations [29], and goal assignments [28]; goal-based risk analysis [27]; conflict management [26]; and behavior model synthesis from goals and scenarios [9].

KAOS was more oriented towards goal satisfaction whereas NFR/I* was more oriented towards goal satisficing. Beyond our complementary approaches, there were parallel efforts towards formal goal-based model checking and animation [15], [42], [38] and security analysis [31], [23].

In this overall setting, this paper focuses on what appears to me among the most important contributions of John's group to RE, namely,

- the introduction and use of soft goals as criteria for evaluating alternative options arising throughout the RE process;
- the exploitation of goal models for evaluating such options through qualitative reasoning schemes.

This choice of focus is inevitably biased by my research interests and by the ways John's work has influenced my own efforts. The purpose of the paper is to provide a

brief personal account of John's work in this area and discuss some continuation aimed at addressing various issues raised by it.

Section 2 reviews the various types of alternative options we can find during requirements elicitation and evaluation. Section 3 briefly discusses goal models together with the role played by soft goals in them. Section 4 outlines the qualitative reasoning technique in [34], illustrates its use on a meeting scheduling system, and discusses some problems we may experience with it. Section 5 presents a lightweight quantitative technique aimed at addressing these problems while sharing the same underlying principles.

2 Alternative Options in Requirements Engineering

In any software project, we need to discover, understand, formulate, analyze and agree on *what* problem should be solved, *why* such problem needs to be solved, and *who* should be involved in the responsibility of solving that problem. The problem arises within some broader context. It is in general rooted in a complex organizational, technical, or physical system. This *system* is made of components such as organizational units, people playing specific roles, devices such as measurement instruments, sensors and actuators, and pre-existing software. The aim of the project is to improve this system by building a software solution to the problem and plugging it into the system. We therefore need to consider two versions of the same system:

- the *system-as-is* is the system as it exists before the software is built into it,
- the *system-to-be* is the system as it should be when the software will be built and operate in it.

Our job as requirements engineers is to explore the desired effects of the software-to-be on its surrounding environment together with the assumptions we need to make about this environment. While doing so, we have to make decisions among alternative options arising at multiple places [25], in particular:

- When we have elicited an objective of the system-to-be and want to decompose it into sub-objectives; different decompositions might be envisioned, and we need to select a "best" one.
- When we have identified some likely and critical risk; different countermeasures might be envisioned, and we need to select a "best" one.
- When we have detected a conflict between objectives or requirements and want to resolve it; different resolutions might be envisioned, and we need to select a "best" one.
- When we operationalize a system objective through some combination of functional features, constraints, and assumptions; different combinations might be envisioned, and we need to select a "best" one.
- When we consider alternative assignments of responsibilities among components of the system-to-be –in particular, alternative software-environment boundaries where more or less functionality is automated; "best" alternatives must eventually be selected.

All such situations involve *system* design decisions (not to be confused with software design decisions). Once such decisions have been made, we need to recursively elicit, evaluate, document, and consolidate new requirements and assumptions based on them. Different decisions result in different system proposals which, in turn, will result in different software architectures.

Consider a meeting scheduling system, for example. The objective of knowing the constraints of invited participants might be decomposed into a sub-objective of knowing these constraints through email requests or, alternatively, a sub-objective of knowing them through access to their electronic agenda. A system based on e-mail communication for getting constraints will be different at places from one based on e-agendas. Likewise, different system proposals will result from an alternative where meeting initiators are taking responsibility for resolving date conflicts and a more automated alternative where the software-to-be is responsible for this.

3 Goal Models and the Role of Soft Goals

The system-to-be is intended to meet a number of objectives. These are highlighted as first-class citizens in a goal model where they are interrelated through positive/negative contribution links. A *goal* is a prescriptive statement of intent the system should satisfy through cooperation of its agents. An *agent* is an active system component having to play some role in goal satisfaction through adequate control of system items [25].

Goal satisfaction may involve a variety of system agents defining the *system scope*. The finer-grained a goal is, the fewer agents are required to satisfy it. A *requirement* is a goal under responsibility of a single agent of the software-to-be. An *expectation* is a goal under responsibility of a single agent in the environment of the software-to-be. Expectations cannot be enforced by the software-to-be; they form one kind of assumption we need to make for the system to satisfy its goals.

To be under the sole responsibility of an agent, a goal must be *realizable* by this agent [28]. This means roughly that the agent must be able to control the state variables constrained by the goal specification and to monitor the state variables to be evaluated in this specification.

While reasoning about goal satisfaction in the RE process, we often need to use *domain properties* [25]. These are descriptive statements about the system unlike goals; the latter are prescriptive. Domain properties are expected to hold invariably regardless of how the system will behave [20], [37]. The distinction between descriptive and prescriptive statements is important. Goals may need to be refined into subgoals, negotiated with stakeholders, assigned to agents responsible for them, weakened in case of conflict, or strengthened or discarded in case of unacceptable exposure to risks. Unlike prescriptive statements, domain properties are not subject to such decisions in the RE process.

A goal is either a behavioral goal or a soft goal. A *behavioral goal* prescribes intended system behaviors declaratively. It implicitly defines a maximal set of admissible behaviors [11]. Behavioral goals can be *Achieve* or *Maintain/Avoid* goals. An *Achieve goal* prescribes some *TargetCondition* to be established sooner or later when some current condition holds. A *Maintain goal* prescribes some *GoodCondition* to be maintained. (Similarly, an *Avoid goal* prescribes some *BadCondition* to be avoided.)

Unlike behavioral goals, a *soft goal* cannot be established in a clear-cut sense [34]. In a meeting scheduling system, for example, we cannot say in a strict sense whether a specic system behavior satisfies the goal of reducing the meeting initiator's load or not. We may however say that one system behavior reduces the initiator's load further than another. Said in more general terms, the phrase "goal satisfaction" should not be taken in a strict sense for a soft goal as we cannot observe that the goal is satisfied by some behaviors and not satisfied by others. The phrase "*goal satisficing*" is sometimes used instead; the degree of satisfaction of a soft goal may be higher in one alternative than in another.

Behavioral goals are therefore used for deriving system operations to satisfy them [11], [29] whereas soft goals are used for comparing alternative options to select best ones [34], [8]. We come back to this in Sections 4 and 5.

The behavioral/soft goal typology should not be confused with a goal categorization into functional goals, underlying system services, and non-functional goals, prescribing qualiy of service. For example, a confidentiality goal *Avoid[SensitiveInformationDisclosed]* is traditionally considered as non-functional; it is not a soft goal though as we can always determine whether or not this goal is satisfied in a clear-cut sense.

A *goal model* is basically an annotated AND/OR graph showing how higher-level goals are satisfied by lower-level ones (goal *refinement*) and, conversely, how lower-level goals contribute to the satisfaction of higher-level ones (goal *abstraction*) [10], [34].

Fig. 1 shows a fragment of a goal model for the meeting scheduling system. The goals appearing there are behavioral goals.

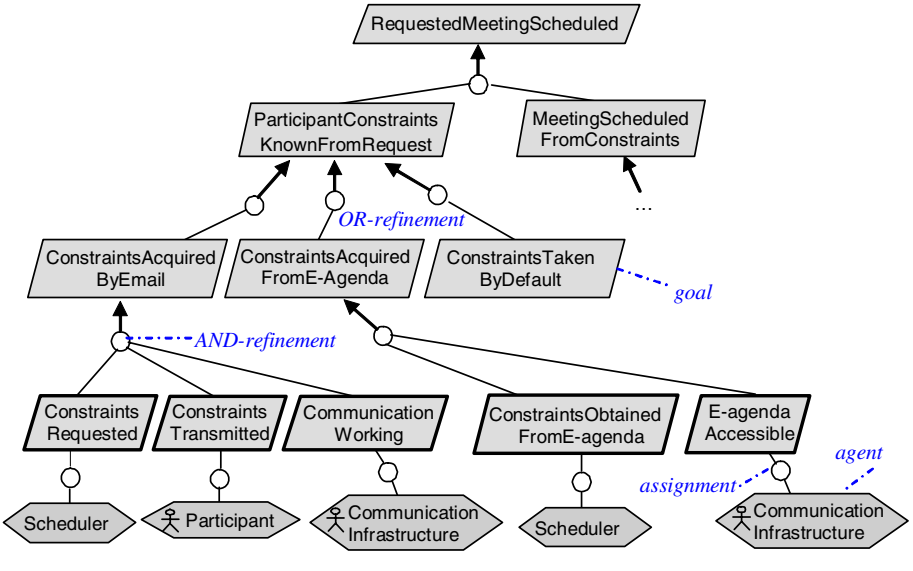

Fig. 1. Goal model fragment for the meeting scheduling system

In a goal model, the top goals are the highest-level ones to be still in the system scope whereas the bottom goals are requirements or expectations assignable to single system agents (see the bottom of Fig. 1). In such graph, an AND-refinement link relates a goal to a set of subgoals called *refinement*. Domain properties may also be included in a refinement. The meaning is that the parent goal can be satisfied/satisficed by satisfying/satisficing all subgoals in the refinement, assuming the domain properties to hold. A goal node can be OR-refined into multiple AND-refinements; each of these is called *alternative* for satisfying/satisficing the parent goal. The meaning of multiple alternative refinements is that the parent goal can be satisfied/satisficed by satisfying/satisficing the conjoined subgoals in any of the alternative refinements.

For example, the goal ParticipantConstraintsKnownFromRequest in Fig. 1 is OR-refined into three alternatives: ConstraintsAcquiredByEmail, ConstraintsAcquiredFromE-Agenda, and ConstraintsTakenByDefault. The goal ConstraintsAcquiredByEmail is in turn AND-refined into three subgoals: ConstraintsRequested, ConstraintsTransmitted, and Communication-Working.

A goal model may also document conflicts that were detected among two or more goals (see Fig. 2 in the next section for an example). A *conflict* arises when behavioral goals cannot be satisfied together or when one of the soft goals contributes negatively to the other goals.

Goal nodes in a goal model are annotated with individual features such as their name and precise specification in natural language, their type, category, priority level, elicitation source, etc. Such annotations act as placeholders for dedicated techniques used in the RE process [25]. For example, priority levels are used for conflict management and requirements prioritization.

The systems *as-is* and *to-be* can both be captured within the same model. The two versions share high-level goals and differ along OR-refinement branches of common parent goals. We can thereby capture multiple variants in a system family.

Goal models can be used for a wide variety of purposes [24], [25], including: the evaluation of alternative options; the structuring and documentation of satisfaction arguments; the checking of the correctness of goal refinements and operationalizations; model animation; the analysis of risks, security threats, and conflicts; requirements prioritization; traceability management; the derivation of software architecture drafts; and the semi-automated generation of the requirements document. In this paper we focus on the use of goal models as a basis for evaluating alternative options.

4 Qualitative Reasoning about Alternative Options

Throughout the RE process we need to explore alternative options of different types, as introduced in Section 2, and select best ones based on specific evaluation criteria. In a GORE framework, options may refer to alternative goal refinements, responsibility assignments, operationalizations, conflict resolutions, risk resolutions, and threat resolutions.

To compare options and select best ones, we need evaluation criteria. The great idea set forth by the NFR framework was to use the soft goals identified in the goal model as evaluation criteria [34]. Different alternatives contribute to different degrees

of satisficing these soft goals. Although originally described in the context of alternative goal refinements, the qualitative evaluation technique in [34] does in fact work for the other types of options as well.

The general idea is to use qualitative estimations for assessing the positive or negative contribution of alternative options to the soft goals [34]. The aim is to determine, in each alternative, a qualitative degree of satisfaction of the top-level soft goals in the goal refinement graph; the option with best degrees of satisfaction is then selected.

To achieve this we need, for each alternative option, to:

- assess its positive or negative influence on each leaf soft goal in the model,
- propagate such influence bottom-up in the goal graph until we reach the top-level soft goals.

Let us make these steps more precise while seeing them in action on our meeting scheduling system. Fig. 2 shows a portion of a goal model with soft goal refinements and conflicts – e.g., the faster the constraint acquisition process or the fewer the interactions with a participant, the less accurate the acquired constraints with respect to the participant's actual constraints.

4.1 Assessing the Qualitative Contribution of Alternative Options to Leaf Soft Goals

We first need to qualitatively assess the extent to which each alternative contributes to the various leaf goals in soft goal refinement trees – e.g., "++" (very positively), "+" (positively), "-" (negatively), "--" (very negatively), "*n*" (neutral). This amounts to putting some *qualitative weight* on refinement and conflict links in the goal model.

Condider our meeting scheduler case study, for example. Fig. 1 highlighted three alternative options for knowing the participant's date constraints:

- acquiring them by email requests,
- acquiring them by access to the participant's electronic agenda,
- taking default constraints (such as working days only).

Table 1 shows the qualitative contribution of these three options to the four leaf soft goals in Fig. 2. For example, the option ConstraintsAcquiredByEmail might

Table 1. Qualitative contributions of options to leaf soft goals

Leaf soft goals	Alternative options		
	Constraints Acquired By Email	Constraints Acquired FromE-Agenda	Constraints aken ByDefault
AccurateConstraints	+	-	-
FastConstraintAcquisition	-	+	+
Minimum Replanning	+	-	-
Minimum Interaction ForGettingConstraints	-	+	-

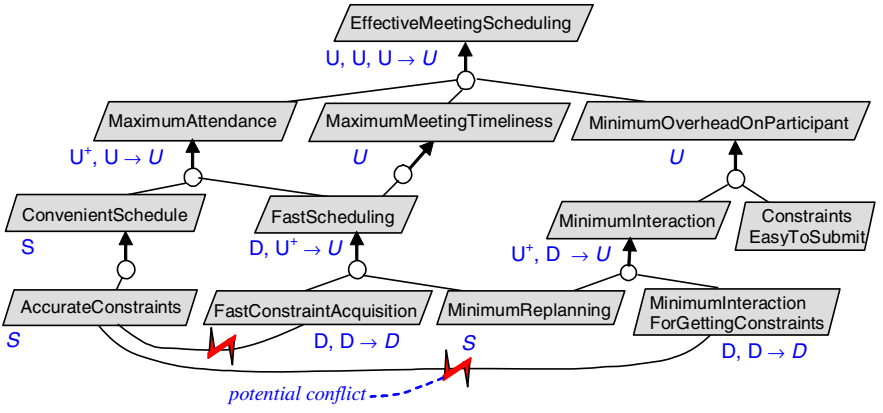

Fig. 2. Upward propagation of satisficing labels in a soft goal refinement graph [25]

contribute negatively to the soft goal FastConstraintAcquisition (as an invited participant might be non-responsive), positively to the soft goal AccurateConstraints (as the participant is likely to know her actual constraints), negatively to the soft goal MinimumInteractionForGettingConstraints (as the participant might get multiple email reminders), etc. On the other hand, the option ConstraintsAcquiredFromE-Agenda might contribute positively to FastConstraintAcquisition, negatively to AccurateConstraints (as the participant's e-agenda is likely to inaccurately reflect her actual availabilities), and positively to MinimumInteractionForGettingConstraints (as no participant interaction is required) .

4.2 Bottom-Up Propagation of Qualitative Contributions

The next step consists of propagating such contributions upwards in the sof goal graph through refinement and conflict links. For this we may use a procedure that assigns qualitative labels to each node in the graph [34]. A node is labelled:

S (satisficed):	if it is satisficeable and not deniable,
D (denied):	if it is deniable but not satisficeable,
C (conflicting):	if it is both satisficeable and deniable,
U (undetermined):	if it is neither satisficeable nor deniable.

The procedure propagates such labels bottom-up along refinement links, marked as "+" or "++" according to the strength of the positive contribution, and along conflict links, marked as "-" or "--" according to the severity of the negative contribution. Additional label values can be assigned at intermediate stages of the procedure, e.g.,

U^+: unconclusive positive support, U^-: unconclusive negative support,
?: user intervention required for getting an adequate label value.

An upward propagation step from offspring nodes to their parent node goes as follows.

1. The individual effect of a weighted link from an offspring to its parent is determined according to propagation rules such as the ones shown in Table 2 (we only consider a few weights to make things simpler). One additional rule states that *if* a node is a refinement node and all its offspring nodes have a *S* label *then* this node gets a *S* label. A node will thus in general have multiple labels –one per incoming link.
2. The various labels thereby collected at a parent node are merged into one single label. The user is first asked to combine the *U+* and *U-* labels into one or more *S*, *D*, *C* and *U* labels. The result is then combined into a single label by choosing the minimal label according to the following ordering:

$$S, D \geq U \geq C.$$

This upward propagation step is applied recursively until the top-level soft goals get a single label.

Let us see how this technique works for the options in Fig. 1 and the soft goal model portion in Fig. 2. We consider the first option ConstraintsAcquiredByEmail in Table 1; it therefore gets a "S" label at the left bottom in Fig. 2. According to Table 2, its "+" contribution links to AccurateConstraints and MinimumReplanning yield a "S" label for these two leaf soft goals (as no other subgoal is shown for the latter goals in the goal model; otherwise they would get a "U+" instead, unless all other subgoals have a "S"). On the other hand, the "-" links from this first option to FastConstraintAcquisition and MinimumInteractionForGettingConstraints yield a "D" label for these two leaf soft goals in Fig. 2.

At the next step, the "S" label of AccurateConstraints yields a "S" label for ConvenientSchedule, through a "+" link, and a "D" label for FastConstraintAcquisition, through the "-" conflict link shown in Fig. 2. Based on the above ordering, the labels "D,D" on the latter goal get merged into a single "D".

At the next step, this "D" label on FastConstraintAcquisition gets propagated through a "+" link to the parent goal FastScheduling as a "D" label again whereas the latter goal also gets a "U^+" label through a "+" link from its offspring MinimumReplanning (see Table 2). After user intervention this "U^+" might become "U" and the resulting label merge yields, according to the predefined label ordering, a single "U" label for FastScheduling. The process goes on upwards until we obtain "U, U, U" labels for the top soft goal EffectiveMeetingScheduling, which get merged into a single "U" label.

Table 2. Qualitative label propagation rules [34]

	Link weight		
	+	**-**	**n**
Offspring label	**Parent label**		
S	U^+	D	U
D	D	U^+	U
C	?	?	U
U	U	U	U

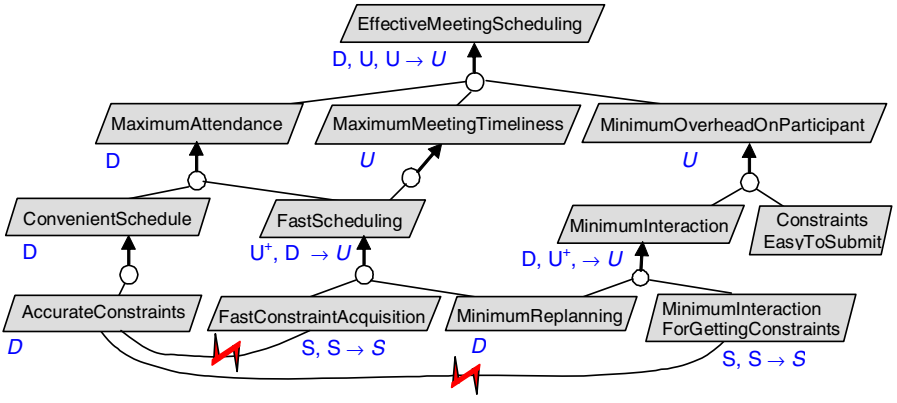

Fig. 3. Propagation of degrees of satisficing for the option Constraints Acquired From E-Agenda

A similar upward propagation for the second option in Table 1, namely, ConstraintsAcquiredFromE-Agenda, yields "D, U, U" labels for the top soft goal, as shown in Fig. 3.

4.3 Discussion

Overall the two options ConstraintsAcquiredByEmail and ConstraintsAcquiredFromE-Agenda seem comparable as they get the same top label after merge, namely, "U" (undetermined). However, the second option has a "D" label (denied) among the top labels which might make it less preferred. The third option of just taking default constraints gets one "D" more at the top which might be a good reason for discarding it.

This simple meeting scheduling example illustrates some limitations of qualitative approaches:

- The propagation rules make labels become rapidly unconclusive as we move up in the soft goal refinement tree. To overcome this, we might refine the qualitative labels, weights, and propagation rules in Table 2 to make them less rough.
- Still, the various types of labels and link weights have no clear meaning in terms of system-specific phenomena. Qualitative reasoning schemes provide some quick and cheap means for rough evaluation in the early stages of the RE process. Their applicability for effective decision support based on accurate arguments appears more questionable.
- All leaf soft goals used as evaluation criteria are considered to have the same importance. This is rarely the case in practice. For example, the soft goal AccurateConstraints is much more important than the soft goal MinimumInteractionForGettingConstraints in view of the key concern of maximum attendance to the meeting.

Regarding the second limitation above, the problem partly arises from the way soft goals are specified. Their specification often violates a basic RE principle stating that goals, requirements and assumptions should be *measurable*. The specification of a

soft goal should therefore be complemented with a *fit criterion* that quantifies the extent to which this goal must be satisfied [39]. For example:

ConvenientSchedule: *The scheduled meeting dates shall meet the date constraints of invited participants as much as possible.*

Fit criterion: *Scheduled dates should fit all actual date constraints of at least 90% of invited participants.*

MinimumInteractionforGettingConstraints: *There should be as little interaction as possible with participants for getting their time constraints.*

Fit criterion: *There should be at most 4 interactions per participant to organize a meeting.*

Measurable soft goal specifications open the way to more accurate evaluation as we discuss now.

5 Lightweight Quantitative Evaluation of Alternative Options

To address the preceding limitations, we may use quantitative estimations for assessing the positive or negative contribution of alternative options to soft goals. The aim is to determine the overall score of each option with respect to all the leaf soft goals in the goal refinement graph, taking their respective importance into account. The option with highest score is then selected.

To achieve this,

- We assign different weights to the leaf soft goals in order to reflect their relative importance.
- We numerically score each option against the leaf soft goals. The scores should be grounded on measurable system phenomena related to the fit criteria of these soft goals.
- We collect the weights and scores in a weighted matrix for overall comparison.

5.1 Quantifying Option Contributions to Leaf Soft Goals through Score Matrices

Weighted matrices are a standard system engineering technique for quantitative decision support. Such matrices bear similarities with those used for risk management within the RE process [14]. They capture estimated scores of each option with respect to the evaluation criteria used.

- As our evaluation criteria are the soft goals from the goal model, we first assign a weight to each leaf soft goal to reflect its importance relatively to others – typically, a numerical proportion. Such weight can be derived from the goal's priority level specified in the goal model [25].
- A matrix cell associated with an option *opt* and a leaf soft goal *lsg* captures the estimated score of the option with respect to this soft goal. A score *X* means that the option is estimated to satisfy the soft goal in *X%* of the cases.

- The last row of the matrix gives the overall score of each option as a weighted summation of its scores with respect to each leaf soft goal:

$$totalScore\,(opt) = \sum_{lsg} (Score\,(opt, lsg) \times Weight\,(lsg))$$

Table 3. Weighted matrix for evaluating alternative options against all leaf soft goals [25]

Leaf soft goals	*Importance weighting*	Option scores		
		Constraints Acquired By Email	*Constraints Acquired FromE-Agenda*	*Constraints Taken ByDefault*
AccurateConstraints	*0.50*	*0.90*	*0.30*	*0.10*
Fast ConstraintAcquisition	*0.30*	*0.50*	*0.90*	*1.00*
MinimumReplanning	*0.10*	*0.80*	*0.30*	*0.10*
MinimumInteraction ForGettingConstraints	*0.10*	*0.50*	*1.00*	*1.00*
TOTAL	**1.00**	**0.73**	**0.55**	**0.46**

Table 3 shows such quantitative evaluation for the options and soft goals evaluated qualitatively in Table 1. In this evaluation, the soft goal AccurateConstraints is considered relatively much more important than the soft goals MinimumInteractionForGettingConstraints and MinimumReplanning (the latter having the same level of limited importance). The first option, ConstraintsAcquiredByEmail, is estimated to satisfy the soft goal AccurateConstraints in *90 %* of the cases –as invited participants are expected to directly express their own constraints. The 10% remaining stand for participants confusing dates or having taken other commitments in the meantime. Overall, this first option outperforms the others in view of the relative weights assigned to each soft goal.

For this approach to work effectively, we need to be able to determine *adequate* option scores against each leaf soft goal in the goal model. Note that the accuracy of an individual score taken in isolation is not what matters the most. We can draw meaningful comparative conclusions as long as *all scores are set consistently, from one alternative to the other, as a common basis for making comparisons*. Nevertheless, to avoid subjective estimations resulting in questionable decisions, the scores should be grounded on domain expertise, experience with similar systems and interpretations in terms of measurable phenomena in the system.

5.2 Deriving Option Scores from Measures of Soft Goal Satisficing

The goal model may be used to estimate option scores that are grounded on measurable system phenomena. The general idea is to: identify gauge variables referred to in the specification of the soft goals and their fit criterion; evaluate such variables along the refinement tree of the various options; and derive options scores from the values obtained.

Identifying and evaluating gauge variables. A *gauge variable* is a variable associated with a specific leaf soft goal in the goal model. It may capture:

- a quantity the soft goal prescribes to *Improve*, *Increase*, *Reduce*, *Maximize*, or *Minimize*;
- the estimated cost of satisficing this soft goal;
- the estimated time taken for satisficing it.

Consider the leaf soft goal *MinimumInteractionForGettingConstraints* in the meeting scheduling system. Its specification was seen to be:

There should be as little interaction as possible with participants for getting their time constraints.

Fit criterion*: There should be at most 4 interactions per participant to organize a meeting.*

The variable ExpectedNumberOfInteractions may be derived from this specification as a gauge variable for this soft goal. It estimates the average number of interactions between a participant and the scheduler to get the participant's constraints. Note that the full specification implicitly specifies an *ideal target value* (*0*) and an *acceptability threshold* (*4*).

To support the evaluation of options based on soft goals grounded on measurable phenomena, a gauge variable should meet the following requirements.

- *Soft goal measure:* To enable comparisons of options with respect to soft goals, the variable should provide some measure of the degree of satisficing of its associated leaf soft goal.
- *Cumulative quantity:* To enable accurate estimations of its values for the different options, the variable should propagate additively along the AND-trees refining these options in the goal model.

Fig. 4 explains what is meant by additive propagation. Let *lsg* denote a leaf soft goal and *gv* a cumulative gauge variable associated with it. In view of the semantics of goal AND-refinement, the value of *gv* at a parent goal *G* in the refinement tree of a specific option is obtained by summing up the values of *gv* at the subgoals *G1* and *G2*. When a value for the variable at some subgoal makes no sense, we just ignore it in the summation.

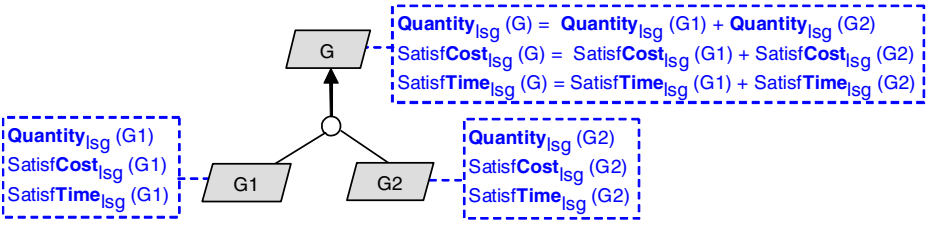

Fig. 4. Cumulative propagation of gauge variables [25]

The merits of an option are obtained by upward propagation, along the option's refinement tree, of the cumulative values of the gauge variables measuring the satisficing of the leaf soft goals in the goal model, starting from the option's own leaf subgoals. The reason for performing such up-propagation is that the values of gauge variables will generally be more easily and accurately estimated for the finer-grained leaf subgoals of the option.

Table 4 illustrates the evaluation of gauge variables by up-propagation from the leaf subgoals in the various options. Each gauge variable there corresponds to a quantity that the associated leaf soft goal in Table 3 prescribes to *Maximize* or *Minimize*.

Table 4. Values of gauge variables for soft goal satisficing by alternative options [25]

Soft-goal gauge variable	Option values		
	Constraints Acquired By Email	*Constraints Acquired FromE-Agenda*	*Constraints Taken ByDefault*
Expected Number of Correct Free Half-Days per Week	*9*	*3*	*1*
Expected Constraint Acquisition Time (in days)	*3*	*1*	*0*
Expected Number of Replannings	*0.5*	*2*	*4*
Expected Number of Interactions	*2*	*0*	*0*

For example, consider the gauge variable *ExpectedNumberOfInteractions* in Table 4, associated with the soft goal MinimumInteractionForGettingConstraints in Table 3. The estimated value "2" for this variable in the option *ConstraintsAcquiredByEmail* is obtained by summing the value "1" for the subgoal ConstraintsRequested in Fig. 1 (one interaction per requested constraints) and the value "1" for the subgoal ConstraintsTransmitted (another interaction per returned constraints). The estimated value "3" in the same column, for the same option and the gauge variable *ExpectedConstraintAcquisitionTime* associated with the soft goal FastConstraintAcquisition, results from "0" (time taken for the scheduler to request constraints) + "3" (estimated average time, in days, for a participant to return her constraints). Each gauge variable in Table 4 is derived from the specification and fit criterion of the corresponding leaf soft goal in Table 3; its values are obtained by up-propagation along the refinement tree of the corresponding option from the estimated values at leaf nodes (see Fig. 4).

Deriving option scores from values of gauge variables. Once the overall gauge values are obtained at the root of the refinement tree of each option by such up-propagation, we can derive the scores of each option as follows.

Let *Score (opt, lsg)* denote the score of option *opt* with respect to the leaf soft goal *lsg*. Let g_V denote the value of the gauge variable associated with *lsg*, obtained at the root of *opt*'s refinement tree by up-propagation from its leaf subgoals. Let g_T denote the ideal target value we would expect for this variable and g_{max} its maximum

acceptable value; g_T and g_{max} are derived from *lsg*'s specification and fit criterion, respectively. We then have:

$$Score\,(opt, lsg) = 1 - (|\, g_T - g_V \,|) / g_{max}$$

As we can see from this formula, the closer the gauge value to its ideal target relatively to its maximum acceptable value, the closer the corresponding score to 1. The more distant the gauge value from its ideal target relatively to its maximum acceptable value, the closer the corresponding score to 0.

Table 5. Evaluation of alternative options against leaf soft goals based on gauge variables

Leaf soft goals	*Importance weighting*	**Option scores**				
		g_T	g_{max}	*Constraints Acquired ByEmail*	*Constraints Acquired FromE-Agenda*	*Constraints Taken ByDefault*
AccurateConstraints	*0.50*	10	10	*0.90*	*0.30*	*0.10*
Fast ConstraintAcquisition	*0.30*	0	6	*0.50*	*0.84* (0.90)	*1.00*
MinimumReplanning	*0.10*	0	4	*0.87* (0.90)	*0.50* (0.30)	*0* (0.10)
Minimum InteractionFor GettingConstraints	*0.10*	0	4	*0.50*	*1.00*	*1.00*
TOTAL	**1.00**			**0.74** (0.73)	**0.55**	**0.45** (0.46)

Table 5 shows another weighted matrix for evaluating alternative options in our meeting scheduling system. Compared to the matrix in Table 3, this one is based on such score derivation from the values of the gauge variables in Table 4. (The numbers in parentheses refer to the corresponding rough estimations in Table 3.)

For example, Table 4 gave a value of "2" for the gauge variable ExpectedNumberOfInteractions in the option ConstraintsAcquiredByEmail. This gauge measures the degree of satisficing of the *Minimize* soft goal MinimumInteractionForGettingConstraints. From the specification and fit criterion of this soft goal, we get the values "0" for g_T and "4" for g_{max}. As the value g_V for this gauge, obtained by up-propagation from the leaf nodes in the refinement tree of the option ConstraintsAcquiredByEmail, is "2", the score obtained according to the preceding formula is "0.50". Similarly, Table 4 gave a value of "9" for the gauge variable ExpectedNumberOfCorrectFreeHalfDaysPerWeek in the option ConstraintsAcquiredByEmail. This gauge measures the degree of satisficing of the *Maximize* soft goal AccurateConstraints; a value *X* for this variable means that, among the 10 working half-days in a week where a participant is *stated* to be free, *X* of these are half-days where she is *actually* free. From the specification and fit criterion of this soft goal, we get the values "10" for g_T and "10" for g_{max}. As the value g_V for this gauge, obtained by up-propagation from the leaf nodes in the refinement tree of the option ConstraintsAcquiredByEmail, is "9", the score obtained according to the preceding formula is "0.90".

In comparison with Table 3, the numbers in Table 5 are not significantly different. They result in the same comparative evaluation yielding the selection of the first

option ConstraintsAcquiredByEmail as best one. There are significant differences, however, in the arguability of conclusions and the way we get to them through gauge variables:

- these conclusions rely on measures of degree of soft goal satisficing that are based on system phenomena;
- they are derived systematically from the specifications of the leaf soft goals in the goal model.

We therefore gain increased confidence in the adequacy of our conclusions.

A more sophisticated approach to quantitative reasoning about alternative options can be found in [30]. Quality variables are used there instead of gauge variables. They are random variables with probability distribution functions. The analysis then is more accurate but more heavyweight as a price to pay.

6 Conclusion

The evaluation of alternative system options is at the heart of the RE process. John Mylopoulos has significantly contributed to the development of concepts, models and techniques for this critical task. The important role played by goal models, soft goals as evaluation criteria, and propagation of positive/negative goal contributions are now much better understood. Others have built upon his results and will continue to explore the directions he has opened.

Beyond the work outlined in this paper, the RE community owes much to John for his contribution to raising the technical standards in the field, his open-minded and interdisciplinary attitude in research, his humility and friendliness in research interactions, and the network of colleagues he has created worldwide.

Acknowledgement. The principle of using measurable fit criteria for quantitative reasoning about soft goals results from joint work with Emmanuel Letier. Warm thanks are due to the reviewer of this paper whose requests for clarification resulted in the introduction of quantified links between option scores and soft-goal gauge variables.

References

1. Alford, M.: A Requirements Engineering Methodology for Real-Time Processing Requirements. IEEE Transactions on Software Engineering 3(1), 60–69 (1977)
2. Bell, T.E., Thayer, T.A.: Software Requirements: Are They Really a Problem? In: Proc. ICSE 1976: 2nd International Conference on Software Enginering, San Francisco, pp. 61–68 (1976)
3. Boehm, B.W.: Software Engineering Economics. Prentice-Hall, Englewood Cliffs (1981)
4. Borgida, A., Mylopoulos, J., Reiter, R.: And Nothing Else Changes: The Frame Problem in Procedure Specifications. In: Proc. ICSE 1993 - 15th International Conference on Software Engineering, Baltimore (May 1993)

5. Brodie, M., Mylopoulos, J., Schmidt, J. (eds.): On Conceptual Modeling: Perspectives from Artificial Intelligence, Databases, and Programming Languages. Springer, Heidelberg (1984)
6. Brooks, F.P.: No Silver Bullet: Essence and Accidents of Software Engineering. IEEE Computer 20(4), 10–19 (1987)
7. Castro, J., Kolp, M., Mylopoulos, J.: Towards Requirements-Driven Information Systems Engineeering: the TROPOS Project. Information Systems 27, 365–389 (2002)
8. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-functional requirements in software engineering. Kluwer Academic, Boston (2000)
9. Damas, C., Lambeau, B., van Lamsweerde, A.: Scenarios, Goals, and State Machines: A Win-Win Partnership for Model Synthesis. In: Proceedings of FSE 2006, 14th ACM International Symp. on the Foundations of Software Engineering, November 2006, Portland, OR (2006)
10. Dardenne, A., Fickas, S.S., van Lamsweerde, A.: Goal-directed Concept Acquisition in Requirements Elicitation. In: Proc. 6th International Workshop on Software Specification and Design, Como, Italy, pp. 14–21 (1991)
11. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-Directed Requirements Acquisition. Science of Computer Programming 20, 3–50 (1993)
12. Darimont, R., van Lamsweerde, A.: Formal Refinement Patterns for Goal-Driven Requirements Elaboration. In: FSE'4 - Fourth ACM SIGSOFT Symp. on the Foundations of Software Engineering, October 1996, pp. 179–190 (1996)
13. DeMarco, T.: Structured Analysis and System Specification. Yourdon Press (1978)
14. Feather, M.S., Cornford, S.L.: Quantitative Risk-Based Requirements Reasoning. Requirements Engineering Journal 8(4), 248–265 (2003)
15. Fuxman, A., Pistore, M., Mylopoulos, J., Traverso, P.: Model Checking Early Requirements Specifications in Tropos. In: Proc. RE 2001 - 5th Intl. Symp. Requirements Engineering, Toronto (August 2001)
16. Greenspan, S.J., Mylopoulos, J., Borgida, A.: Capturing More World Knowledge in the Requirements Specification. In: Proc. ICSE-1982: 6th Intl. Conf. on Software Enginering, Tokyo (1982)
17. Greenspan, S.J., Borgida, A., Mylopoulos, J.: A Requirements Modeling Language and its Logic. Information Systems 11(1), 9–23 (1986)
18. Heninger, K.L.: Specifying Software Requirements for Complex Systems: New Techniques and their Application. IEEE Transactions on Software Engineering 6(1), 2–13 (1980)
19. Hui, B., Laiskos, S., Mylopoulos, J.: Requirements Analysis for Customizable Software: A Goals Skills Preferences Framework. In: Proc. RE 2003: 11th IEEE Joint Intl. Requirements Engineering Conference, Monterey, CA, September 2003, pp. 117–126 (2003)
20. Jackson, M.: Software Requirements & Specifications - A Lexicon of Practice, Principles and Prejudices. ACM Press, Addison-Wesley (1995)
21. van Lamsweerde, A., Dardenne, A., Delcourt, B., Dubisy, F.: The KAOS Project: Knowledge Acquisition in Automated Specification of Software. In: Proc. AAAI Spring Symposium Series, Track: Design of Composite Systems, Stanford University, American Association for Artificial Intelligence, March 1991, pp. 59–62 (1991)
22. van Lamsweerde, A.: Requirements Engineering in the Year 00: A Research Perspective. In: Proc. ICSE 2000: 22nd International Conference on Software Engineering, pp. 5–19. ACM Press, New York (2000) (invited keynote paper)
23. van Lamsweerde, A.: Elaborating Security Requirements by Construction of Intentional Anti-Models. In: Proc. ICSE 2004, 26th Intl. Conference on Software Engineering, Edinburgh, May 2004, pp. 148–157. ACM-IEEE (2004)

24. van Lamsweerde, A.: Requirements Engineering: From Craft to Discipline. Invited Paper for the ACM Sigsoft Outstanding Research Award, Proc FSE-16: 16th ACM Conference on the Foundations of Software Engineering, Atlanta (November 2008)
25. van Lamsweerde, A.: Requirements Engineering: From System Goals to UML Models to Software Specifications, January 2009. Wiley, Chichester (2009)
26. van Lamsweerde, A., Darimont, R., Letier, E.: Managing Conflicts in Goal-Driven Requirements Engineering. IEEE Transactions on Sofware Engineering 24(11), 908–926 (1998)
27. van Lamsweerde, A., Letier, E.: Handling Obstacles in Goal-Oriented Requirements Engineering. IEEE Transactions on Software Engineering 26(10), 978–1005 (2000)
28. Letier, E., van Lamsweerde, A.: Agent-Based Tactics for Goal-Oriented Requirements Elaboration. In: Proc. ICSE 2002: 24th Intl. Conf. on Software Engineering, May 2002. ACM-IEEE (2002)
29. Letier, E., van Lamsweerde, A.: Deriving Operational Software Specifications from System Goals. In: Proc. FSE'10: 10 th ACM Symp. Foundations of Software Engineering, Charleston (November 2002)
30. Letier, E., van Lamsweerde, A.: Reasoning about Partial Goal Satisfaction for Requirements and Design Engineering. In: Proc. FSE 2004, 12th ACM International Symp. on the Foundations of Software Engineering, Newport Beach, CA, November 2004, pp. 53–62 (2004)
31. Liu, L., Yu, E., Mylopoulos, J.: Security and Privacy Requirements Analysis within a Social Setting. In: Proc. RE 2003: Intl. Requirements Engineering Conf. (September 2003)
32. Mylopoulos, J.: Information Modeling in the Time of the Revolution, Invited Review. Information Systems 23(3-4), 127–155 (1998)
33. Mylopoulos, J., Bernstein, P., Wong, H.: A Language Facility for Designing Interactive Database-Intensive Applications. ACM Transactions on Database Systems 5(2) (June 1980)
34. Mylopoulos, J., Chung, L., Nixon, B.: Representing and Using Nonfunctional Requirements: A Process-Oriented Approach. IEEE Transactions on Sofware Engineering 18(6), 483–497 (1992)
35. Mylopoulos, J., Chung, L., Liao, S., Wong, H., Yu, E.: Exploring Alternatives During Requirements Analysis. IEEE Software 18(1), 92–96 (2001)
36. Mylopoulos, J., Chung, L.L., Yu, E.E.: From Object-Oriented to Goal-Oriented Requirements Analysis. Communications of the ACM 42(1), 31–37 (1999)
37. Parnas, D.L., Madey, J.: Functional Documents for Computer Systems. Science of Computer Programming 25, 41–61 (1995)
38. Ponsard, C., Massonet, P., Molderez, J.F., Rifaut, A., van Lamsweerde, A.: Early Verification and Validation of Mission-Critical Systems. Formal Methods in System Design 30(3), 233–247 (2007)
39. Robertson, S., Robertson, J.: Mastering the Requirements Process. ACM Press, Addison-Wesley (1999)
40. Ross, D.T., Schoman, K.E.: Structured Analysis (SA): A Language for Communicating Ideas. IEEE Transactions on Software Engineering 3(1), 16–34 (1977)
41. Roussopoulos, N., Mylopoulos, J.: Using Semantic Networks for Database Management. In: Proc. 1st Conf. on Very Large Databases (VLDB), September 1975, pp. 144–172 (1975)
42. Tran Van, H., van Lamsweerde, A., Massonet, P., Ponsard, C.: Goal-Oriented Requirements Animation. In: Proc. RE 2004, 12th IEEE Joint International Requirements Engineering Conference, Kyoto, September 2004, pp. 218–228 (2004)
43. Yu, E.S.K.: Modelling Organizations for Information Systems Requirements Engineering. In: Proc. RE 1993 - 1st Intl Symp. on Requirements Engineering, pp. 34–41 (1993)