A Qualitative, Interactive Evaluation Procedure for Goal- and Agent-Oriented Models

Jennifer Horkoff[1], Eric Yu[2]

[1]Department of Computer Science, [2]Faculty of Information,
University of Toronto, Canada
jenhork @ cs.utoronto.ca, yu @ ischool.utoronto.ca

Abstract. Early modeling of domain intentions has emerged as a method for deriving early-phase requirements. Although much understanding can be gained through the creation of such models, we want to find ways to get more use out of models and the modeling process, increasing the return in the investment of modeling time. Applying systematic analysis procedures to early requirements models can test the satisfaction of stakeholder goals and facilitate an evaluation of design alternatives. In this work, we introduce a qualitative and interactive model evaluation procedure for the i* Framework. We use a qualitative approach to compensate for the lack of detail in early requirement analysis. In applying the procedure to a variety of case studies, including strategic modeling for a large social service organization, we found that the interactive nature of the procedure prompts semantic checks, producing models that better reflect the intended perspective and increasing domain knowledge.

Keywords: Goal-Oriented Requirements Engineering, Agent-Oriented Modeling, Model Analysis, Model Reasoning

1 Introduction

Use of goal- and agent-oriented models has emerged as a way to help system developers ensure that they have elicited the 'right', or an adequate set, of requirements. In these approaches, high-level system functionality is mapped to goals of particular stakeholders via positive or negative contributions to goal satisfaction, allowing for the assessment of system functionality in terms of stakeholder objectives, and taking into account tradeoffs between non-functional requirements.

The approach introduced by goal- and agent-oriented modeling frameworks, such as i* [1], has been referred to as "Early" Requirements Analysis, advocating for modeling and exploration of the socio-technical domain before focusing on detailed system functionality. Such models form a complex web of relationships and dependencies. In order to perform analysis over such models, systematic analysis procedures are needed, considering the chain of effects of goals and functionalities throughout the network in a consistent way. The immediate aim of such analysis is to determine whether stakeholder's goals can be achieved, given domain assumptions.

Various analysis procedures aimed for goal models have been introduced (for example [2], [3], [4], and [5]). Despite the availability of these procedures in the literature, when goal-oriented techniques are used, explicit analysis procedures are often not applied. For example, work in [6] uses synchronization with i* and other models to find omissions in the domain, but does not apply analysis to i* models individually. Work in [7] uses comparisons of i*-like models to value models for a requirements analysis of enterprise networks without applying explicit analysis to goal models. These omissions can be partially attributed to the lack of emphasis on analysis in earlier presentations of goal and agent-oriented models [1].

As goal models are used in the early stages of the system development process, the resulting models have an intrinsic level of informality. Many domain concepts are subject to interpretation, especially quality attributes. It is hard to judge when the model is sufficiently complete to reflect the real-world complexity of the domain. Nevertheless, an analysis of the social aspects of the domain is often critical to the success of a system.

In this work, we argue that the informal and incomplete nature of goal models calls for analysis procedures which are interactive, qualitative, and simple to apply. To this end we introduce a qualitative, interactive evaluation procedure for goal-and agent-oriented models. We describe the specifics of the procedure in terms of the i* Framework, although the procedure can be applied to other goal- and agent-oriented models in general. The procedure is adapted from an evaluation procedure included in the NFR Framework [8], which was only described briefly in prose. We expand on this work by specifying the algorithm in precise pseudocode, providing more formal definitions for evaluation terminology, taking into account agent-related features, and considering several issues not covered in the original procedure. The original procedure was advocated as a way to compare design alternatives. Here, we go beyond this purpose, describing additional benefits, including model iteration.

The procedure has been tested via application to a number of detailed case studies. Application to a long-term project involving a large social-service application allowed us to effectively evaluate alternative system designs represented within large-scale models [9] [10]. The procedure has been used to depict and analyze differences between viewpoints concerning technological intentions [11]. It has been used in an agent-oriented approach to analyzing knowledge transfer effectiveness [12]. We use excerpts from these and other examples in this work.

This paper is organized as follows: Section 2 describes the type of analysis needed for early requirements models, Section 3 briefly describes the i* Framework, Section 4 describes the evaluation procedure introduced in this work, Section 5 describes the benefits of this procedure, Section 6 contains discussion, Section 7 reviews related work, while Section 8 outlines conclusions and future work.

2 Analysis of Early Requirements Models

As the flexibility of goal- and agent-oriented modeling allows application in many stages of system development, different analysis approaches may be more appropriate for different stages of system development, for different purposes. For later stages of

system analysis, where quantitative information is known, where models contain more specific detail concerning stakeholder actions and system functionality, and where models are relatively stable, automated and quantitative evaluation can be appropriate.

For early-stage modeling, qualitative, interactive evaluation is more appropriate. As argued in [8], when considering non-functional desires such as privacy, maintainability, or usability in the early phases of system analysis, before detailed requirements are developed, concrete quantitative information is often not available to express non-functional notions precisely. However, given the importance of addressing non-functional requirements, there are benefits to capturing, representing and reasoning about these desires before they can be quantifiable. The use of qualitative measures allows for analysis in this early stage.

Although we are interested in capturing the social and personal motivations which affect the success of a system, it is difficult to completely capture all stakeholder goals and social phenomena which may affect or be affected by a software system. While a fully automated evaluation procedure may appear to require the least effort from the user, it is not necessarily preferable for early requirements analysis, due to the informal and potentially incomplete nature of the socio-technical models. An interactive procedure allows expert judgment to be applied at appropriate points, to compensate for the inherent incompleteness. Interactive evaluation forces the modeller to examine the model in greater detail, following the reasoning steps, questioning the qualitative results and the underlying semantics of the model. The gaps in knowledge revealed by this process prompt the modeller to pursue further domain elicitation. Our experience with case studies has shown that the investigations and changes prompted by this iterative process can help to improve the quality of the model, and increase the modellers' knowledge of the domain. In order to support interactivity and gain the trust of the modeller, analysis procedures should be simple to use and transparent.

The imprecise nature of goal-oriented models makes it difficult to judge the correctness of analysis results in a formal way. Instead we focus on judging the utility of the analysis in helping to understand and communicate aspects of the system domain, and in helping to support decision making in high-level system design.

3 Analysis with the i* Framework

The i* Framework is intended to facilitate exploration of the system domain with an emphasis on social aspects by providing a graphical depiction of system actors including their intentions, dependencies, responsibilities, alternatives and vulnerabilities [1].

The social aspect of i* is represented by *actors*, including *agents* and *roles*, and the associations between them. Actors depend upon each other for the accomplishment of *tasks*, the provision of *resources*, the satisfaction of *goals* and *softgoals*. *Softgoals* are goals without clear-cut criteria for satisfaction. Actors can be "opened-up" using *actor boundaries* containing the goals, softgoals, tasks, and resources explicitly desired by the actors. The interrelationships between elements inside an actor are

depicted with *Decomposition* links, showing the elements which are necessary in order to accomplish a task; *Means-Ends* links, showing the alternative tasks which can accomplish a goal; and *Contribution* links, showing the effects of softgoals, goals, and tasks on softgoals. Positive/negative contributions representing evidence which is sufficient enough to satisfy/deny a softgoal are represented by *Make*/*Break* links, respectively. Contributions with positive/negative evidence that is not sufficient to satisfy/deny a softgoal are represented by *Help*/*Hurt* links. Positive/negative evidence of unknown strength is represented by *Some*+/*Some*- links.

When analyzing i* models, one can visualize the effects of one element on others element for one or two steps. However, more complex questions may involve many propagation steps along the graph, cycles, multiple paths, and multiple sources of evidence. Performing this process in an ad hoc manner with no intermediate storage of results is difficult and can be prone to errors and inconsistencies. It is clear that a systematic evaluation procedure is needed.

4 A Qualitative, Interactive Evaluation Procedure for the i* Framework

The proposed procedure works as follows. The modeller raises a question such as "How effective is this design option with respect to the desired goals?" The procedure starts with initial values (typically satisfied or denied) assigned to model elements representing the design option. These values are propagated through the model links using defined rules. Human judgement is needed when multiple conflicting or partial values must be combined to determine the satisfaction or denial of a softgoal. The final satisfaction and denial values for the elements of each actor are analyzed in light of the original question. An assessment is made as to whether the design choice is satisficed ("good enough"), likely stimulating further exploration. During this process, if the evaluation values seem counter-intuitive, the model can be examined to determine if it captures the domain to a sufficient level of accuracy and completeness, triggering iterative cycles of model refinement. Details of this procedure are described in the following sections, with more detail found in [13].

Although the description of the procedure below may seem complex, the precise detail is intended to allow a reproducible implementation. This complexity is hidden from the user in the implementation. In fact, most applications of the procedure have been performed manually, attesting to its simplicity.

4.1 Introducing a More Precise Description of i* Concepts

Given the informal nature of i*, it is not our aim to introduce a complete formalization of the i* Framework. However, to clearly describe the evaluation algorithm introduced in this work, we introduce some more precise definitions of i* concepts.

Definition: An i* model. *An i* model is a tuple $\mathcal{M} = <\mathcal{E}, \mathcal{L}, \mathcal{A}, \mathcal{AL}>$, where $\mathcal{E}$ is a set of elements, each having a type attribute e.type, with a value of the set {Softgoal, Goal, Task, Resource}, $\mathcal{L}$ is a set of links, each having a type attribute l.type with*

value being one of {ME, DEC, DEP, CONT}, (means-ends, decomposition, dependency, and contribution, respectively) where CONT can be further divided into {Make, Some+, Help, Unknown, Hurt, Some-, Break}, $\mathcal{A}$ is a set of actors, and $\mathcal{AL}$ is a set of actor association links.

Definition: Links. *DEP and CONT links are "binary", relating one element to another element, and can be expressed as $\mathcal{L}: \mathcal{E} \rightarrow \mathcal{E}$. For example, in HELP: $e_1 \rightarrow e_2$, e_1 is the* ***source*** *and contributes to e_2, the* ***destination****. In DEP: $e_1 \rightarrow e_2$, e_2 depends on e_1, meaning that e_1 is the source and e_2 is the destination. The remaining links, ME and DEC are "(n+1)-ary", $\mathcal{L}: \mathcal{E} \times ... \times \mathcal{E} \rightarrow \mathcal{E}$, meaning that for ME/DEC:$(e_1, ..., e_n) \rightarrow e$, the evaluation values of a set of elements $(e_1, ..., e_n) \in \mathcal{E} \times ... \times \mathcal{E}$ determines the evaluation value of $e \in \mathcal{E}$.*

Given these more precise definitions, we can describe some restrictions on the construction of a valid i* model, as per the style described in [1].

Syntax Restrictions on an i* model.

a. *Each element has at most one Decomposition or Means-Ends relation, i.e. for each $e_d \in \mathcal{E}$, only one of DEC:$e_s \rightarrow e_d$ or ME:$e_s \rightarrow e_d$ hold.*
b. *For each DEC:$(e_1, ..., e_n) \rightarrow e$, e.type = Task, while $(e_1, ..., e_n)$.type = any type.*
c. *For each ME:$(e_1, ..., e_n) \rightarrow e$, e.type = Goal, while $(e_1, ..., e_n)$.type=Task.*
d. *For each CONT:$e_s \rightarrow e_d$ relation, e_d.type = Softgoal, e_s.type = any type.*

4.2 Qualitative Evaluation Labels

For simplicity and recognition, we adopt the qualitative labels used in NFR evaluation and previous examples of i* evaluation, replacing "weakly" with "partially". The *(Partially) Satisfied* label represents the presence of evidence which is *(insufficient)* sufficient to satisfy a goal. *Partially* denied and *denied* have the same definition with respect to negative evidence. *Conflict* indicates the presence of both positive and negative evidence of roughly the same strength. *Unknown* represents the situation where there is evidence, but its effect is unknown. We introduce the "None" label to indicate a lack of any label. These labels are shown in the Table 1.

Definition: Evaluation Labels/Values. *We can define qualitative evaluation labels as an attribute of $e \in \mathcal{E}$, expressed as e.v, where $v \in \mathcal{V}$ and $\mathcal{V}$ ={S, PS, U, C, PD, D, N}, (Satisfied, Partially Satisfied, Unknown, Conflict, Partially Denied, Denied, or None, respectively).*

We adopt the use of partial labels for tasks, resources, and goals to allow for greater expressiveness. For consistency, we use the term satisfied and not *satisficed* when talking about element satisfaction. For a discussion of i* semantics, see [13].

4.3 Initial Evaluation Values

In order to start an evaluation of a model, a set of initial evaluation values must be placed on the model, reflecting a particular situation.

Definition: Initial Evaluation Values. *Initial evaluation values are each $e.v \in \mathcal{V}$, $e \in \mathcal{E}$, which manually receives a value to initialize an evaluation.*

It is useful to define our use of the term "alternative". Often, when referring to i* models, "alternative" is used to mean choices in a means-ends relationship.

Definition: An Alternative. *Alternative means to an end. For ME: $(e_1, \ldots e_n) \rightarrow e$, each possible combination of setting $(e_1.v, \ldots, e_n.v)$ to values $v_1, \ldots, v_n$, $v_i \in \mathcal{V}$.*

In addition to explicit alternatives, the choice of whether or not to implement or select elements not involved in mean-ends relationships can be made. In order to distinguish between individual alternatives and a selection over multiple elements within a model, we will call the latter a *strategy* after [3]. This refers to a strategy in the domain to be evaluated, and not to a particular way of doing evaluation.

Definition: A Strategy. *A strategy is a set of choices which initialize a particular evaluation. A set, $\mathcal{S}$, of $(e_1, \ldots, e_n)$ such that $e \in \mathcal{E}$ and $\mathcal{S} \subseteq \mathcal{E}$, where each $e_i \in \mathcal{S}$ is given a value $e.v = v \in \mathcal{V}$, where v is not none.*

It is also useful to define exactly what is meant by "an evaluation".

Definition: An Evaluation of a Model. *An i* evaluation starts with a set $\mathcal{S}$, an evaluation strategy, created to reflect an analysis question. These labels are propagated throughout the model using the procedure described in the following sections. Results are analyzed.*

Typically, several evaluations are applied to a single model, each exploring a different strategy. Often changes are made to the model in between evaluations.

4.4 Evaluation Propagation Rules

We define rules in order to facilitate a standard propagation of values given a link type and contributing label. As contribution links in i* are similar to such links in the NFR Framework, we adopt the contribution links propagation rules from this procedure. These rules intuitively reflect the semantics of contribution links.

Propagation: Contribution Rules(l, e_s.v). *For l: $e_s \rightarrow e_d$, given $e_s.v$ and a link $l \in CONT$, a resulting evaluation value, $v \in \mathcal{V}$, is produced as per Table 1.*

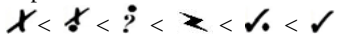
Table 1. Propagation Rules Showing Resulting Labels for *Contribution* Links

Source Label (e_s.v)		***Contribution* Link Type *(l.type)***						
	Name	**Make**	**Help**	**Some+**	**Break**	**Hurt**	**Some-**	**Unkn.**
✓	Satisfied (S)	✓	✓.	✓.	✗	✗.	✗.	?
✓.	Partially Satisfied (PS)	✓.	✓.	✓.	✗.	✗.	✗.	?
↯	Conflict (C)	↯	↯	↯	↯	↯	↯	?
?	Unknown (U)	?	?	?	?	?	?	?
✗.	Partially Denied (PD)	✗.	✗.	✗.	✓.	✓.	✓.	?
✗	Denied (D)	✗	✗.	✗.	✓.	✓.	✓.	?

It is left to define how evaluation values should be propagated through link types that are in i* but not in the NFR framework, namely, *Means-Ends*, *Decomposition*, and *Dependency* links. The nature of a dependency indicates that if the *dependee* is satisfied then the *dependum* will be satisfied, and so will the *depender*.

Propagation: Dependency Rule(e_s, e_d). *For DEP: $e_s \rightarrow e_d$, the result is $e_s.v$.*

In i*, Decomposition links depict the elements necessary to accomplish a task, indicating the use of an AND relationship. AND indicates the selection of the "minimum" value amongst all of the values in the AND relation. Similarly, the Means-Ends link depicts the alternative tasks which are able to satisfy a goal, indicating an OR relationship, taking the maximum values of elements in the relation. To increase flexibility, the OR is interpreted to be inclusive. We expand the order of the values presented in the NFR Framework to allow for partial values, producing:

 (1)

Here the ordering between unknown and conflict is somewhat arbitrary. From these concepts we can define the following functions and rules:

Propagation: Decomposition Rule(e_1.v,…,e_n.v). *For DEC: (e_1.v,…,e_n.v) → e_d, the resulting value is the min $v_i \in (e_1.v,\ldots,e_n.v)$ using (1).*

Propagation: Means-Ends Rule(e_1.v,…,e_n.v). *For ME: (e_1.v,…,e_n.v) → e_d, the resulting value is the max $v_i \in (e_1.v,\ldots,e_n.v)$ using (1).*

Note that the N value is ignored in the above rules, favoring any other value. Only if all v_i =N, $v_i \in (e_1.v,\ldots,e_n.v)$, is the result of min or max N.

Combinations of Links. Models in i* often contain situations where multiple kinds of links affect a single node. This special case complicates an otherwise simple procedure. Hard links, which are Decomposition, Means-Ends and Dependency links, are combined using an AND of the final results of each link type.

Propagation: Mixture of "Hard" Links. *For $l_1,..,l_n$ such that l_i:$e_{si} \rightarrow e_d$, each link shares the same destination, $(l_1,..,l_n)$.type ≠ CONT, then for l_1 mapping to ME or DEC, l_1 = ME/DEC: (e_1.v,…,e_k.v) → e_d, and $l_2,\ldots,l_n$ mapping to DEP, e_d.v is set to Min(Means-Ends Rule/Decomposition Rule(e_1.v, …, e_k.v), for each $l_i \in l_2,\ldots,l_n$, l_i: e_{si} → e_d, Dependency Rule(e_{si}, e_d)). Note that due to syntax restriction a), $l_1,..,l_n$ will only have one type equal to ME or one type equal to DEC.*

If Contribution and Dependency links are mixed, the result of the Dependency links are treated as a Make contribution, and considered in conjunction with the other contributions when the final label is determined.

Resolving Multiple Contributions. Most often, softgoals in i* are the destination of multiple contribution links. We adopt the notion of a "Label Bag" from the NFR Framework. This is needed to store all incoming labels for a particular softgoal. Its content is presented to the user when prompting for human judgment, discussed in the next section. We produce a more precise definition of this concept:

Definition: Label Bag. *Each $e \in \mathcal{E}$ such that e.type = Softgoal contains an additional attribute, expressed as e.lb. We define lb as a set consisting of tuples <v, e_s> where $v \in \mathcal{V}$ is an evaluation label received by e and $e_s \in \mathcal{E}$ is the source of the evaluation label. We keep track of whether the current label bag has been resolved with a human judgment situation with a Boolean e.lb.r.*

Labels in the bag are combined into a single, resulting label, either through automatic rules, as described in Table 2, or by human judgment. The choices in Table 2 allow the evaluator more control than was originally given in the NFR procedure. We define a function which applies the automatic cases in Table 2.

Propagation: Automatic Cases(e.lb). *For e.lb, the label bag, determine if one of the cases described in Table 2 applies, and if so, produce the resulting value. If one of the cases does not apply, then produce a temporary value of N (None).*

Table 2. Cases where Overall Softgoal Labels can be Automatically Determined

Label Bag Contents	Resulting Label
1. The bag has only one label : {<v, e_s>}	the label: v
2. The bag has multiple full labels of the same polarity, and no other labels. Ex: {✓, ✓, ✓} or {✗, ✗}	the full label: ✓or ✗
3. All labels in the bag are of the same polarity, and a full label is present. Ex: {✓•, ✓, ✓•} or {✗, ✗•}	the full label: ✓or ✗
4. The previous human judgment produced ✓ or ✗, and a new contribution is of the same polarity	the full label: ✓or ✗

4.5 Human Judgment in Evaluation

As a result of the inherent incompleteness and abstraction of i* models, human intervention may be needed in order to make some evaluation decisions. Specifically, human judgment is used to decide on an overall label for softgoals in the cases not covered in Table 2. Human judgment may be as simple as promoting partial values to a full value, or may involve the combination of many sources of conflicting evidence. In applying our algorithm, the domain concepts represented by the name of each element are not taken into consideration, except when human judgment is applied. When determining a final label for an element, domain knowledge related to the identities of the destination and source elements should be used.

In the evaluation algorithm, we choose to keep track of the human judgment situations that have occurred, in order to avoid repetition. We store the human judgment situations collected during successive evaluations of a model in a set, $\mathcal{HJ}$.

Definition: Human Judgment Situation. *A human judgment situation is a tuple <e, lb, v>, where $e \in \mathcal{E}$ is the element for which the situation applied, e.type = Softgoal, lb is a copy of the label bag of e at the time the human judgment was made, and $v \in \mathcal{V}$ is the evaluation label result provided by the user.*

4.6 The Evaluation Algorithm

The algorithm adopts the structure outlined in the NFR procedure by including iteration over two steps: propagation and value resolution. In the first step, all present labels are propagated through all outgoing links using the rules described in the previous section. In the second step, the resulting evaluation values for softgoals are determined, using either the automatic cases in Table 2, or human judgment.

Once the values for all elements have been determined in the second step of the algorithm, the cycle starts again. The labels to be propagated are kept track of using a queue of elements to which the labels are assigned, $\mathcal{LQ}$, starting with the initial labels, and adding each final label produced in step 1 and 2. The algorithm will terminate when all labels have been propagated and this queue is empty.

As the procedure allows the placement of initial values, $(e_1.v,\ldots,e_2.v)$, $e \in \mathcal{S}$, on non-leaf nodes, it is necessary to define how these values are affected by subsequent

propagation. In the case of hard elements (non-softgoals), subsequent propagation is ignored, favoring the initial value. In the case of softgoals, initial values are placed in the bag of labels, leaving conflicts between initial and propagated values to human judgment. Element initial values are retained in an attribute $e.i$, $e \in \mathcal{E}$.

Pseudocode describing the evaluation algorithm is shown in Table 3. The procedure for resolving a mix of hard links could be simplified using attributes and data structures for hard elements. Here, we chose a repetitive method of examining all hard links for each incoming value; we leave this to be optimized in the implementation.

4.7 Example Evaluation

To demonstrate the procedure, we provide a simplified example from the Trusted Computing (TC) Case Study [11] in Fig. 1. This model depicts a simplistic view of the TC domain, showing the intentions of the PC User, the PC Product Provider and the Data Pirate. In our example evaluation, we ask: "If the PC User Obtains PC Products from the Data Pirate how does this affect the PC Product Provider's ability to Sell PC Products for Profit?" Initial values are circled and human judgment is annotated in the model.

In this example, when PC Products are Obtained from the Data Pirate, PC Products are Obtained Affordably, but the PC Product Provider does not Sell PC Products for Profit. Further rounds of evaluation and model iteration are needed. Ideally, a solution would be found where the PC Product Provider can make a Profit and the PC User can have Affordable products while Abiding by Licensing Regulations.

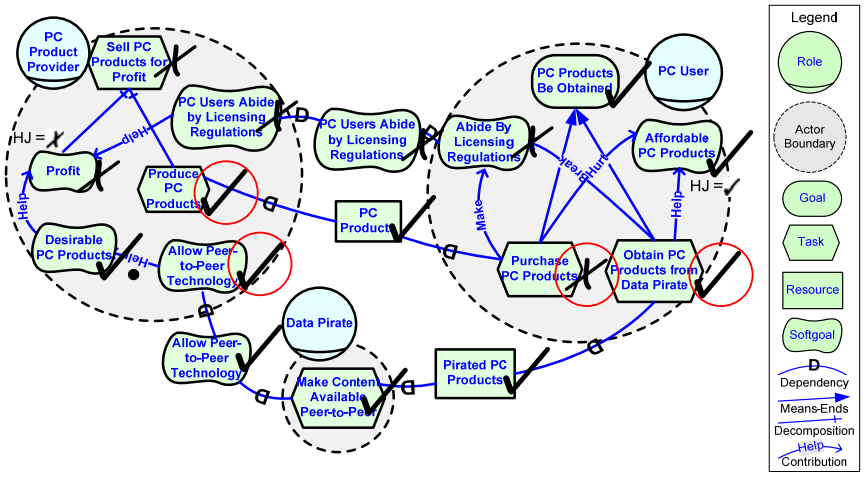


Fig. 1. Simplified TC Example showing Final Evaluation Results

Table 3. Pseudocode of the i* Evaluation Algorithm

```
qualitativeInteractiveEvaluation(E, L, S):
   init(LQ, S);
   While !LQ.empty(){ step1(LQ); step2(LQ); }
init(LQ, S):
   For each e ∈ S
      e.i = e.v; LQ.push(e);
      If e.type == Softgoal { e.LB.addToBag(v, "Initial") }
step1(LQ):
   LQ2 = LQ;
   While !LQ2.empty() {
      e_s = LQ2.pop(); LQ.pop();
      For each l ∈L s.t. l:e_s → e_d {
         Label v = findResultingEvalValue(l, e_s, e_d);
         If e_d.type = Softgoal  { e_d.LB.addToBag(v, e_s)}
         Else if e_d.i == N and v != N { e_d.v = v }
         If e_d ∉ LQ  { LQ.push(e_d)}}}
step2(LQ):
   For each e ∈ E s.t. e.type == Softgoal and e.LB.r == false {
      e.v =AutomaticCases(e.LB)
      If (e.v == N) {
         If <e, e.LB, v> ∈HJ  { e.v = v  }
         Else if <e, e.LB, *> ∉HJ {
            e.v = PromptUser(e.LB) ; HJ.add(<e, e.LB, e.v>); } }
      If e ∉LQ { LQ.push(e)}
      e.LB.r = true }
findResultEvalValue(l, e_s, e_d):
   If l∈CONT {Label v = ContributionRules(l, e_s)}
   If e_d.type == softgoal {Label v = e_s.v }
   Else {Label v = resolveMixofHardLinks(e_d)}
   return v
resolveMixofHardLinks(e_d)
   Label v
   For each l ∈L s.t. l: (e_1,…,e_n) → e_d and l∉CONT {
      v = min(v, resolveSingleHardLinks(l)) }
   return v
resolveSingleHardLinks(l)
   If l == ME
      Label v = Means-Ends Rule(e_1.v,…,e_n.v) where ME: (e_1,…,e_n) → e_d
   If l == DEC
      Label v = Decomposition Rule(e_1.v,…,e_n.v) where DEC: (e_1,…,e_n) → e_d
   If l == DEP { Label v = e_s.v }
   return v
addToBag(v, e):
  { e.LB.remove(<*, e>); e.LB.add(<e.v, e>); e.LB.r = false}
```

4.8 Convergence, Termination, and Correctness

Goal models often contain cycles. To avoid infinite loops, we store a count of each of the combinations of source elements that have been placed in the Label Queue along with their current labels at the time they were added, for example <e_1, PS> may have occurred three times. Once this count has reached a certain fixed number, *n*, the same

source and label combination cannot be placed in the Label Queue again. This allows some fluctuation without infinite loops. In this way, if there are *r* elements in the model, with *6* labels and a cap of *n*, the Label Queue has a maximum lifetime size of *6rn*, and the algorithm must terminate. For simplicity, this is not reflected in Table 3.

Given that the semantic richness of an i* model lies in the informal naming of elements, it is not formally possible to state correctness criteria for the semantic results of the algorithm. Instead, we focus on evaluating the utility of performing the procedure. Regarding correctness of the algorithm in Table 3, we say that the algorithm is correct if all initial evaluation labels are propagated throughout the model according to the propagation rules given. Due to the complexity and detail involved, such a proof is omitted from this work.

5 Benefits of Qualitative, Interactive Evaluation

Benefits of performing qualitative, interactive analysis on goal- and agent-oriented models include both the ability to answer strategic questions and the means to iterate upon and improve the quality of the model and subsequent domain understanding.

Analysis. A significant benefit of i* evaluation is its ability to facilitate analysis by providing answers to strategic questions, and to compare the effectiveness of high-level design strategies. For example, in a case study involving a large social service organization, [10], the evaluation procedure described in this work was applied manually to large models in order to evaluate the situational effectiveness of a variety of technologies, including wikis and discussions forums. It was discovered that the features of a wiki were not effective in satisfying the goals of the organization, while a discussion forum, with a set of specific features, showed more promise.

Analysis can also be used as a means of understanding, justifying and explaining complex situations. Examples of this type can be found in [11], where evaluation is used to describe the motivations behind stakeholders involved in Trusted Computing.

Evaluation and Model Iteration. When evaluation results are unexpected, this can often reveal the presence of missing information or semantic flaws. In the TC case study, although the model appeared to be sufficiently complete, one of the first rounds of analysis of the TC Opponent point of view revealed that PC Users would not buy TC Technology. Although this may be case for some Users, obviously the makers of TC Technology envisioned some way in which users would accept their product. These results lead the modeller to include factors such as product lock-in, more accurately reflecting the domain.

In another example, based on work describing Agents in E-Commerce Environments [15], issues were noticed during the propagation of evaluation values, as shown in Fig. 2, part a). Due to space restrictions we show only small segments of these larger models. First, Rich Consumer Profile depends solely on Produce Consumer Profile, when it seems that just producing a profile alone would not be enough to ensure Richness, and second Accurate Consumer Profile has no effect on any other element. Upon further consideration of the source paper, it was determined that Accurate Consumer Profile has a positive effect on several aspects, including Rich Consumer Profile. Changes to the model are shown in Figure 2, part b).

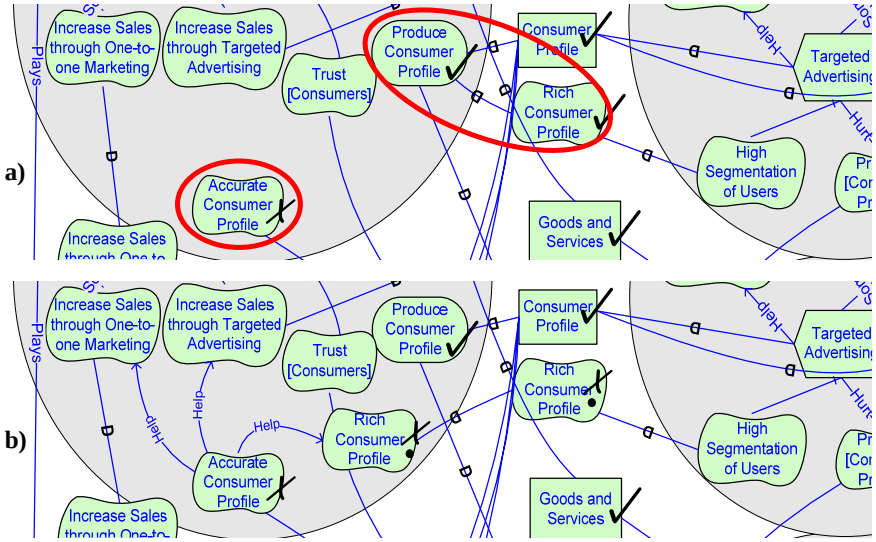


Fig. 2. Excerpts of Model Changes from Agents in E-Commerce Case Study: a) Before changes prompted by evaluation, b) After changes prompted by evaluation.

6 Discussion

When considering the model changes made during evaluation, the argument could be made that if the evaluator is continually adjusting the model to correspond with his/her notion of the domain. As a result, subsequent evaluation will not produce results which bring revelations or new information, only what the modeller already knows. We argue that as the modeller adjusts the model to conform to his/her perception of the domain, the modellers perception of the domain is also altered, and for the better, as the entire interactive process forces the modeller to ask and find answers for specific questions about the relationships between entities. These questions prompt the modeller to return to the system stakeholders, filling in the gaps from previous rounds of elicitation.

In the case studies referred to in this work, the evaluation procedure was applied manually. A version of the algorithm was implemented in the OpenOME tool, an open-source, Eclipse-based, conceptual modeling tool [16]. We are currently working on modifying this implementation to work with the latest version of OpenOME, which makes use of several Eclipse-based frameworks including EMF, GMF, and GEF.

When using qualitative evaluation, it may be useful to apply more fine-grained qualitative labels. For example, weak partially satisfied or strong partially satisfied. In order to retain the simplicity of the procedure, we use only a simple set of values. Additional qualitative values could be appended by adding to the ordering of (1). Future implementations can include the ability to select amongst qualitative scales.

7 Related Work

Several methods also aim to measure the satisfaction of goals. These include evaluation in the NFR Framework, [8], on which this work was based. The NFR method is applied to non-agent goal models, allows less freedom for human judgment, and was intended only for the evaluation of model alternatives. Previous work has used evaluation from the NFR Framework in order evaluate i* models, see [17] for example. Such work assumed that the NFR procedure could be easily extended for use with i*, without describing the necessary extensions or additional benefits.

Giorgini et al have introduced qualitative and quantitative procedures for goal model analysis [2]. This approach separates negative and positive evidence throughout the procedure, is fully automated, and works in a forwards and backwards direction. The most recent definition of GRL [3], a variant of i*, describes a variety of associated evaluation methods which range from quantitative to qualitative, and involve an overall measure of actor satisfaction. This approach uses differing propagation rules, favoring undecided values, and a differing algorithm, propagating in stages based on the type of link, i.e. first through all decomposition links, then through all contribution links, etc.

Methods have been proposed to evaluate the satisfaction of goals in the KAOS Framework [4]. A notion of partial goal satisfaction is introduced via the creation of a probabilistic model, where the satisfaction level of a goal corresponds to the likelihood of its satisfaction, given concrete domain evidence.

All existing qualitative methods which explore goal satisfaction, with the exception of [8], are fully automated, either separating negative or positive evidence or using arbitrary rules to decide values for softgoals, often resulting in the proliferation of conflicts or partial values. We chose to focus on an interactive procedure which gives greater control to the modeller and promotes model iteration.

In quantitative analysis methods, unless the numbers used in the evaluation are derived from measures in the domain, they can be viewed as fine-grained qualitative analysis. There is a danger that users may place an undeserved amount of confidence in the results, as numerical results are often associated with mathematical precision.

The evaluation of metrics over goal- and agent- oriented models has been proposed. For example, in [5], Franch outlines a framework which uses the structure of i* models as a means to measure desired properties such as security, completeness, modifiability and predictability. However, it is difficult to determine if the results of metrics correspond well to real-life phenomena. In addition, the semi-formal nature of i* constructs, and the variance in i* styles, makes it difficult place a high degree of confidence in metrics based on the structure of i* models.

Methods have applied planning and simulation to goal- and agent-oriented models in order to find a satisfactory plan or explore model interactions (for example, [18] and [19]). Methods which check properties over such models have been explored [20]. The detailed information used in these approaches makes them more appropriate for later stage models. Work has tried to improve the quality of goal- and agent- oriented models by introducing methodologies to direct their creation [21], by cross-checking these models with other models [6], [7], or by involving stakeholders in the modeling process [6]. We are not aware of approaches which focus on improving the quality of non-temporal goal models after their initial construction.

8 Conclusions and Future Work

In this work, we have identified the need for a systematic method of performing early, qualitative analysis of Early Requirement Models. We have introduced a relatively simple procedure which builds on the NFR procedure, specifying the algorithm in precise pseudocode, expanding the algorithm to deal with i*-specific constructs, precisely defining the meanings of several evaluation-related terms, exploring issues not covered in [8], and considering convergence, termination, and correctness. We have explored the benefits of such a procedure, including analysis, argumentation, and model iteration and elicitation, using examples from several case studies. Issues brought to light by our exploration have been discussed.

This work can be expanded in several ways, for example, considering the effect of actor boundaries in evaluation, evaluating the satisfaction of actors, as in [3], incorporation textual arguments, similar to work in [22], expanding to "top-down" or backwards analysis, explored in [23], allowing for evaluation constraints as in [2], facilitating the traceability of evidence, giving users selection over different qualitative scales, and supporting views over evaluations, as is done in [24].

Acknowledgements. Financial support has been provided by Bell University Laboratories and the Ontario Graduate Scholarship Program. The KHP project was conducted along with Steve Easterbrook, Jorge Aranda, Yuntian Fan, Marcel Leica, Rifat Abdul Qadir, and Markus Strohmaier. The TC case study was conducted along with Lin Liu. Thanks to Steve, Jorge, Markus, Golnaz Elahi, Sotirios Liaskos, Samer Abdulhadi, and Marsha Chechik for helpful discussions and comments.

References

1. Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In 3rd IEEE International Symposium on Requirements Engineering (RE'97), pp. 226--235. IEEE Press, New York (1997)
2. Giorgini, P., Mylopoulos, J., Sebastiani, R.: Simple and Minimum-Cost Satisfiability for Goal Models. In: Persson, A., Stirna, J. (eds) CAiSE 2004. LNCS, vol. 3084, pp. 20-3-5. Springer, Heidelberg (2004)
3. International Telecommunication Union, Telecommunication Standardization Sector, Study Group 17: Z.151: User Requirements Notation (URN), http://jucmnav.softwareengineering.ca/twiki/bin/view/UCM/DraftZ151Standard (2008)
4. Letier, E., Lamsweerde, A. van: Reasoning about Partial Goal Satisfaction for Requirements and Design Engineering, In: 12th ACM International Symposium on the Foundations of Software Engineering (FSE'04), pp. 53--62. ACM, New York (2004)
5. Franch, X.: On the Quantitative Analysis of Agent-Oriented Models: In Dubois, E., Pohl, K. (eds) CAiSE'06. LNCS, vol. 4001, pp. 495--509. Springer, Heidelberg (2006)
6. Maiden, N. A. M., Jones, S. V., Manning, S., Greenwood, J., Renou, L.: Model-Driven Requirements Engineering: Synchronising Models in an Air Traffic Management Case Study. In: Persson, A., Stirna, J. (eds) CAiSE'04, LNCS, vol. 3084, pp. 367--383, Springer, Heidelberg (2004)

7. Gordijn, J.; Petit, M.; Wieringa, R.: Understanding Business Strategies of Networked Value Constellations Using Goal- and Value Modeling. In: 14th IEEE International Conference on Requirements Engineering (RE'06). pp. 129--138. IEEE Press, New York (2006)
8. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Norwell, MA (2000)
9. Easterbrook, S. M., Yu, E., Aranda, J., Fan, Y., Horkoff, J., Leica, M., Qadir, R. A.: Do Viewpoints Lead to Better Conceptual Models? An Exploratory Case Study. In: 13th IEEE International Requirements Engineering Conference (RE'05). pp. 199--208. IEEE Press, New York (2005)
10. Strohmaier, M., Horkoff, J., Yu, E., Aranda, J., Easterbrook, S.: Can Patterns improve i* Modeling? Two Exploratory Studies. In: Paech, B., Rolland, C. (eds) REFSQ'08. LNCS, vol. 5025, pp. 153--167. Springer, Heidelberg (2008)
11. Horkoff, J., Yu, E., Liu, L.: Analyzing Trust in Technology Strategies. In: International Conference on Privacy, Security and Trust (PST 2006). pp. 21--32. (2006)
12. Strohmaier, M., Yu, E., Horkoff, J., Aranda, J., Easterbrook, S.: Analyzing Knowledge Transfer Effectiveness - An Agent-Oriented Approach. In: 40th Hawaii International Conference on Systems Science (HICSS-40 2007), pp. 188b. IEEE Computer Society, Washington DC (2007)
13. Horkoff, J.: An Evaluation Algorithm for the i* Framework. Master's Thesis, Department of Computer Science, University of Toronto (2006)
14. Horkoff, J., Elahi, G., Abdulhadi, S., Yu. E.: Reflective Analysis of the Syntax and Semantics of the i* Framework. In: Song, I-Y et al. (eds) RIGiM'08. LNCS, vol. 5232, pp. 249--260. Springer, Heidelberg (2008).
15. Spiekermann, S., Dickinson, I., Günther, O., Reynolds, D.: User Agents in E-commerce Environments: Industry vs. Consumer Perspectives on Data Exchange. In: Eder, J., Missikoff, M. (eds) CAiSE'03. LNCS, vol. 2681, pp. 1029. Springer, Heidelberg (2003)
16. OpenOME, an open-source requirements engineering tool, https://se.cs.toronto.edu/trac/ome/wiki
17. Liu, L., Yu, E.: Designing Information Systems in Social Context: A Goal and Scenario Modelling Approach. Information Systems. 29(2), 187--203 (2004)
18. Gans, G., Jarke, M., Lakemeyer, G., Schmitz, D.: Deliberation in a Modeling and Simulation Environment for Inter-organizational Networks. In: Eder J., Miisikoff M. (eds), Proceedings CAiSE'03. LNCS, vol. 2681, pp. 242--257, Springer, Heidelberg (2003)
19. Wang, X., Lesperance, Y.: Agent-oriented requirements engineering using ConGolog and i*. In: Workshop on Agent-Oriented Information Systems (AOIS'01) co-located with CAiSE'01, pp. 59--78 http://www.aois.org/ (2001)
20. Fuxman, A., Pistore, A., Mylopoulos, J., Traverso, P.: Model Checking Early Requirements Specifications in Tropos. In Fifth IEEE International Symposium on Requirements Engineering (RE01). pp. 174-181. IEE Press, New York (2001)
21. Grau, G., Franch, X., Maiden, N.A.M.: PRiM: an i*-based process reengineering method for information systems specification. Information and Software Technology. 50(1-2), 76--100 (2008)
22. Maiden, N.A.M., Lockerbie, J., Randall, D., Jones, S., Bush, D.: Using Satisfaction Arguments to Enhance i* Modelling of an Air Traffic Management System. In: 15th IEEE International Conference on Requirements Engineering (RE'07). pp.49--52. IEEE Press, New York (2007)
23. Horkoff, J., Yu, E.: Qualitative, Interactive, Backward Analysis of i* Models. In: Castro, J., Franch, X., Perini, A., Yu, E. (eds) 3rd International i* Workshop. pp. 43—46, CEUR-WS.org (2008)
24. jUCMNav - Eclipse plugin for the User Requirements Notation, http://jucmnav.softwareengineering.ca/jucmnav/