Detecting Judgment Inconsistencies to Encourage Model Iteration in Interactive i* Analysis

Jennifer Horkoff[1], Eric Yu[2]

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

Abstract. Model analysis procedures which prompt stakeholder interaction and continuous model improvement are especially useful in Early RE elicitation. Previous work has introduced qualitative, interactive forward and backward analysis procedures for i* models. Studies with experienced modelers in complex domains have shown that this type of analysis prompts beneficial iterative revisions on the models. However, studies of novice modelers applying this type of analysis do not show a difference between semi-automatic analysis and ad-hoc analysis (not following any systematic procedure). In this work, we encode knowledge of the modeling syntax (modeling expertise) in the analysis procedure by performing consistency checks using the interactive judgments provided by users. We believe such checks will encourage beneficial model iteration as part of interactive analysis for both experienced and novice i* modelers.

Keywords: Goal-and Agent-Oriented Models, Early Requirements Engineering, Model Analysis, Interactive Analysis, Judgment Consistency.

1 Introduction and Motivation

Modeling and analysis can be challenging in Early Requirements Engineering (RE), where high-level system requirements are discovered. In this stage, hard-to-measure non-functional requirements are critical, and understanding the interactions between systems and stakeholders is a key to system success. Because of the high-level, social nature of Early RE models, it is important to provide procedures which prompt stakeholder involvement (interaction) and model improvement (iteration). To this end, our previous work has introduced interactive, qualitative analysis procedures over agent-goal models (specifically, i* models) which aim to promote model iteration and convergent understanding [1-5]. These procedures are interactive in that, where partial or conflicting analysis labels appear in the model, users are asked to provide a human input as resolution before the procedure proceeds further.

Experiences with skilled i* modelers in complex case studies have provided evidence that interactive analysis prompts further elicitation and beneficial model iteration [1,3]. However, case studies comparing ad-hoc to semi-automated interactive analysis using novice participants showed that model iteration was not necessarily a consequence of systematic interactive analysis, but of careful

examination of the model prompted by analysis in general [6]. We concluded that the positive iterative effects of interactive analysis found in previous case studies were dependent upon modeling expertise (the ability to notice when analysis results were inconsistent with the model), domain expertise (the ability to notice when results differed from the modeler's understanding of the world), and interest in the domain being modeled (caring enough about the modeling process to improve the model).

One consequence of these results would be to recommend that interactive analysis be performed by, or in the presence of, someone with significant knowledge of i*. However, this is often not a reasonable expectation, as many i* modelers may be new to the notation and modeling technique, and will want to be guided by evaluation procedures in analyzing the model. As a result, we aim to embed some modeling expertise into the analysis procedure and corresponding tool support by detecting inconsistencies using the results of interactive judgments.

Case study experiences show that making judgments over the model can lead the modeler to revise the model when the decision made using domain knowledge differs from what is suggested by the model. For instance, in the simple example model for Implement Password System in Fig. 1, if the application Asks for Secret Question but does not Restrict Structure of Password, model analysis would suggest that Usability would be at least partially satisfied. If instead, the modeler thinks that Usability should be partially denied, this means the model is inaccurate or insufficient in some way. Perhaps, for example, Usability also requires hints about permitted password structure.

However, in our student study we found several occasions where novice modelers made judgments that were inconsistent with the structure of the model, and did not use these opportunities to make changes or additions to the model. To place this situation in the context of our previous example, if the Application Asks for Secret Question but does not Restrict Password the student may have decided that Usability was still partially denied, continuing the evaluation without modifying the model to be consistent with their judgment.

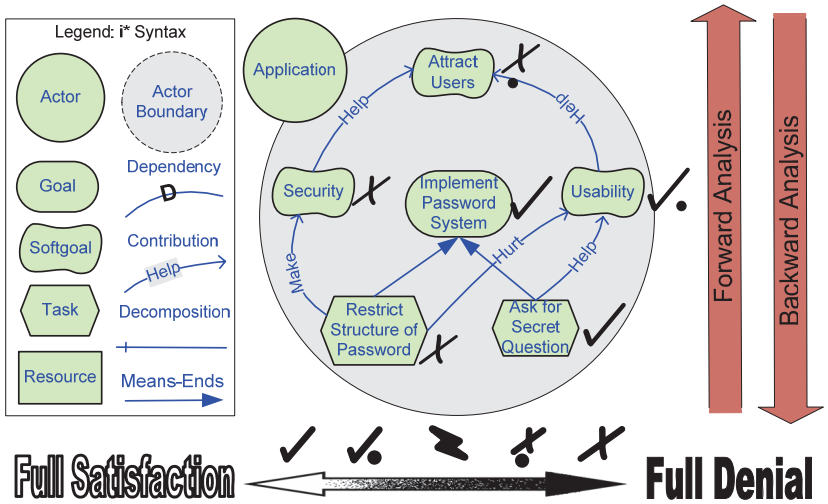

Fig. 1. Simple Example of an i* Model for a Password System from [2,11]

Similarly, our studies and experiences showed that it is easy to forget previous judgments over an intention element and to make new judgments which are inconsistent with previous judgments. For example, a user may decide that if Security is partially denied and Usability is partially satisfied, Attract Users is partially denied. In another round of analysis, if they are presented with an identical situation, they may now decide that Attract Users has a conflict.

We use these observations to guide us in embedding modeling expertise into interactive i* analysis by detecting inconsistencies using judgments. We distinguish and check for two types of inconsistencies: inconsistencies with the structure of the model and inconsistencies with judgment history. In this work, we take the initial steps of describing these checks formally and through examples. Future work will test the practical effectiveness of these checks in encouraging beneficial i* model iteration.

2 Background

We assume the reader is familiar with the i* Framework. The evaluation procedures and their extensions described in this work use the syntax defined in [7]. More information can also be found on the i* Wiki Guidelines [8].

In order to more precisely define the consistency checks introduced in this work, we summarize the formalization of the i* framework presented in [2]. The definitions use the → notation to represent relationships between elements, so if $(i_1,i_2) \in R$ we write this as $R{:}i_1 \rightarrow i_2$.

Definition: i*model. *An i* model is a tuple <I, R, A>, where I is a set of intentions, R is a set of relations between intentions, and A is a set of actors. Each intention maps to one type in {Softgoal, Goal, Task, Resource}. Each relation maps to one type in* $\{R^{me}, R^{dec}, R^{dep}, R^{c}\}$*, means-ends, decomposition, dependency, and contribution links, respectively.*

Analysis labels are used in i* to represent the degree of satisfaction or denial of an intention. We use the formal definition of analysis predicates from [2], adapted from [9]:

Definition: analysis predicates. *We express agent-goal model analysis labels using a set of predicates, V, over* $i \in I$*. Each* $v(i) \in V$ *maps to one of {S(i), PS(i), C(i), U(i), PD(i), D(i)} where S(i)/PS(i) represents full/partial satisfaction, C(i) represents conflict, U(i) represents unknown, and D(i)/PD(i) represents full/partial denial.*

In addition, we have defined a conceptually useful total order where $v_1 > v_2$ implies that v_1 is more desirable (or "higher") than v_2. This order is as follows:

$$S(i) > PS(i) > U(i) > C(i) > PD(i) > D(i) \quad (1)$$

The framework for interactive goal model analysis summarized in [10] currently provides two types of analysis procedures: forward (from alternative solutions to goals) [1,3,4] and backward (from goals to solutions) [2,5]. Generally, the procedures start from initial labels expressing the analysis questions, e.g. what if the Application Restricts Structure of Password and Asks for Secret Question? (forward) or is it possible for Attract Users to be at least partially satisfied? (backward). Propagation is automatic following rules defined in our previous work. Propagation can be described via the forward and backward propagation axioms described in [2]. Generally, for an intention $i \in I$, where i the destination of one to many relationships, $r \in R : i_1 \text{ x } \ldots \text{ x } i_n \rightarrow i$, these predicates take on the form:

Forward Propagation:
(Some combination of $v(i_1) \wedge \ldots \wedge v(i_n), v \in V) \rightarrow v(i)$
Backward Propagation:
$v(i) \rightarrow$ *(Some combination of* $v(i_1) \wedge \ldots \wedge v(i_n), v \in V)$

The interactive nature of the procedures comes when human judgment is needed to resolve incoming partial or conflicting labels (forward) or to provide feasible combinations of incoming labels to produce a target label (backward). New judgments are added to the model formalization by replacing the axioms defined above for an intention with new axioms of the same form, describing the judgment. For example, given S(Restrict Structure of Password) and S(Ask for Secret Question) (both alternatives are satisfied),we decide that Usability has a conflict, C(Usability), we would remove all axioms having Usability as a target or source and add:

Forward: S(Restrict Structure of Password) ∧ S(Ask for Secret Question) → C(Usability)
Backward: C(Usability) → S(Restrict Structure of Password) ∧ S(Ask for Secret Question)

For simplicity, in this work we will refer to the left side of the forward propagation axioms as a *combination of labels, CL*, and the right side as the *individual label, IL*. Forward judgments then consist of $CL \rightarrow IL$ and backward judgments consist of $IL \rightarrow CL$.

3 Detecting Inconsistencies in Interactive Judgments

In this section we define two types of inconsistencies using human judgments.

3.1 Inconsistencies with Model

When considering inconsistencies between a judgment and the model, we compare the contents of the combinations of labels (*CL*) to the individual label (*IL*), looking for inconsistencies. For example, if the combination of labels has no positive labels (S, PS) and the *IL* is positive, we classify this as inconsistent (Case 3). We enumerate the following cases which we define as inconsistent, summarizing each case in after the "//" symbols:

For a judgment CL → IL or IL → CL over i ∈ I:
//there are no unknown labels in the CL, but the IL is unknown
Case 1: *for all $v_j(i_j)$ in CL, $v_j \neq U$ and IL = U(i)*
//there are no negative labels in the CL, but the IL is negative
Case 2: *for all $v_j(i_j)$ in CL, ,$v_j \neq$ PD or D and IL = PD(i) or D(i)*
//there are no positive labels in the CL, but the IL is positive
Case 3: *for all $v_j(i_j)$ in CL, , $v_j \neq$ PS or S and IL = PS(i) or S(i)*
//the CL is all positive or all negative, but the IL is a conflict
Case 4: *for all $v_j(i_j)$ in CL, (v_j = PS or S) or (v_j = PD or D) and IL = C(i)*

In the forward case, the combination of labels can be said to represent evidence from the model, while the individual label is the user judgment. In the backward case, the individual label is the required evidence in the model, while a permissible combination of labels is the user judgment applied to the model structure.

3.2 Inconsistencies with Judgment History

When considering inconsistencies with between old and new judgments over the same intentions, we compare the combination of labels (*CL*) in the new and previous judgments, looking for cases when the combination of labels is the same, is clearly more positive, or more negative, using the ordering of labels from (1). We use this comparison to decide whether the new individual label (*IL*) is consistent with the old individual label. An example of the case is described in Section 1, when the combination of labels is equal, but the individual label is not. In another example, the user decides that with incoming labels of *PS(Security)* and *PD(Usability)*, Attract Users is *C(Attract Users)*. In the next round of evaluation, incoming labels may be *PS(Security)* and *C(Usability)*. The new combination of labels is more positive than the previous, as *C > PD*, so the individual label should not be less than the previous individual label, C, i.e. not *U*, *PD*, or *D*.

To aid in our definition of these cases we will refer to *ILnew* and *Cnew*, the most recent judgment for *i ∈ I*, and *ILprev* and *Cprev* , the previous judgments for *i*. We define psuedocode to check for these types of inconsistencies as follows:

For a judgment CLnew → ILnew (backward: ILnew → CLnew) over i ∈ I:
For each previous judgment CLprev → ILprev over i ∈ I:
//compare labels in previous CLs to labels in new CL
For each $v_j(i_j)$ ∈ CLprev,
For $v_k(i_j)$ ∈ CLnew, compare $v_j(i_j)$ to $v_k(i_j)$
Classify as: >, =, or <
CLnew → ILnew is inconsistent with CLprev → ILprev if:
//The new CL is more positive, but the IL is more negative
All classifications are > or =, and ILnew < ILprev
//The new CL is more negative, but the IL is more positive
All classifications are < or =, and ILnew > ILprev
//The new and old CLs are identical, but the IL has changed
All classification are = (CLprev = CLnew), and ILnew ≠ ILprev

4 Discussion, Conclusions and Future Work

This work reinforces the semantics of i* by embedding rules into the iterative analysis procedures which check for consistency amongst and between user judgments in the model. We have been very flexible and permissive in defining our judgments, only defining cases which are clearly inconsistent. For example, we could include rules to measure when a CL is mostly negative (many more negative labels than positive), and check that the IL is at least partially negative.

Although we have defined inconsistent judgment situations, we have not specified what actions to take when inconsistencies are found. In order to provide flexibility, we do not recommend preventing users from making inconsistent judgments, but instead suggest warning users, either when the judgment is made, or after the fact using a judgment consistency checker. This feature would work similarly to a built-in model syntax checker. Both the judgment consistency and model syntax checks are currently being implemented in the OpenOME tool [12]. The GMF meta-model of the tool has been expanded to include judgment and evaluation alternatives.

As we are aiming for model iteration, future work should adapt these checks to take frequent model changes into account. . Studies involving experienced and new i* users are needed to test the effectiveness of these checks in encouraging model iteration through interactive analysis.

References

[1] J. Horkoff and E. Yu, "Interactive Analysis of Agent-Goal Models in Enterprise Modeling," International Journal of Information System Modeling and Design (IJISMD), vol. 1, 2010, pp. 1-23.

[2] J. Horkoff and E. Yu, "Finding Solutions in Goal Models: An Interactive Backward Reasoning Approach," 29th International Conference on Conceptual Modeling, Springer-Verlag New York Inc, 2010, p. 59.

[3] J. Horkoff and E. Yu, "Evaluating Goal Achievement in Enterprise Modeling – An Interactive Procedure and Experiences," The Practice of Enterprise Modeling, Springer, 2009, pp. 145-160.

[4] J. Horkoff and E. Yu, "A Qualitative, Interactive Evaluation Procedure for Goal- and Agent-Oriented Models," CAiSE 09 Forum, Vol-453, CEUR-WS.org, 2009, pp. 19-24.

[5] J. Horkoff and E. Yu, "Qualitative, Interactive, Backward Analysis of i* Models," 3rd International i* Workshop, CEUR-WS.org, 2008, pp. 4-46.

[6] J. Horkoff and E. Yu, "Interactive Goal Model Analysis Applied - Systematic Procedures versus Ad hoc Analysis," The Practice of Enterprise Modeling, 3rd IFIP WG8.1 (PoEM'10), 2010.

[7] E. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," Proceedings of ISRE 97 3rd IEEE International Symposium on Requirements Engineering, vol. 97, 1997, pp. 226-235.

[8] i* Wiki," http://istarwiki.org", 2010.

[9] P. Giorgini, J. Mylopoulos, and R. Sebastiani, "Goal-oriented requirements analysis and reasoning in the Tropos methodology," Engineering Applications of Artificial Intelligence, vol. 18, 2005, pp. 159-171.

[10] J. Horkoff and E. Yu, "A Framework for Iterative, Interactive Analysis of Agent-Goal Models in Early Requirements Engineering," 4th International i* Workshop, submitted, 2010.

[11] OpenOME, an open-source requirements engineering tool, http://www.cs.toronto.edu/km/openome/", 2010.