Adding Temporal Intention Dynamics to Goal Modeling: A Position Paper

Alicia M. Grubb
University of Toronto, amgrubb@smith.edu

Follow this and additional works at: https://scholarworks.smith.edu/csc_facpubs

Part of the Computer Sciences Commons

Recommended Citation

This Conference Proceeding has been accepted for inclusion in Computer Science: Faculty Publications by an authorized administrator of Smith ScholarWorks. For more information, please contact scholarworks@smith.edu
Adding Temporal Intention Dynamics to Goal Modeling: A Position Paper

Alicia M. Grubb
Department of Computer Science
University of Toronto, Toronto, Canada
amgrubb@cs.toronto.edu

Abstract—Goal models for early phase requirements enable modelers to elicit stakeholders’ intentions, analyze dependencies and select preferred alternatives. Standard analysis techniques provide options for analysis of static goal models but do not consider the dynamic environment that the model represents and do not evaluate the intentions over time. In this position paper, we illustrate that goal model analysis for early phase requirements can be improved by explicitly considering the intention dynamics of a potential system across multiple time scales.

I. INTRODUCTION

Early-phase requirements engineering focuses on understanding the environment in which a software system exists. Goal modeling addresses the needs of stakeholders and end-users by illustrating potential scenarios, and testing phenomena. Given lower-level phenomena (which we will refer to as intentions), goal modeling is able to predict how high-level goals will be impacted by different evaluations of the intentions. Several notations have been developed for modeling goals [1] [2] [3] [4] [5]. In choosing between alternative scenarios, prior work in design-time goal modeling assumes that intention evaluations do not change once assigned, and therefore the analysis is static and unable to answer questions about what happens on different time scales and which alternative is better at a specific point in time. The goal of our project is to understand the impacts of dynamically changing intentions on decision making. Specifically, we enrich goal models with the capability of expressing intentions with dynamically changing evaluations and temporally delayed dependency relationships. Our work focuses on extending existing i* analysis techniques (such as forward analysis [6]) to predict how these changes will affect the satisfaction of high-level goals across multiple time scales. In this paper, we illustrate, by way of example, the need for analysis across multiple time scales, our framework for describing dynamic goal models, and our extension to standard goal modeling techniques.

The remainder of this paper is organized into sections. Section II introduces the motivating example and relevant background. Section III discusses how we introduce dynamic behaviour into i*. Section IV discusses our framework for facilitating the analysis. Section V connects our approach to related work. Finally, we conclude in Section VI.

II. BACKGROUND: FORWARD ANALYSIS & EARLY REQUIREMENTS

We introduce relevant background with an example. Consider a bank looking to enter the loan market (this example was inspired by Asnar et al. [7]). The bank wants to develop a system to accept, assess, and approve loan applications. One key decision in this process is whether to assess the applications in-house or outsource the assessments to a credit agency. Figure 1 shows the partial i* goal model [1] of this decision, consisting of intentions: goals, tasks, and softgoals; and connections between the intentions: means-ends, decomposition, and contribution links (see Legend for representations of intentions and connections). The root goal Assess Application is OR-decomposed (represented by Means-Ends links) into the goals Assessed In-house and Assess by Credit Bureau. Assess by Credit Bureau is then AND-decomposed (represented by Decomposition links) into the Receive Assessments goal and the Receive Quality Assessments softgoal. Assessed In-house is AND-decomposed into

![Fig. 1. Assess loan application sub-model with forward analysis performed, based on the label assignments of leaf-level intentions (in bold).](image-url)

---

1We ignore the cost of each option for simplicity.
the Produce Quality Assessments softgoal and Determine Assessments task, which is further AND-decomposed into the Build Assessment System task. Finally, Understand Monetary Conditions and Audit Assessments Procedures are connected by contribution links (affecting the satisfaction or fulfillment of a softgoal) to Produce Quality Assessments. Makes/Helps contribution links pass the evaluation label from the link source to the link target.

Given our i* model, we can evaluate the tradeoffs within Assess Application’s alternatives using current goal modeling techniques, specifically, forward analysis [6]. Forward analysis allows modelers to assign satisfaction evaluations to the leaf-level nodes (intentions) in a goal model and have child evaluation labels propagate up to give values to their parent node, eventually giving values to root nodes (goals). For example (as shown in Figure 1), if Understand Monetary Conditions, Audit Assessments Procedures, and Build Assessment System are assigned the value Fully Satisfied (or ✓), and Receive Assessments and Receive Quality Assessments are assigned Fully Denied (or ✗), then parents Produce Quality Assessment and Determine Assessments receive the value ✓, and Assess by Credit Bureau receives the value ✗. Next, Assessed In-house receives the value ✓ from its children. Finally, Assess Application receives the value ✓. By following this propagation, the modeler can determine that the bank can satisfy its Assess Application goal by assessing the applications in-house.

Suppose instead that the modelers assigned ✓ to Receive Assessments and Receive Quality Assessments; and assigned ✗ to Build Assessment System. Applying forward analysis to propagate these leaf-level values results in Assess Application obtaining the ✓ label, only this time the satisfaction label propagates from the Assess by Credit Bureau node. From these initial values, the bank should choose to use the credit bureau for their assessments. By trying different scenarios, the modelers can understand the impacts of each tradeoff.

III. DYNAMICALLY CHANGING INTENTIONS: SIMULATION & ANALYSIS

Standard goal modeling assumes that once an evaluation label of an intention is assigned by the modeler, it remains constant; goal modeling only discusses static models. Once a decision between the alternatives is made, it is also assumed to be static. Returning to our example, the bank is likely to determine the satisfaction of leaf-level goals based on the current situation of the bank. However, using only information about the current situation may lead to suboptimal decisions. The above analysis indicated that if the bank has an assessment system built (Build Assessment System labeled ✓), then it should assess the applications in-house; otherwise, the bank should use the Credit Bureau to assess the applications. However, this may seem counterproductive in early-phase requirements, since it is unlikely that a bank, looking to enter the loan market, has a pre-existing assessment system but it might be willing to build one. Given it is apparent that Build Assessment System and other intentions in the model change over time, at what point should the banks decision also change. We look at the dynamics of intentions, goals, and connections both at single points in time and across time periods.

A. Inputs

We first consider the changes involved in modeling intentions as dynamic entities, whose satisfaction level changes over time. Returning to the decision of how to assess loan applications (Figure 1), we define several possible types of dynamic functions that can be applied to intentions in Table I and illustrate the expression of these functions in Figure 2. The Set-Stay-Set Positive (SSS+) function describes intentions that will become ✓, and once they are achieved, they stay achieved. The achievement of SSS+ intentions can mark the end of a process, such as Build Assessment System or Receive Assessments. By making Receive Assessments SSS+, we are denoting that once an agreement is reached with the credit bureau, the credit bureau will continue to provide assessments. In Figure 2, Receive Assessments becomes ✓ at t1 and remains ✓ throughout the simulation. Build Assessment System is also modeled as SSS+ because once the system is built, it will always remain built. The Monotonic Positive (M+) function is similar to SSS+: once the intention reaches the ✓ state, it will stay achieved. M+ differs from SSS+ in that prior to this achievement, the SSS+’s evaluation is stochastic, whereas the M+ intention will never decrease in satisfaction. Audit Assessments Procedures is understood to be M+ because the procedures can be partially implemented and improved over time. In Figure 2, the monotonic progression of Audit Assessments Procedures occurs between t3 and t5.

Although not shown in this example, contribution links pass negative (or opposite) evaluation labels through Breaks/Hurts.

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stochastic (RND)</td>
<td>changes in satisfaction level are non-deterministic or random</td>
</tr>
<tr>
<td>Set-Stay-Set Positive (SSS+)</td>
<td>stochastically changing until a static-state of ✓ is reached</td>
</tr>
<tr>
<td>Set-Stay-Set Negative (SSS−)</td>
<td>stochastically changing until a static-state of ✗ is reached</td>
</tr>
<tr>
<td>Monotonic Positive (M+)</td>
<td>its value will be “more true” or trend toward ✓ as time progresses</td>
</tr>
<tr>
<td>Monotonic Negative (M−)</td>
<td>its value will be “less true” or trend toward ✗ as time progresses</td>
</tr>
<tr>
<td>User Defined</td>
<td>its value is a stepwise function defined by the modeler</td>
</tr>
</tbody>
</table>
When intentions are known to be dynamic but their particular dynamic behaviour is unknown, we model these intentions as *Stochastic* (RND). Understand Monetary Conditions and Receive Quality Assessments can change their values at any time, and are therefore initially modeled as RND. Although we have not illustrated every dynamic function type in the loan example, there are other alternatives (also denoted in Table I). Set-Stay-Set Negative (SSS−) and Monotonic Negative (M−) are the complementary functions to SSS+ and M+: they progress toward and result in the final state of X instead of ✓. Finally, we introduce the *User Defined* function allowing modelers to define and create their own stepwise functions, thus enabling greater flexibility in dynamic functions.

We allow modelers to constrain the order in which intentions can become satisfied. Constraints can involve known relationships between elements, such as “one happens before another” or “one changes faster than another”. For example, in Figure 2, we added the constraint that Receive Assessment must be satisfied before Receive Quality Assessment, which is encoded as Receive Assessment >= Receive Quality Assessment.

Notice that the intention values, in Figure 2, change between X and ✓ at various times (t0...t11). We rely on the modeler to declare timing points of interest. For example, t2 is a notable time point because it is the first time that Assess Application is assigned a value of ✓. These time points mark the boundaries of epochs to reason over and enable us to answer questions such as: “what is happening now?” “what will happen soon?”, and “what will happen in the distant future?”.

Next, we extend the standard i* notation to enable propagation delays across links and introduce an augmented label. When the satisfaction label of a child node changes, there may be a delay in this change propagating to the parent node (indicated by a “*” on the label). For example, a Makes* contribution link uses the same propagation rules as its Makes counterpart but with the added possibility of a delay. Referring to Figure 1, if Understand Monetary Conditions had a Makes* relationship with Produce Quality Assessments instead of Makes, then when the Understand Monetary Conditions value changes, the value of Produce Quality Assessments would not necessarily change immediately. In Figure 2, this delay in propagation is shown in the value of Produce Quality Assessments between t3-t5 and t8-t9. Similarly, Build Assessment System is connected with Determine Assessments by a Decomposition* link instead of Decomposition. The propagation of the changes in Build Assessment System at t3 and t7 are delayed in Determine Assessments.

### B. Analysis

Given our i* model extended with dynamic intentions and relationships described above, we can temporally extend i* analysis over both points and functions. In this section we only discuss forward analysis, which reasons about goals given intentions as inputs, but the same extension can be applied to backward analysis [8], which uses goals to reason about intentions. We show results of both simulation and analysis for our loan example diagrammatically in Figure 2. The top five items are the leaf-level intentions (or inputs) which are propagated to the root-goal (or output). The low, middle, and high signals give the values X, Unknown, and ✓, respectively.

We can simulate goal model satisfaction by varying leaf-level intentions and propagating their values to goals. For example, we illustrate two points of...
We require the modeler to make the dynamic behaviour of goals in the system explicit. Our analysis takes i* models (including initial satisfaction levels), such as the one we described in the previous sections. In addition to the standard i* model, modelers have the option of specifying a dynamic function type (defined in Table I) for each intention as well as relationship dynamics and timing constraints. To facilitate user interactions with our simulator, we created a simple constraint language to articulate points of interest (or breakpoints) within a trace. A breakpoint can be specified as when an intention reaches a declared satisfaction level, or after a discrete number of simulation steps.

**B. Analysis**

In our Model Simulation, we simulate the satisfaction of dynamic intentions with and without delayed relationships...
for a given goal model over a series of time steps. Our simulation uses polling to check for changes in the model and uses extended forward analysis over points at each step to update the model. The **Model Simulation** outputs the **Timing Visualization** as both a text file (shown in Figure 4) and a signal diagram (presented in Figure 2) for a given single short-term and long-term path. The simulation output (see Figure 4) prints the information presented in the \( i^* \) input file (the intentions, goals, and their respective types and initial values). Each line of the simulation presents the intentions/goals vertically and the current satisfaction level horizontally (represented by the numbers 0 (\( X \)), 1 (Partially Denied), 2 (Unknown), 3 (Partially Satisfied) and 4 (\( \checkmark \))). The simulation stops and prints whenever a constraint or breakpoint is triggered. Finally, the simulation ends with short-term and long-term recommendations for any tradeoffs in the model.

The **Static Analysis** module consists of extensions to standard \( i^* \) analysis techniques across functions: forward analysis as described above, and backward analysis. We will develop constraints for the permutations of the dynamic function types and temporal relationships discussed above, to allow for the propagation of functions. We will connect our analysis with an SMT solver [9] to compute the resulting functions for a given model. If all the constraints can be satisfied, our Static Analysis will output the Function Computation for the goals (in forward analysis) and the intentions (in backward analysis), otherwise it will indicate that the model is over-constrained.

### C. Status

At this time, we have built an initial **Model Simulation**. We have completed the extended forward analysis over points and have incorporated some of the breakpoint analysis. We have not implemented extended backward analysis over points. Our simulator is solely text based at this time. We hope to implement a graphical interface soon. Our next step is to implement the Static Analysis initially for extended forward analysis over functions with the goal of determining the root-goal functions if the model is valid.

### V. Related Work

We are not the first to investigate the temporal nature of goals. Regev and Wegmann [10] discuss how goals change in their achievement over time and classify goals into types (i.e., achievement goals, maintenance goals, softgoals, beliefs, and constraints) for the purpose of developing heuristics to improve goal identification. Achievement goals map to our \( SSS^+ \) dynamic function and constraints map to \( SSS^- \). Other goal types could be represented as \( RND \) or **User Defined**.

In addition to Regev and Wegmann’s goal types, Asnar et al. [7] proposed a framework for understanding and mitigating the risks of events that affect goal satisfaction when selecting between alternative designs (within the Tropos framework). Using KAOS, Letier et al. [11] considered goals that are real-time timing properties of a system and developed a formal approach to construct software specifications from these goals.

They defined the goals in temporal logic and connected timed event-based models [12] with state-based formalisms. In our work, we look at the evolution of goals in the model rather than looking at concrete timing requirements, or concrete events within the model. The propagation delays in our system are not explicit measurable time delays; they are left unspecified to allow for uncertainty in the model. We use a simple constraint language rather than temporal logic to reduce the complexity of our static analysis and simulation. Cheong and Winikoff [13] focused on connecting time dependent goals within a decomposition (denoted by precondition links), allowing for the coordination and achievement of agents’ plans. Gans et al. [14] add temporal ordering to \( i^* \) models inorder to simulate goal alternatives. We use relative or ordered dependancy constraints specified by the modeler, which is similar to Cheong and Winikoff.

Goals as dynamic entities have been used outside the requirements phase for runtime analysis [15] [16], self-adaptive systems [17], and context-aware systems [18]. Dalpiaz et al. distinguished between design-time goal models and runtime goal models [19]. Throughout our simulation and analysis,
our models remain design-time goal models at the class level. Robinson et al. [15] convert goal models into components that monitor states in the system through runtime monitoring. Baresi et al. [17] add adaptive goals to goal models in monitoring the system at runtime. These do not monitor how the requirements in the model change, but rather monitor properties of the system.

In terms of how propagation impacts goals, we have discussed using forward analysis (a bottom-up approach) with qualitative evaluation. Others have developed and used complementary approaches [2] [4] [20]. Our analysis techniques could be adapted for quantitative analysis (as in [3] [4] [5]) or hybrid analysis (as in [20]). We have yet to investigate applying our technique to backward analysis (a top-down approach) [4] [8], but see this as useful and complementary to forward analysis.

VI. CONCLUSIONS & FUTURE WORK

In this position paper, we introduced dynamic intentions for goal models as well as dynamic function types with the goal of modeling alternatives on multiple time scales. We believe that goal model analysis for early phase requirements can be improved by explicitly considering these dynamics. We presented a framework for doing both static analysis and simulation of dynamic intentions and explored the technique through an example of a bank entering the loan market deciding whether to assess loan applications in-house or outsource the assessments to a credit agency.

In future work, we will complete our simulator for analysis across points and implement the Static Analysis component. Once we have validated our methodology on the qualitative analysis of i*, we will then investigate how it can be incorporated into quantitative and hybrid analysis techniques presented in other goal modeling languages.

VII. ACKNOWLEDGEMENTS

We would like to thank the Software Engineering Modeling Group at the University of Toronto as well as Marsha Chechik and Fabiano Dalpiaz for their contributions to this work.

REFERENCES