12-2-2016

Looking into the Crystal Ball: Requirements Evolution over Time

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

Marsha Chechik
University of Toronto

Looking into the Crystal Ball: Requirements Evolution over Time

Alicia M. Grubb, Marsha Chechik
Department of Computer Science
University of Toronto, Toronto, Canada
{amgrubb, chechik}@cs.toronto.edu

***Abstract*—Goal modeling has long been used in the literature to model and reason about system requirements, constraints within the domain and environment, and stakeholders' goals. Goal model analysis helps stakeholders answer 'what if' questions enabling them to make tradeoff decisions about their project requirements. However, questions concerning the evolution over time of stakeholder requirements or changes in actor intentionality are not explicitly addressed by current approaches. In this paper, we tackle this problem by presenting a method for specifying changes in intentions over time, and a technique that uses simulation for asking a variety of 'what if' questions about such models. Using the development of a web-based modeling tool as an example, we demonstrate that this technique is effective for debugging goal models and answering stakeholder questions.**

I. Introduction

In early-phase requirements engineering (RE), goal modeling helps stakeholders understand and evaluate potential project scenarios, and the system's interactions with its surrounding environment. Specifically, goal modeling elicits and connects stakeholders' *intentions* and social needs with technical requirements. Goal modeling has long been used in the literature to model and reason about system requirements, constraints within the domain and environment, and stakeholders' goals. Within the goal-oriented requirements engineering community, different approaches and notations have been developed [1]–[5]. These approaches differ in their purpose and application, for example, [1] and [2] deal with only qualitative information, while the rest can also evaluate quantitative information. [4] and [5] do not support dependencies between actors, but have formal semantics. [3] and [4] have automated analysis procedures. [6] gives a more comprehensive comparison of these approaches.

In early-phase RE, these approaches do not explicitly model the evolution of goal evaluations over time. When planning for changing project scenarios, stakeholders can, in principle, reason about time in an ad-hoc manner, by considering the state of the model before or after the change. But this is insufficient to understand trends over the model. Thus, goal models without an explicit modeling of time may lead to incorrect results because they do not account for variations in intentions' satisfaction. It is important to provide analysis to allow stakeholders to consider alternatives over time.

Motivating Scenario: Waste Management Example (WME). Consider a city evaluating its waste management infrastructure. The city currently has a landfill (dump) that has not reached capacity and is considering investing in building a new one as well as a recycling and composting facility. Fig. 1[1] shows the partial Strategic Rationale iStar goal model [1] of the City and its Citizens, consisting of their intentional elements (intentions): goals, tasks, resources, and soft-goals; and connections between their intentions: dependency, decomposition, and contribution links (see Legend for representations of intentions and connections). The city wants to satisfy Manage City Waste (MCW), Comply with Standards, Reduce Operating Costs (ROC), Have Workers Union Contract, and for their citizens to Enjoy City (HC) (i.e., their *root-level* goals).

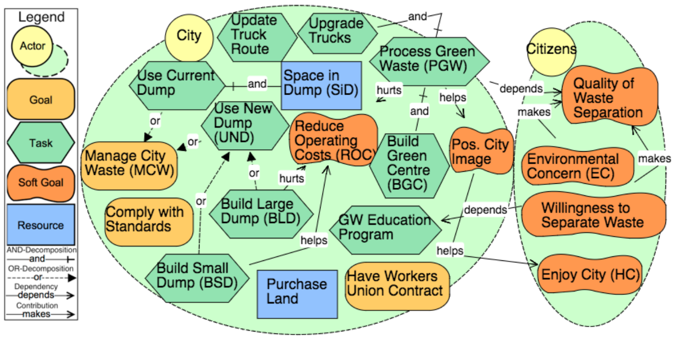


Fig. 1: Early RE model for the Waste Management Example.

Several elements in this model change over time. For example, the Willingness to Separate Waste will become more satisfied over time, but Environmental Concern (EC) will vary substantially depending on external factors. As the dump gets used, the Space in Dump (SiD) will become less satisfied, and if the city decides to build a new dump at some point, it will need to Purchase Land either near the city where prices are high or far away from the city, where prices are lower, or both. Buying land farther away from the city results in increased garbage transportation costs.

The city wants to understand how these goals changing over time impact its decision making, by exploring the following seven questions: (EQ1) Is it feasible to first Build Green Centre (BGC) and then build a dump (Build Large Dump (BLD) or Build Small Dump (BSD))? (EQ2) Does the order of these developments (Process Green Waste (PGW) and Use New Dump (UND)) matter? (EQ3) Which possible scenarios always satisfy MCW even if SiD becomes denied in the future? (EQ4)

[1]Dependums have been removed from the model to save space.

How do changes in EC affect the city's root-level goals over time? (EQ5) Can all root-level goals eventually be satisfied? (EQ6) Given the initial results from answering EQ1-EQ5, is there another way to answer these questions?

Current Approaches. Many techniques for analyzing goal models already exist [7]. Generally, approaches either determine the evaluation of root-level intentions given evaluation assignments to leaf-level intentions, called *forward analysis*, or determine possible evaluations of leaf-level intentions given evaluation assignments of root-level intentions, called *backwards analysis*. Variations of forward and backwards analysis have been created for interactive qualitative analysis [7], and automated qualitative and quantitative analysis [4], [8].

Current goal model analysis approaches can answer some of the City's questions to a limited extent. For example, we could use forward analysis to answer EQ1 (and similarly EQ2) by creating an initial model before anything is built and additional models for each combination of BGC, BSD, and BLD, but manual comparison of these models is fraught with errors and does not account for the elapsed time between models. Forward analysis could also be used for EQ4 by exhaustively propagating evaluation labels, but again does not say how the individual labels connect, and comparison between propagated models is difficult. Backwards analysis would partially answer EQ3, by providing possible solutions with and without the denial of SiD, and EQ5 could also be answered by setting all root-level goals to be satisfied and finding values for the remaining intentions using backwards analysis. However, backwards analysis would not be able to inform stakeholders how (or in what order) to satisfy these goals. In each of these and other cases, standard techniques cannot explicitly answer the city's questions and cannot provide a path to achieve their satisfaction values. Furthermore, no current approaches explicitly allow the exclusion of multiple previous solutions (EQ6), which is vital in grasping the full range of tradeoffs.

Strategies for Analysis. We believe goal model analysis can be improved by allowing stakeholders to consider alternatives over time through path-based strategies. Based on the time-based questions we have encountered, we consider three simulation strategies: (AR1) create a random path given initial states in the model; (AR2) create a path given desired properties of the intermediate state (with optional properties over the initial or final state); (AR3) create a path which is different than the previously seen path over the same constraints.

In order to understand the effect of building a green centre and then building a dump (EQ1), we need an exploration of the model, handled by AR1. We can also use AR1 to explore how changes in EC affect the city's root-level goals (EQ4). But in order to find simulations that always result in the future satisfaction of MCW (EQ3), we need to find a path given a desired final state, which is the approach of AR2. Similarly, we use AR2 when we want to find a path where all root-level goals are *FS* in the final state (EQ5). AR2 can also be used for intermediate states to test if the ordering of events matters (EQ2). Finally, if we have an initial result and want to know if there is another answer to our questions (EQ6), then we need a strategy that takes previous results into account; this is handled by AR3. We discuss these mappings further in Sect. IV.

Contributions and Organization. We contribute terminology that enables stakeholders to ask questions about model dynamics over time, and strategies that enable answering time-based questions. By developing a simulator that implements these strategies, we demonstrate the effectiveness and scalability of our approach by evaluating it with a variety of goal models.

The remainder of this paper is organized as follows. Sect. II introduces relevant goal modeling background. Sect. III explores representation of dynamic intentions and relationships. Sect. IV introduces our simulation strategies to answer stakeholder questions. Sect. V reports on the effectiveness and scalability of our approach. Sect. VI connects our approach to related work. We conclude in Sect. VII.

II. Background

We introduce relevant goal modeling background. An *iStar model* is a tuple $M = < I, R, A, V >$, where I is a set of intentions, R is a set of relations between intentions, and A is a set of actors [9], [10]. V is a set of initial qualitative evaluation labels. Intentions (i.e., goals, tasks, resources, and soft-goals) are the elements in the model that can be evaluated using the qualitative evaluation labels *Fully Satisfied* (*FS*), *Partially Satisfied* (*PS*), *Partially Denied* (*PD*), *Fully Denied* (*FD*), *Conflict* (*CF*), *Unknown* (*U*), and *None* (*N*) in the absence of other labels. Evaluation labels can be assigned by modelers (i.e., in the set V) and can be propagated throughout the model via the relations between intentions (i.e., links). Propagation occurs through either forward analysis or backwards analysis (discussed in Sect. I).

Decomposition links separate intentions into sub-components. A single intention can be decomposed with either *AND-Decomposition* or *OR-Decomposition* links but not both. In propagation, *AND-Decomposition* propagates the minimum values of the sub-components, and *OR-Decomposition* propagates the maximum values. Dependency links connect intentions between actors and propagate the minimum satisfaction value from the dependee to the depender. Contribution links propagate values to soft-goals. There are four types of contribution links: *Makes* (*Breaks*) propagates sufficient evidence in support (against) of a soft-goal and *Helps* (*Hurts*) propagates insufficient evidence in support (against) of a soft-goal.

Actors can have relationships between them based on their type (i.e., Role or Agent). While we do not use actor types or relationships in our analysis, we do present them for clarity in visual models.

III. Expressing Dynamics in Goal Models

In order to apply simulation strategies, we must first identify how dynamism occurs in goal models and develop a way of specifying it. We illustrate this process using the Waste Management Example (WME).

TABLE I: Definitions of Dynamic Function Types for Intentions and Relationships

Elementary Functions	
Constant (*C*)	the satisfaction evaluation remains constant at *constantValue*
Increase (*I*)	changes in satisfaction evaluation become "more true" to a *maxValue* as time progresses
Decrease (*D*)	changes in satisfaction evaluation become "less true" to a *minValue* as time progresses
Stochastic (*R*)	changes in satisfaction evaluation are stochastic or random
General Compound Function	
User-Defined (*UD*)	its value is a stepwise function defined by a sequence of other functions, repeating behaviour can be specified over a subset of the function
Common Compound Functions	
Satisfied-Denied (*SD*)	the satisfaction evaluation remains *FS* until t_i and then remains *FD*
Denied-Satisfied (*DS*)	the satisfaction evaluation remains *FD* until t_i and then remains *FS*
Stochastic-Constant (*RC*)	changes in satisfaction evaluation are stochastic or random until t_i and then remains constant at *constantValue*
Constant-Stochastic (*CR*)	the satisfaction evaluation remains constant at *constantValue* until t_i and then changes in evaluation are stochastic or random
Monotonic Positive (*MP*)	changes in satisfaction evaluation become "more true" to a *maxValue* at t_i and then remains constant at *constantValue*
Monotonic Negative (*MN*)	changes in satisfaction evaluation become "less true" to a *minValue* at t_i and then remains constant at *constantValue*

Relationship Dynamics	
Multi-Relationship	the link between *source* and *destination* is *relationshipA* until t_i, at which point it becomes *relationshipB*

A. Definition of Epoch and Epoch Boundary

For the purpose of defining dynamic functions, we discretize time into intervals, called *Epochs*, defined as the period from the start time t_{start} (inclusive) to the end time t_{end} (exclusive). An *Epoch Boundary* (EB) is an identifiable time point (i.e., t_{end}) where there is a transition between epochs. For example, t_{BGC} defines the time at which Build Green Centre (BGC) is assigned a *FS* satisfaction label for the first time. When creating dynamic functions, we reserve t_0 to denote the start of the function (initial evaluation) and t_∞ to denote the end of the function (final evaluation).

B. Defining Dynamic Functions for Intentions

Here we introduce *dynamic function types* for intentions – ways of describing changes in the evaluation of intentions over time (see Tbl. I). A preliminary version of some of these has been presented in [11].

Elementary Functions. Between two consecutive EBs, a value of the intention can become more true, or *Increase* (*I*), become more false, or *Decrease* (*D*), remain *Constant* (*C*), or change randomly, displaying a *Stochastic* (*R*) pattern. For example, UCD remains constant from the initial time point t_0

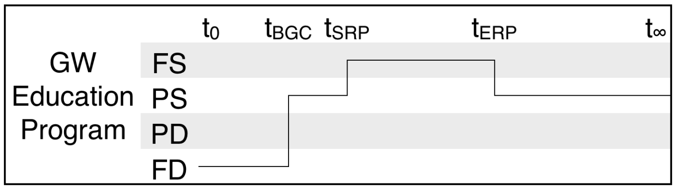


Fig. 2: An illustration of the *UD* function type for GW Education Program.

to t_{UCD} when a decision is made not to use the dump. Yet EC's state is not known over any Epoch, so we model it as *R*.

General Compound Function. We enable modelers to create compound functions by defining an ordered list of elementary functions using the *User-Defined* (*UD*) function type. For example, BGC is *C* with the value *FD* from t_0 until t_{BGC} and then *C* with the value *FS* until t_∞. For GW Education Program, the function (illustrated in Fig. 2) is *C* with the value *FD* from t_0 until t_{BGC}, *C* with the value *PS* until t_{SRP}, *C* with the value *FS* until t_{ERP}, and *C* with the value *PS* until t_∞.

Functions with Repeating Behaviour. When creating a *UD* function, modelers have the option to specify repeating behaviour over the evaluation of an intention. Suppose a resource is available (*FS*) for an epoch, then not available (*FD*) for another epoch, then again available (*FS*) and so on. Have Workers Union Contract follows this pattern. Currently, we do not allow users to specify the duration or the number of iterations for a repeating interval, but we do allow them to specify EBs to start and stop the repeating behaviour. This gives modelers the option of specifying non-repeating behaviour before and after a single repeating segment by specifying the EBs of the repeating segments. If t_0 and t_∞ are given as the repeating segment, it is assumed that the behaviour repeats for the entirety of the simulation. The union contract could be specified as *C* with *FS* for t_0 to t_{R} and *C* with *FD* for t_{R} to t_∞, with t_0 and t_∞ as the repeating EBs. Modelers can use repeating functions to understand cyclic behaviour, such as changes in the season and quarterly business cycles.

Common Compound Functions. Finally, we can define functions that capture common patterns in goal models. For example, the pattern for BGC discussed above is common, and we define it as *Denied-Satisfied* (*DS*). Another common pattern is *Stochastic-Constant* (*RC*) where the value is unknown (*R*) until an EB and then it is known with a user-specified constant value (*C*). When the evaluation value of an intention increases in satisfaction until it reaches its maximum value and then remains constant at that value, we define this as *Monotonic Positive* (*MP*). See Tbl. I for a full list of common compound functions. We enhanced these functions (over the presentation in [11]) to enable stakeholders to declare the min, max, or constant evaluation label within a function. For example, we can define an *MP* function with the maximum value *PS*.

Completeness of Function Types. Our set of function types is not complete. For example, a stakeholder may want to describe a function with a repeating function until an EB, followed by an epoch with non-repeating behaviour and then a second unique repeating function. However, we have not yet encountered such phenomena. We expect the usability of function types to be improved as new common compound function types are defined through case studies.

C. Other Definitions and Process

Defining Relationship Dynamics and Model Evolution. We understand relationship dynamics as changing the structure of the relationship over time. We allow modelers to specify this multi-relationship by extending iStar link types as specified in Tbl. I and adding the *No-Link* type to indicate that no relationship exists across this link. However, in implementing these new relationships, we restrict the relationship combinations to valid iStar relationships. For example, all relationship types (i.e., *Dependency*, *AND-Decomposition*, *OR-Decomposition*, and *Contribution Links*) can be combined with the *No-Link* type, but only *Contribution Links* (i.e., *Makes*, *Helps*, *Hurts*, *Breaks*) can be combined with each other.

We indirectly represent model evolution where goals are removed from the model and/or new goals appear in the model through dynamic functions and relationships. For example, if we want the goal Comply with Standards to appear at t_{CWS}, we add the element to the original model and give it a function type (such as *CR*) where the initial epoch is a constant evaluation of N until t_{CWS}. We also add that Comply with Standards to MCW has no relationship (*No-Link* type) until t_{CWS} and a *Dependency* relationship after. Thus, we use the N evaluation and the *No-Link* relationship to represent intentions not present over specific epochs.

Defining Constraints between Intentions. In order to achieve more realistic model evolutions, we allow users to specify an ordering over symbolic constants associated with EBs. We know that Update Truck Route is completed (*FS*) before either the UND or the PGW tasks are *FS*, but neither depend on an updated route. Given that Update Truck Route is modeled as *RC* with the value *FS* at t_{UTR}, and UND is modeled as *RC* with the value *FS* at t_{UND}, we can add the constraint that t_{UTR} occurs before t_{UND}. Similarly, we can say that two EBs occur at the same time. Symbolic constraints can be ordered using operators from the set $\{=, <>, <, <=\}$, enabling stakeholders to express constraints in chronological order from left to right. These constraints should not be used to specify dependency or decomposition relationships. Their purpose is to communicate correlations between timed events for the purpose of simulation.

Defining Queries between Intentions. Not all interactions between intentions are known in early-phase requirements. We enable modelers to construct queries over constrains in the model using the ? symbol. For example, to answer EQ2 (see Sect. I), the modeler wants to know if the order between PGW and UND matters and if satisfying solutions exist for each order. If PGW is *FS* at t_{PGW} and UND is *FS* at t_{UND}, then we ask (t_{PGW} ? t_{UND}).

Elicitation Process. We elicit these dynamics through meetings with stakeholders. Once initial actors and dependencies have been established, we help stakeholders discover dynamicity and expand their model by asking four questions: (1) Which evaluation labels are currently correct (matching the real world values) and which are not? (2) For those currently correct, are there any guarantees that they will stay static? (3) Identify which elements in this model change over time? (4) How do each of these elements change?

IV. Analysis Process

In this section, we describe the process of analyzing goal models with dynamics.

A. Path Definition

In order to define the notion of a path, we extend the definition of a goal model discussed in Sect. II, such that each intention $i \in I$ also maps to one Dynamic Function Type in the set, $I \mapsto$ *DynamicFunction*, where *DynamicFunction* = {*I*, *D*, *R*, *C*, *UD*, *SD*, *DS*, *RC*, *CR*, *MP*, *MN* } (see Sect. III-B). See the supplemental material online[2] for a discussion of the iStar meta-model extension. We define *an evaluation state* $w(t)$ as an evaluation of a goal model M at time t.

A *path* is a sequence of evaluation states $w(t_i)$ for an ordered sequence of times $(t_1, t_2, t_3, \ldots)$ with transitions between states. We compute paths by selecting times and evaluations of the model at those times. Times are selected randomly while respecting declared constraints within the model's dynamic functions.

B. Encoding

We encode an iStar goal model into Constraint Solving Problem (CSP) as a series of constraints over an array of IntVar[] for each intention. Each intention has an IntVar for each time step of the form `N`<*IntentionID*>_<*TimeStep*>. Intention evaluations are mapped to integers: *FD*-0, *PD*-1, *PS*-2, *FS*-3, etc. having the range {0..6}. Epoch boundaries (`E`<*EpochID*>) have the number of time steps as their range. The initial state is specified for all intentions, and the dynamic functions are encoded over the intention set with identified EBs for transitions between elementary functions. Then the model structure, i.e., links between intentions, is added. Finally, constraints between EBs are added, specifying a partial order over the EBs. For example, three encoding fragments of the WME are shown in Fig. 3. Fragment (1) shows the initial state encoding for Use Current Dump (`N0001`). It is assigned the value *FS*-3 for time step 0. To create a *DS* function (shown as fragment (2)), an EB is created (`E0011`) and a constraint is added for each time step (in this case Build Green Centre `N0011` for time step 1). The constraint says that if the EB is greater than or equal to the current time step, then the intention is assigned *FD*-0, else the intention is assigned *FS*-3. A constraint between two EBs is directly mapped into CSP. Fragment (3) says that Update Truck Route's `E0019` is constrained to be less than Use New Dump's `E0002`. The complete encoding of the WME is described online[2].

[2] http://www.cs.toronto.edu/~amgrubb/archive/RE16-Supplement

```
(1) XeqC3:XeqC (N0001_0::{0..6}, 3)

(2) IfThenElse2:IfThenElse (
 XgteqC13:XgteqC (E0011::{0..10}, 1),
 XeqC50:XeqC (N0011_1::{0..6}, 0),
 XeqC51:XeqC (N0011_1::{0..6}, 3))

(3) XltY2:XltY (E0019::{0..10}, E0002::{0..10})
```

Fig. 3: Segment of CSP encoding from WME.

Once we have encoded the model, CSP determines a total order over the EBs and assigns values to every intention evaluation at every time point. We do not encode any actor information in CSP as it is not needed for the analysis. We allow the modeler to configure CSP to either pick the evaluation labels for each intention randomly from *AnalysisLabels* or select the maximum valid value based on the ordering ($FS > PS > PD > FD > CF > U > N$).

C. Strategies

Given our definition of a path, we discuss the implementation of our strategies below. We show how to use the strategies to answer questions asked in the motivating example.

1) AR1 – A random path given initial states: The first strategy is to create a random path given initial states in the model, with the goal of giving stakeholders a sense of model evolution over time. Modelers see one possible outcome of their model and can try different dynamic functions for leaf-level goals. By running multiple simulations, modelers can see possible scenarios that may result, as well as experiment with multiple dynamic function types to better understand how the goal will become satisfied.

This strategy creates a path starting from the initial state. Given a state, we calculate the next state by randomly selecting intentions to update, given their dynamic functions, and then check the resulting values against the model constraints, to ensure that the path is valid. Finally, we propagate the values throughout the goal model. If no values in the model can be further updated, or we have reached the requested number of simulation steps, we complete and return the calculated path.

We can use AR1 to answer the question EQ1 (Is it feasible to first BGC and then build a dump BSD/BLD?), by initializing values and adding the constraints $t_{\text{BGW}} < t_{\text{BLD}}$ and $t_{\text{BGW}} < t_{\text{BSD}}$. The model already had $t_{\text{UTR}} < t_{\text{UND}}$ and $t_{\text{UTR}} < t_{\text{PGW}}$ as constraints. The strategy generates a path where BGC happens before BSD or BLD. This answer shows that it is feasible to first BGC and it results in the *FS* of the root-level goals with the exception of EC which becomes *PS*. We also answer EQ4 (How do changes in EC affect the city's root-level goals over time?) by initializing values for the leaf nodes and following the strategy. EC affects the city's root-level goals ROC and HC through PGW. Stakeholders can see in the resulting path that when EC becomes *FD*, QWS is affected. Since WSW also has a *Makes* relationship with QWS, QWS becomes *FD* or *U* affecting ROC and HC through propagation. It does not affect the other root-level goals, thus answering EQ4.

2) AR2 – A path given desired properties of the intermediate state(s): The second strategy is to create a path given desired properties of the intermediate state(s) (with optional properties over the initial or final state). This strategy allows stakeholders to set desired goals and see paths that satisfy those goals, if they exist.

In addition to model M, a mapping from intentions to dynamic functions, a list of constraints between intentions, the initial evaluation state, the maximum number of simulation steps (*maxTime*), and the maximum value of an EB (*maxEB*), this strategy takes selection type (i.e., random or maximizing goals) and optional queries (introduced in Sect. III-C).

We encode *maxTime* and *maxEB* into CSP. CSP then finds a model evaluation for each time step, so if *maxTime* = *maxEB*, all EBs are guaranteed to occur within the path, and the final state is found given t_{∞} for the dynamic function type for each intention. This is important in enabling modelers to generate shorter paths, by making *maxTime* < *maxEB*, which is important in understanding short-term impacts.

After encoding the rest of the inputs into CSP, as discussed in Sect. IV-B, we then ask CSP to find a total order over the constraints and values for each state in the path. If the modeler includes any queries, then CSP is repeatedly called with each query. A path for each valid option is returned to the modeler.

This strategy produces a valid path when one is available. Depending on the constraints specified by the user, a valid path may not be possible. EQ2 asks whether the order of PGW and UND matters. Using the query function and this strategy, we generate solutions for each ordering. The solution where UND became *FS* first (via BSD), resulted in the model being satisfied. The solution where PGW became *FS* first, resulted in SiD becoming *FD*, which in turn resulted in MCW becoming *FD*. Answering EQ2, stakeholders learn that the order does not matter, and that SiD is not fully represented in the model.

Since PGW has an impact on SiD which is not represented by the model, stakeholders ask EQ3: Which possible scenarios always satisfy MCW even if SiD becomes denied in the future? This strategy shows that one of BSD or BLD must be *FS* prior to the denial of SiD. Since BSD results in the *PS* of ROC and *PD* of BLD, BSD becomes slightly preferred.

Finally, we used this strategy to understand if all root-level goals can eventually be satisfied (EQ5). Encoding this question resulted in no valid path because we included the requirement that HC and ROC became *FS*, which is not possible given the model's structure. Once relaxing these desired values to *PS*, we did not find a solution that resulted in the *PS* of ROC and HC at the same time. We could achieve one but not both, thus not all root-level goals can be satisfied. Stakeholders can use this information to evaluate priorities between root-level goals.

3) AR3 – Path given a previous path: The third strategy is to create a path which is different from the previously seen path over the same constraints. Allowing stakeholders to construct new alternative paths enables them to understand other possible evolutions that exist. This is especially useful when dealing with stochastic variables and to understand the ordering of epochs.

This strategy works the same way as AR2 but has the added input of the previous path(s). The previous path's EBs are negated to tell CSP not to pick them again. If a new valid path exists, CSP returns it to the modeler. Each intention gets a slightly different negation constraint based on the intentions' dynamic function. For example, a C intention can not be negated because it will break the constraints of the model.

As we discussed EQ2 above, we noted that several different solutions existed for the ordering of PGW and UND. We can use AR3 to understand what other results (if any) exist as asked in EQ6: Given the initial results EQ1-EQ5, is there another way to answer the question? When selecting PGW, this strategy showed stakeholders alternative solutions to whether or not each of BSD and BLD became satisfied (answering EQ6). This strategy is especially useful when $maxTime < maxEB$, and not all EBs occur by the end of analysis. Also, using AR3 improves the modeling process because it removes the need for manual comparison of models when generating unique solutions.

4) Summary: In Sect. I, we introduced and mapped six questions from the motivating example onto our strategies (AR1-3). Using AR1 on our example, we found out that it would be feasible to build a green centre prior to building a dump (EQ1). We then used AR2 to determine that the success is not affected by the order of these events (EQ2) but that either a small or a large dump must be built before the space in the current dump runs out (EQ3). By stochastically changing the citizens' environmental concern in a random path simulation (AR1), we were able to determine that it affects two root-level goals: (1) to have citizens enjoy the city, and (2) for the city to reduce its operating costs (EQ4). At this point, we used AR2 to determine that all root-level goals could not be eventually satisfied (EQ5). Thus, stakeholders must make the tradeoff decisions between reducing operating costs and having their citizens enjoy the city, a typical trade off for many cities. We then used AR3 to explore what other simulations could result when we process green waste prior to using a new dump, and concluded that it would be best to build a green centre followed by a small dump (EQ6).

V. Validation

In this section, we discuss the effectiveness of our approach (by analyzing a large example) and show its scalability (by justifying that using CSP as the underlying solver is feasible for realistic models). Throughout Sect. IV, we showed how AR1-3 effectively answer EQ1-6 from the Waste Management example and so we depart from this example to further demonstrate effectiveness.

In order to evaluate our approach, we built a new web-based goal modeling and analysis tool [12] available at http://www.cs.toronto.edu/~amgrubb/growing-leaf.

A. Effectiveness: GrowingLeaf Analysis

In the absence of a real-world case study, we show our analysis with a goal model for developing our tool *GrowingLeaf*.

Example Background. In order to evaluate our time-based analysis, we decided to build a new iStar goal modeling tool replacing OpenOME [13] and had already decided not to build it on top of other existing goal modeling tools. The primary functionality goals of the tool were to enable users to build goal models, run forward analysis over models, run time-based simulation over the models, and save/share the models between sessions. We wanted to evaluate alternatives for the tool based on five qualities: development speed, ease of installation, effectiveness, usability, and maintainability. We had to decide whether to build the tool as an Eclipse plugin or as an online tool in the browser. If we chose the Eclipse plugin option for development then we would be restricted to interacting with the Eclipse framework. If we chose the browser approach, we could completely separate the analysis from the front-end, but would need to have a way to connect them back.

We modeled the intentionality of the first author and possible users of *GrowingLeaf* using iStar (not shown). After analyzing the models, we found that using Eclipse would be the best option given the expertise available in the group. Later, we discussed our plans with others and they told us of the difficulties of collaborating and extending Eclipse-based tools. In our previous analysis, we had not considered how our tool could evolve over time.

Applying Dynamics. We considered the motivations of potential collaborators and the types of web technologies to use. We explicitly considered how evolution of the environment, development expertise, and relationships between stakeholders might change over time. An iStar model with some of these new dynamics is shown in Fig. 4[3] and has 9 actors, 74 intentions, and ~100 links. We evaluated our model based on three scenarios in which we and our collaborators (a) chose Eclipse, (b) chose Web-based technologies, and (c) chose other technologies. Fig. 4 shows the scenario where we both picked Web-based technologies.

We capture the dynamic function of an intention using the format (*initialValue*, *dynamicFunction(max/min/const.Value)*). The Lead Dev had expertise in Java $(FS, C(FS))$ and had no expertise in web development but was willing to learn $(FD, MP(FS))$. Eclipse development expertise was available through another developer who was willing to help (FS, R) (but can be more precisely modeled as a UD function with repeating SD behaviour). Each of the tasks in the Leaf actor can be modeled as (FD, SD).

Analyzing the Model. Next we show that simulation produces meaningful results by discussing some of the questions we explored while developing the tool.

Our first question is "If the Local Developer's Eclipse Expertise has a repeated UD dynamic function where they are available FS and then not available FD repeatedly, what will be the outcome of Build Tool and the top-level goals?" To understand this change over time we model the progression of the tool development given the variability of Eclipse Expertise

[3]Dependums have been removed from the model to save space.

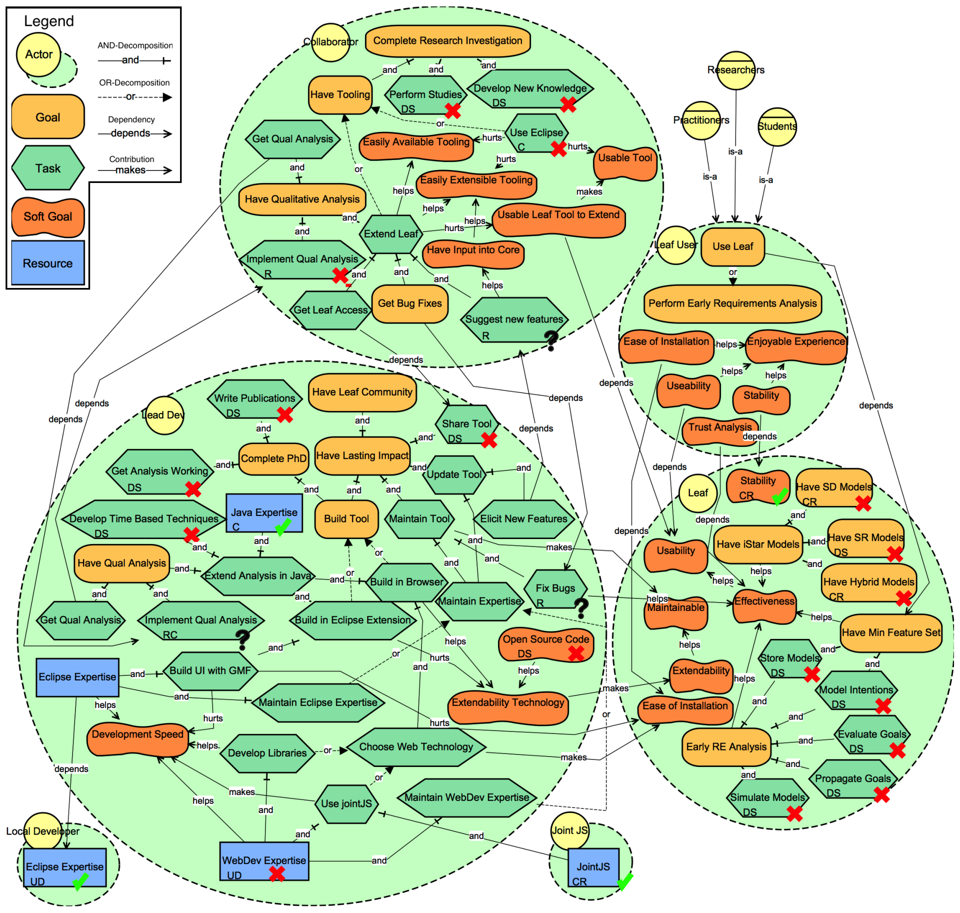


Fig. 4: iStar model of *GrowingLeaf* example with selected analysis labels.

with AR1. We assigned initial values to leaf-level intentions and dynamics to intermediate nodes, for example applying $(FD,\ MP(FS))$ to Build UI with GMF. The resulting path showed a progression of satisfying Build Tool but was not able to satisfy all goals. This simulation showed us that establishing expertise in the project is required for it's success, and our initial understanding using current approaches was incorrect.

We are considering sharing our code/tool and wanted to know "What might happen if we don't share the tool until after Complete PhD is satisfied?" This question is asking for a possible path given intermediate constraints (AR2). We add the constraint Complete PhD $<$ Share Tool. In the resulting path, both Have Leaf Community and Have Lasting Impact do not have the value *FS*. Also, in paths where the Collaborator's intention Use Eclipse was *FD*, the Collaborator's root-level goals were denied. Thus, not sharing the tool until Complete PhD is satisfied is not favourable.

We also looked at the question, "In the long-term (identified as when Complete PhD becomes *FS*), we want Leaf's Effectiveness, Usability, and Maintainability to be at least *PS*. What are possible solutions that will lead to these goals?" Given intermediate and final constraints, we first used AR2 to generate an initial path, and then used AR3 to generate additional unique paths. Seeing multiple paths showed that the order of completing some intentions did not matter, for example the satisfaction order of Have SD Models and Have Hybrid Models is irrelevant. Whereas, the order for completing other intentions mattered, specifically Share Tool should happen after Have Min Feature Set. As a result of this analysis we added further constraints to the model, in order to predict a *PS* outcome for Effectiveness and Usability in the long-term.

Asking these and other questions and considering the analysis results helped us debug the model and understand it. We found errors in the links and dynamics we initially assigned. More importantly, doing time-based modeling clarified the potential dependencies between Lead Dev and Collaborator.

Summary. We built this model to understand trade-offs in selecting development technologies, over our goals to have a functional, effective, useable, and maintainable tool. Standard analysis led us to the wrong decision because we assumed that the availability of Eclipse expertise is static. Our analysis allowed us to validate our decision to choose scenario (b): web-based technologies for development. The ability to generate multiple paths allowed us to determine which tasks must be completed in a prescribed order and which were independent. Overall, these strategies were effective in helping us understand possible evolutions of our project.

B. Scalability

We show the scalability of our approach by analyzing the run-times for each strategy on the examples discussed in this paper and on randomly generated examples. All scalability tests were completed on a 3.2 GHz Intel Core i5 processor with 8 GB 1600 MHz DDR3 of memory running OSX 10.11.5.

Our first scalability experiment explored the question: "How does the length of the generated path affect the computation time in AR1 and AR2?" We explored this question with the Waste Management example (`WME`) consisting of 21 Intentions and 18 Links, and the *GrowingLeaf* example (`Tool`) consisting of 74 intentions and 100 links. Each test was run ten times. Fig. 5a shows the mean run-times (in milliseconds) with min/max bars for `WME` and `Tool` for paths consisting of 5, 10, 25, 50, 75, 100, 150, 200, 300, 400, 500, 600, 700, 800, 900, and 1000 states. The maximum times for `WME` and `Tool` were 68 and 226 ms, respectively, and the shape of the graph illustrates that the run-time of AR1 appears to be growing polynomially with the length of the generated path.

Fig. 5b shows the mean run-times (in seconds) with min/max bars that strategy AR2 took for `WME` and `Tool` for the same set of path lengths. The maximum times for `WME` and `Tool` were 13.52 and 91.89 sec which is significantly longer than AR1. This may be a concern for larger models, with over 800 steps, because a run-time longer than one minute may cause the server call to timeout and the modeler to get frustrated. Further experiments with industrial models would help us understand and mitigate these limitations.

Given the lengthy run-times for AR2 in the first experiment, our second experiment considered the question: "How does the number of intentions in a model affect the computation time in AR2?" To evaluate variations in the number of intentions, we created a set of models by linking all intentions in a tree structure and varying the number of intentions in the models as follows: 25, 51, 75, 101, 125, 151, 175, and 201. For each model, we assigned a dynamic function type (see Sect. IV-B) to portions of the model intentions. For each model, we calculated the mean run-time (in seconds) with min/max bars for generating paths of length 5, 10, 25, 50, 75, 100, 150, and 200. Fig. 5c presents the results of this experiment. Some of the experiments initially failed due to run-time stack-overflow errors, requiring us to increase the default stack size. Fig. 5c shows that model size does impact the run-time of our analysis and appears to have a greater effect than the size of the path. These experiments show that AR2 (and CSP as the underlying solver) are expected to be scalable to realistic goal models. We have not attempted to optimize our CSP encoding to improve the run-time, but believe optimizations exist.

Since AR3 generates a path based on previous paths, our final experiment asked: "How does the number of previous paths used affect the computation time in AR3?" It is not meaningful to evaluate the scalability based on the models generated for random dynamic functions. Instead, we again used the `WME` and `Tool` examples to calculate the average run-times to generate additional paths. In the experiment, we used path lengths of 15 and 30 states for each example, and generated three new paths given the previous paths. Fig. 5d shows the mean results of the AR3 experiment which was repeated five times for each new generated path. The maximum times for `WME` and `Tool` were 34 and 139 milliseconds, respectively, but these times were for a single previous path (where there was the most variability). The data suggests that additional previous paths reduce the run-time of AR3, making it more scalable than AR2. We believe this is due to the additional constraints reducing the CSP search space (and thus its run-time).

Summary. To summarize, we conducted scalability experiments for our three strategies. For AR2, we used both case studies and generated models. From these results, we conclude that our strategies are scalable enough to be applied to the analysis of realistic goal models.

C. Threats to Validity

Construct Validity. We operationalized intentions' changing evaluations with our dynamic functions (see Sect. III-B), but did not establish that this was the best representation. We have explored instantiating each goal to create a model of objects, as in [14], but we believe that this approach is too complex for early RE. If modelers wish to explore specific instantiations of a goal, they can add them to the model as though they are an evolution of the model (see Sect. III-C).

Theoretical Reliability. We recognize that 'research bias' may be an issue in our effectiveness evaluation in Sect. V-A, because as researchers we were studying ourselves and we are biased towards the success of our technique. To mitigate this and ensure the accuracy of our models, we elicited feedback from other researchers and discussed the model in a workshop with our research group. We also tested our approach with goal models shared by other researchers in the domains of network maintenance, software supply chains, and sustainability. We observed that we can add dynamics to previously created models, but not all goal models contain dynamics.

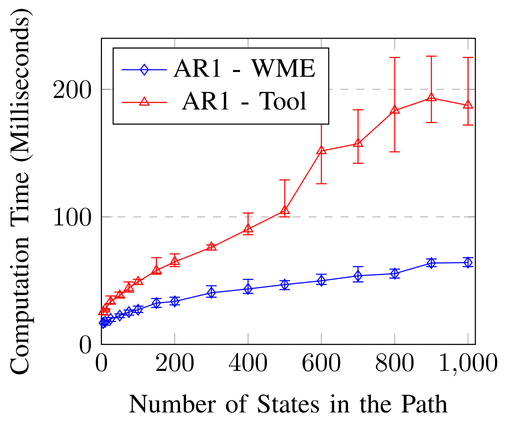
Internal Validity. Given the research bias discussed, there is a risk that the *GrowingLeaf* model may not be representative of other iStar goal models. Effectiveness (see Sect. V-A) is also threatened since we presented only one fully-worked out example in the paper. However, the example presented is quite large and we have continued to evaluate other large examples. Given that our scalability tests (see Sect. V-B) used randomly generated models, there may have been errors in the model or in the analysis. We mitigated this by manually reviewing the results for some of the generated models.

External Validity. Several other iStar experts tried out our tool and analysis at various stages of the project, with very encouraging results. However, stakeholders and non-expert modelers might not have the skills to build models with dynamics or understand how to configure our analysis. We will mitigate this threat in the future by evaluating our approach with an external case study. We also hope to evaluate what additional training is required to use our approach effectively.

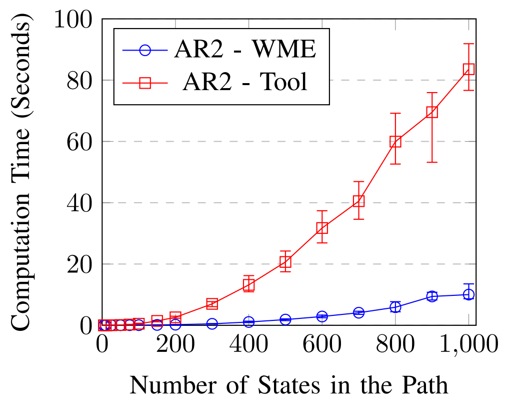
VI. Related Work

In this section, we compare our approach with related work.

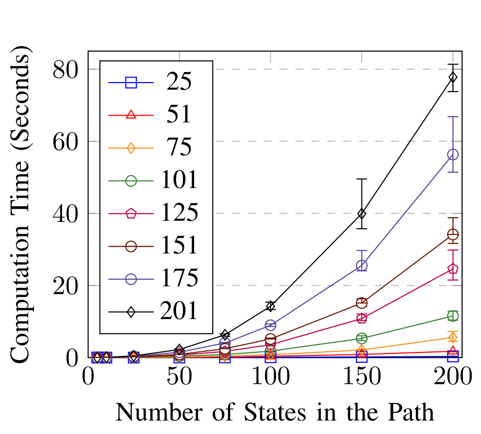
Changing Goals. [15] discusses how goals change in their achievement over time for the purpose of developing heuristics



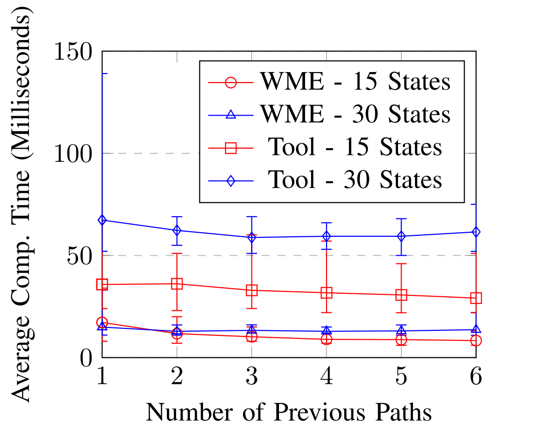
(a) Results of changing the path length for AR1.



(b) Results of changing the path length for AR2.



(c) Results of changing the model size for AR2.



(d) Results of changing no. of prev. paths for AR3.

Fig. 5: Scalability tests for AR1-AR3.

to improve goal identification. Using KAOS, [16] considered goals that are real-time 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 [17] with state-based formalisms. [18] looked at conditional non-monotonic temporal beliefs and goals. They represented and reasoned with them using an ASP solver and an LTL reasoner. In our work, we look at the evolution of goals rather than looking at concrete timing requirements, or concrete events within the model.

Simulation of Agents. Agent-based approaches, such as [19], discuss the simulation and operationalization of agents' goals and multi-agent systems. In our work, we are considering scenarios about top-level goals and actor intentionality, rather than the tasks completed by collections of agents. [20], [21] focused on connecting time dependent goals within a decomposition (denoted by precondition links), allowing for the coordination and achievement of agents' plans. [22] combined iStar with ConGolog to create a utility based temporal ordering for agents to execute tasks in order to simulate goal alternatives. Instead, we use relative or ordered dependency constraints specified by the modeler.

Requirements Evolution. [23] also looked at developing requirements for evolution, but with the goal of developing assets for reuse between systems by modeling variation points. [24] considered the evolution of goals with the goal of redesigning legacy systems. Instead, we analyze evolution of stand-alone systems prior to them being built.

Goals outside early RE. Goals as dynamic entities have been used outside the requirements phase for runtime analysis and monitoring [25]–[27], self-adaptive systems [28]–[30], and context-aware systems [31]. [32] distinguished between design-time goal models and run-time goal models. Our models remain design-time entities throughout our analysis. [25] converted goal models into components that monitor states in the system through runtime monitoring. [28] added adaptive goals to goal models. These approaches do not monitor how the requirements in the model change, but rather check properties of the system as it runs.

Analysis Techniques and Encoding. While [7] argues that exploratory tools capture imprecise and incomplete information and latter-stage models are better fit for automated analysis, we believe that automated analysis does yield benefits in the early-phase requirements, even with incomplete information. Different techniques for automated-forward analysis of qualitative evaluations exist [8], [33]. These techniques were designed with distinct purposes and differ in their resolution of soft-goal labels, [33] selects the minimum label while [8] does an averaging of the labels. We have implemented both resolution techniques in our tool for both forward analysis and backwards analysis. Others have developed and used complementary approaches for forward analysis [2], [4], [8] and backwards analysis [4], [9]. Our analysis techniques can in principle be adapted for quantitative analysis (as in [3]–[5]) or hybrid analysis (as in [8]).

We were not the first to use CSP for analyzing goal models. [34] used CSP (and JaCoP [35]) to solve quantitative constraints in GRL. We used this work to guide our implementation and extended it by enabling analysis over qualitative values. Our earlier work [11] used SAT to encode the goal models (similar to [36] and [9]) but we found the proposition-based encoding of evaluation labels compromised our analysis.

VII. Conclusions and Future Work

In this paper, we argued that goal modeling for early-phase RE can be improved by explicitly modeling and analyzing intention evaluations over time. We enabled stakeholders to answer questions with model dynamics over time. We presented strategies for simulating goal model dynamics and showed that our tool *GrowingLeaf* allowed effective and scalable application of these strategies to analyze realistic goal models.

In future work, we will evaluate the effectiveness of our approach with an external industrial case study. This paper dealt with only "relative" times and can be extended by adding "wall clock time" to our analysis. With access to large real-world models, we can also further test the feasibility of our approach and find optimizations for the AR2 CSP encoding. We also plan to formally specify our extension to iStar and consider other qualitative evaluation approaches [4]. Finally, our analysis and tooling can be combined with other methods such as the delphi method and creativity techniques [37] to further improve stakeholders' decision making processes.

Acknowledgments. We thank Jennifer Horkoff, Michalis Famelis, Daniel Amyot, Fabiano Dalpiaz, John Mylopoulos, Carlo Ghezzi, members of the Modeling group in Toronto, and our anonymous referees for helping improve this work.

References

[1] E. Yu, "Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering," in *Proc. of RE'97*, 1997, pp. 226–235.
[2] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. Kluwer, 2000.
[3] D. Amyot, "Introduction to the User Requirements Notation: Learning by Example," *Comput. Netw.*, vol. 42, no. 3, pp. 285–301, Jun. 2003.
[4] P. Giorgini, J. Mylopoulos, and R. Sebastiani, "Goal-oriented Requirements Analysis and Reasoning in the Tropos Methodology," *Eng. Appl. Artif. Intell.*, vol. 18, no. 2, pp. 159–171, 2005.
[5] A. van Lamsweerde, *Requirements Engineering - From System Goals to UML Models to Software Specifications*. Wiley, 2009.
[6] J. Horkoff and E. Yu, "Analyzing Goal Models: Different Approaches and How to Choose Among Them," in *Proc. of SAC'11*, 2011.
[7] ——, "Comparison and Evaluation of Goal-Oriented Satisfaction Analysis Techniques," *Requir. Eng.*, vol. 18, no. 3, pp. 199–222, 2013.
[8] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, and E. Yu, "Evaluating Goal Models Within the Goal-Oriented Requirement Language," *Int. J. of Intell. Syst.*, vol. 25, no. 8, pp. 841–877, 2010.
[9] J. Horkoff and E. Yu, "Finding Solutions in Goal Models: An Interactive Backward Reasoning Approach," in *Proc. of RE'10*, 2010, pp. 59–75.
[10] ——, "Interactive Goal Model Analysis For Early Requirements Engineering," *Requir. Eng.*, pp. 1–33, August 2014.
[11] A. M. Grubb, "Adding Temporal Intention Dynamics to Goal Modeling: A Position Paper," in *MiSE@ICSE'15*, 2015, pp. 66–71.
[12] A. M. Grubb, G. Song, and M. Chechik, "GrowingLeaf: Supporting Requirements Evolution over Time," 2016, submitted.
[13] J. Horkoff, Y. Yu, and E. Yu, "OpenOME: An Open-source Goal and Agent-Oriented Model Drawing and Analysis Tool," in *Proc. of i* Wrksp'11*, 2011, pp. 154–156.
[14] F. Dalpiaz, P. Giorgini, and J. Mylopoulos, "Adaptive Socio-Technical Systems: A Requirements-Based Approach," *Requir. Eng.*, vol. 18, no. 1, pp. 1–24, 2013.
[15] G. Regev and A. Wegmann, "Where Do Goals Come From: The Underlying Principles of Goal-Oriented Requirements Engineering," in *Proc. of RE'05*, 2005, pp. 353–362.
[16] E. Letier and A. van Lamsweerde, "Deriving Operational Software Specifications from System Goals," in *Proc. of FSE'02*, 2002.
[17] E. Letier, J. Kramer, J. Magee, and S. Uchitel, "Fluent Temporal Logic for Discrete-time Event-based Models," *SIGSOFT Softw. Eng. Notes*, vol. 30, no. 5, pp. 70–79, 2005.
[18] R. Gonçalves, M. Knorr, J. Leite, and M. Slota, "Non-monotonic Temporal Goals," in *Proc. of LPNMR'13*, 2013, pp. 374–386.
[19] G. Gans, M. Jarke, G. Lakemeyer, and D. Schmitz, "Deliberation in a Metadata-Based Modeling and Simulation Environment for Inter-Organizational Networks," *J.Inf.Syst.*, vol. 30, no. 7, pp. 587–607, 2005.
[20] C. Cheong and M. Winikoff, "Hermes: A Methodology for Goal Oriented Agent Interactions," in *Proc. of AAMAS'05*, 2005.
[21] ——, "Hermes: Designing Goal-Oriented Agent Interactions," in *Proc. of AOSE'05*, ser. LNCS, vol. 3950, 2006, pp. 16–27.
[22] G. Gans, M. Jarke, G. Lakemeyer, and D. Schmitz, "Deliberation in a Modeling and Simulation Environment for Inter-Organizational Networks," in *Proc. of CAiSE'03*, 2003, pp. 242–257.
[23] D. Webber and H. Gomaa, "Modeling Variability with the Variation Point Model," in *Proc. of ICSR'02*, 2002, pp. 109–122.
[24] A. Anton and C. Potts, "The Use of Goals to Surface Requirements for Evolving Systems," in *Proc. of ICSE'98*, 1998, pp. 157–166.
[25] N. Robinson, "A Requirements Monitoring Framework for Enterprise Systems," *Requir. Eng.*, vol. 11, no. 1, pp. 17–41, 2005.
[26] N. Bencomo, J. Whittle, P. Sawyer, A. Finkelstein, and E. Letier, "Requirements Reflection: Requirements As Runtime Entities," in *Proc. of ICSE'10*, 2010, pp. 199–202.
[27] X. Franch, R. Kenett, F. Mancinelli, A. Susi, D. Ameller, M. C. Annosi, R. Ben-Jacob, Y. Blumenfeld, O. H. Franco, D. Gross, L. Lopez, M. Morandini, M. Oriol, and A. Siena, "The RISCOSS Platform for Risk Management in Open Source Software Adoption," in *Proc. of OSS'15*, 2015, pp. 124–133.
[28] L. Baresi, L. Pasquale, and P. Spoletini, "Fuzzy Goals for Requirements-Driven Adaptation," in *Proc. of RE'10*, 2010, pp. 125–134.
[29] P. Sawyer, N. Bencomo, D. Hughes, P. Grace, H. J. Goldsby, and B. H. C. Cheng, "Visualizing the Analysis of Dynamically Adaptive Systems Using i* and DSLs," in *Proc. of REV'07*, 2007, p. 3.
[30] A. Knauss, D. Damian, X. Franch, A. Rook, H. A. Müller, and A. Thomo, "ACon: A Learning-Based Approach to Deal with Uncertainty in Contextual Requirements at Runtime," *J. Information & Software Technology*, vol. 70, pp. 85–99, 2016.
[31] M. Vrbaski, G. Mussbacher, D. Petriu, and D. Amyot, "Goal Models As Run-time Entities in Context-aware Systems," in *Proc. of MRT'12*, 2012, pp. 3–8.
[32] F. Dalpiaz, A. Borgida, J. Horkoff, and J. Mylopoulos, "Runtime Goal Models: Keynote," in *Proc. of RCIS'13*, 2013, pp. 1–11.
[33] J. Horkoff, R. Salay, M. Chechik, and A. D. Sandro, "Supporting Early Decision-Making in the Presence of Uncertainty," in *Proc. of RE'14*, 2014, pp. 33 – 42.
[34] H. Luo and D. Amyot, "Towards a Declarative, Constraint-Oriented Semantics with a Generic Evaluation Algorithm for GRL," in *Proc. of i* Wksp'11*, 2011, pp. 26–31.
[35] K. Kuchcinski and R. Szymanek, "JaCoP - Java Constraint Programming solver," http://jacop.osolpro.com, 2016, accessed: 2016-02-21.
[36] R. Sebastiani, P. Giorgini, and J. Mylopoulos, "Simple and Minimum-Cost Satisfiability for Goal Models," in *Proc. of CaiSE'04*, 2004, pp. 20–35.
[37] J. Horkoff, N. A. M. Maiden, and J. Lockerbie, "Creativity and Goal Modeling for Software Requirements Engineering," in *Proc. of C&C'15*, 2015, pp. 165–168.