2023-5

Aiding Users in Requirements Analysis Tasks Through Color and Filtering

Yesugen Baatartogtokh
Smith College

Aiding Users in Requirements Analysis Tasks through Color and Filtering

Yesugen Baatartogtokh

Submitted to the Department of Computer Science
of Smith College
in partial fulfillment
of the requirements for the degree of
Bachelor of Arts

Alicia M. Grubb, Honors Thesis Adviser

May 2023

Contents

List of Figures

List of Tables

Abstract

Goal-oriented requirements engineering (GORE) allows stakeholders to visualize project scenarios and evaluate their changes over time, which assists them in making early trade-off decisions. However, goal models use formal notation that raises stakeholder concerns about comprehension when evaluating a model's evolution over time or exploring other potential solutions. This thesis aims to assist users in interpreting the results of automated analysis in answering time-based questions over goal models. To do so, we validate a previous approach that uses color visualization to improve model comprehension, extend it to assist users in exploring alternate scenarios, and implement filtering to allow stakeholders to interpret models and generate their own scenarios.

Acknowledgments

I would like to give my warmest thanks to my adviser, Professor Alicia M. Grubb, who made this thesis possible. The completion of this thesis could not have been possible without their guidance and encouragement, and I would like to express my gratitude for over two rewarding years in the Grubb Lab. Many thanks to my fellow lab members, who were a pleasure to work with. I am especially grateful to my lab partner, Irene Foster, for being a vital part of this research and for being a great friend. I could not have undertaken this journey without her.

Thank you to Professor Gunter Mussbacher for providing excellent feedback as a second reader. His thoughtful comments were incredibly valuable in improving my writing. I would also like to thank the McKinley Honors Fellowship Program for funding this thesis. Finally, words cannot express my gratitude to my friends and family for their support.

Chapter 1

Introduction

Requirements engineering (RE) is the initial step in the development and design of software products. Well-defined requirements are essential to making sure projects are completed on time, within scope, under budget, and of sufficient quality [39]. Goal-oriented requirements engineering (GORE) is an element of requirements engineering that assists stakeholders in decision-making in the early stages of project planning through the use of goal models, which allow them to model the requirements and intentions of the actors in their project [19, 2, 27]. Users create models for the purpose of considering how alternative project proposals impact the needs and motivations of stakeholders, who depend on the system. Goal model analysis enables stakeholders to understand and answer "what-if" questions by exploring alternate scenarios [30], including evolving scenarios [3, 25]. In this thesis, we use the Tropos goal modeling language and the Evolving Intentions framework to specify and analyze evolving scenarios.

In Tropos, an intention may be assigned an evidence pair as valuation (also known as satisfaction values), which consists of evidence for and against the fulfillment of the intention. In the Evolving Intentions framework, intentions can also be assigned a function that defines how an intention's evidence pair valuations evolve over time. However, these formal evidence pair labels make goal models difficult to interpret, as stakeholders have to manually review and keep track of each intention as it evolves over time. In this thesis, we investigate

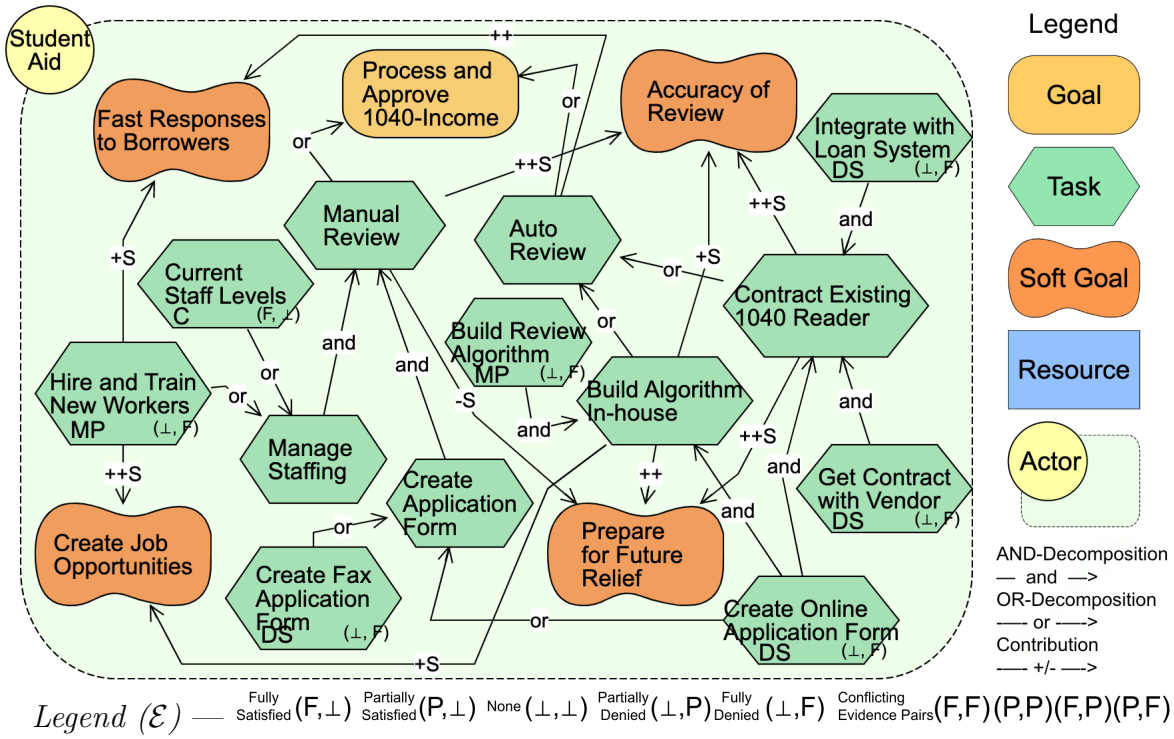
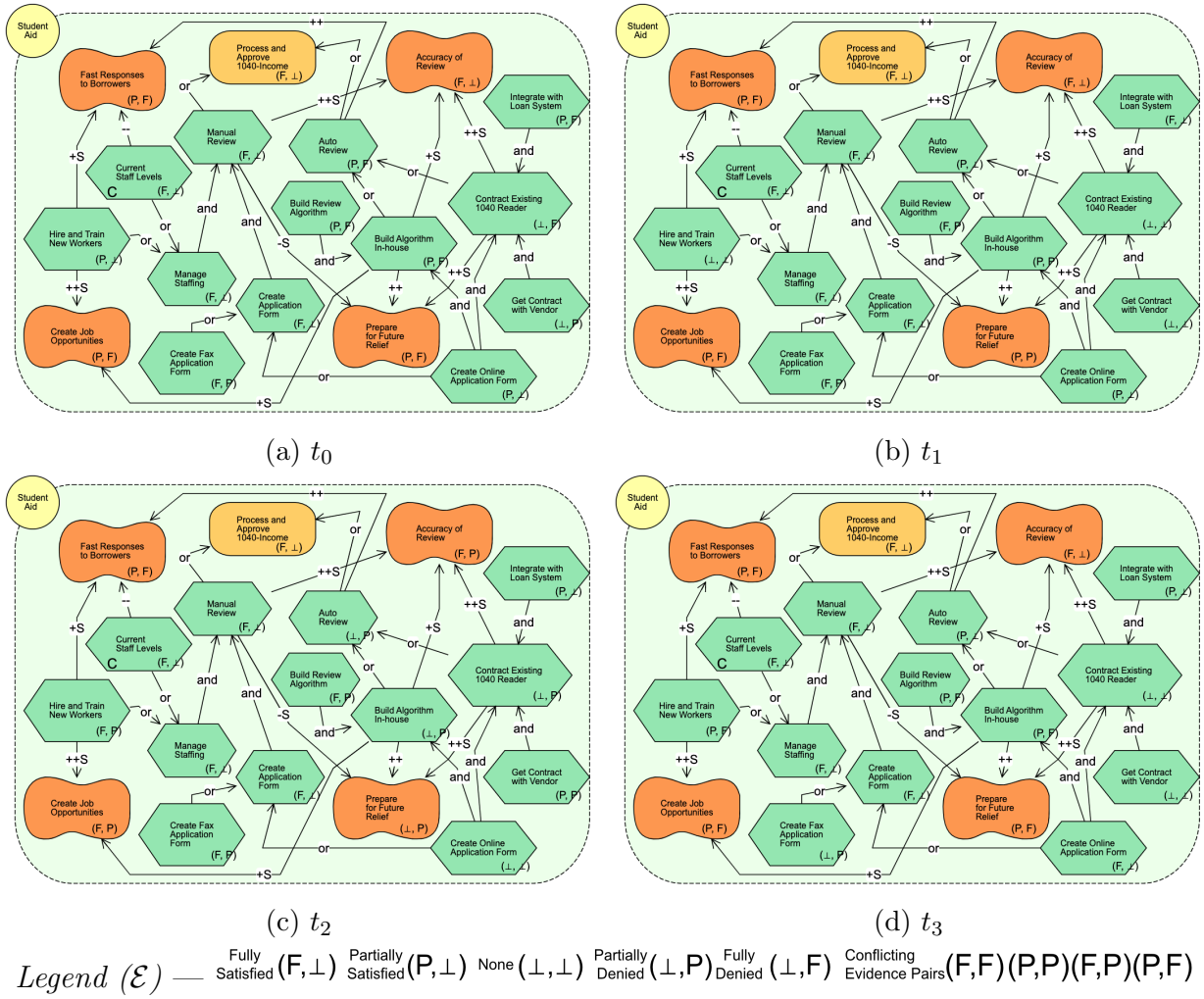


Figure 1.1: Fragment of goal model for processing and approving income limits of student loan borrowers via IRS Form 1040.

improving the interpretability of analysis results within this framework.

1.1 Illustrative Example

Consider the software team at the US Department of Education working to implement the *2022 Biden-Harris Administration Student Debt Relief Plan* [12]. Fig. 1.1 shows the goal model created in BloomingLeaf, a web-based goal modeling analysis tool, by analysts as part of their requirements process. As part of this temporary program, roughly 32 million borrowers are required to send in their most recent tax return (via IRS Form 1040) to verify they are within the income limit and qualify for the relief payment [12, 35]. The team is evaluating trade-offs for processing the 1040 forms. This decision is illustrated by the top-level goal Process and Approve 1040-Income being or-decomposed into Manual Review and Auto Review, with each of these options being further decomposed into their sub-tasks. We further discuss goal modeling notation in Chapter 2.



(a) t_0 (b) t_1

(c) t_2 (d) t_3

Figure 1.2: Simulation path of the Student Aid model.

In simulating this model over time, the analysts want to answer the question, "Is conducting a Manual Review, Auto Review, or a combination of the two best to ensure that the applications are processed correctly (i.e., satisfying Accuracy of Review) with fast turn-around for borrowers (i.e., satisfying Fast Responses to Borrowers)?", where the Manual Review option is satisfied first (see t_1 in Fig. 1.2(b)). At this time, Accuracy of Review is satisfied, denoted by the $(F, \bot)$, and Fast Responses to Debt Holders is conflicted, denoted by the (P, F), which prompts the team to want to know whether both these soft goals can be satisfied. Additionally, the team wants to explore creating two paths, one where Get Contract with Vendors is satisfied and the other where Build Algorithm In-house is satisfied. BloomingLeaf allows users to answer these types of questions through the *Next States* view, where users can get all

possible states for a given time point in the model evolution. However, other than clicking through each possible state, it is unclear how the user can make an informed choice, and analysts will have to keep track of individual intentions and their satisfaction values.

The simulation in BloomingLeaf returns a path of eight steps (four of these are shown in Fig. 1.2) and gives an answer. The analyst who is trying to interpret the path would have to go through each time point of the model and evaluate the evidence pairs of the intentions they are interested in. Due to the formal notation of Tropos evidence pairs, evaluating a goal model over time is a labor-intensive challenge for analysts. At this point, they cannot verify the existence of a path or generate their own scenarios. While BloomingLeaf's *Next States* window provides analysts with the ability to examine all paths and discover alternate scenarios, there are too many permutations for them to analyze manually.

1.2 State of the Art and Problem

The comprehensibility of Tropos evidence pairs has already been investigated in the literature. Hadar et al. compared Tropos and Use Case models and found that Tropos models seem to be more comprehensible with respect to some requirements analysis tasks, although Tropos models were found to be more time consuming [26]. In a replication of Hadar et al.'s work, Siqueira found no difference in model comprehensibility and effort between Tropos and Use Case models, when those models have equivalent complexity [51]. The issue of ease of comprehension means that analysts will have difficulty both understanding the initial state of the model and its evolution over time.

To assist users and improve the interpretability of model evaluations and analysis, previous work proposed using colors to visualize the information about the fulfillment of evaluation labels for intentions (i.e., evidence pairs) [52]. This tool, called Evaluation Visualization Overlay (EVO), provides users with a high-level overview of the contents of a goal model and its evolution over time. However, this does not allow analysts to comprehend the solution space of a model's scenario or select custom states from it.

In this thesis, we explore the tension between model comprehension and automated analysis techniques for evolving goal models. We consider two problems: (1) the interpretability of analysis results, and (2) the analyst's ability to generate a custom path when there are many permutations.

The first problem is that when a path is simulated, each intention is assigned an evidence pair at each time point which the user must evaluate in order to understand the analysis results and make decisions. On smaller models or on scenarios that cover fewer time points evaluating evidence pair labels may not be so tedious, but in reality, the complexity of many projects means that the associated goal models will often be large or span over multiple time points. Thus, the stakeholder is left with many intentions whose evolutions they need to keep track of over time.

The second problem is when a user is unable to generate and interpret their own custom paths from a solution space due to its size and complexity. As a scenario becomes bigger and more complex, the state space of the problem (i.e., the number of possible *Next States*) grows exponentially in the worst case, creating a state explosion problem. This scope makes it difficult for the user to find a preferred path and customize their simulation path, as they would have to manually evaluate each state.

1.3 Contributions

In this thesis, we explore the problem of model comprehension and custom path generation when reviewing goal models that evolve over time, as well as their permutations. We report on an IRB-approved between-subjects experiment on the efficacy of using color visualization. We extend the color visualization approach so that stakeholders can visualize the contents of a solution space in order to assist them in selecting possible states, when the content of each state is an entire model. Finally, we implement valuation-based filtering to better allow stakeholders to generate their own path from a solution space.

In the remainder of this thesis, we review relevant background in Chapter 2. Chapter 3

presents our experiment on the efficacy of using color visualization. Chapter 4 describes our work in intention valuation coloring and filtering to enable state space exploration. We conclude and describe future work in Chapter 5.

Chapter 2

Background

In this chapter, we review the goal modeling notation and visualization overlay used in this thesis. We use the *2022 Biden-Harris Administration Student Debt Relief Plan* model shown in Fig. 1.1 to illustrate our notation.

2.1 Goal Model

A goal model consists of actors, intentions, and links. Intentions, as the name suggests, describe the intentionality of each actor and consist of four types: goals, soft goals, tasks, and resources. For example, Fig. 1.1 contains one actor, named Student Aid, and eighteen intentions that describe the Student Aid team's goals and motivations.

Intentions can be decomposed or contribute to the fulfillment of one another via links, forming one or more graphs of nodes in the model. Decomposition links (i.e., and, or) decompose an intention into subsequent or child nodes. An intention with an and-decomposition requires all of its children to be fulfilled, while an or-decomposition requires only one to be fulfilled. In Fig. 1.1, the Student Aid's only goal is to Process and Approve 1040-Income, which is or-decomposed into two alternate tasks Manual Review and Auto Review. Contribution links (e.g., +, -, ++S, –S) indicate that an intention has influence on another intention, with an S contributing only "for" evidence. For example, Build Algorithm In-house (see Fig. 1.1) propa-

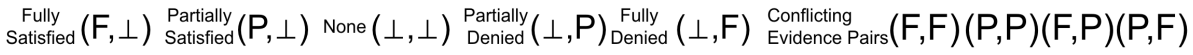


Figure 2.1: Evaluation Labels.

gates all evidence to Prepare for Future Relief via a ++ link, while Manual Review propagates all "for" evidence to Accuracy of Review via a ++S.

The fulfillment of an intention is evaluated qualitatively using an *evidence pair* (s, d), which separates evidence *for* and *against* the fulfillment of the intention. Both s and d consist of one of three values: F represents full evidence, P represents partial evidence, and $\perp$ represents no evidence, where $\perp \leq P \leq F$. Thus, goals can have one of five initial values: [*Fully*] *Satisfied* $(F, \perp)$, *Partially Satisfied* $(P, \perp)$, *Partially Denied* $(\perp, P)$, [*Fully*] *Denied* $(\perp, F)$, and *None* $(\perp, \perp)$; as well as four conflicting values that may result from propagation: (F, F), (F, P), (P, F), and (P, P). For clarity, we list these evidence pairs in Fig. 2.1. In Fig. 1.1, the task Build Review Algorithm is assigned the value *Denied* $(\perp, F)$ because the actor Student Aid has not yet completed the task.

2.2 Simulating Models over Time

We use the Evolving Intentions framework [25] to simulate how a model's fulfillment changes over time. The framework allows users to specify one or more stepwise functions (called *User-Defined* (*UD*) functions) describing how the evidence pair assignment for an intention changes over time. Over any time interval, the valuation of an intention can *Increase* (I), *Decrease* (D), remain *Constant* (C), or be random or *Stochastic* (R). In Fig. 1.1, the resource Current Staff Levels remains CONSTANT with the valuation of *Satisfied* $(F, \perp)$ over time. These atomic function types can be combined in to create compound function types that capture common patterns in goal models. The *MP* label on Build Review Algorithm indicates a *Monotonic Positive* function, meaning that the valuation will become more fulfilled until it is fully satisfied and then it will remain constant with that value. Other compound functions are: (*Denied-Satisfied* (*DS*)) the satisfaction evaluation remains *Denied* $(\perp, F)$ until t and

then remains *Satisfied* $(F, \perp)$; (*Satisfied-Denied* (*SD*)) the satisfaction evaluation remains *Satisfied* $(F, \perp)$ until t and then remains *Denied* $(\perp, F)$; (*Stochastic-Constant* (*RC*)) changes in satisfaction evaluation are stochastic or random until t and then remains constant with a given evidence pair; (*Constant-Stochastic* (*CR*)) the satisfaction evaluation remains constant at a given evidence pair until t and then changes in evaluation are stochastic. (*Monotonic Negative* (*MN*)) where the valuation will become more denied until it is fully denied and then it will remain constant with that value.

After a path has been simulated, all of the intentions in the model are assigned an evidence pair label for each time point. Intentions that are not assigned evolving functions receive their valuations via propagation. Thus, one of the contributions of the framework is to allow users to make trade-off decisions about future states of the model by stepping through each time point and reviewing the evidence pair assignments of each intention. For example, Fig. 1.2 gives the first four time points for a simulation path of the debt relief model in Fig. 1.1. Note that unlike the initial state in Fig. 1.1, all intentions have been assigned an evidence pair at t_0 (see Fig. 1.2(a)).

2.3 Next State Analysis

When considering trade-off decisions, simulating paths of the model over time might not be sufficient for an analyst to understand the domain. In BloomingLeaf, *Next State* analysis allows users to step into any time point in the path solution and visualize all the possible next states for the model. Analysts may want to guide the generation of the path when one state is more favored than another state, or check if the proposed answer is ideal or believable, given the provided constraints. Analysts can either explore the possible states for the next state in the path, or generate the remainder of the path with the selected state.

In BloomingLeaf, the simulation paths and next states analysis are formulated as a constraint satisfaction problem (CSP) and determined using the JaCoP constraint solver [33]. To find all possible next states, the solver takes the goal model, the current time point and

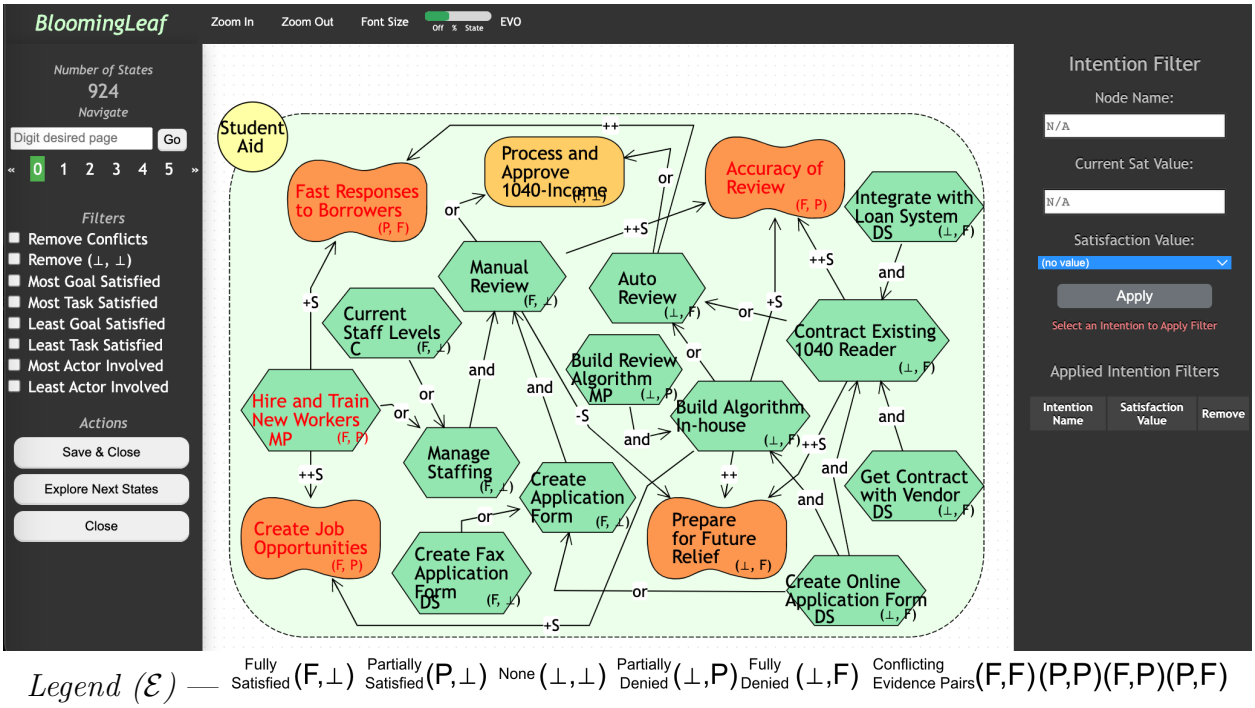


Figure 2.2: BloomingLeaf's Next States view of Student Aid model.

the partial timeline generated by the single path analysis as inputs and it outputs all the possible states with satisfying assignments to each variable. BloomingLeaf then displays the first option for the selected time point in a pop-up window (see Fig. 2.2). As a model becomes bigger, the state space of this problem grows exponentially in the worst case, especially when the scenario has not been fully explored by the user. In Fig. 2.2, the debt relief model originally returned 924 possible states.

Within GORE, Hu and Grubb proposed a simple set of filters to reduce the domain and solution space [31]. These filters allow state-reduction based on the whole model, such as removing any states that contain a conflicted satisfaction value, but users cannot filter by individual intentions. Thus, while these filters eliminate unhelpful states, they are generic and do not provide insights or assist the user in selecting a future state. Prior work has shown that humans have difficulty making choices as the number of options increases [46, 32]. For scenarios with more than one possible solution, this prior work helps in removing unwanted states but is limited in assisting the users in finding a future state.

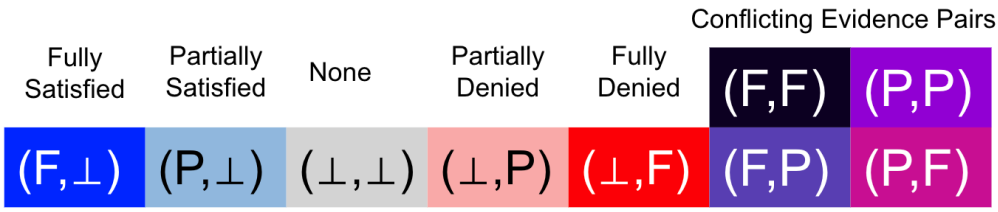


Figure 2.3: Evaluation Labels with EVO.

2.4 EVO: Evaluation Visualization Overlay

To assist users and improve the interpretability of model evaluations and analysis, previous work proposed using colors to visualize the information about the fulfillment of evaluation labels for intentions, which we have implemented in BloomingLeaf [52]. This color visualization tool, called Evaluation Visualization Overlay (EVO), presents evaluation label information both statically and as they change over time. Within the Evolving Intentions framework introduced above, it is difficult for a user to remember all of the different valuations of each intention at each time point, much less synthesize them all together to act upon the given information. The use of color provides users with a high-level overview of a solution, without having to individually evaluate each intention and its evaluation label.

Each evidence pair (s, d) label is assigned a color (see legend in Fig. 2.3), where blue denotes evidence for (i.e., the s value), red denotes evidence against (i.e., the d value), and purple denotes conflicting evidence. The more saturated (or darker) the color shade, the stronger the evidence (i.e., F is darker than P). Observe that (F, F) is a very dark shade of purple, whereas (P, P) is a lighter shade of purple. For (P, F) there is both blue and red present, making it purple, but because there is more evidence for denial, it is more red-purple, with the inverse being true for (F, P) (see Fig. 2.3).

During modeling activities, when EVO is enabled the color of each intention corresponds to any initial assignment, while unassigned intentions retain their original color. This provides an overall visualization of the model's initial state. For example, Fig. 2.4 gives the initial state of the Student Aid model with EVO turned on. There, Current Staff Levels is

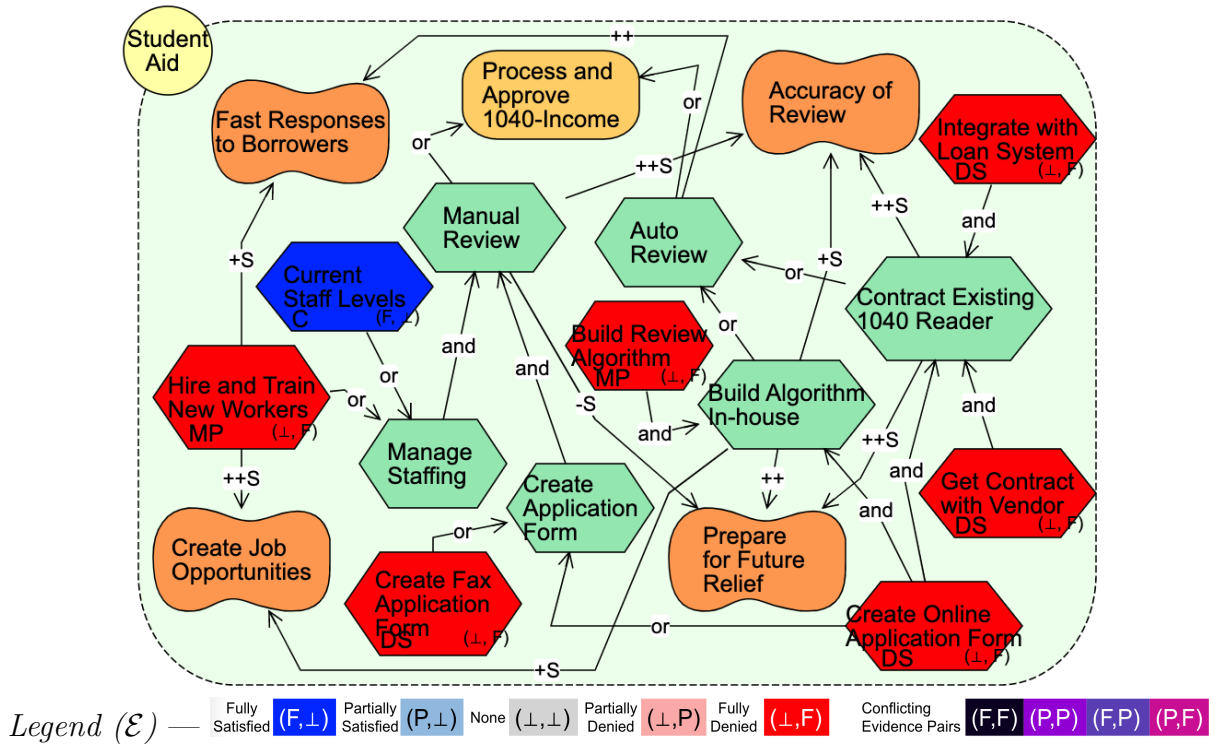


Figure 2.4: BloomingLeaf's Initial State of Student Aid model, with EVO turned on.

colored dark blue because it has been assigned the $(F, \bot)$ label while Build Review Algorithm is colored dark red because it has been assigned the $(\bot, F)$ label. Other intentions retain their original color because they have not been assigned an evidence pair.

EVO provides three modes to visualize simulations: State, Time, and Percent. To introduce these modes, we consider only the evolution of the Create Job Opportunities intention from the *Student Aid* model. Fig. 2.5 shows the color and evidence pair assignments for Create Job Opportunities at time points 0–4. *State* mode shows the current time point of the model, with the background of each intention colored based on their assigned evidence pair. *Time* mode shows the valuations over the entire path in one view. In Fig. 2.5, each of the stripes on Create Job Opportunities represents the colors of each state shown above. Finally, *Percent* mode colors by overall evaluation percentages, making the background of each intention colored with the percentage of states in the simulation where the intention has each evidence pair assignment. The width of each colored stripe corresponds to the percentage of time points that it holds a specific evidence pair, ordered based on level of fulfillment.

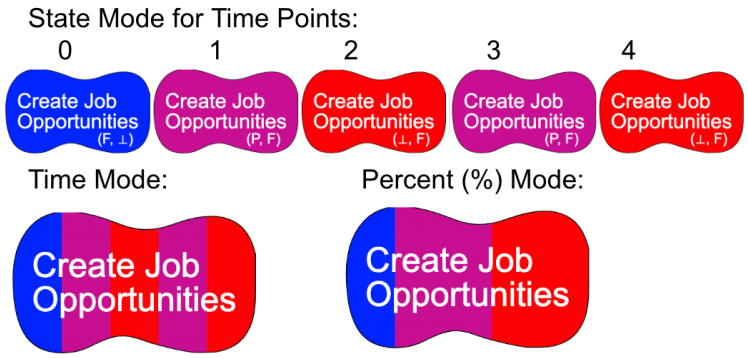
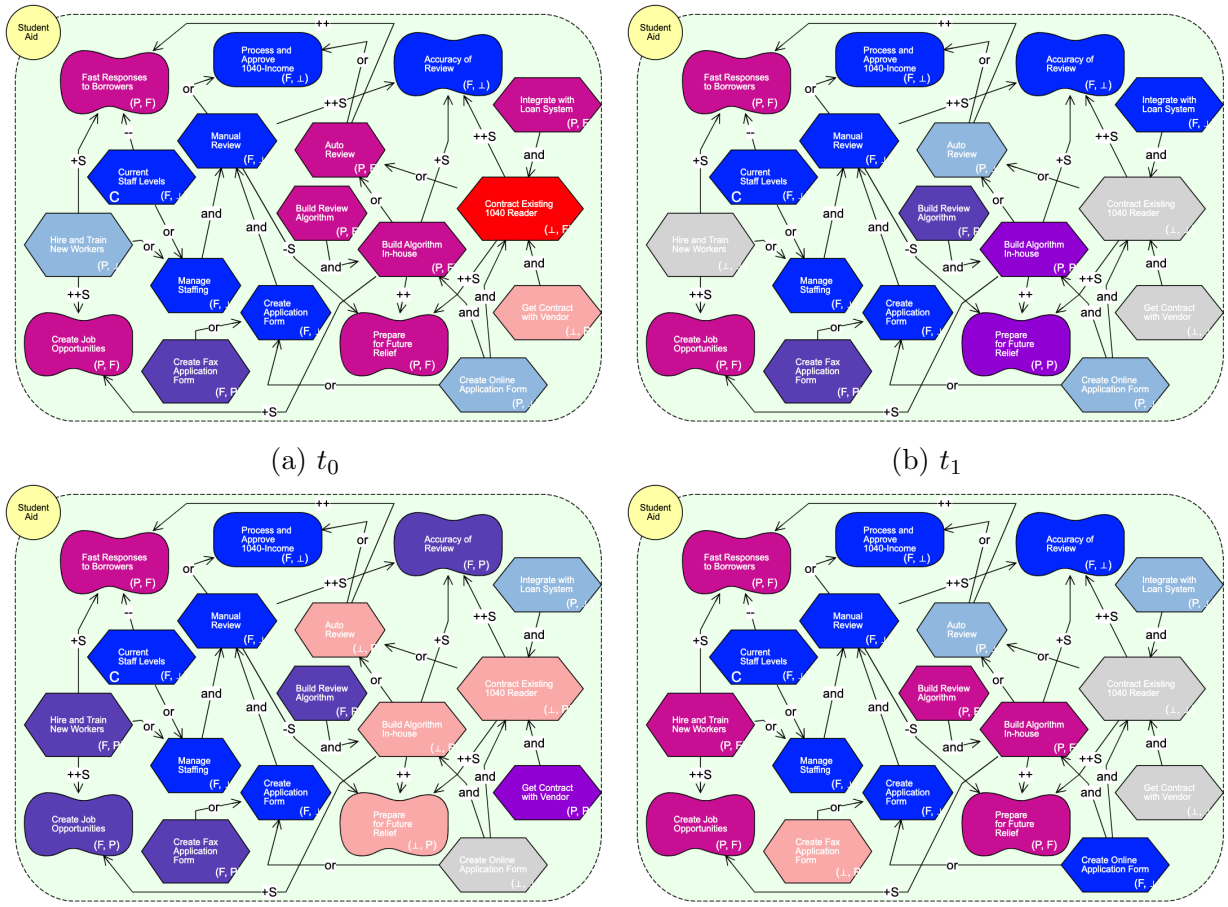


Figure 2.5: EVO modes showing only Create Job Opportunities.

Returning to the simulation results introduced in Fig. 1.2, we apply EVO *State* mode to the results in Fig. 2.6. The addition of color means that analysts are able to keep track of intentions and their evaluation labels more efficiently, as they are given a visual summary that is less cognitively demanding than evaluating individual labels.



(a) t_0 (b) t_1

(c) t_2 (d) t_3

Figure 2.6: Simulation path of the Student Aid model, with EVO color visualization applied.

Chapter 3

Validating Color Visualization

3.1 Introduction

In qualitative goal models, there are multiple methods for evaluating intentions. For example, i* and GRL use visual labels (e.g., checkmarks and Xs), while Tropos uses evidence pairs (e.g., (F, P)). In comparing these approaches, the visual labels in i* are more understandable to end-users but lack formal semantics, while the evidence pairs in Tropos allow for automation but are hard for users to understand.

This tension between model comprehension and automated analysis is further exacerbated by evaluating models over time [23] and with families of models [1], where users evaluate collections of models instead of single instances. Given the potential for automating analysis of goal models [36] and connecting them with downstream activities [28], we aim to improve the interpretability of Tropos evidence pairs, making them more accessible to end-users.

We investigate to what extent, if any, does using EVO (i.e., the Evaluation Visualization Overlay) affects how individuals understand and make decisions about goal models with timing information, using Tropos evidence pairs. We report on an IRB-approved between-subjects experiment conducted with 32 undergraduate students. We aim to answer four research questions:

RQ0 Do modelers across treatment groups perform similarly on basic goal modeling and simulation tasks, given a consistent training protocol?

RQ1 To what extent are subjects able to use EVO, and does using EVO affect subjects' understanding and reasoning about goal models in comparison to the control?

RQ2 How does EVO compare with the control in terms of time and subjects' perceptions?

RQ3 How do subjects rate the study instruments and experience?

From this experiment, we conclude that our subjects, with minimal prior training in goal modeling, were able to learn and use the EVO extension to make decisions. We found no evidence that EVO altered the quality of understanding or decision making, either positively or negatively. However, we found that EVO significantly decreased the time required to make decisions. Finally, the subjects responded positively to EVO and the study protocol.

3.2 Methodology

In this section, we describe our methodology for conducting this study, which was approved by the institutional review board at Smith College (IRB Protocol #20-026). Our supplemental materials are available online[1].

3.2.1 Experiment Design

Our primary objective in designing this experiment was to measure the effects of EVO. The original EVO proposal was implemented as an extension to BloomingLeaf [24]. We did not intend to evaluate the usability of BloomingLeaf; instead, we wanted to test EVO in isolation without the confounding variables of tooling, making our study tool agnostic. Additionally, we wanted to collect timing information in an accurate way. Thus, we designed the study instrument to be completed via our institution's browser-based Qualtrics® XM platform. We used the BloomingLeaf git repository [24] only for the purpose of creating our study

[1]See *https://doi.org/10.35482/csc.002.2023* for online supplement.

materials and models.

In designing this experiment, our main consideration was ensuring that we measured the appropriate elements (i.e., construct validity), and controlled for the risks of variability between subjects' tasks, subjects' natural performance, and any learning or fatigue effects Our desire to control for these variables resulted in us using a nested 2x2 design [57]. To measure the impacts of using EVO we compared measurements of subjects analyzing a model with and without having access to EVO, which requires subjects to answer questions over two models. To mitigate any learning effects, we controlled for the order that subjects were exposed to EVO. Previous investigations have demonstrated that task equivalency is an important factor in analyzing model comprehensibility [51]. We've designed our comprehension tasks to be similar but not identical. To understand any effects that may result from variations between models, groups of subjects answer both modeling questions with EVO and without EVO

We explored conducting the study as either a between- or within-subjects comparison. Ideally, our study would be analyzed in-subjects. This would control for natural variations in individual performance, model variability, and EVO ordering. Yet, analyzing this design requires the use of ANOVA, for which we were unsure we could get sufficient subjects. Instead, we planned our analysis to be performed between-subjects, but this has the downside of not being able to control individual subject variability

3.2.2 Materials: Models and Videos

In this study, we used four models: the Course model (see Fig. 3.1), the Employment model (see Fig. 3.2), the Summer model (see Fig. 3.3), and the Bike model (Fig. 3.4). We list these models and their associated metrics in Tbl. 3.1.

The Course model describes the process of a student (and their advisor) trying to decide whether the student should take a fun and interesting or practical and unexciting elective in the next semester, with the top-level goal of Have an Elective. The Employment model (see

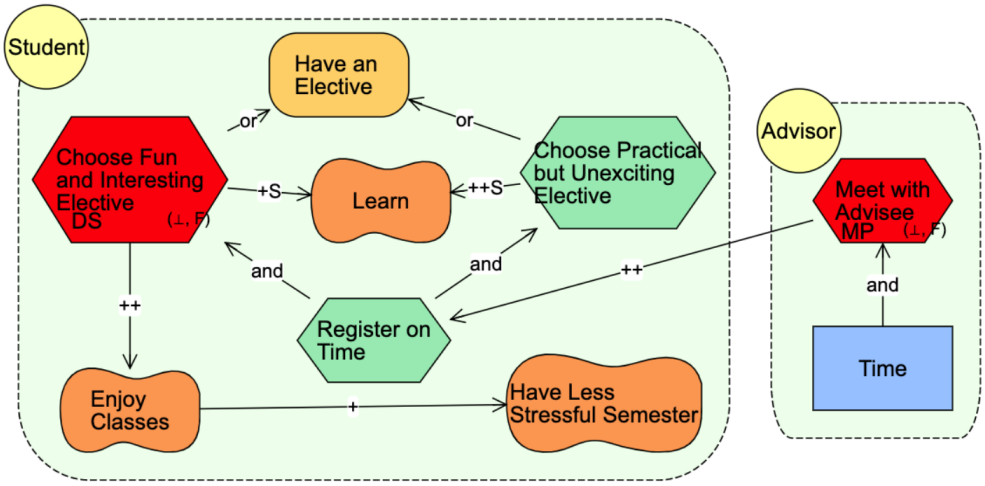


Figure 3.1: Course Model with EVO

Table 3.1: Study Models

Models	Figure	Actors	Intentions	Links	Evolving Functions
Course	Fig. 3.1	2	9	10	2
Employment	Fig. 3.2	1	9	10	3
Summer	Fig. 3.3	1	14	17	8
Bike	Fig. 3.4	1	16	20	7

Fig. 3.2) describes an employee, who is debating between working from home or working in an office, with the top-level goal of Have Employment.

We created both an EVO and control version of all models. In the Summer model (see Fig. 3.3), the actor Joy wants to have a summer activity, with choices between tasks Join Book Club, Join Community Center, and Join Soccer Team. These tasks are and-decomposed into sets of tasks which must be satisfied. In the Bike model shown in Fig. 3.4, the City actor wants to construct bike lanes, with the top-level goal Have Bike Lanes, for which they must have satisfied both sub-goals Have Design Plans and Have Build Plans. These two goals are or-decomposed into tasks they must choose from. While the Bike model has more intentions and links, the evolving functions are simpler than the Summer model.

Our study consisted of three training videos (the transcripts are available in the online supplement[1]). The first video, *Goal Models in Tropos* (VidGM), reviews goal modeling and

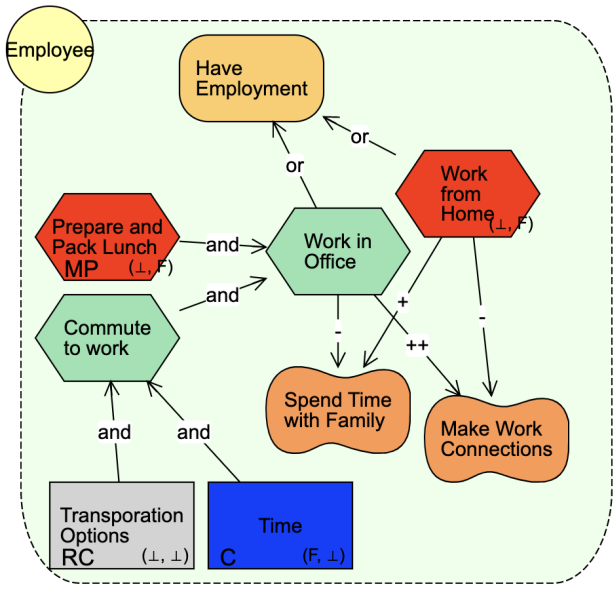


Figure 3.2: Employment Model with EVO

Table 3.2: Study Protocol

	Treatment Groups			
	Order: EVO		Order: Control (CTL)	
Part	1.EVO:Summer	2.EVO:Bike	3.CTL:Bike	4.CTL:Summer
0	Consent, Color Test, and Subject Background			
1	Training: Goal Modeling and Simulation			
2	Training: EVO	Training: EVO	Summer Control	Bike Control
3	Summer EVO	Bike EVO	Training: EVO	Training: EVO
4	Bike Control	Summer Control	Bike EVO	Summer EVO
5	Debrief			

explains Tropos evidence pairs and links. *Introduction to Simulation Over Time* (VidSim) introduces function types and evolving intentions, describing what it means to simulate a model over time. The last video (VidEVO), introduces the *EVO* color scheme for evidence pairs and goes over its three possible modes: *State*, *Time*, and *Percent*.

3.2.3 Procedure: Conducting the Experiment

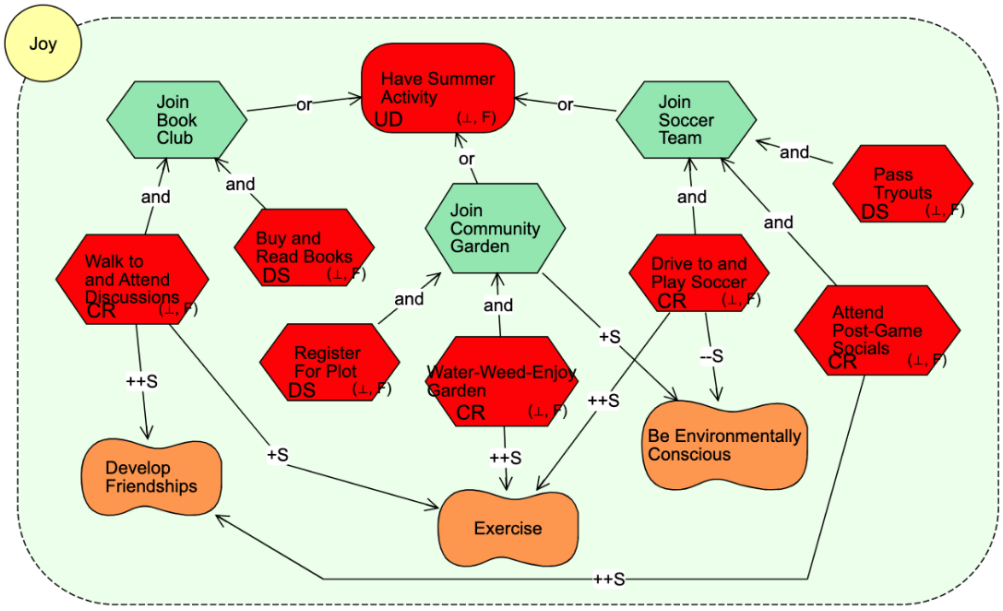


Figure 3.3: Summer Model with EVO

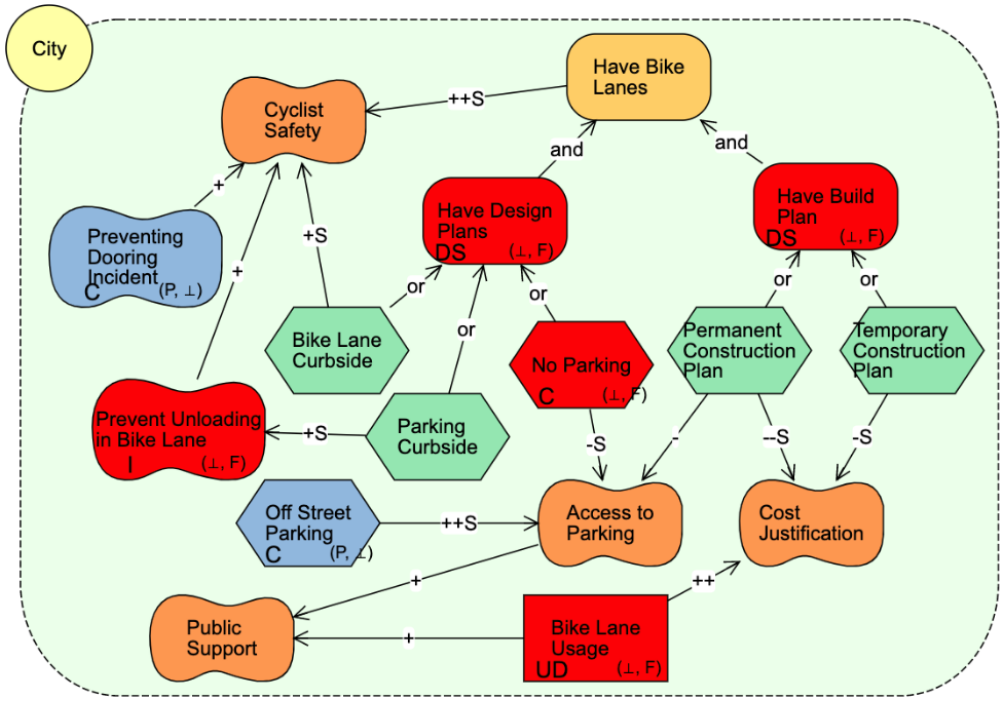


Figure 3.4: Bike Model with EVO

Table 3.3: Summer and Bike Questions

Page	Num	Summer Model	Bike Model
P1	Q1	What is the initial satisfaction value of "Pass Tryouts"?	What is the initial satisfaction value of "Prevent Dooring Incident"?
P1	Q2	What is the initial satisfaction value of "Exercise"?	What is the initial satisfaction value of "Bike Lane Usage"?
P1	Q3	Is the initial state of the model more satisfied, denied, or conflicted?	Is the initial state of the model more satisfied, denied, or conflicted?
P2	Q4	For each of the elements listed below, how many times over the simulation does the element become Fully Satisfied? (a) Have Summer Activity, (b) Pass Tryouts, (c) Exercise	For each of the elements listed below, how many times over the simulation does the element become Fully Satisfied? (a) Bike Lane Curbside, (b) Temporary Construction Plan, (c) Public Support
P2	Q5	How does "Join Soccer Team" generally evolve over the simulation?	How does "Public Support" generally evolve over the simulation?
P2	Q6	For each of the following satisfaction values, at which time point in the simulation do the most number of elements have the value. Note: In the event of a tie, choose the later time point (higher number). (a) Fully Satisfied, (b) Fully Denied, (c) Any Conflicted Value	For each of the following satisfaction values, at which time point in the simulation do the most number of elements have the value. Note: In the event of a tie, choose the later time point (higher number). (a) Fully Satisfied, (b) Fully Denied, (c) Any Conflicted Value
P2	Q7	Which intentions are Partially Denied at Time Point 1?	Which intentions are Partially Satisfied at Time Point 1?
P3	Q8	Which intention would you choose to satisfy to make "Exercise" Fully Satisfied?	Which intention would you choose to satisfy to make "Prevent Unloading in Bike Lane" Fully Satisfied?
P4	Q9	On the previous page, we ask the question: 'Which intention would you choose to satisfy to make "Exercise" Fully Satisfied?' You answered [insert Q8 choice]. Please explain your answer to this question.	On the previous page, we ask the question: 'Which intention would you choose to satisfy to make "Prevent Unloading in Bike Lane" Fully Satisfied?' You answered [insert Q8 choice]. Please explain your answer to this question.
P4	Q10	How would assigning "Drive to and Play Soccer" the value Fully Satisfied influence the model?	How would assigning "Parking Curbside" and "Temporary Construction Plan" the value Fully Satisfied influence the model?
P5	Q11	Click here for a PDF to compare three different scenarios of the Summer model. Should you choose to join a book club, community garden, or soccer team?	Click here for a PDF to compare different scenarios of the Bike Lanes model. How should you construct the bike lanes?
P6	Q12	On the previous page, we asked you to compare three different scenarios of the Summer model and answer the question: 'Should you choose to join a book club, community garden, or soccer team?' You answered [insert Q11 choice]. Please explain your answer to the previous question.	On the previous page, we asked you to compare different scenarios of the Bike Lanes model and answer the question: 'How should you construct the bike lanes?' You answered [insert Q11 choice]. Please explain your answer to the previous question.

Tbl. 3.2 lists the steps in our protocol for each treatment group. Parts 0, 1, and 5 are common among all subjects. In Part 0, we obtained *informed consent* from all subjects and had them rate their previous experience with goal modeling. In this step we also had them complete a short (seven question) color deficiency test to ensure subjects met the inclusion criteria (see Sect. 3.2.4). In Part 1, subjects completed two training modules, one introducing goal modeling more generally using VidGM, and the other introducing the minimal required subset of the Evolving Intentions framework (using VidSim) to complete the study. Here, we used the Course model and Employment model (see Tbl. 3.2, described in Sect. 3.2.3). Specifically, the Course model was used as part of our training materials, including videos, to introduce new concepts, whereas, we used the Employment model as the experimental test for Part 1. After each module, subjects were asked questions to test their understanding of each respective topic: goal modeling, and simulations. These questions allowed us to establish a base-level competency of our subjects when completing goal model tasks. In Part 5, we asked for feedback from the participants and collected remuneration and followup information.

Parts 2–4 (see Tbl. 3.2) varied based on the subjects' randomly assigned treatment group. All subjects completed the 'Training: EVO' module, which used the Employment model in Part 2 and 3. Afterwards, subjects answered questions about the Bike and Summer models (see Tbl. 3.3) after examining each model. What varied is which model (i.e., Bike or Summer) they answered questions about using EVO and whether they answered questions about a model before or after completing the EVO training. This allowed us to control for both variations in the models and a learning effect.

3.2.4 Experimental Conditions and Subject Information

We conducted our investigation in early 2023. All subjects were required to be proficient in English and be known to not have a color vision deficiency (i.e., colorblindness), as well as

Table 3.4: Subjects' Reported Familiarity with Topics

Subject Group	Median Familiarity (0: None, 10: Complete)				
	English	RE	iStar	Tropos	GRL
1.EVO:Summer	10	1	0	0	0
2.EVO:Bike	10	0.5	2.5	0	0
3.CTL:Bike	10	0.5	0	0	0
4.CTL:Summer	10	0.5	0	0	0

apply to participate in the study. Subjects were also required to be enrolled as undergraduates at Smith College and have completed our data structures and algorithms course, which is open to all majors. Subjects were excluded if they did not meet the inclusion criteria or had a prior relationship (i.e., conflict of interest) with our lab. Thus, we recruited subjects through a department mailing list and flyers were posted in the science buildings on campus, see the online supplement[1] for details.

Once subjects applied for the study, they were brought into the lab to complete the one-hour study in-person on our lab machine in a soundproof room. Since the subjects were not required to have training in goal modeling, one researcher was on hand to answer any questions after each training module.

We recruited 32 undergraduate students to participate. All subjects achieved a perfect score on the color vision test. During Part 0 of our protocol (see Tbl. 3.2), we asked subjects to rate their familiarity with written English, requirements engineering (RE), and three GORE languages (where 0 is no familiarity and 10 is complete familiarity). Tbl. 3.4 reports the median familiarity score for each treatment group. Subjects rated themselves highly with respect to English. One subject in each of Groups 1, 3, and 4 rated their familiarity with English between six and nine, while all other subjects selected ten. The median scores for RE and iStar were low but non-zero. It is likely that some of our participants have completed the software engineering course at our institution. While this course varies in its coverage of RE each semester, a (non-lab associated) instructor at our institution has explicitly taught iStar in two recent semesters. However, we did not distribute the participants with previous iStar experience across treatment groups as we had no reliable measure of whether subjects had

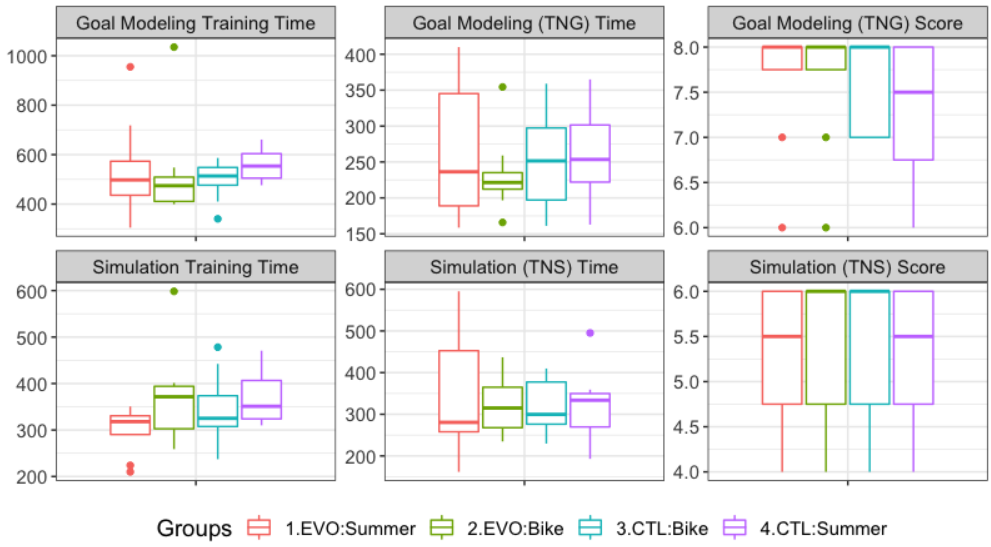


Figure 3.5: Scores (counts) and timing data (in seconds) for the goal modeling and simulation training. Maximum score for goal modeling was 8, while maximum simulation score was 6.

retained any goal modeling knowledge at the time of our study. We did not expect subjects to have any familiarity with Tropos or GRL, but include these scores for completeness.

3.3 Results

In this section, we answer our research questions using data collected in our investigation.

3.3.1 RQ0: Establishing a Baseline for Comparison

We begin by answering RQ0: Do modelers perform similarly on basic cognition tests, given a consistent training protocol? All data collected during Part 1 of our protocol (see Tbl. 3.2), was used to establish a baseline both to compare between subjects and evaluate to what extent subjects understood the training.

First, subjects watched VidGM video and answered eight questions about goal modeling (TNG), and then they watched VidSim and answered six questions (plus one qualitative question) about simulation of models over time (TNS), see the online supplement[1] for questions. All answers were scored as correct or incorrect. Fig. 3.5 reports box plots for subjects'

training time, test time, and test scores (from left to right), for both the goal modeling and simulation training. Each box plot is sorted by treatment group and times are reported in seconds. For the goal model training (see first row in Fig. 3.5), most subjects spent 8–9.5 minutes on the initial training (i.e., rounded first to third quantile), which included a 7.5 minute video), most subjects took 3–5 minutes to answer the TNG questions, achieving scores between 6–8. For the simulation training (see second row), subjects completed the initial training (including a 5 minute video) in 5–6.5 minutes. They then answered the TNS questions in 5–6.5 minutes, achieving scores between 4–6. From the box plots, we cannot observe any meaningful difference between treatment groups. For completeness, we used the *Kruskal-Wallis Rank Sum* (KWRS) test [42] to test for any variability between treatment groups. Our null hypothesis was that the treatment groups performed equally well on the questions, both in terms of score and time. We failed to reject our null hypothesis ($p \not< 0.1$), meaning that we could not detect a difference between the treatment groups.

Additionally, subjects were asked to document any questions they had after reviewing the training videos (and associated documents). For the goal modeling training (TNG), eighteen subjects left a substantive question. These questions were most commonly about the evidence pairs, differences in contribution link types, and specific choices made by the modeler of the example. There were two questions about the differences between the training materials and iStar. For the simulation training, fourteen subjects asked a question. The vast majority of them were about choice and usage of evolving functions. Specifically, to explain the behavior of an intention without an assigned evolving function. Anecdotally, based on our experience teaching goal modeling, these questions are consistent with those asked in the classroom. Since subjects were not trained modelers, researchers answered subjects' questions before proceeding to the next part of the study.

We conclude that subjects performed similarly on basic cognition tests, given a consistent training protocol.

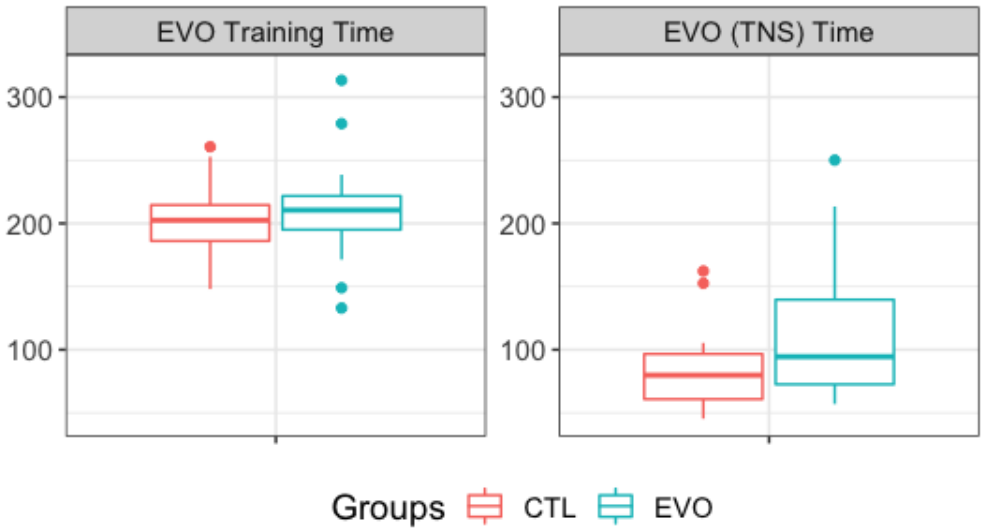


Figure 3.6: Timing data (in seconds) for the EVO training.

Table 3.5: EVO training score frequencies, grouped by order.

	Score Freq.		
Group	0-4	5	6
CTL	0	3	13
EVO	0	4	12

3.3.2 RQ1: Subjects' Use of EVO

Second, we consider RQ1: To what extent are subjects able to use EVO to understand and reason about goal models. Given our results of RQ0, we investigate this question between-subjects, using a nested 2x2 design, where in Parts 2–4 (see Tbl. 3.2), each subject completed the EVO Training module and answered questions about the Bike and Summer models (see Tbl. 3.3), one using the EVO feature and one without. Thus, we compare the EVO training module and the results of each model separately. We divide RQ1 into two sub-questions: (a) Is our training sufficient for learning how to use EVO? and (b) To what extent were subjects able to answer modeling and simulation questions with and without EVO?

(a) EVO Training. All subjects completed a common EVO training module consisting of six questions. We matched treatment groups 1 & 2 (EVO) and 3 & 4 (Control or CTL), to understand if there were any effects in reviewing one of the experimental models (i.e., Bike

or Summer) first. Tbl. 3.5 lists the score data for the EVO training. All subjects achieved a score of 5 or 6 (out of a possible 6), and the groups are not distinguishable. Fig. 3.6 shows the box plots for the training and test times for the EVO Module. Subjects took between two and five and a half minutes to review the training materials and between one and four and a half minutes for the EVO questions. Our null hypothesis is that there is no significant variation between groups. We fail to reject this hypothesis (KWES, $p \not< 0.1$), unable to detect variations between groups.

Again, subjects were asked to document any questions they had after reviewing the EVO training, with nine subjects asking a question. Questions focused on understanding the simulation results and the differences between the EVO modes. Two subjects asked about the order of the percent (%) mode, which was further clarified. Thus, subjects learned and demonstrated proficiency in using EVO in under ten minutes.

(b) Answering Questions with EVO. We now review subjects' ability to answer the model questions listed in Tbl. 3.3. Q4 and Q6 were each scored out of 3, one for each sub-question. Q9 and Q12 were excluded from scores as they were used to validate the answers of Q8 and Q11, respectively. Thus, each model was scored out of 14.

Tbl. 3.6 lists median scores for each treatment group. Scores ranged between eight and fourteen for the Bike model, with a median score of thirteen. While for the summer model, scores ranged between nine and fourteen, with a median score of twelve. Combining the medians for the EVO and non-EVO users for each of the two models, we found that medians for EVO users were 13.25 for the Bike model and 11.75 for the Summer model, while the medians for non-EVO users was 12.5 for the Bike model and 12.75 for the Summer model. Thus, EVO produced a slightly better median for the Bike model but also a slightly worse median for the Summer model. The questions answered best by subjects were Q1, Q3, and Q5 (see Tbl. 3.3), with only one subject incorrectly answering each question between both the Bike and Summer models combined. The worst performing question was Q6(b) for the Summer model and Q6(a) for the Bike model. The phrasing of Q6 can be improved (see

Table 3.6: Median Scores for Bike and Summer Questions

Group	Bike Median	Summer Median
1.EVO:Summer	12	12
2.EVO:Bike	13	12.5
3.CTL:Bike	13.5	13
4.CTL:Summer	13	11.5

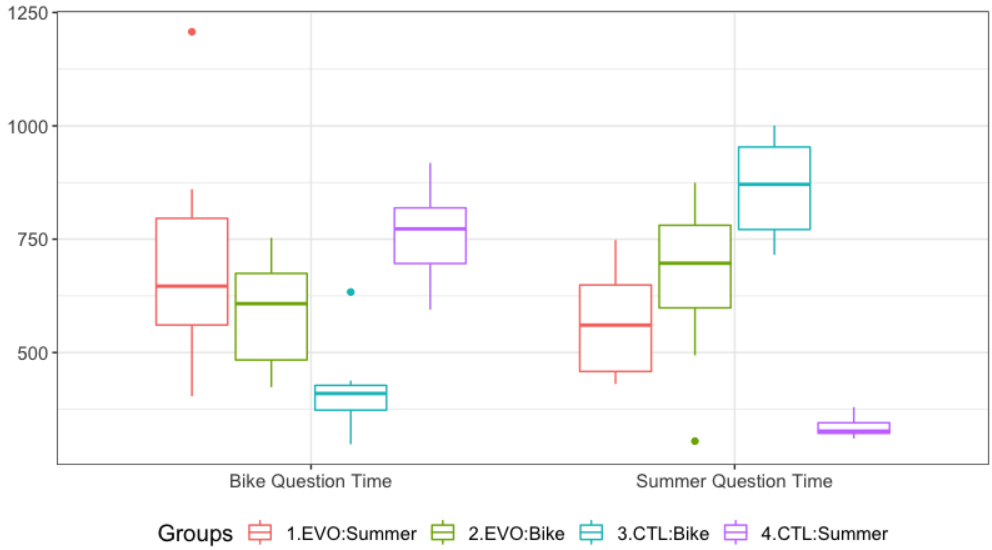


Figure 3.7: Timing Data (in seconds) for answering Bike and Summer questions (see Tbl. 3.3).

Sect. 3.4.1 for a discussion). Given the score data in Tbl. 3.6, we did not expect to find variations between groups (i.e., our null hypothesis) and, in fact, did not find any statistical difference between treatment groups (i.e., KWES, $p \not< 0.1$) with respect to the subjects' scores for Bike and Summer model questions.

We conclude that subjects were able to use EVO to understand and reason about goal models with a few errors, which is consistent with new learners.

3.3.3 RQ2: Comparing EVO with the Control

Next, we consider RQ2: How does using EVO compare with the control? We again break this research question into two sub-questions: (a) Does EVO help subjects make decisions faster? and (b) How did subjects perceive EVO?

(a) Bike and Summer Times. To measure subject completion times, we added their times from Pages 1, 2, 3, and 5 (see Tbl. 3.3). Pages 4 and 6 were excluded because they contained solely free form answers where subjects' time depended on the length of their answer.

The times for both models are comparable, ranging from five to twenty minutes. Fig. 3.7 gives the box plot for each treatment group for the Bike and Summer model question time. In the Bike model (left side), groups 2 (green) and 3 (blue) used EVO to answer the questions and visibly lower time. Again, our null hypothesis is that there is no difference between treatment groups. Using the KWES test, we find the times for the Bike model to be significantly faster ($p < 0.01$). In the Summer model (right side), groups 1 (red) and 4 (purple) used EVO to answer the questions and also have visibly lower time. Again using the KWES test, we find the times for the Summer model to be significantly faster ($p < 0.001$).

Upon further inspection of Fig. 3.7, we observe a possible learning effect—the results are more pronounced when the control group used EVO (i.e., group 3 (blue) for the Bike model and group 4 (purple) for the Summer model). Yet, when we conduct a pair-wise comparison based on treatment group order and EVO, we do not find a significant difference with respect to order but we do find one with respect to using EVO; thus, we hypothesize that the interaction of subjects being in the control group and using EVO may contribute to this additional benefit. Therefore, we found a significant effect between the treatment groups with respect to the time required to answer the Bike and Summer questions.

(b) Qualitative Perspectives. Finally, we performed a qualitative analysis on the question, "Compare and contrast the colored views with the non-colored views, which do you prefer? Why?". All subjects preferred the EVO view over the control. More than half said that EVO was faster and/or easier to use. Other comments include that EVO was more intuitive, better for comparing models, and improved subjects' high-level understanding of the model. While no critiques of EVO were present in this question, we discuss subjects' recommendations for improving EVO in Sect. 3.3.4.

Table 3.7: Recommendations for Improvement

EVO Improvements

- Add ticks or an outline to time mode. (x4)
- Choose prettier colors (and better fonts). (x2)
- Better contrast between text color and EVO color. (x2)
- Change conflict colors:
  - All conflicts the same color.
  - (P, P) should be grey, reduce visual noise.
  - Use green/yellow for conflicting evidence pairs.
- Left to right arrow on time mode.
- Eliminate possible left-right bias in % mode.
- Colors may not be accessible to all users. (x2)

Goal Modeling Improvements
- Add goal prioritization in models. - Organize models as decision tree. - Improve visualization of links (maybe with color). - Create model-level metrics (in a table). - Distinguish between OR and XOR links. - Make evolving functions more explicit. - Add more possible values for (s, d).
Study Instrument Improvements
- Clarify difference between + and $+S$. (x2) - Better explain evolving functions. - Clarify difference between initial state and time point 0. (x2) - Clarify difference between % and Time mode. - Organize handout landscape with models left to right. - Text too crowded/overlap, make images simpler/larger. (x2) - Change "become Fully Satisfied" wording in Q6. - (F, F) looks black, not dark purple. - Add progress bar to questionnaire.

We conclude that subjects preferred using EVO over the control. Subjects completion times were measurably faster with EVO.

3.3.4 Improvements and Recommendations

Finally, we address RQ3: How do subjects rate the study instruments and experience? To answer this question, we collected optional quantitative ratings after each module and qualitative reports at the end.

Table 3.8: Average (mean) subjects' rating of their difficulty with three study aspects (where 0 was no difficulty and 10 was complete difficulty): understanding the scenario description, understanding the model, and answering the questions.

	Scenario	Model	Questions
Phase 1	3.7	5.0	4.8
EVO	2.6	2.6	2.3
Summer	3.4	4.2	4.1
Bike	3.6	4.2	4.6

For each of Parts 1–4 in Tbl. 3.2 (i.e., the initial training sequence, the EVO training, the Summer model, and the Bike model), subjects rated their experience completing each part. They were asked to rate their difficulty with the three aspects (where 0 was no difficulty and 10 was complete difficulty): (i) understanding the scenario description, (ii) understanding the model, (iii) answering the questions. Tbl. 3.8 gives the average difficulty rating for each aspect and each part. Subjects had the most difficulty during the initial training phase, which seems appropriate because subjects had very limited familiarity with RE and goal modeling (see Tbl. 3.4, discussed in Sect. 3.2.4). Subjects perceived the Bike scenario and questions as slightly more difficult than the Summer model but perceived the models similarly. The EVO training was rated as the least difficult part, with average scores of 2.3-2.6. While this provides additional data for our assertions in RQ1, comparing between the scores in Tbl. 3.8 is confounded by the fact that the EVO training was the shortest module and built on the Phase 1 training.

Finally, we ask subjects for suggestions and additional comments. Specifically, to gather suggestions, we asked the question: "what suggestions or changes would you recommend to the developers of this goal modeling language (and tool)?" Tbl. 3.7 lists the recommendations provided by subjects, organized into three categories: improvements that can be made to EVO, goal modeling, and our study instrumentation.

Subjects made a variety of recommendations about improving the look and feel of EVO—from changing the colors of conflicting evidence pairs to adding ticks to show time points in the Time mode. We are aware of the accessibility issues associated with red-blue color

vision deficiencies (see Chapter 5 for details).

Since this study was conducted in isolation from tooling and other approaches, many of the goal modeling recommendations have already been investigated by other approaches. For example, goal prioritization, XOR links, model-level metrics, and quantitative valuations have all been investigated by researchers [14, 13, 2, 5]. We found the recommendation about improving the visual aspects of the links of interest and may pursue this in future work.

Finally, subjects recommended improvements to our study instrument. Subjects recommended clarifying the differences between link types, evolving function types, and the difference between the initial state and time point 0. Specifically, with respect to EVO, one subject thought more explanation was required to understand the difference between % and Time mode. Other comments included adding a progress bar and improving our study handouts and questions. Three subjects (excluded from Tbl. 3.7) encouraged the developers to implement the EVO feature.

Six subjects provided additional comments. Of these responses, three mentioned that the survey was long/hard, one said that they do not like goal modeling, one thought that (F, F) is the color black, and the final comment explained an inconsistency in the subject's answer to a previous question.

We conclude that subjects rated the study instruments and experience as suitable and not overly difficult; yet, roughly 10% reported that the study was long or hard. Subjects found the initial training most difficult and the EVO training easiest.

3.4 Discussion

Next, we describe our lessons learned, compare the bike and summer model, and discuss the validity of our experiment.

3.4.1 Lessons Learned and Implications for Research

Subject Background and Recruitment. We developed this study instrument over a six-month period. We first iterated the instrument with individuals in our lab, then completed a small pilot with four subjects. The purpose of the pilot was to evaluate the quality of our instrument and understand what timing data was generated from our Qualtrics$_{®}$ XM platform. The pilot helped us improve the quality of the data we collected. We added opportunities for subjects to take breaks and originally collected one timing value for Q1-12 in Tbl. 3.3. We discovered these values varied dramatically based on how much text subjects entered in the free form questions. As listed in Tbl. 3.3, we separated these questions across six pages (see Page column) and added timing information to each page. It was extremely difficult to recruit subjects for a survey that took a full hour. Due to the tax legislation in our country and policies at our institution, we were not able to offer remuneration in an amount over $20 USD. We launched three separate iterations of the study. Our first emailed researchers within the goal modeling community and targeted trained modelers. We received five responses and of these, only one completed the study instrument. Our second attempt was to recruit subjects within a large software engineering class with Tropos instruction at another institution, again receiving only one completed session. After two unsuccessful attempts, we pivoted to recruiting subjects for in-person sessions. We updated our protocol to include additional training and recruited students as described in Sect. 3.2.4. There is something cognitively different about coming into a research lab for an hour as opposed to completing a survey online at home for an hour, even when remuneration amounts are consistent. We had sufficient volunteers for our in-person version and felt this was an important lesson learned.

Improvements to the Study Instrument. We reviewed the questions and supplemental information from the study by Hadar et al. [26] and iteratively developed our study instrument. We encourage other researchers to use and adapt our survey instruments. In completing this investigation, we discovered areas of improvement. For example, in question

Q6 (for both the Bike and Summer models, see Tbl. 3.3), we asked "how many times over the simulation does the element become Fully Satisfied" which would have been better rephrased as, "how many time point(s) over the simulation is the element Fully Satisfied".

It was sometimes difficult to achieve task equivalency. For example, the tasks in question Q8 (see Tbl. 3.3) are not exactly matched between models. The correct Q8 answer for Bike model was *none of the above* because no intentions fulfill Prevent Unloading in Bike Lane. To satisfy Exercise in the Summer model requires either Water-Weed-Enjoy Garden or Drive to and Play Soccer, but we did not include Drive to and Play Soccer as an option, intending subjects to select Water-Weed-Enjoy Garden. Since the Bike model had a *none of the above*, we included the same for the Summer question, yet this resulted in subjects choosing it because they wanted to select Drive to and Play Soccer. In a future iteration of this instrument we would change the selected intention for the Bike model and remove the *none of the above* option.

In our analysis, we were unable to detect any differences between scores on the models with or without EVO. Future work is required to determine whether our study instrument is sufficiently discriminatory. One of the aspects we iterated on was the length and complexity of the questions we asked in this study. We opted for a balance in these factors to ensure that subjects would complete the study in one hour, which we agreed upon as a reasonable upper bound.

Statistical Methods. Given our per group sample size, any statistical test will have lower power to make conclusions (see Sect. 3.4.3). In Sect. 3.3, we used the Kruskal-Wallis rank sum (KWRS) test to evaluate if there are distinct groupings within our sample data [42]. The KWRS test is valuable for small sample sized data because it does not make assumptions about the distribution of the data and is not influenced by data points that vary greatly in magnitude, which is useful for time data.

3.4.2 Comparing Bike and Summer Models

As introduced in Sect. 3.2.1, we explored our research questions between-subjects. In Sect. 3.3, we found a statistically significant difference between using EVO and the control in the time it took subjects to answer questions about both the Bike and Summer model. Yet, in this test we cannot directly compare the times associated with the Bike and Summer model or understand the impacts of variations between individuals.

We briefly explore variations of the time it took subjects to answer the test questions (i.e., our *dependent variable*). We compare test times given three factors (*independent variables*): (i) whether the subject used EVO, (ii) whether it was the first or second measurement for that subject, (iii) whether the measurement was of the Bike or Summer model. In order to identify which factors are significant, we compared within subjects by fitting multiple linear mixed-effects models and then conducted a model comparison with repeated measures data using a likelihood ratio test (i.e., ANOVA) [38]. We used a linear mixed-effects model to account for non-independence (i.e., there were two measurements for each subject).

Comparing the full model to one with interactions between factors showed that the interaction terms in the model are not significant ($p > 0.05$). We found the EVO factor to be significant ($p < 0.001$), meaning that within-subjects there was a difference in the time it took subjects to answer questions with EVO as opposed to without EVO. The order of whether subjects were given the control or the treatment first was significant ($p < 0.001$), implying that there was a learning effect over time. Which model is measured was not significant ($p > 0.05$), meaning that there is no significant difference in the times for the Summer and Bike models. Since there is no significant difference between models and no interaction effect, we can analyze this as a two-way ANOVA where using EVO and order of EVO presentation are the two factors. Using a statistical power test for repeated measures ANOVA within-subjects with a medium effect size, we found that the minimum sample size using G*Power [11] for our experiment was 56. Thus, we have low statistical power. We did not find any difference between the Bike and Summer models and found the presence of a

learning effect within subjects.

3.4.3 Threats to Validity

We discuss threats to validity using the categories in [57].

Conclusion Validity. Our main threat in this experiment is low sample size. Having 32 subjects spanning four treatment groups is considered a low sample size. Thus, we chose to conduct our main analysis between-subjects to mitigate this threat. We may have experienced a reliability of measures threat, as subjects asked questions about the wording of Q6 (see Sect. 3.4.1). We wrote scripts to analyze our data wherever possible and automatically recorded page completion times to ensure reliable measurements. Qualitative data was randomized before review and categorization. Different researchers conducted the in-person and data analysis components to reduce researcher bias. To mitigate variations in treatment implementation, we standardized the experimental setup by using our online platform, videos, and pdf handouts to ensure that the subjects had equivalent training materials (see Sect. 3.2.2), and maintained our laboratory setup throughout the study period, to ensure a consistent in-person experience. We do not believe there is a random heterogeneity of subjects risk, since our population was homogeneous, having similar knowledge, abilities, and previous experience with English, Tropos, and RE (see Tbl. 3.4). In a future study, we would collect data about subjects' year in the undergraduate program (e.g., first-year, seniors) to further mitigate this risk.

Internal Validity. We explicitly designed our study to control for a learning effect or maturation risk (i.e., where one group learns a treatment faster than another). We gave opportunities for subjects to take breaks if they were fatigued and shortened the instrument wherever possible. We controlled for an instrumentation effect in our 2x2 design; yet, the Bike model questions may have been slightly harder (see Sect. 3.3.2). Our voluntary study with cash remuneration may have experienced a selection effect. To our knowledge, no subjects used BloomingLeaf or EVO prior to the study.

Construct Validity. We conducted multiple pilot mini studies (not discussed in this thesis) to ensure that our study instrument was measuring our intended constructs. In one such study, we found that our unit of time measure was inaccurate because it included too many questions; hence, we divided the questions across multiple pages as listed in Tbl. 3.3 and isolated qualitative questions. We collected data in multiple forms (e.g., scores and times) and asked different types of questions to mitigate mono-method and mono-operation biases. As always, we have threats of *hypothesis guessing* and *evaluation apprehension*. Some subjects expressed nervousness asking if they needed to review data structures or read about goal modeling before participating. Some students who took a software engineering course may have scored better overall; yet, our common training protocol may have limited this threat.

External Validity. Our setting was not reflective of the use of EVO in the "real world". We conducted the experiment one-on-one in our lab using a survey, instead of embedding EVO within a goal modeling tool (e.g., BloomingLeaf). We were limited by the size of the models we used, as we wanted to keep the time it took to complete the study to around an hour. Thus, we were unable to validate EVO on large models that are more reflective of complex "real world" scenarios. Our largely homogeneous population of undergraduate students means that we cannot generalize to the broader population of individuals engaged in RE activities, but given the limited prior knowledge of our subjects (see Tbl. 3.4), these results may in fact, generalize. Additional experiments with different populations, problem domains, and larger models for scalability are required to generalize these results.

3.5 Related Work

Recent work has critiqued the adaptability of GORE approaches [37]. In this chapter, we address this gap by improving the interpretability of models and analysis. As already introduced in Sect. 3.1, Hadar et al. [26] and Siqueira [51] studied the comprehensibility of Tropos models with respect to Use Case models. While it is difficult to compare our results

with these studies because we only evaluate Tropos models, these studies were influential in the design of our study and the importance of controlling for the use of different models, while investigating the performance of subjects on analysis tasks.

Using color as a technique to improve visualizations of goal models has been a topic of recent interest within the community. In GRL, Amyot et al. has used colors to visualize analysis results using the jUCMNav tool [2], while TimedGRL used color in heat maps to visualize evolving GRL goal models [3]. In Tropos, Varnum et al. proposed using colors to help stakeholders interpret the evidence pairs used for intention evaluations [52]. At the same time, Oliveira and Leite proposed mapping the primary colors (i.e., red, blue, green) onto NFR soft goal labels and contribution links, allowing color values to be quantitatively calculated and propagated throughout the model [40]. Reddivari et al. investigated using visual analytics techniques to help in requirements negotiation [41]. Horkoff and Yu first investigated highlighting root/leaf nodes and conflicting alternatives in goal graphs to assist users in understanding analysis tasks [29]. Varnum et al. used a static set of colors whereas Oliveira and Leite use a large range of colors calculated dynamically. In reviewing these approaches, we chose to first validate the coloring approach of Varnum et al. because of its static nature, which made it easier to evaluate experimentally and understand whether color was an effective approach. Further research is required to validate the choice of colors in both approaches, and whether the dynamic nature of Oliveira and Leite's approach causes an additional cognitive load that reduces the overall effectiveness.

We built on the methodology of similar studies in RE for our between-subjects experiment and followed the guidance in [57] and [47]. Winkler et al. reported on a between-subjects 2x2 design similar to ours with sixteen subjects [56]. The authors assumed that the treatment group had increased precision and a reduction in time to complete the tasks due to working with direct output from the tool; whereas, the control group completed the task manually. We attempted to control for differences in tool usage by providing both groups with direct output from BloomingLeaf. Ghazi et al. reported a study comparing two nav-

igation techniques for requirements modeling tools [18]. They used time limits to motivate the participants to work as fast as they would on real tasks in industry, giving the subjects about five minutes to try out the tool. However, this may force subjects to work faster, which may result in worse results. To prevent this, we let the subjects take the time needed to review the training documents since our population comprised new learners. Santos et al. presented a quasi-experiment to explore the interpretability of iStar models given different concrete syntax [44]. Subjects were tasked with identifying defects in a goal model, a task we did not include in our study as it may have been too difficult for new learners and increased their fatigue.

3.6 Summary

In this chapter, we explored how using EVO to visualize evidence pairs impacts an individual's ability to reason with goal models that evolve over time. To do so, we conducted an IRB-approved between-subjects experiment with 32 undergraduate students. We found that when given a consistent training protocol for goal modeling and simulation, each treatment group performed equally well on the initial training modules, establishing a baseline for comparison between treatment groups. Subjects were able to learn EVO in under ten minutes and use the extension to make decisions. From this experiment, we concluded that subjects were able to answer goal modeling comprehension questions with EVO faster than without EVO but we did not find a significant difference between the scores of subjects who answered questions with and without EVO. Thus, there was no evidence that EVO has an impact on an individual's understanding of goal models. However, subjects had a positive response to EVO and all preferred the EVO view over the control, with most saying that EVO was faster or easier to use. Finally, our subjects, without prior training in GORE, were able to complete the instrument without much difficulty. By demonstrating the impacts of EVO, we increase the potential of automated analysis techniques in Tropos.

Chapter 4

Visualizing Solution Spaces

4.1 Introduction

In this chapter, we introduce the methods we implemented to assist users in exploring and selecting states from a solution space more efficiently. In Chapter 3, we validated that EVO provided users with a faster comprehension of goal models and their results after simulation. We extend the EVO coloring tool for the Next State view in BloomingLeaf, which assists users in understanding a solution space.

As briefly introduced in Sect. 2.3, BloomingLeaf uses a CSP solver to simulate paths and allow users to explore these paths. The model structure, as well as the intention evaluation labels, act as the constraints placed upon the solver. When a black-box solver produces only a single solution, users may be suspicious about whether the proposed answer is ideal or believable, given the provided constraints. As well, users may want to explore other results that satisfy the same criteria or create their own custom paths. Ideally, users would like to explore the potential solutions but depending on the size of the solution space (i.e., state explosion problem [9]) and complexity of any individual state, current visualization techniques (e.g., [50, 4]) are insufficient.

In this chapter, we investigate the problem of solution space exploration when the content of each state is an entire model. We propose the use of valuation-based filtering and coloring

to assist users in exploring and selecting states from a solution space more efficiently. We implemented the EVO State mode for the Next States view to allow users to review individual states more efficiently, and we implement Percent mode to allow for a high-level overview of the contents of the entirety of the state space. Additionally, we introduce valuation-based filtering to allow users to reduce the solution space to a more manageable size in order to select custom states from it, as receiving an overview of the solution space with EVO is not enough to fully explore alternate scenarios. We demonstrate how these techniques can be used on a fully worked out example to allow custom state selection. We conduct initial measurements of the time savings and state space reduction created by the valuations and color filtering, and discuss future directions of this project.

4.2 Background and Constraint Satisfaction Problems

In this section, we return to our description of a goal model from Sect. 2.1. Here, we formally define an evolving goal model and describe how it is captured as a CSP. Within the Evolving Intentions framework [25], an *evolving goal graph* M is a tuple $\langle A, G, R, EF, C, maxTime \rangle$, where A is a set of actors, G is a set of intentions, R is a set of relationships over intentions, EF is a set of evolving functions (e.g., C, DS), C is a set of constraints over the time points in the graph, and $maxTime \in \mathbb{N}^+$ is the maximum absolute time over which any time point is defined (adapted from [25]).

For the purposes of this chapter, M is a graph (henceforth called a *model*) consisting of intentions $g \in G$ and these intentions are assigned evidence pairs as valuations. An *evidence pair* is a pair (s, d) where $s \in \{Full\ (F), Partial\ (P), None\ (\bot)\}$ is the level of evidence **for** and $d \in \{F, P, \bot\}$ is the level of evidence **against** the fulfillment of g. The cross product of s and d results in nine valuations, denoted as the universe $\mathcal{E}$ (see legend in Fig. 2.2). The evidence pair assignments for intentions are determined by the model constraints defined in R, EF, and C.

The model can then be simulated over a series of time points Π, known as a time point

path [25]. For a given time point $t \in \Pi$, the evaluation of g at t, is a mapping $G \times \Pi \rightarrow \mathcal{E} \cup \{\perp\}$. Thus, a complete evaluation of M at t is the total mapping $G \times t \rightarrow \mathcal{E}$ resulting from the repeated application of all constraints within R, EF, and C. Simply put, a complete evaluation path is the complete evaluation of M at each time point in Π, which we represent as a CSP.

General purpose CSP solvers determine if a satisfying solution exists for a given CSP and if so, provide a possible solution. They can be used to describe singular problems (i.e., states) and multistep or temporal problems (i.e., paths). Solvers have been used to efficiently find a single solution or find all possible solutions (i.e., collections of possible states or paths through the solution space). Given a state space, where each state has a set of variables that are given values, and a set of constraints on the variables, a constraint satisfaction problem (CSP) is defined as solved when all variables are assigned values that satisfy the given constraints.

A CSP is defined as a triple $\langle V, D, Q \rangle$, where V is a set of variables, D is a set of the respective domains of values, and Q is a set of constraints [43]. When constructed as a CSP, our aim is to find an evaluation for each intention in the graph ($g \in G$) at each time point ($t \in \Pi$). Thus, the CSP variables V are a set of intentions at all time points (i.e., $|V| = |G| * |\Pi|$). Initially, the domain d of each variable v is the set of evidence pairs ($\mathcal{E}$). Each element in R, EF, and C forms constraints Q over the elements in V. Thus, the valuation of all variables in $|V|$ is a complete evaluation path of model M.

4.3 Filters & Visualizations

In this section, we describe two techniques to assist users in creating their own path from a solution space of an evolving model. Fig. 4.1 shows a screenshot of our extension of BloomingLeaf in the state exploration view (called the *Next States* window, see Chapter 2) with filters and visualizations applied. BloomingLeaf invokes the JaCoP constraint solver [33] to find satisfying assignments for each variable and displays the first option for the selected

time point (t_3) on the center canvas of the Next States window. The left panel shows the existing filters, as presented in [31]. The user can select from the many possible states or incrementally navigate through each state with arrows in the left panel. The new Color Visualization is shown in the top toolbar (called EVO, '%' selected) and the valuation filtering is shown in the right panel.

4.3.1 Color Visualization.

We introduce two color overlays (EVO's State and Percent modes) to assist users in understanding the model and selecting future states. The first (see Fig. 4.2) colors the background of each intention in the graph with the color associated with the assigned evidence pair. This view is similar to the coloring applied to the model path in Fig. 2.6. Seeing the colors for each intention allows users to more quickly understand the valuations in the viewed state without reading each of the evidence pairs. The second is the percentage mode, which is selected in the EVO slider on the top toolbar and shown on the canvas in Fig. 4.3. When percent mode is selected, the background of each intention is colored with the percentage of states in the solution space where the intention has each evidence pair assignment.

4.3.2 Filtering Intention Valuations.

To allow users to sort through and find desired solutions, we added an additional panel on the right-side of the window that enables the user to add and remove filters, which we called Intention Filters. Unlike the filter on the left panel (from [23]), it filters solutions based on the valuation of individual intentions, returning only solutions where intentions have the specified evidence pair. This act reduces the solution space, allowing users to focus only on the options that meet their aims without having to manually evaluate all permutations of an intention. For example, when Build Review Algorithm is selected on the center canvas in Fig. 4.1(a) (see red box), then the Intention Filter panel is populated with the information for this element and allows the user to add a new filter.

4.3.3 Usage Scenario.

While each of these filters may seem simplistic on their own, taken together they form a powerful mechanism for users to create and validate their own paths. In the illustrative example, after generating the path (see Fig. 2.6 and Chapter 2) and exploring all possible states for t_3, there are 924 states (see Fig. 2.2). Fig. 4.1 shows a step-by-step demonstration of how to handle this scenario. First, we recommend turning on EVO % mode to see a high-level view of the solution space. Since the user is looking for a state where Build Review Algorithm is satisfied, they select the node and choose Satisfied $(F, \perp)$ in the right panel, limiting the number of states to 132 (see Fig. 4.1(a)). With a second valuation filter applied to Get Contract with Vendors, there are 44 possible states (see Fig. 4.1(b)). From here, the user can see that much of the variability in the remaining states exists within a few elements. If the user had applied the valuation of None $(\perp, \perp)$ to Hire and Train New Workers, then only 6 similar states remain (see Fig. 4.1(c)). Finally, the user turns on EVO in State mode and clicks through the remaining states, and selects #5 as their preferred one (see Fig. 4.1(d)). By selecting Save & Close (see Fig. 4.1) the CSP solver generates the remainder of the path given this updated state. Alternatively, the user could click Explore Next States to generate all possible solutions for the next time point (t_4), given the updated choices for t_3.

4.4 Discussion

Next, we describe the limitations of this approach, show our initial evaluation, and discuss its validity.

4.4.1 Benefits.

Our work allows for the more efficient exploration of a solution space. The intention valuation filters reduce the number of states in the solution space that need to be manually considered. Valuation-based coloring through EVO State mode allows for the quicker cognitive processing

of evidence pairs in states, since users do not need to read each evidence pair. The EVO % mode shows an overview of the composition of the state space and allows users to verify the absence of a state. For example, given the filters applied in Fig. 4.1(b), no state exists where Get Contract with Vendor is Denied (i.e., colored red).

4.4.2 Limitations.

Our current implementation is limited to the Evolving Intentions framework and needs to be expanded to work with other CSPs. Within this framework, we require users to generate a full path before exploring the state space. Another design choice would allow users to explore states from the initial model. Further, the color palette may not be intuitive across an international audience. We are extending our implementation to enable users to select the palette colors, including a palette for individuals with a color vision deficiency. Finally, there is an upper bound on the number of states our solver will generate before timing out. The illustrative example without any MP and DS functions produces 1,740,288 states if strong conflict (F, F) is removed from the domain d of each variable v and a memory error if kept. In these situations, it is often the case where the scenario has not been fully explored by the users. We investigate a methodology to help users explore state spaces more efficiently.

4.4.3 Initial Evaluation.

Our initial evaluation work focused on demonstrating the feasibility and effectiveness of our approach. We took measurements of the time savings and state space reduction created by the valuations and color filtering.

Our initial trials to validate EVO measured the time it took to physically click through the states in search of a solution with and without EVO State mode enabled. We measured exploring the entirety of the solution space as well as finding a solution with specific evidence pairs for one, two, or three intentions, including two intentions that were close and distant from one another.

Table 4.1: Rate of state review with and without EVO.

	Rate of Review (states per second)				
	0	1	2-Close	2-Distant	3
Base	5.37	4.08	4.03	2.08	0.66
EVO	5.78	5.64	4.89	4.98	4.69

Table 4.2: Best and worst case percentage reduction in states using one or two valuation filters.

Model		Percent Reduction			
#	States	1-Worst	1-Best	2-Worst	2-Best
1	68178	80.0%	98.8%	99.2%	99.6%
2	1005	6.2%	98.8%	30.4%	99.7%
3	1232	37.6%	96.1%	68.8%	98.7%
4	3122	88.3%	98.9%	97.9%	99.6%
5	608	50.0%	96.0%	75.0%	98.6%

Comparing the speed of searching for close and distant intentions allowed us to test cognitive processing delays. Tbl. 4.1 lists the averaged number of states that were reviewed per second. In all cases, it was faster to iterate through the states with EVO. As the number of evidence pairs requiring review increased the effects of the EVO state view improved. Users could review double the states when reviewing two distant or three intentions.

To test the effectiveness of applying intention valuation filters, we measured the percentage reductions in the size of the solution space after applying one to two filters on five different models. Depending on the number of constraints in Q, the number of initial states could be anywhere between one hundred and a million. Tbl. 4.2 lists the percentage of states that are removed after applying one and two filters in the best case and worst case. We chose these cases based on visual inspection of the EVO % mode. From Tbl. 4.2, we observe that with a variety of models, in the best case, we reduce the number of states by 96-98% with one intention and 98-99% with two intentions. The worse case reductions were as low as 6%, with averages between 50 and 75%, which indicates that much of the reduction comes from the user choosing an appropriate intention.

In summary, using intention valuation filters creates a greater than 95% reduction in the number of states in the best case, and using EVO more than doubles the rate of manual

review when reviewing multiple intentions.

4.4.4 Threats to Validity for Initial Evaluation.

Finally, we discuss threats to validity [57]. Our main threat is that our initial validation was completed by the researchers of this project, who are experts in goal modeling and the BloomingLeaf tool. The rate of state review values listed in Tbl. 4.1 may vary with individuals who are less familiar with BloomingLeaf. This fact threatens conclusion validity, as we cannot say there is a significant relationship since we did not have plausible subjects. As well, our setting was not reflective of how stakeholders would learn to use and apply these methods in the "real world", posing a threat to external validity. Further experiments with different populations, problem domains, and contexts will be needed, as well as validating EVO Percent mode (see Sect. 4.6 for further discussion).

We were limited by the size of the models we used, as we wanted to keep the time it took to complete the study to around an hour. Thus, we were unable to validate EVO on large and complex models that are more reflective of "real world" scenarios.

4.5 Related Work

In this section, we describe related work in state space visualizations. As already stated in Sect. 2.3, we reimagine the work of Hu and Grubb, who reduced the size of a CSP domain and solution space with generic filters [31].

CP and CSPs are found in many domains within the broader software engineering (SE) community, from automating software design [53] to debugging [58] and verification [21, 6], but have seen limited use in requirements engineering (RE) and GORE [28, 25]. Within GORE, the *propositional satisfiability problem* (i.e., SAT problem), and associated solvers, are more prominent [36, 30, 17]. Both SAT and CSP solvers have high computational complexity and should only be used for problems not solvable in polynomial time, such as

backwards analysis or scenario generation.

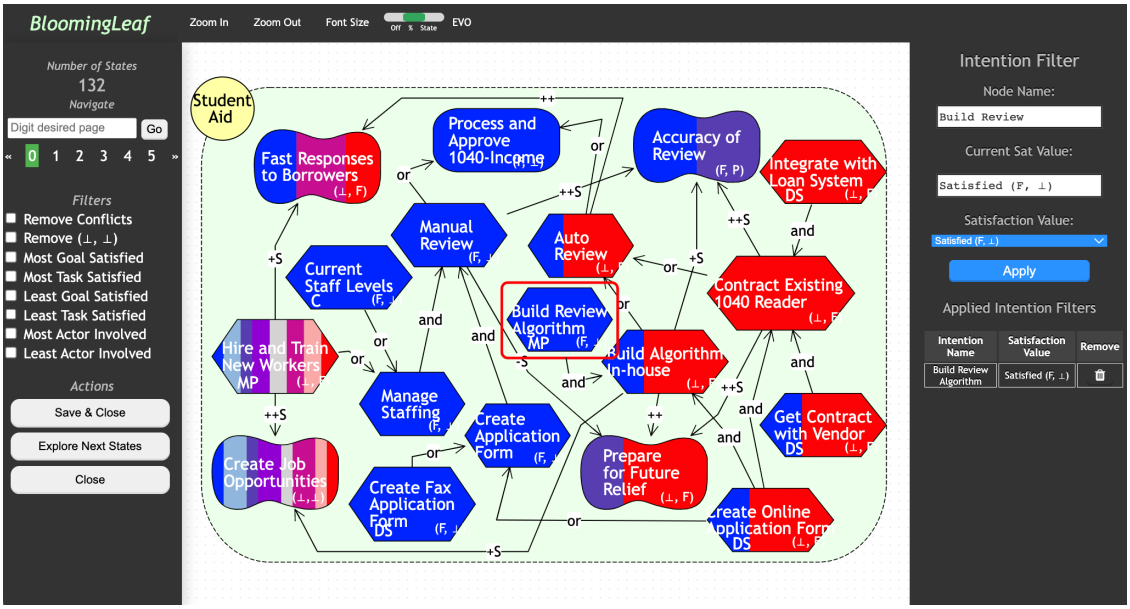
Since the late 1990s, there has been significant work on visualizing and debugging constraint programs [45, 20]. Researchers investigated visualizing search trees (e.g., [8, 48]) and global constraints (e.g., [7, 22, 49]), with later work focusing on explaining and comparing various algorithms for CP. Freuder et al. generated explanations for solving methods in the form of trees [15, 16]. Dooms et al. created a generic approach to visualize constraint-based local-search [10]. Li and Epstein provided high-level visuals of the search space to inform the guided search of CSPs [34]. Simonis et al. created a generic visualization of CP problems for postmortem analysis using the output of the solver (i.e., the search tree, constraint, and variable logs) [50]. Finally, Ayub et al. compared how different algorithmic approaches (e.g., backtracking, arc consistency) explore the search space of CSPs [4]. Closer to our investigation, Verbeek et al. enabled users to explore state spaces via the attributes of the system and "get a feeling" for their behavior [54, 55]. In addition to visualizing the graph of states, Verbeek et al. generated Petri net models from the state space to give users a deeper understanding of the valuations of attributes. Thus, while there has been a consistent effort to visualize the decisions of the solver, there is limited work on visualizations for the purpose of making human decisions, a significant area for future exploration.

4.6 Summary

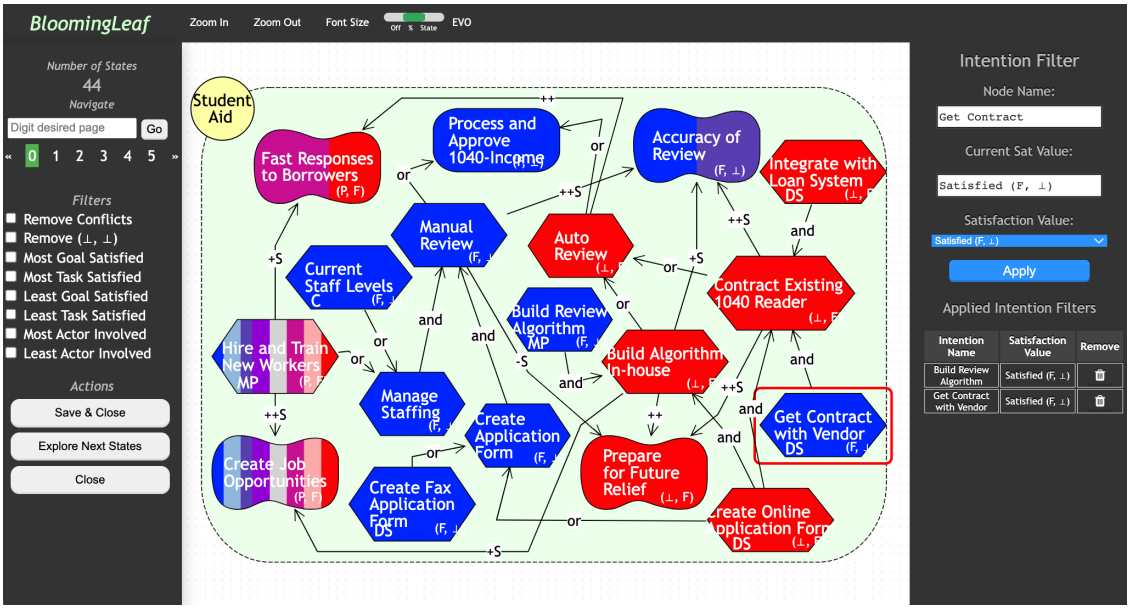
In this chapter, we introduced the use of valuation-based filtering and coloring to assist users in exploring a solution space and selecting custom states from it. We demonstrated the use of these techniques through our illustration of the Student Aid model. Our initial measurements showed a time savings when reviewing states using the color visualization, and up to a 95% reduction in the number of states viewable by the user with valuation filtering. Given the threats described in Sect. 4.4, further study is required to validate our initial claims. In this work, we mitigate the issue of navigating explosive state spaces when the content of each node is an entire model. We see these visualizations as new avenues for

interpreting solutions in other areas of GORE.

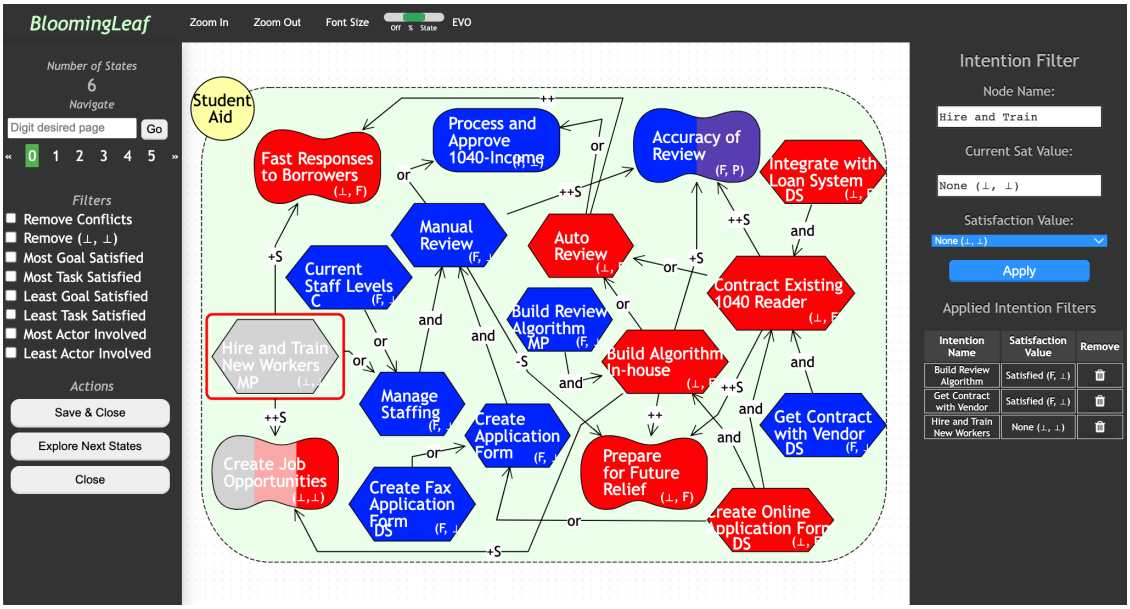
The current implementation is limited to the Evolving Intentions framework, which requires users to generate a full path before exploring the state space. Additionally, there is an upper bound on the number of states our solver will generate before timing out due to incomplete scenario specifications. We will explore a methodology to assist users in fully describing scenarios and tooling to generate paths from the initial state. We also intend to conduct a user-validation study to measure the efficiency gains of these filters and visualization techniques on the Evolving Intentions framework, as well as the usability of our extensions to BloomingLeaf. This study will include EVO Percent mode, which was not part of our initial evaluation, to get a complete overview of the efficacy of EVO.


BloomingLeaf
Zoom In
Zoom Out
Font Size
EVO
Number of States
132
Navigate
Digit desired page
Go
Filters
Remove Conflicts
Remove (⊥, ⊥)
Most Goal Satisfied
Most Task Satisfied
Least Goal Satisfied
Least Task Satisfied
Most Actor Involved
Least Actor Involved
Actions
Save & Close
Explore Next States
Close
Student Aid
Fast Responses to Borrowers
Process and Approve 1040-Income
Accuracy of Review
Integrate with Loan System DS
Manual Review
Auto Review
Current Staff Levels
Build Review Algorithm MP
Contract Existing 1040 Reader
Hire and Train New Workers MP
Manage Staffing
Create Application Form
Build Algorithm In-house
Get Contract with Vendor DS
Create Job Opportunities
Create Fax Application Form DS
Prepare for Future Relief
Create Online Application Form DS
Intention Filter
Node Name:
Build Review
Current Sat Value:
Satisfied (F, ⊥)
Satisfaction Value:
Apply
Applied Intention Filters
Intention Name
Satisfaction Value
Remove
Build Review Algorithm
Satisfied (F, ⊥)

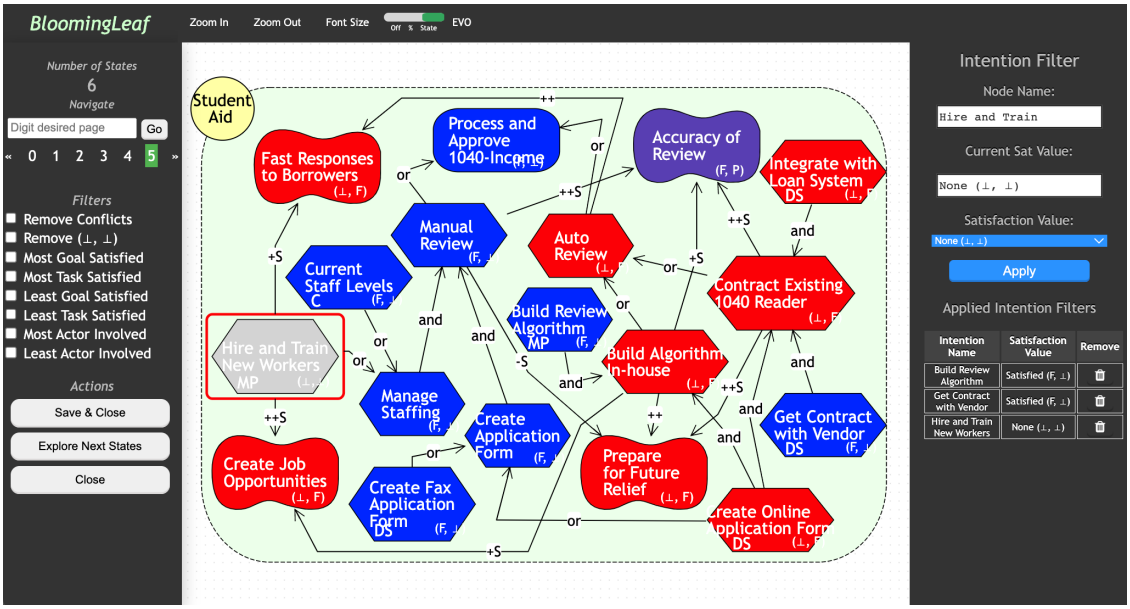
(a) t_0


BloomingLeaf
Zoom In
Zoom Out
Font Size
EVO
Number of States
44
Navigate
Digit desired page
Go
Filters
Remove Conflicts
Remove (⊥, ⊥)
Most Goal Satisfied
Most Task Satisfied
Least Goal Satisfied
Least Task Satisfied
Most Actor Involved
Least Actor Involved
Actions
Save & Close
Explore Next States
Close
Student Aid
Fast Responses to Borrowers
Process and Approve 1040-Income
Accuracy of Review
Integrate with Loan System DS
Manual Review
Auto Review
Current Staff Levels
Build Review Algorithm MP
Contract Existing 1040 Reader
Hire and Train New Workers MP
Manage Staffing
Create Application Form
Build Algorithm In-house
Get Contract with Vendor DS
Create Job Opportunities
Create Fax Application Form DS
Prepare for Future Relief
Create Online Application Form DS
Intention Filter
Node Name:
Get Contract
Current Sat Value:
Satisfied (F, ⊥)
Satisfaction Value:
Apply
Applied Intention Filters
Intention Name
Satisfaction Value
Remove
Build Review Algorithm
Satisfied (F, ⊥)
Get Contract with Vendor
Satisfied (F, ⊥)

(b) t_1



(c) t_2



(d) t_3

Figure 4.1: A Usage Scenario of applying filters and EVO to BloomingLeaf's Next States view of Student Aid model.

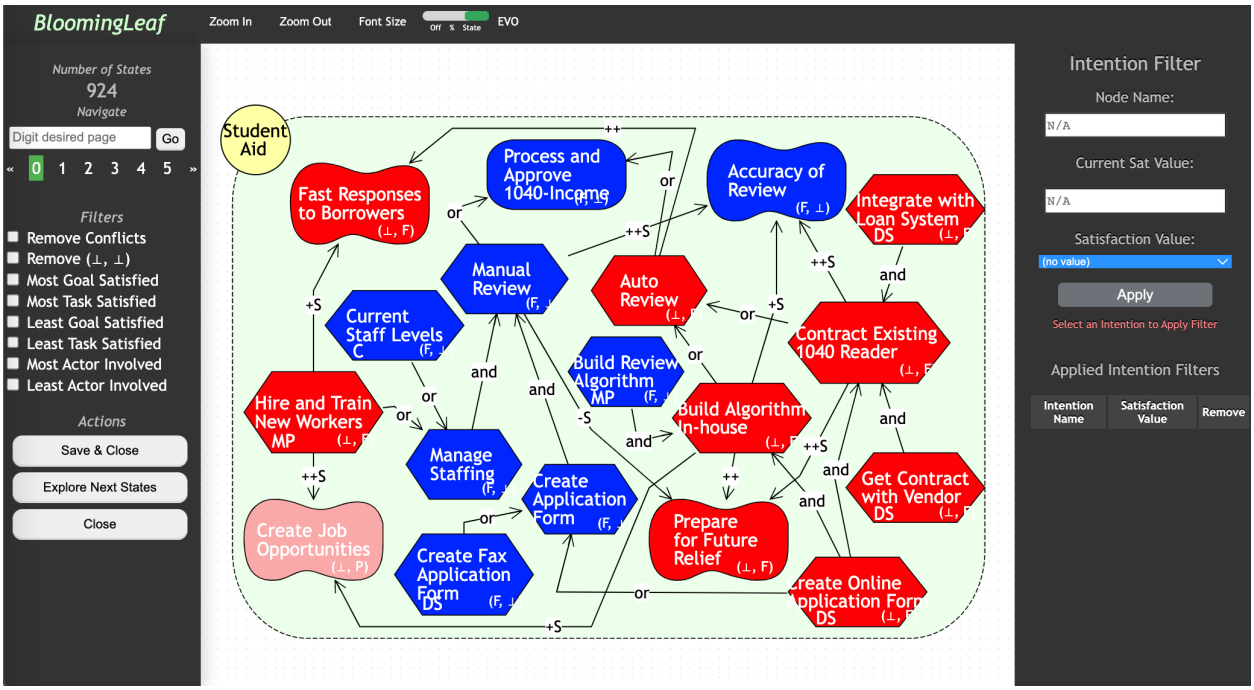


Figure 4.2: BloomingLeaf's Next States view of Student Aid model, with EVO State mode selected.

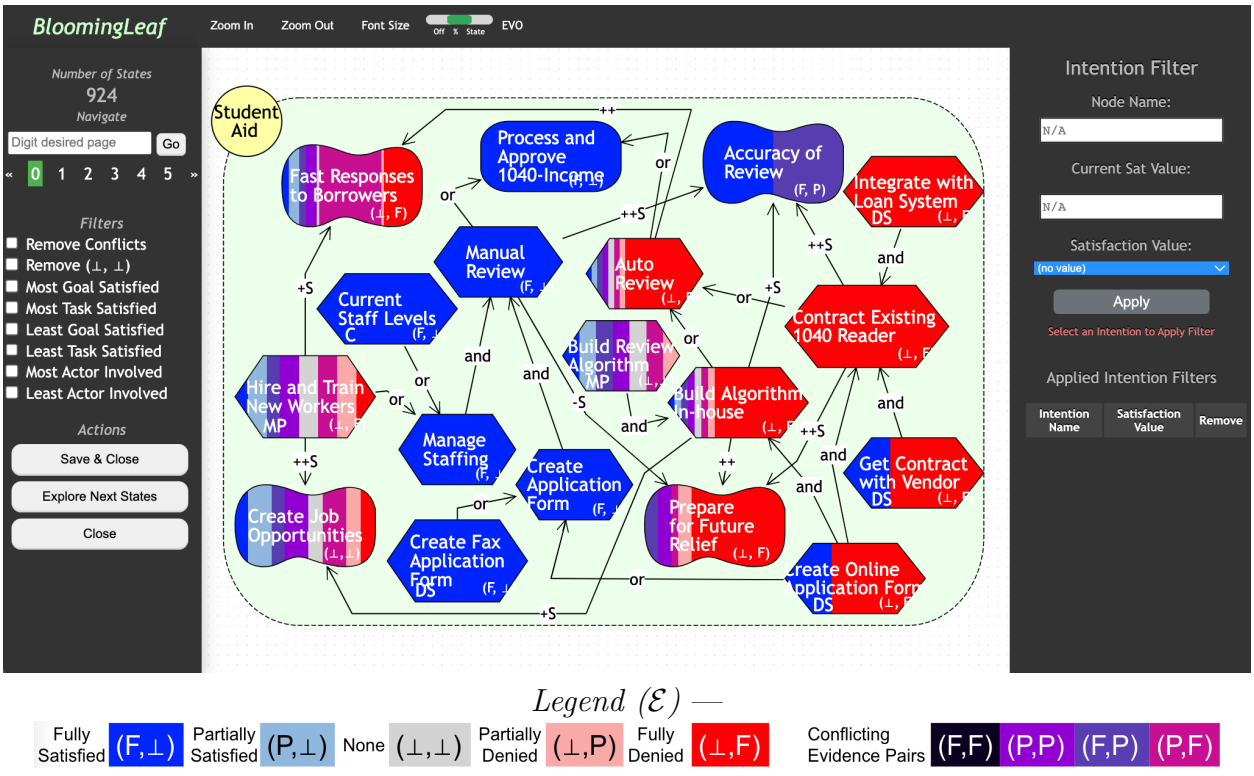


Figure 4.3: BloomingLeaf's Next States view of Student Aid model, with EVO % mode selected.

Chapter 5

Conclusion and Future Work

In this thesis, we explored the problem of visualizing goal models for easier comprehension and the feasibility of exploring a solution space to support stakeholders in their analysis. As goal models grow in complexity, it becomes more difficult for stakeholders to keep track of and evaluate individual nodes and their respective evaluation labels. We reported on the efficacy of color visualization, validating prior work, and extended this approach to assist users in understanding the contents of solution spaces when trying to generate a custom path and select a desired state. Finally, we implemented valuation-based filtering to reduce the size of the solution space to tackle the state explosion problem.

There has been an interest in goal modeling and solution space visualization in the RE community. Previous work has introduced color to improve the visualizations of goal models [52], while generic filters have been used to reduce the size of a CSP domain and solution space [31]. Our IRB-approved validation study of EVO showed that using color significantly decreased the time required to make decisions about goal models that evolve, but found no evidence that EVO altered the quality of understanding or decision making, either positively or negatively. The initial evaluation of EVO and Intention Filtering in Next State demonstrated the time savings and solution space reduction of this approach. Our initial measurements showed time savings when reviewing states using the color visualization and up to a 95% reduction in the number of states viewable by the user with valuation filtering.

A future avenue for consideration in BloomingLeaf is to implement actor evaluations, as explored by Franch et al. [14]. Future work on EVO includes validating the selected colors of blue, red, and purple, as well as implementing alternative color palettes, such as palettes for colorblind users. Further validation is needed to explore the efficacy of EVO in real groups in early-phase RE and to validate EVO with other types of analysis. We can replicate our EVO validation study in order to establish external validity (see Sect. 3.4.3). In terms of solution space exploration, further work includes conducting a user-validation study to measure the efficiency gains of these filters and visualization techniques on the Evolving Intentions framework, as well as the usability of our extensions to BloomingLeaf. This would include validating EVO Percent mode in solution space exploration, which was not part of our initial evaluation. We would also like to explore the scalability of our color visualization and filtering approach on larger models that are more reflective of complex"real-world" scenarios.

Bibliography

[1] S. Alwidian and D. Amyot. "Union is Power": Analyzing Families of Goal Models Using Union Models. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS '20)*, pages 252–262, 2020.

[2] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, and E. Yu. Evaluating Goal Models Within the Goal-Oriented Requirement Language. *International Journal of Intelligent Systems*, 25(8):841–877, 2010.

[3] Aprajita and G. Mussbacher. TimedGRL: Specifying Goal Models Over Time. In *Proceedings of the the Sixth International Workshop on Model-Driven Requirements Engineering (MoDRE'16)*, 2016.

[4] M. A. Ayub, K. A. Kalpoma, H. T. Proma, S. M. Kabir, and R. I. H. Chowdhury. Exhaustive study of essential constraint satisfaction problem techniques based on n-queens problem. In *2017 20th International Conference of Computer and Information Technology (ICCIT)*, pages 1–6, 2017.

[5] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, May 2004.

[6] J. Cabot, R. Clarisó, and D. Riera. Umltocsp: a tool for the formal verification of uml/ocl models using constraint programming. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pages 547–548, 2007.

[7] M. Carro and M. Hermenegildo. Tools for constraint visualisation: The vifid/trifid tool. In *Analysis and Visualization Tools for Constraint Programming*, pages 253–272. Springer, 2000.

[8] M. Carro and M. Hermenegildo. Tools for search-tree visualisation: The apt tool. In *Analysis and Visualization Tools for Constraint Programming*, pages 237–252. Springer, 2000.

[9] E. M. Clarke, W. Klieber, M. Novacek, and P. Zuliani. Model Checking and the State Explosion Problem. In B. Meyer and M. Nordio, editors, *Tools for Practical Software Verification*, volume 7682 of *Lecture Notes in Computer Science*, pages 1–30. Springer Berlin Heidelberg, 2012.

[10] G. Dooms, P. Van Hentenryck, and L. Michel. Model-driven visualizations of constraint-based local search. In C. Bessière, editor, *Principles and Practice of Constraint Programming – CP 2007*, pages 271–285, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[11] F. Faul, E. Erdfelder, A.-G. Lang, and A. Buchner. G* Power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences. *Behavior research methods*, 39(2):175–191, 2007.

[12] Federal Student Aid, An Office of the U.S. Department of Education. One-Time Student Loan Debt Relief. Online at https://studentaid.gov/debt-relief-announcement/one-time-cancellation, 2022. Accessed 10/01/2022.

[13] X. Franch. On the Quantitative Analysis of Agent-oriented Models. In *Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE'06)*, pages 495–509, 2006.

[14] X. Franch, G. Grau, and C. Quer. A Framework for the Definition of Metrics for Actor-Dependency Models. In *Proceedings of the Requirements Engineering Conference, 12th IEEE International*, RE '04, pages 348–349, Washington, DC, USA, 2004. IEEE Computer Society.

[15] E. C. Freuder, C. Likitvivatanavong, and R. J. Wallace. A case study in explanation and implication. In *CP2000 workshop on analysis and visualization of constraint programs and solvers*, 2000.

[16] E. C. Freuder, C. Likitvivatanavong, and R. J. Wallace. Deriving explanations and implications for constraint satisfaction problems. In T. Walsh, editor, *Principles and Practice of Constraint Programming — CP 2001*, pages 585–589, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[17] A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri, and P. Traverso. Specifying and Analyzing Early Requirements in Tropos. *Requirements Engineering*, 9(2):132–150, May 2004.

[18] P. Ghazi and M. Glinz. An experimental comparison of two navigation techniques for requirements modeling tools. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 240–250, 2018.

[19] P. Giorgini, J. Mylopoulos, and R. Sebastiani. Goal-oriented Requirements Analysis and Reasoning in the Tropos Methodology. *Engineering Applications of Artificial Intelligence*, 18(2):159–171, 2005.

[20] P. Godefroid. Model checking for programming languages using verisoft. In *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 174–186, 1997.

[21] C. A. González, F. Büttner, R. Clarisó, and J. Cabot. Emftocsp: A tool for the lightweight verification of emf models. In *2012 First International Workshop on Formal*

Methods in Software Engineering: Rigorous and Agile Approaches (FormSERA), pages 44–50. IEEE, 2012.

[22] F. Goualard and F. Benhamou. Debugging constraint programs by store inspection. In *Analysis and Visualization Tools for Constraint Programming*, pages 273–297. Springer, 2000.

[23] A. M. Grubb. *Evolving Intentions: Support for Modeling and Reasoning about Requirements that Change over Time*. PhD thesis, University of Toronto, 2019.

[24] A. M. Grubb and M. Chechik. BloomingLeaf: A Formal Tool for Requirements Evolution over Time. In *Proceedings of the 26th IEEE International Requirements Engineering Conference (RE'18): Posters & Tool Demos*, pages 490–491, 2018.

[25] A. M. Grubb and M. Chechik. Formal Reasoning for Analyzing Goal Models that Evolve over Time. *Requirements Engineering*, 26(3):423–457, 2021.

[26] I. Hadar, I. Reinhartz-Berger, T. Kuflik, A. Perini, F. Ricca, and A. Susi. Comparing the comprehensibility of requirements models expressed in use case and tropos: Results from a family of experiments (article) comparing the comprehensibility of requirements models expressed in use case and tropos: Results from a family of experiments (article) author. *Information and Software Technology*, 55(10):1823–1843, 2013.

[27] J. Horkoff, F. B. Aydemir, E. Cardoso, T. Li, A. Maté, E. Paja, M. Salnitri, L. Piras, J. Mylopoulos, and P. Giorgini. Goal-oriented requirements engineering: an extended systematic mapping study. *Requirements Engineering*, 24(2):133–160, 2019.

[28] J. Horkoff, T. Li, F.-L. Li, M. Salnitri, E. Cardoso, P. Giorgini, J. Mylopoulos, and J. Pimentel. Taking goal models downstream: A systematic roadmap. In *Proceedings of the IEEE 8th International Conference on Research Challenges in Information Science (RCIS'14)*, pages 1–12, May 2014.

[29] J. Horkoff and E. Yu. Visualizations to support interactive goal model analysis. In *2010 Fifth International Workshop on Requirements Engineering Visualization*, pages 1–10, 2010.

[30] J. Horkoff and E. Yu. Interactive Goal Model Analysis For Early Requirements Engineering. *Requirements Engineering*, 21(1):29–61, 2016.

[31] B. C. Hu and A. M. Grubb. Support for User Generated Evolutions of Goal Models. In *Proceedings of the 11th International Workshop on Modeling in Software Engineering*, pages 1–7, 2019.

[32] S. Iyengar. *The Art of Choosing*. Little, Brown Book Group, 2010.

[33] K. Kuchcinski and R. Szymanek. JaCoP - Java Constraint Programming solver. `http://jacop.osolpro.com`, 2016. Accessed: 2016-02-21.

[34] X. Li and S. L. Epstein. Visualization for structured constraint satisfaction problems. In *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[35] A. Lu. For Biden's Loan-Forgiveness Plan, a Flurry of Lawsuits, a Rollback, and a New Sticker Price. The Chronicle of Higher Education, Online at *https://www.chronicle.com/article/for-bidens-loan-forgiveness-plan-a-flurry-of-lawsuits-a-rollback-and-a-new-sticker-price*, Sep. 30 2022.

[36] G. Mathew, T. Menzies, N. Ernst, and J. Klein. Shorter Reasoning About Larger Requirements Models. In *Proceedings of the 25th IEEE International Requirements Engineering Conference (RE'17)*, 2017.

[37] A. Mavin, P. Wilkinson, S. Teufl, H. Femmer, J. Eckhardt, and J. Mund. Does Goal-Oriented Requirements Engineering Achieve Its Goal? In *In Proceedings of the 25th IEEE International Requirements Engineering Conference (RE'17)*, pages 174–183, September 2017.

[38] L. Meier. *ANOVA and Mixed Models: A Short Introduction Using R.* CRC Press, 2022.

[39] S. Nuseibeh, B.; Easterbrook. Requirements engineering: A roadmap. *Proceedings of the conference on the future of Software engineering*, 2000.

[40] R. F. Oliveira and J. C. S. do Prado Leite. Using colorimetric concepts for the evaluation of goal models. In *Proceedings of the Tenth International Model-Driven Requirements Engineering (MoDRE)*, pages 39–48, 2020.

[41] S. Reddivari, S. Rad, T. Bhowmik, N. Cain, and N. Niu. Visual requirements analytics: a framework and case study. *Requirements Engineering*, 19(3):257–279, 2014.

[42] R. P. Runyon. *Nonparametric Statistics: A Contemporary Approach.* Addison-Wesley, 1977.

[43] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, chapter 6 Constraint Satisfaction Problems. Prentice Hall, 2010.

[44] M. Santos, C. Gralha, M. Goulão, J. Araújo, and A. Moreira. On the Impact of Semantic Transparency on Understanding and Reviewing Social Goal Models. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 228–239, 2018.

[45] C. Schulte. Oz explorer: A visual constraint programming tool. In *International Symposium on Programming Language Implementation and Logic Programming*, pages 477–478. Springer, 1996.

[46] B. Schwartz. *The Paradox of Choice: Why More Is Less.* Harper Collins, 2003.

[47] F. Shull, J. Singer, and D. I. Sjøberg. *Guide to Advanced Empirical Software Engineering.* Springer-Verlag New York, Inc., 2007.

[48] H. Simonis and A. Aggoun. Search-tree visualisation. In *Analysis and Visualization Tools for Constraint Programming*, pages 191–208. Springer, 2000.

[49] H. Simonis, A. Aggoun, N. Beldiceanu, and E. Bourreau. Complex constraint abstraction: Global constraint visualisation. In *Analysis and Visualization Tools for Constraint Programming*, pages 299–317. Springer, 2000.

[50] H. Simonis, P. Davern, J. Feldman, D. Mehta, L. Quesada, and M. Carlsson. A generic visualization platform for cp. In D. Cohen, editor, *Principles and Practice of Constraint Programming – CP 2010*, pages 460–474, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[51] F. L. Siqueira. Comparing the comprehensibility of requirements models: An experiment replication. *Information and Software Technology*, 96:1–13, 2018.

[52] M. H. Varnum, K. M. B. Spencer, and A. M. Grubb. Towards an Evaluation Visualization with Color. In *Proceedings of the 13th International i* Workshop (iStar)*, pages 79–84, 2020.

[53] S. Vathsavayi, O. Sievi-Korte, K. Koskimies, and K. Systä. Using constraint satisfaction and optimization for pattern-based software design. In *2014 23rd Australian Software Engineering Conference*, pages 29–37. IEEE, 2014.

[54] H. Verbeek, A. Pretorius, W. van der Aalst, and J. van Wijk. On petri-net synthesis and attribute-based visualization. In *Proc. Workshop on Petri Nets and Software Engineering (PNSE 2007)*, pages 127–141, 2007.

[55] H. Verbeek, A. Pretorius, W. Van der Aalst, and J. van Wijk. Visualizing state spaces with petri nets. *Computer Science Report*, 7(01), 2007.

[56] J. P. Winkler, J. Grönberg, and A. Vogelsang. Optimizing for recall in automatic requirements classification: An empirical study. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pages 40–50, 2019.

[57] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering*. Springer Berlin Heidelberg, 2012.

[58] F. Wotawa, M. Nica, and I. Moraru. Automated debugging based on a constraint model of the program and a test case. *The Journal of Logic and Algebraic Programming*, 81(4):390–407, 2012.