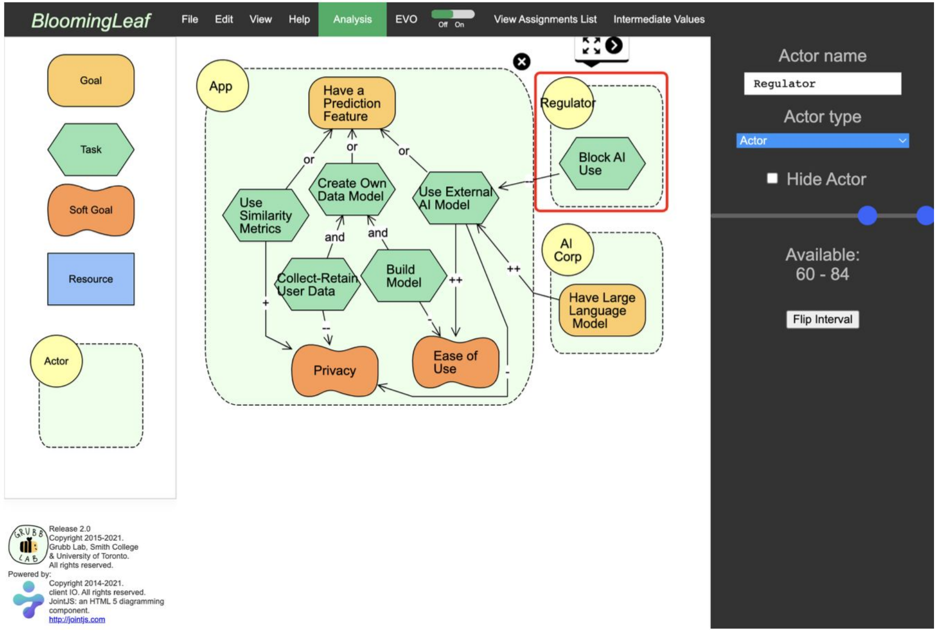
Incorporating Presence Conditions into Goal Models that Evolve Over Time



Actor name: Regulator

Actor type: Actor

□ Hide Actor

Available: 60 - 84

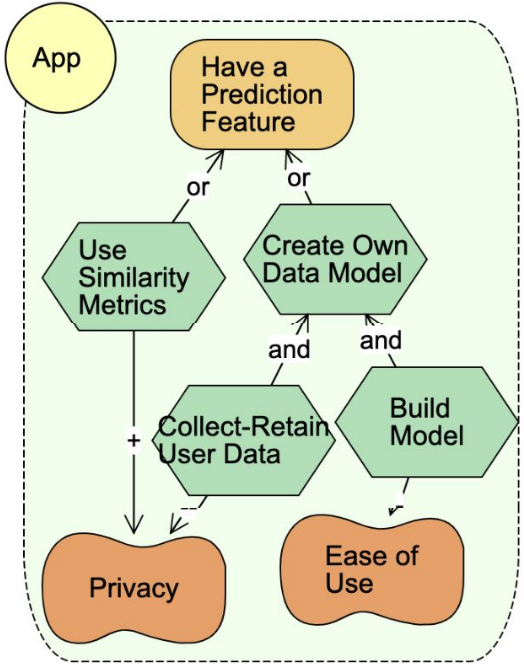
Release 2.0
Copyright 2015-2021.
Grubb Lab, Smith College
& University of Toronto.
All rights reserved.

Powered by:
Copyright 2014-2021.
client IO. All rights reserved.
JointJS: an HTML 5 diagramming component.
http://jointjs.com

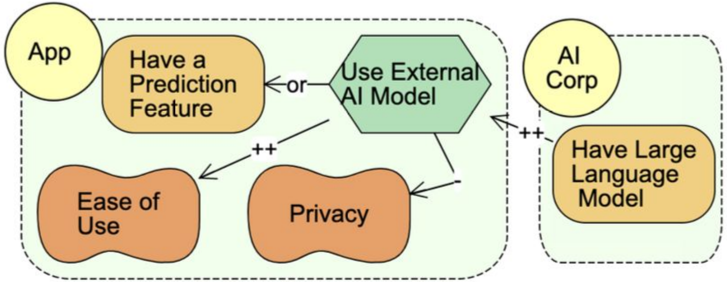
Xinran Bi and Alicia M. Grubb
Smith College, United States

September 5, 2023
Model-Driven Requirements Engineering Workshop (MoDRE)

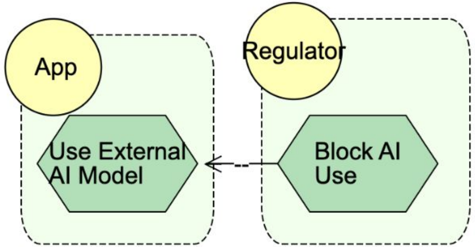
Why presence conditions? Example of the Predictive App



Year 2020

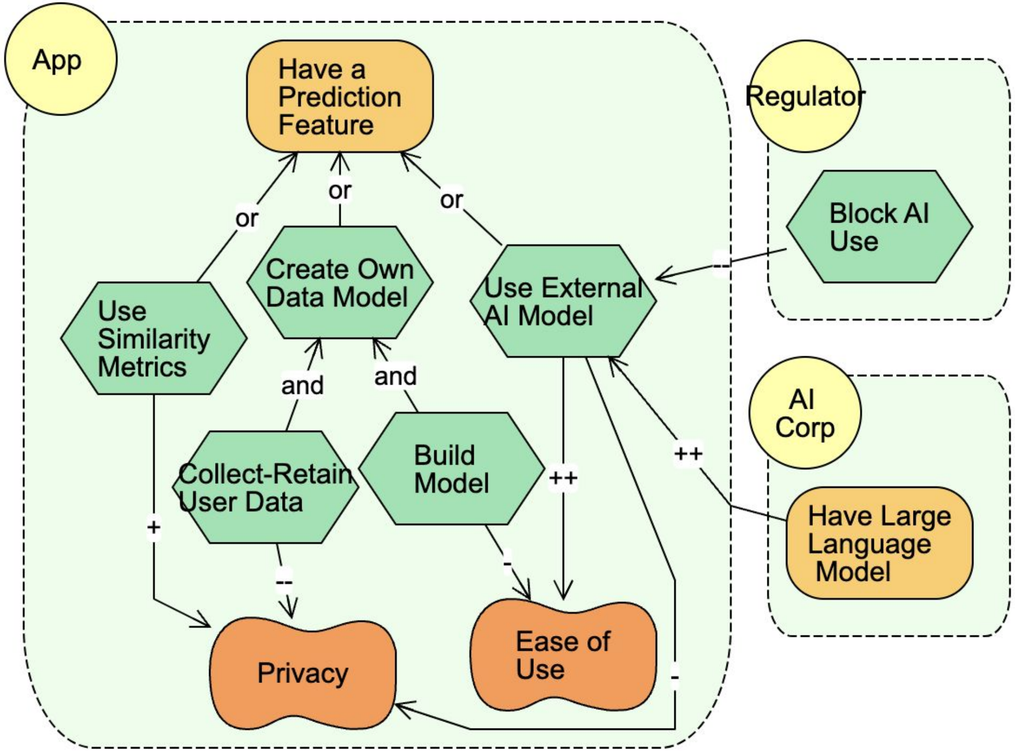


Year 2023

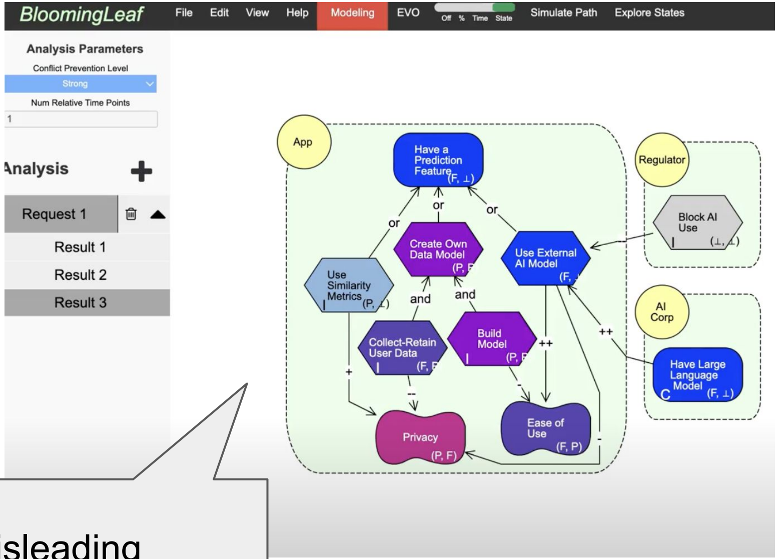


Year 2025

Why presence conditions? Example of the Predictive App



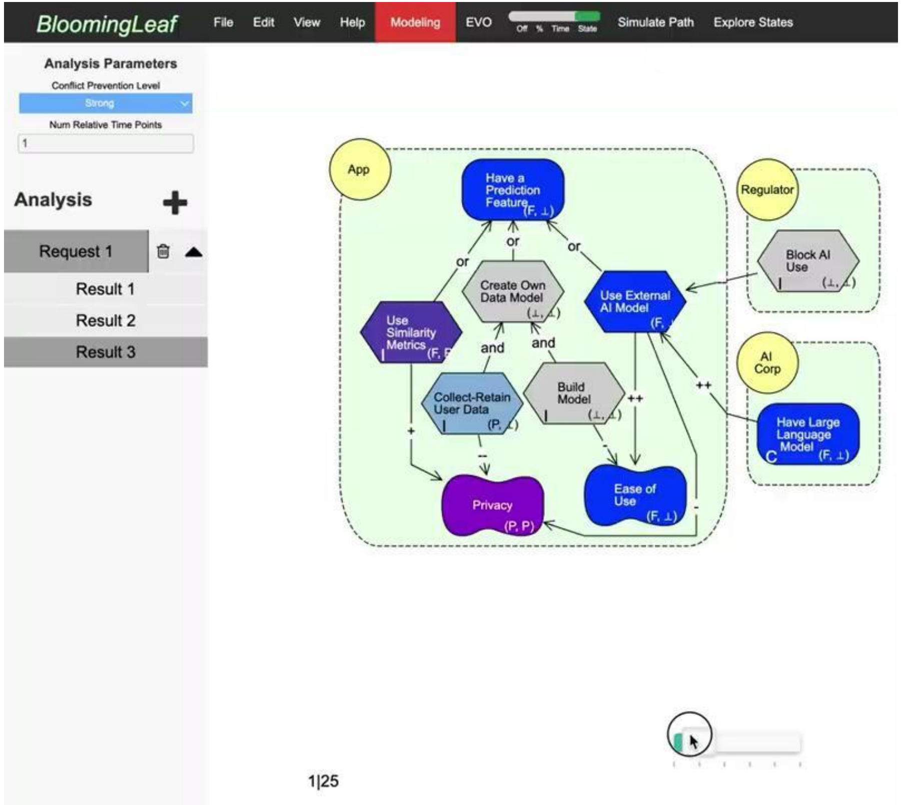
Simulation without presence conditions



Analysis Parameters

Conflict Prevention Level: Strong

Num Relative Time Points: 1



Analysis Parameters

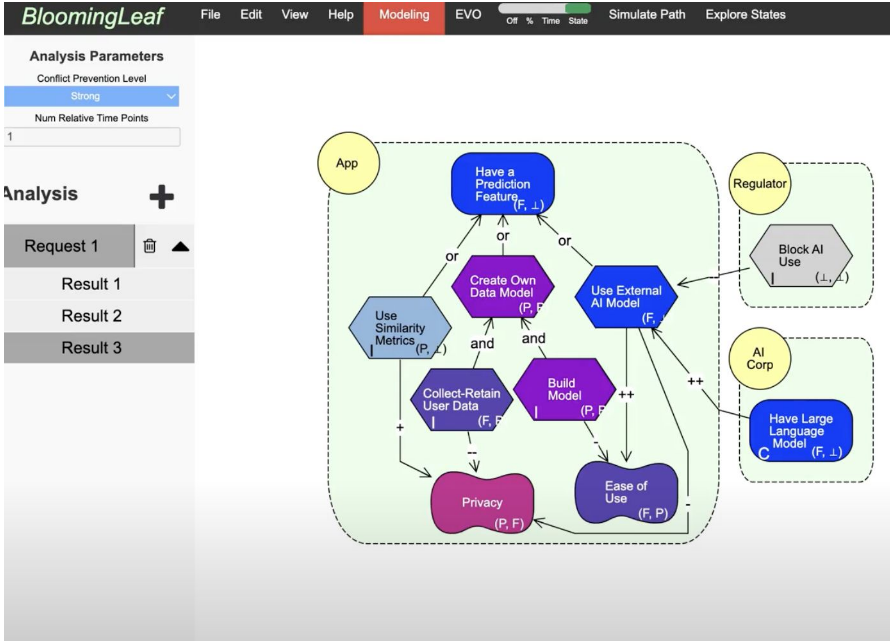
Conflict Prevention Level: Strong

Num Relative Time Points: 1

Misleading simulation visualization!

Research Problem:

Avoid nonsensical visualization after merging model segments from different time points



Analysis Parameters

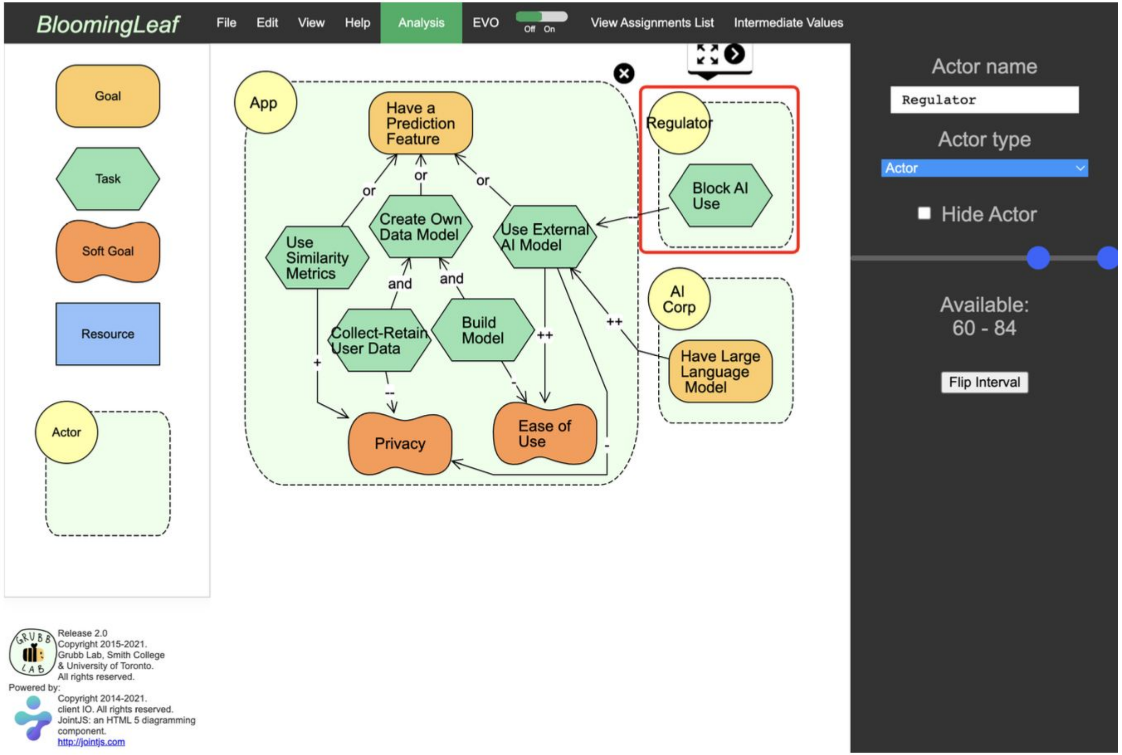
Conflict Prevention Level: Strong

Num Relative Time Points: 1

Analysis
Request 1
Result 1
Result 2
Result 3
App
Have a Prediction Feature (F, ⊥)
or
or
or
Create Own Data Model (P, F
Use External AI Model (F,
Use Similarity Metrics (P, ⊥)
and
and
Collect-Retain User Data (F, F
Build Model (P, F
++
++
+
--
-
-
Privacy (P, F)
Ease of Use (F, P)
Regulator
Block AI Use (⊥, ⊥)
AI Corp
Have Large Language Model (F, ⊥)
C

Misleading simulation visualization!

Our contribution



Actor name: Regulator

Actor type: Actor

□ Hide Actor

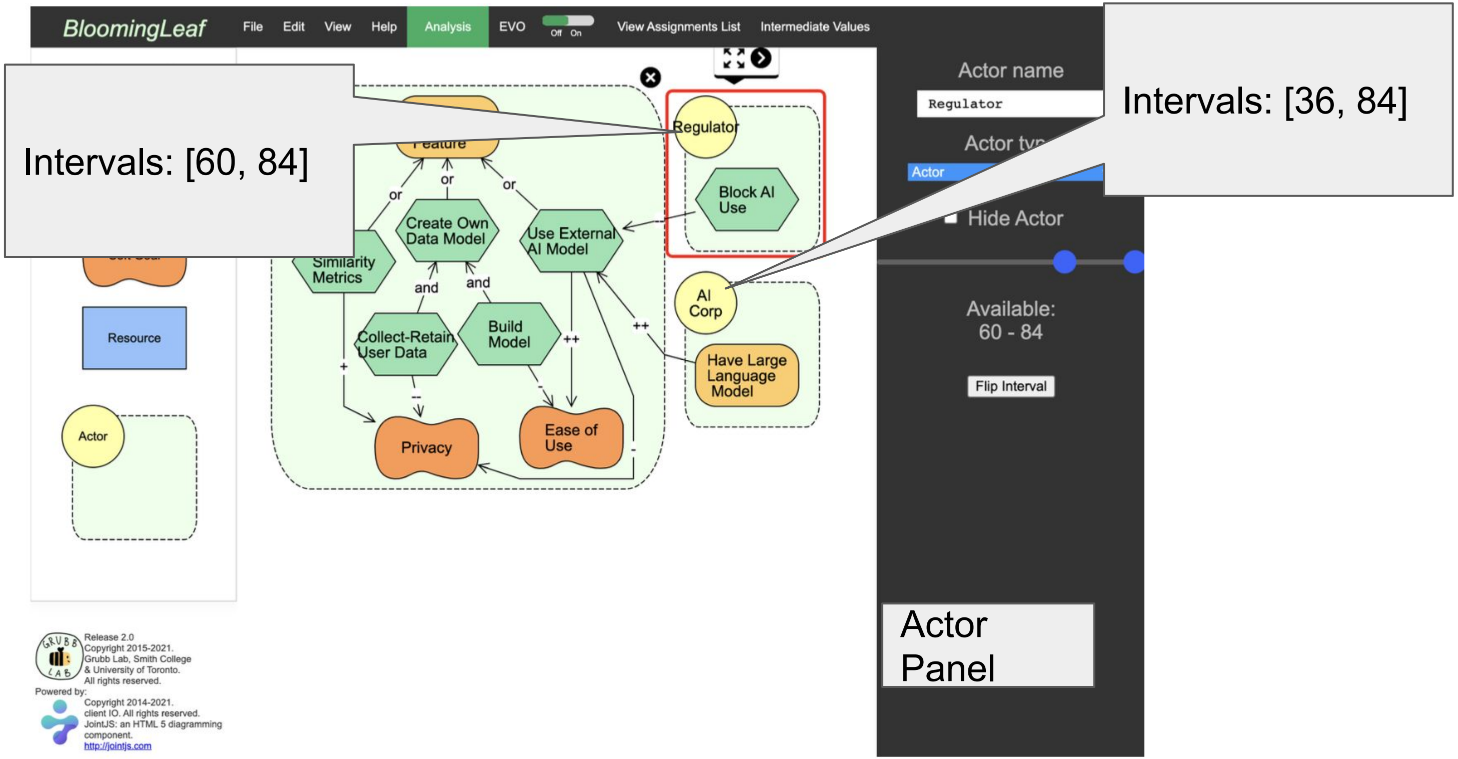
Available: 60 - 84

Release 2.0
Copyright 2015-2021.
Grubb Lab, Smith College
& University of Toronto.
All rights reserved.

Powered by:
Copyright 2014-2021.
client IO. All rights reserved.
JointJS: an HTML 5 diagramming component.
http://jointjs.com

Incorporated presence conditions into the Evolving Intentions framework to avoid nonsensical visualization

Presences conditions are implemented as part of BloomingLeaf in the front-end actor



Intervals: [60, 84]

Intervals: [36, 84]

Actor Panel

Actor name: Regulator

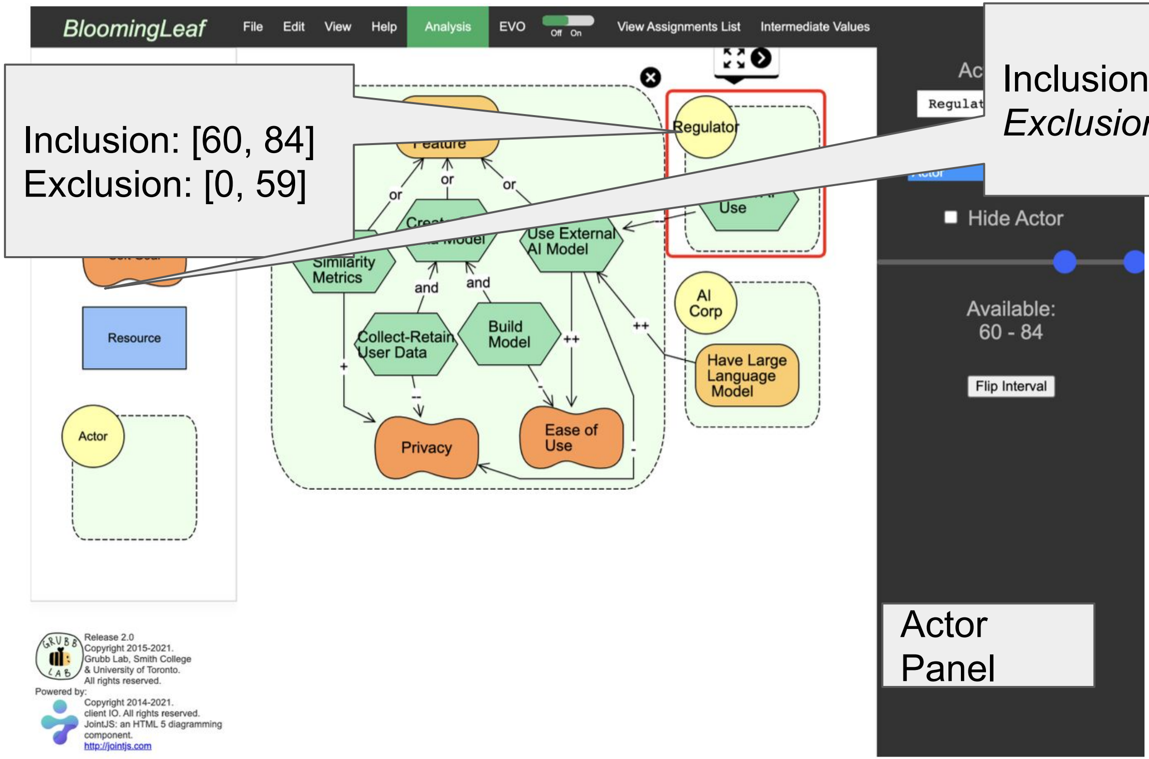
Actor type: Actor

Available: 60 - 84

Release 2.0
Copyright 2015-2021.
Grubb Lab, Smith College
& University of Toronto.
All rights reserved.

Powered by:
Copyright 2014-2021.
client IO. All rights reserved.
JointJS: an HTML 5 diagramming component.
http://jointjs.com

Presence conditions specifications



Inclusion: [60, 84]
Exclusion: [0, 59]

Inclusion: [36, 84] (User Perspective)
Exclusion: [0, 35] (System Specification)

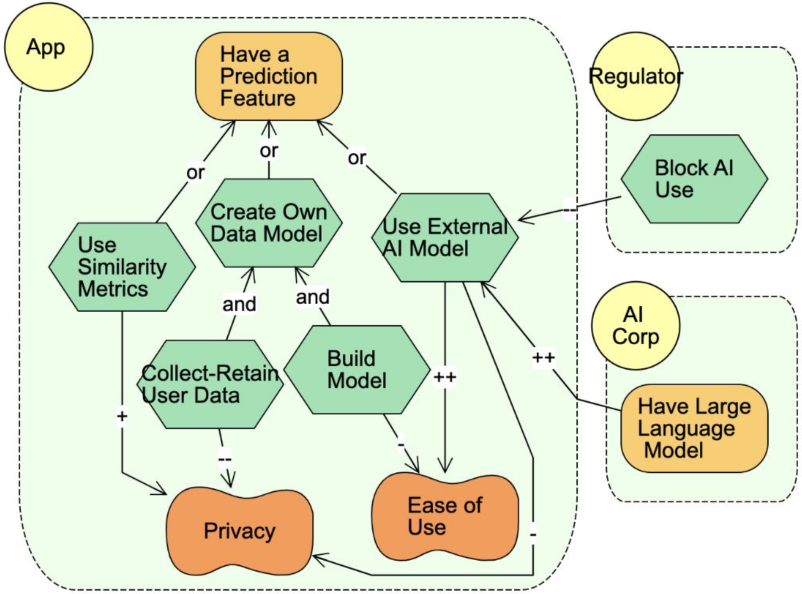
Available:
60 - 84

Actor Panel

Presence conditions were specified as **exclusion periods** to enable backward compatibility

Presence conditions specifications - Actor

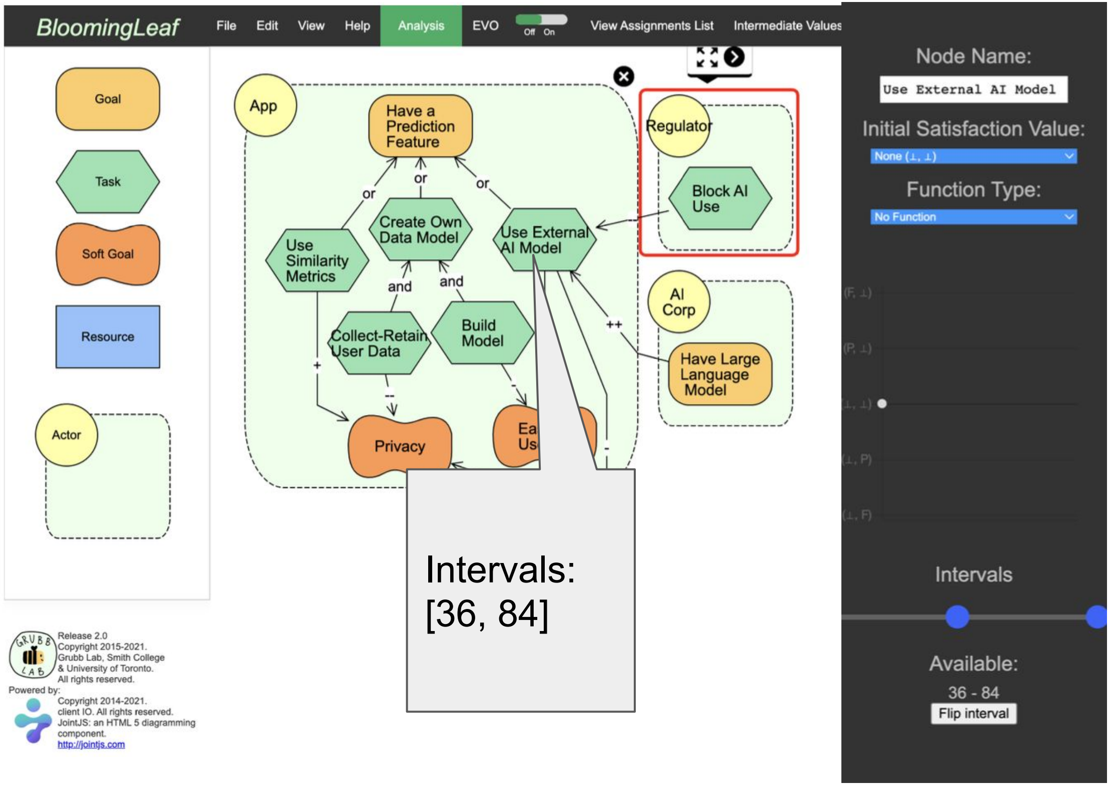
actor a ∈ A: ⟨*name*, *type*, *intention-set*, ***exclusion-set (optional)***⟩



Inclusion: [0, 84], 84 is **Maximum absolute time**
Exclusion: []

Inclusion: *User perspective*
Exclusion: *System specification*

Presences conditions are implemented as part of BloomingLeaf in the front-end intention



Node Name: Use External AI Model

Initial Satisfaction Value: None (⊥, ⊥)

Function Type: No Function

Intervals

Available: 36 - 84

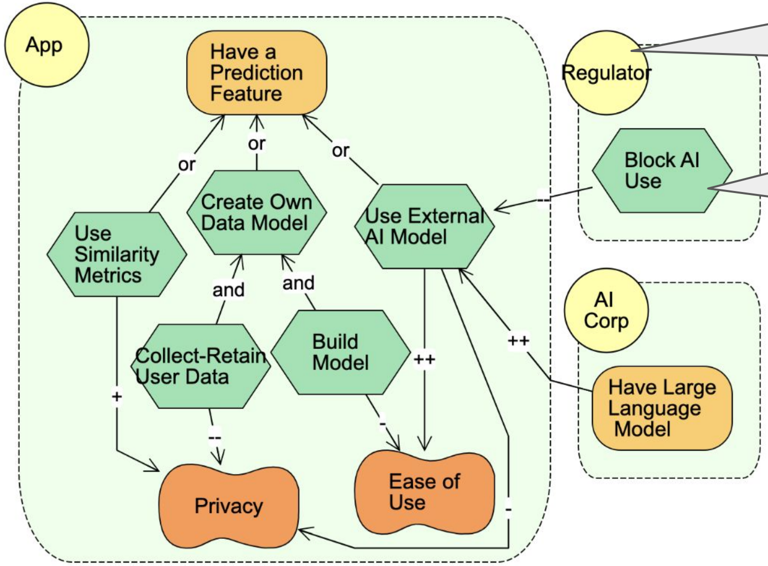
Flip interval

Intervals: [36, 84]

Intention Panel

Presence conditions specifications - Intention

intention g ∈ G: ⟨*name*, *type*, *intention-set*, ***exclusion-set (optional)***⟩



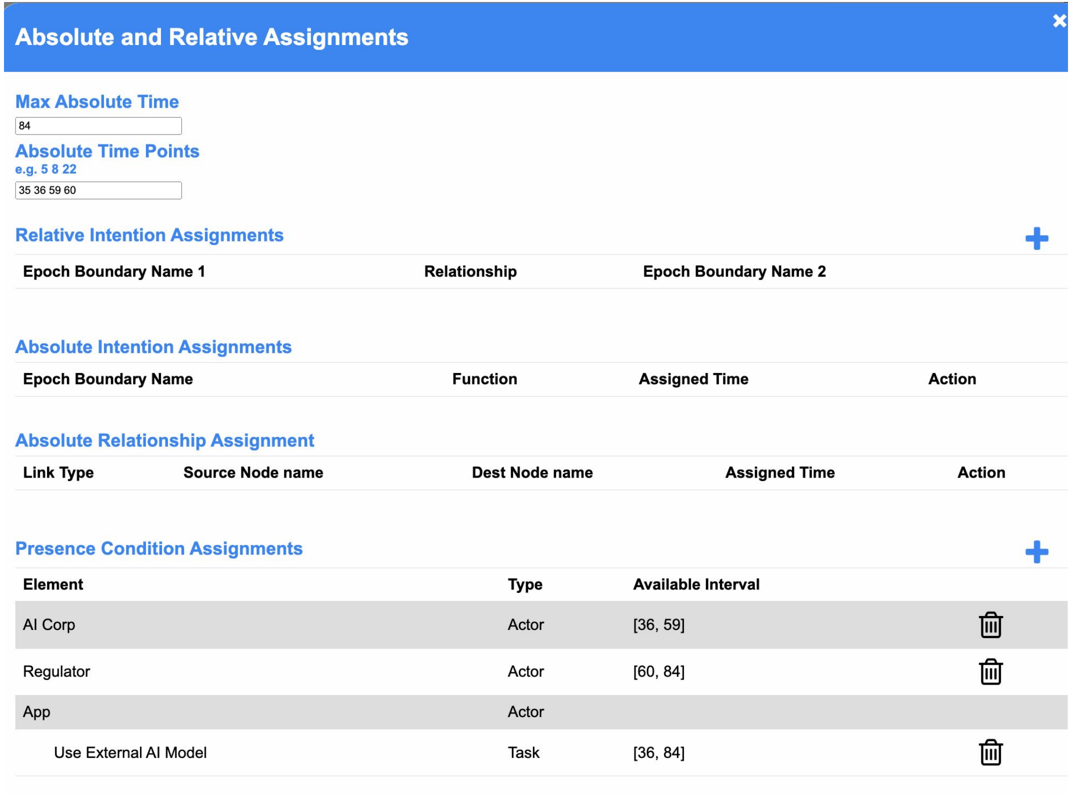
Inclusion: [60, 84],
Exclusion: [0,59]

Exclusion can be:
[0, 62]

Intention's can have a **more restrictive** exclusion interval than its actor's

Inclusion: *User perspective*
Exclusion: *System specification*

Model-level view of presence conditions assignment



Absolute and Relative Assignments

Max Absolute Time

84

Absolute Time Points

e.g. 5 8 22

35 36 59 60

Relative Intention Assignments

Epoch Boundary Name 1	Relationship	Epoch Boundary Name 2

Absolute Intention Assignments

Epoch Boundary Name	Function	Assigned Time	Action

Absolute Relationship Assignment

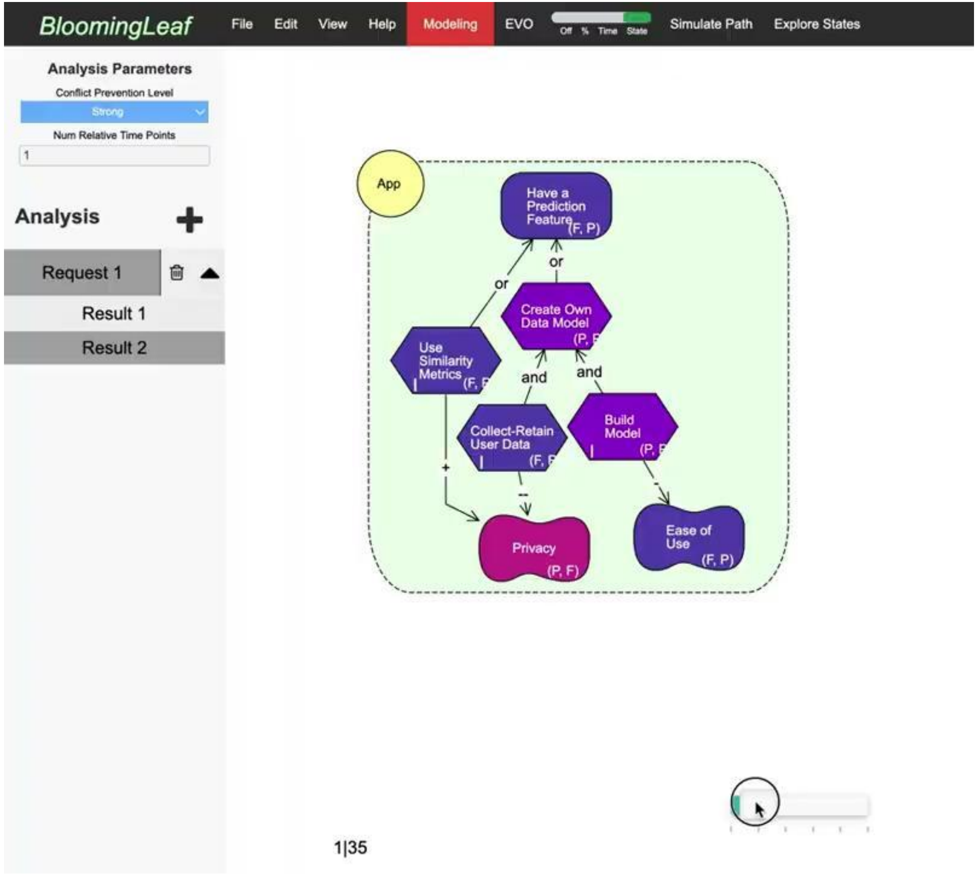
Link Type	Source Node name	Dest Node name	Assigned Time	Action

Presence Condition Assignments

Element	Type	Available Interval	
AI Corp	Actor	[36, 59]	🗑
Regulator	Actor	[60, 84]	🗑
App	Actor		
Use External AI Model	Task	[36, 84]	🗑

Help user keep track of every elements’ presence conditions specification

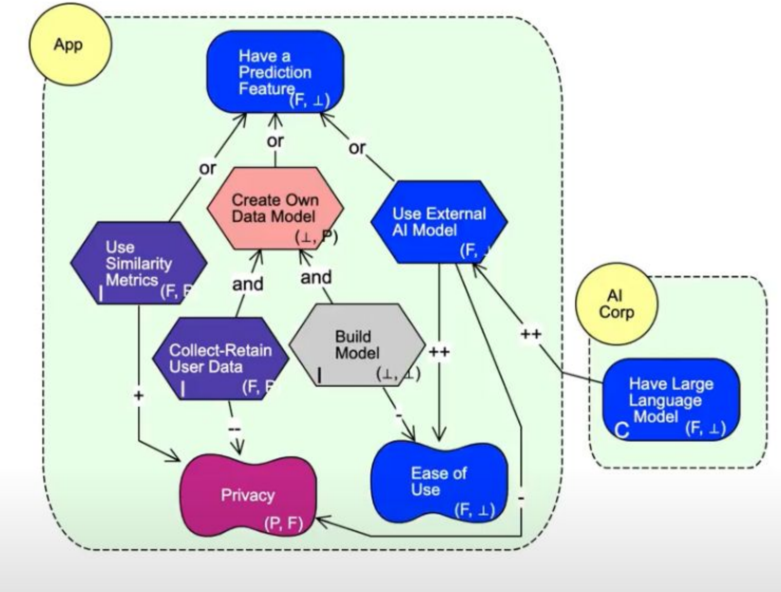
Simulation with Presence Conditions



Analysis Parameters

Conflict Prevention Level: Strong

Num Relative Time Points: 1

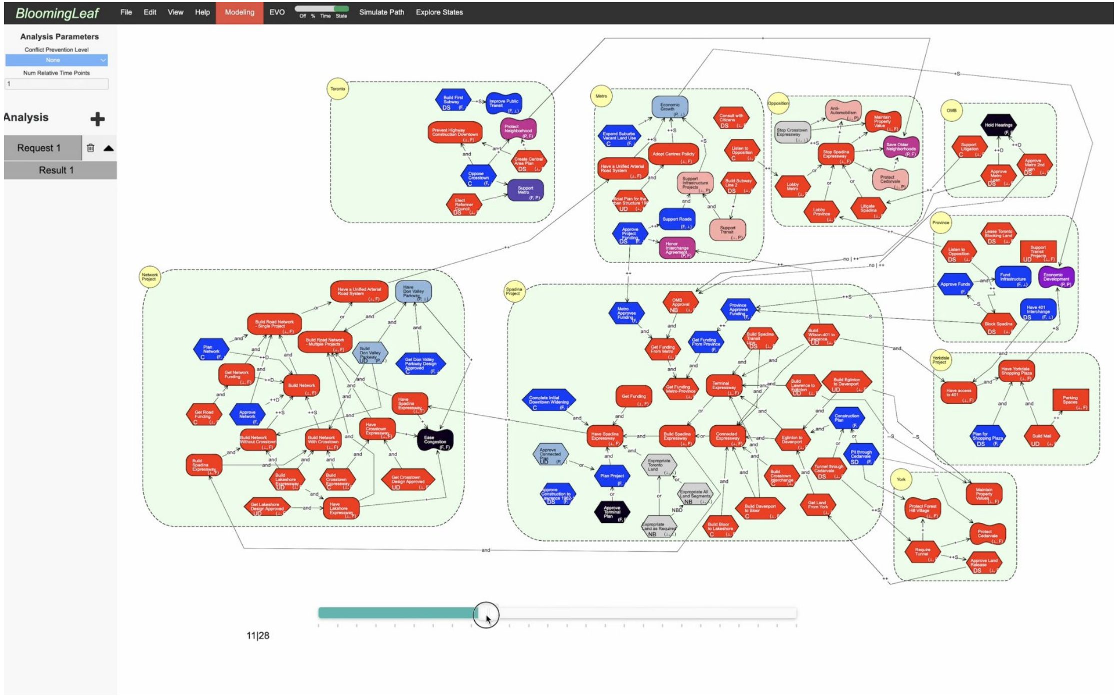


App
Have a Prediction Feature (F, ⊥)
or
or
or
Create Own Data Model (⊥, P)
Use External AI Model (F, ⊥
Use Similarity Metrics (F, F
and
and
Collect-Retain User Data (F, F
Build Model (⊥, ⊥)
++
++
+
--
-
-
Privacy (P, F)
Ease of Use (F, ⊥)
AI Corp
Have Large Language Model (F, ⊥)

Has a better representation of when does certain actors come into play

Evaluation - Our approach is scalable, tested by a model over 100 intentions

Model of a large road construction project*



Analysis Parameters

Conflict Prevention Level: None

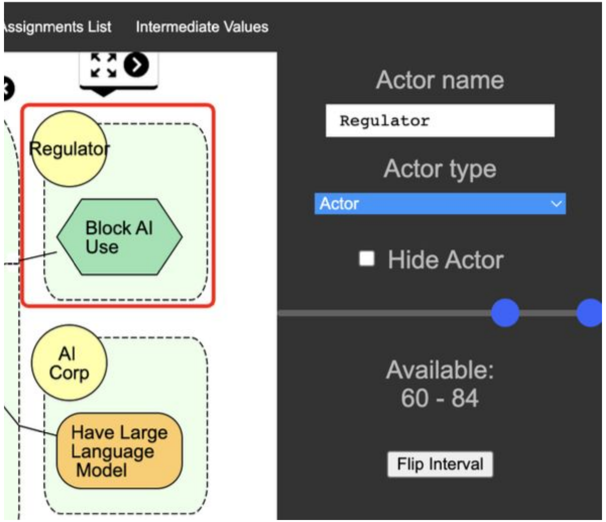
Num Relative Time Points: 1

Can be improved by allowing to select multiple intentions at one time

* A.M. Grubb and M. Chechik. Reconstructing the Past: The Case of the Spadina Expressway. *Requirements Engineering* (**REJ**), 25(2):253–272, 2020.

Model slicing

Using presence conditions as model slices allows us to empirically observe how stakeholders interpret large models.



Actor name: Regulator

Actor type: Actor

□ Hide Actor

Available: 60 - 84

Similar ‘Hide’ function in the analysis mode
Allow us to study how users explore large models

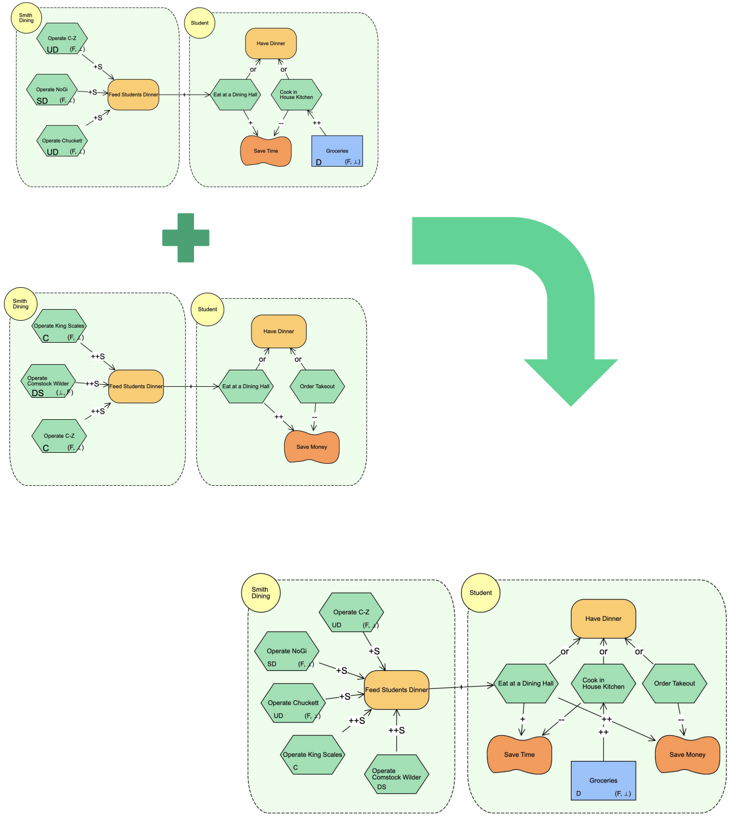
Ongoing Work & Future Work

Next steps:

Improve backend analysis to take presence conditions into account

Incorporate presence conditions with model **merging and slicing**

Generalizing our approach to other goal modeling languages



Related Work

Used for specifying software product lines in model-driven engineering

- Software product lines variability
- Domain Specific Languages (DSLs) for mapping feature models

Goal modeling:

- TimeURN

While we specify presence conditions as **temporal ranges**, not boolean conditions

Summary

We extended the Evolving Intentions framework to enable analysts to specify presence conditions

We integrated this extension in one of the goal modeling tools, BloomingLeaf

Acknowledgements:

We thank the members of our lab who helped in tool development and gave us feedback on this project: Venus Nguyen, Emily Kung, Karenna Kung, Thu Tran, and Caroline Zouloumian

This material is based upon work supported by the National Science Foundation under Award No. 2104732.