Visual Consistency in Formal Modeling: An Empirical Evaluation

YILIANG LIANG, Carnegie Mellon University, USA
AVINASH PALLIYIL, Georgia Institute of Technology, USA
EUNSUK KANG, Carnegie Mellon University, USA
JOSHUA SUNSHINE, Carnegie Mellon University, USA

Formal modeling tools rely on visualizations to help users explore models, identify errors, and build confidence in system correctness. However, visualizers often produce inconsistent layouts across instances, making it difficult to track structural similarities and changes. In this paper, we present the first human-subject study of visual consistency in formal modeling, examining how consistent layouts influence users' ability to understand and debug models. Through a controlled experiment, participants completed tasks using either manually created consistent visualizations or visualizations created by the Alloy formal modeling tool. Our findings show that full (strict) visual consistency significantly improves both the speed and correctness of bug detection and localization. We also observe that partial consistency – where diagrams need not be fully identical but preserve enough recognizable structure for users to perceive similarities – offers a similar set of improvements to user performance, offering a practical alternative to full consistency. Subjective findings through post-experiment interviews reveal that consistency lowers cognitive overhead by helping participants more naturally and efficiently detect differences and track corresponding elements across states.

CCS Concepts: • **Human-centered computing** → **Graph drawings**; **Empirical studies in visualization**; **Visualization design and evaluation methods**; • **Software and its engineering** → **Specification languages**; **System modeling languages**.

Additional Key Words and Phrases: formal modeling, visualization, visual consistency, model comprehension, empirical study, human-subject experiment

ACM Reference Format:
Yiliang Liang, Avinash Palliyil, Eunsuk Kang, and Joshua Sunshine. 2018. Visual Consistency in Formal Modeling: An Empirical Evaluation. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 30 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Formal modeling uses rigorous logic to represent complex systems and help stakeholders understand, explore, and verify the correctness and reliability of system design. Under this framework, users would write an abstract description of the system in a modeling language. An analysis tool then leverages advanced solvers to explore valid configurations of the system and verify that the system satisfies some desirable properties.

These benefits, however, come with a steep learning curve and usability challenges. The abstract nature of formal models and the often technical notations used can obscure their underlying semantics. In short, formal models of complex

Authors' Contact Information: Yiliang Liang, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, yiliangl@andrew.cmu.edu; Avinash Palliyil, Georgia Institute of Technology, Atlanta, Georgia, USA; Eunsuk Kang, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA; Joshua Sunshine, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

Manuscript submitted to ACM

systems can be hard to learn, hard to understand, hard to construct, and hard to validate [29, 39, 50]. Visualizations can help bridge this gap by providing intuitive representations that ground complex concepts in more tangible views [31], enabling stakeholders to interpret formal models more easily. Under this assumption, many recent formal modeling works have visualization support – for example, the B modeling language has a visualization framework known as VisB [52], and the Alloy modeling language [26] has a built-in instance visualizer.

In the spirit of Krishnamurthi et al. [30], whether or not these visualizations are helpful requires empirical evaluations. Indeed, there have been some prior studies on how visualizations help formal modeling stakeholders and how they do not. In our prior work [32], we conducted an interview study with expert formal modelers. This interview study concluded that while visualizations are extremely helpful for "understanding, validating, and presenting formal models," their effectiveness is overshadowed by two issues:

- the lack of domain specificity – where visualizations do not accurately reflect the "user's mental model of how the system works" [13]; and
- the **lack of visual consistency** – where, when viewing formal models with multiple system states, visualizations evolve in a way that makes it hard for stakeholders to correspond the visual elements across the two states.

The first challenge of using visualizations, domain specificity, has received substantial attention in academia with tools such as Lightning [19], Sterling [16], Cope-and-Drag [42], and our work on Penlloy [32]. Many of these tools come with empirical evaluations, from light-weight expert evaluations for Penlloy [32], to full-fledged classroom studies for Sterling (via Forge [37]) and Cope-and-Drag [42].

In contrast, the second challenge, **visual consistency**, has seen little empirical investigation. Some engineering progress has been made, from Couto et al. [10] proposing the use of "transition managers" to enable consistency, to our prior work [32] formalizing different variants of visual consistency as constrained optimization problems solvable under the Penlloy framework. However, we still lack empirical evidence on whether incorporating visual consistency truly improves stakeholders' ability to understand and work with formal models.

To address this gap, in this paper, we designed and conducted a human-subject study that directly evaluated the role of visual consistency in model understanding and debugging. In short, participants used visualizations equipped with different levels of visual consistency to work through model debugging tasks across two formal models written in the Alloy specification language. Our goal was to assess whether, and in what ways, consistency improves stakeholders' ability to reason with visualizations. The study was structured around three research questions:

RQ1 How does the incorporation of *fully-consistent* visualizations, as opposed to the Alloy default visualizations which do not incorporate consistency, affect users' speed and accuracy in diagnosing model bugs?

RQ2 How does user performance in diagnosing model bugs differ when using *fully-consistent* versus *partially-consistent* visualizations?

RQ3 What roles does visual consistency play in users' model understanding and debugging from a *subjective* perspective?

In summary, our results show that visual consistency substantially improves efficiency and accuracy in debugging tasks (RQ1), with partial consistency offering benefits comparable to full consistency (RQ2). Moreover, post-experiment interviews revealed that consistency lowers cognitive overhead by helping participants track corresponding elements across states, while remaining unobtrusive to their workflow. The interviews also highlighted cases where consistency can have limited benefits, suggesting opportunities for combining consistency with other visual principles (RQ3).

The remainder of this paper is structured as follows: Sec. 2 provides background and related work on formal modeling in the Alloy modeling language, visualizations of instances of Alloy models, and the visual consistency problem; Sec. 3 outlines our study design; Sec. 4 presents the results of the experiments; Sec. 5 discusses the implications of these findings and the potential threats to validity of our study, and proposes future work; and Sec. 6 concludes the paper.

2 Background and Related Work

2.1 Formal Modeling

Formal modeling provides a mathematically rigorous framework for describing and reasoning about complex systems. Unlike natural language or informal diagrams which can be ambiguous, formal models are specified using a precise logical notation with only one interpretation. This precision enables systematic exploration of system behavior, automated checking of invariants, and early detection of design flaws. Formal modeling is especially valuable in domains where correctness is critical, such as distributed systems, safety-critical software, and cyber-physical systems because they can uncover subtle errors long before implementation.

Both academia and industry have successfully applied formal modeling to critical systems. For example, Zave used formal modeling to identify flaws in the Chord distributed protocol [54–56]; Cunha et al. modeled an aircraft arrivals manager to expose vulnerabilities in its human–machine interaction model [11]; and Amazon's adoption of formal modeling has "prevent[ed] subtle but serious bugs from reaching production" [38].

2.2 Modeling in Alloy

Alloy [25–27] is a lightweight formal specification language designed to model properties and behaviors of software systems, both static and dynamic. Based on first-order relational logic, Alloy offers a concise syntax for specifying a system as a set of logical constraints. Its key strength lies in the Alloy Analyzer, which automatically generates *instances* that satisfy a given set of constraints using an off-the-shelf Boolean satisfiability (SAT) solver. These instances supports two types of analyses: *model exploration* (akin to asking the analyzer to "generate a sample execution of the system") and *model checking* ("find an execution of the system that violates a given property").

Since these instances are essentially simulations of the system behavior, inspecting and understanding these instances allows stakeholders to explore the implications of their specifications, surface unintended behaviors, and build confidence in the correctness and reliability of the system design.

In the following sections, we illustrate the Alloy modeling workflow, discuss how visualizations help or do not help with the modeling process, and motivate the visual consistency problem.

2.3 Example: A Ring Network Model

To illustrate the Alloy modeling workflow, we present a simple "ring network" model, one of the example models used in our experiments. A ring network is a collection of nodes arranged in a single cycle (ring structure), where each node may be associated with one or more data elements. The full textual description of the ring network model can be found in Appendix A.1.

Upon reading this verbal description, a model specifier may start with a simple Alloy model, as follows:

```
1 sig Node {} // a sig (nature) is a basic element of the system being modeled
2 sig Data {}
3
```

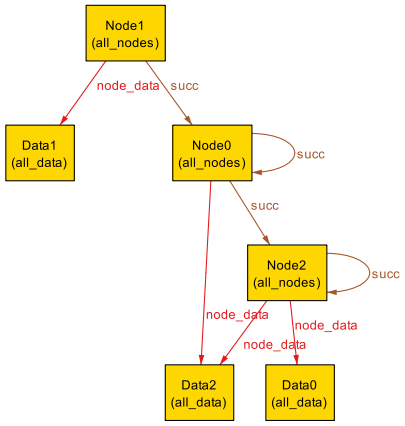
```
sig Network {                          // This sig organizes a network structure into:
  all_nodes: set Node,                 //   - the set of all nodes in the network,
  all_data: set Data,                  //   - the set of all data in the network,
  succ: Node -> Node,                  //   - the successor relation between nodes, and
  node_data: Node -> Data              //   - the association between nodes and data,
} {                                    // with additional constraints:
  succ in all_nodes -> all_nodes       //   - the successor relation only involves nodes in the network, and
  node_data in all_nodes -> all_data   //   - the association between nodes and data only involve nodes and
                                       //     data in the network.
}
```

2.4 Visualizations for Iterative Model Development

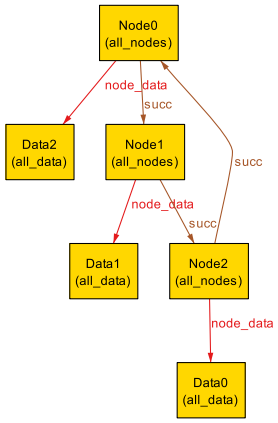
Even in this simple form, the model can be analyzed with the Alloy Analyzer. For example,

```
run {} for 3 but exactly 1 Network
```

This command asks Alloy to produce an example instance of this model, with one network and up to three nodes and data items. The Alloy Analyzer generates candidate instances, which are graphically depicted by the built-in visualizer in Alloy. Applying common settings (e.g., showing one network state at a time and hiding nodes in the instance but not in the network), the visualizer renders diagrams such as in Fig. 1a.



(a) Visualization of an instance of the buggy model without the invariant. This instance involves three nodes (`Node0`, `Node1`, and `Node2`) and three pieces of data (`Data0`, `Data1`, and `Data2`). Notice that the nodes are not arranged into a single ring, as required by the written description of the model. The visualization reveals this bug in an intuitive way.



(b) Visualization of an instance of the model with the invariant. Here, we have the same nodes and data as in Fig. 1a, but the nodes are indeed arranged into a single ring as required by the written description.

Fig. 1. Alloy-generated visualizations of instances of the ring network model, without (left) and with (right) the invariant that enforces the ring network properties (every node has exactly one successor, every node belongs to a ring, and all nodes share a single ring). In general, visualizations of Alloy instances can show unintended behaviors (bugs such as under-specifications and mis-specifications) of the underlying Alloy model.

In the visualization in Fig. 1a, three nodes (`Node0`, `Node1`, and `Node2`) and three data items (`Data0`, `Data1`, and `Data2`) appear as rectangular nodes, with relations between the entities (`succ` and `node_data`) rendered as edges between the rectangular nodes. However, the `succ` relation shows that the nodes are not arranged in a single ring. The visualization makes the modeling flaw immediately apparent: The current model does not enforce the ring structure.

This aligns with the results of our prior interview study [32], where participants emphasized the important role of visualization in surfacing unintended behaviors such as under-specifications and mis-specifications. As one participant put it,

> "*[I use visualizations to] see scenarios of the executions of my system, and be confronted with instances which are possibly not envisioned.*" (P10 of [32])

To address the under-specification bug of not enforcing the ring structure, the model specifier can add the following invariant that enforces the single-ring structure:

```
pred Invariant [net: Network] {
  // Every node has exactly one successor and exactly one predessor.
  all node : net.all_nodes | one node.(net.succ) and one (net.succ).node

  // Every node belongs to a ring.
  all node : net.all_nodes | node in node.^(net.succ)

  // Every node is reachable to every other node, so all node share a single ring.
  all node1, node2 : net.all_nodes | node1 in node2.^(net.succ)
}
```

With this invariant, Alloy generates only valid ring networks (Fig. 1b):

```
run {
  some net: Network |    // net is a network, such that:
    Invariant[net]       //  - net satisfies the invariant
} for 3 but exactly 1 Network
```

This iterative loop of modeling, visualizing, and refining reflects how practitioners often work with visualizations in Alloy [32]:

> '*I would build [a model] and visualize it along the way. I would start with a simple structure, visualize it, and I would say 'well that looks wrong, let's fix that'… I do this in an incremental loop.*" (P2 of [32])

2.5 Modeling and Visualizing Multiple States

Alloy can also model dynamic behaviors of systems through state transitions. Model specifiers can either model states manually, or leverage built-in support for dynamic behaviors in newer Alloy versions [7]. State transitions are then modeled as logical formulas that define what the new states should be.

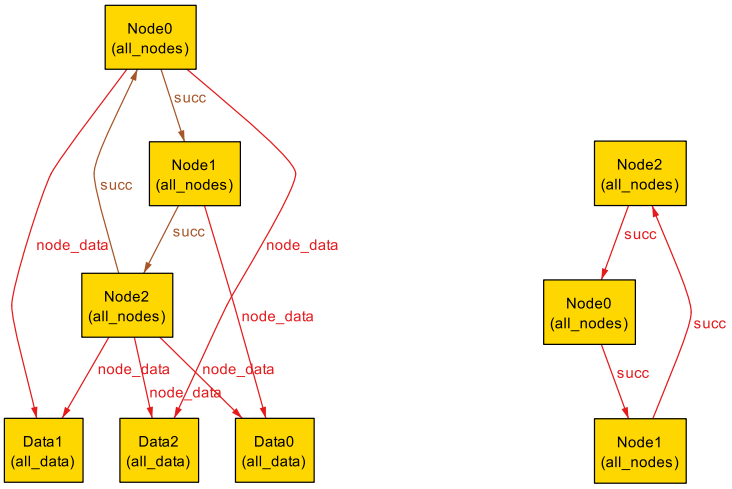
In our ring network model example, the states are just different network configurations, and the state transitions are operations that transform one network configuration into another one. For example, consider a simple operation `RemoveAllData` which removes all the data from the network. In Alloy, this operation would be implemented as the following predicate between two network configurations:

```
pred RemoveAllData[net0, net1: Network] {
  // State net0 is the pre-operation state; state net1 is the post-state.

  // The nodes and successor relations remain unchanged.
  net1.all_nodes = net0.all_nodes
  net1.succ = net0.succ

  // All the data and the associations from the nodes to the data are removed.
  net1.all_data = none
  net1.node_data = none -> none
}
```

Just like how Alloy allows users to explore and visualize valid instances of the network, we can also explore and visualize the executions of the operations:

```
run {
  some disj net0, net1: Network { // States net0 and net1 are different network states, such that:
    Invariant[net0] and           //   - net0 is a valid network configuration, and
    RemoveAllData[net0, net1]     //   - net1 is the result of removing all nodes in net0.
  }
} for 3 but exactly 2 Network     // This instance consists of two network states
```

Running this operation yields something like in Fig. 2. The visualization shows two states: on the left (`net0`), three nodes form a ring with three data items; on the right (`net1`), the same ring remains but all data is removed, as expected.



(a) Pre-operation state. (b) Post-operation state.

Fig. 2. Alloy visualization of the execution of the `RemoveAllData` operation. The pre-operation state `net0` is on the left; the post-operation state `net1` is on the right. This figure exemplifies the lack of visual consistency – the positions of the `Node` entities have changed across the two states.

2.6 The Visual Consistency Problem

Alloy-generated visualizations with multiple frames (e.g., Fig. 2) suffer from the problem of the **lack of visual consistency**, wherein the visualizations evolve in a way that makes it hard for stakeholders to precisely tell what has changed. Chiplunkar et al. [8] described visual consistency (which they dub "continuity") as: When a diagram evolves over multiple states, it should change "in such a way that the stakeholder can easily tell what has changed, and not more than it has to."

In Alloy modeling, the lack of visual consistency manifests through the positions of the corresponding nodes varying between the two frames. In Fig. 2, observe the ring structure formed by `Node0`, `Node1`, and `Node2` through the `succ` relation. Between the two frames, `Node2` moves from the bottom of the ring structure to the top; `Node1` from middle to bottom; and `Node0` from top to middle.

The lack of visual consistency makes understanding and analyzing model behaviors challenging in two ways, both of which we have observed in our Alloy modeling tasks:

- **The visualization appears unchanged despite node repositioning.** Fig. 2 serves as an example: The topology formed by `Node0`, `Node1`, and `Node2` remains a ring, but the nodes' positions have swapped. This may make it more likely for stakeholders to miss the potential critical change between the two frames, as the preservation of the ring structure gives the illusion of continuity. Indeed, prior cognitive science research on change blindness [47] concluded that people often "fail to notice changes to scenes when they do not produce a motion in our retina that attracts attention."
- **The visualization appears drastically different despite relatively minor changes.** An example is Fig. 3: The second frame is semantically the same as the first frame, except the removal of two edges. Because of how the built-in layout algorithm in Alloy lays the nodes out, even small semantic changes in the instance can propagates into a disruptive diagram-wide change, which may overwhelm the stakeholders and make it harder to pinpoint the exact semantically meaningful change. This, in turn, makes it harder to understand the behavior of the operation. Indeed, Rensink's work on change detection [44] concluded that radical changes between states can prevent stakeholders from detecting critical changes and can create confounding factors for their perceptive capability.

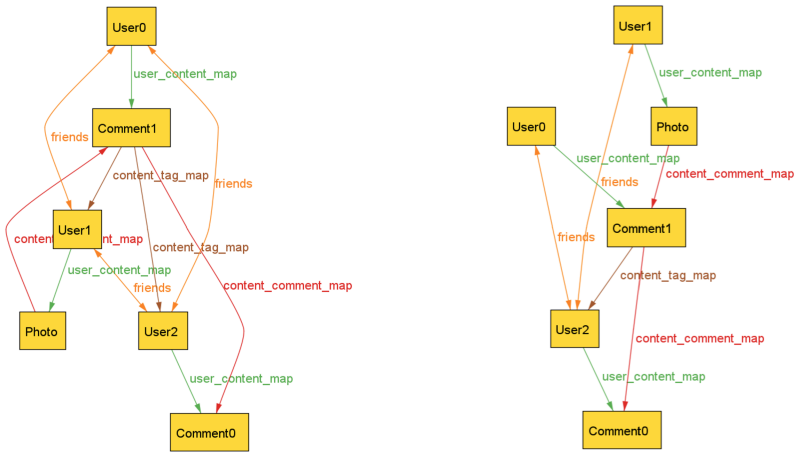
In both cases, the lack of visual consistency creates confusion and frustration among stakeholders. Indeed, participants of our interview study [32] lamented at the mental overhead necessary to overcome the lack of visual consistency:

> "*I don't like the fact that this structure has been rotated in a way that I have to look carefully to understand why it has been rotated. That's weird.*" (P2 of [32])

In another case-study literature, Zave [55] further claimed that while Alloy has "excellent visualization tools," the lack of visual consistency overshadows the benefits of Alloy's visualization tools and causes Zave to instead create hand-drawn visualizations for the generated instances of the Chord distributed protocol:

> "*The nodes move from [state] to [state]. Understanding Chord requires a fixed node layout in which the nodes are arranged in a circle in identifier order. ... I ended up drawing a picture ... by hand for each example or counterexample.*" (Sec. 5 of [55])

Prior literature has studied the use of visual consistency in broader visualization domains. For example, Archambault and Purchase [2] concluded that visual consistency (which they dubbed "mental map preservation") helps with graph-orientation tasks like "finding a specific location or route between two locations" in graphs. Bridgeman and Tamassia



(a) Pre-operation state. (b) Post-operation state.

Fig. 3. Another example of inconsistency in Alloy-generated visualizations. The post-operation state (right) differs from the pre-operation state (left) only by the removal of two edges. Yet such a small semantic change propagates into a drastically different diagram.

[5] explored different metrics of "graph similarity" and evaluated whether or not these metrics correspond to users' perceptions. Qu and Hullman [43] studied how visualization designers author visualizations with or without consistency. These works inspired further engineering efforts to alleviate the lack of visual consistency, such as new algorithms to preserve graph topology [1, 15]. Applied to formal modeling specifically, Couto et al.'s work in improving Alloy visualizations [10] identified visual consistency as a requirement for good visualizations, and proposed the use of transition managers to enable consistency. Our prior work [32] conducted an exploratory study that taxonomized visual consistency into multiple categories, and formalized these categories as numerical optimization problems solvable by Penlloy, our constraint-based visualization framework for Alloy models.

2.6.1 Full and partial consistencies. Our prior work [32] summarized visual consistency into two taxonomies:

Hard and soft consistencies decide how "important" visual consistency is in the full layout problem of the diagram. Hard consistency models consistency as hard constraints that must be satisfied. On the other hand, soft consistency models consistency as soft constraints that the layout system tries to satisfy, but can be violated for the overall aesthetics of the diagram.

Positional and relative consistencies distinguish on the definitions of the objective functions for the aforementioned hard/soft constraints. The objective function for positional consistency measures how each node's position change between states. On the other hand, the objective function for relative consistency measures the changes to the relative positions between nodes related by edges.

These definitions combine into hard positional, hard relative, soft positional, and soft relative consistencies [32]. We further observe that these four categories can also be summarized as two "levels" of consistency, whose effects we study in this paper:

Full consistency requires that the positions of the corresponding nodes across the two frames to remain fixed. This corresponds to hard positional consistency mentioned above.

Partial consistency requires key structural or visual elements to be recognizably preserved across diagrams, but allows elements to "float" around. In other words, partial consistency requires some recognizable commonalities between the two diagrams. This corresponds to soft positional, hard relative, and soft relative consistencies mentioned above.

2.7 Empirical Evaluations of Formal Methods Usability

In the spirit of Krishnamurthi et al. [30], whether or not supposed improvements to formal modeling actually benefit formal modelers cannot be assumed and must be subjected to empirical evaluations. Some aspects of formal modeling have already been tested in this way. Mansoor et al. [34] examined the usability of Alloy, finding that even non-novice users struggle to interpret generated instances. Greenman et al. [21] analyzed the understandability of linear temporal logic, uncovering persistent misconceptions across both learners and practitioners. Complementing these findings, Shevrin et al. [46] demonstrated how syntactic properties of Boolean formulas systematically affect comprehension, and Ma'ayan et al. [33] evaluated reactive synthesis through the Spectra modeling language [35] in classroom contexts, highlighting both its pedagogical benefits and the challenges students encounter when reasoning about synthesized systems. Many of these prior works converge on the same theme: Formal modeling has great potentials, but comes with usability challenges.

2.7.1 Evaluations of visualizations. A myriad of work studies the usefulness of visualizations in the context of understanding complex systems. Classically, Larkin and Simon's seminal paper [31] argued that while informationally equivalent, because visualizations can support more efficient information search operators, "a diagram can be superior to a verbal [textual] description." Since then, visualization research has emphasized the importance of design principles that make visual representations more effective for human cognition. For example, the Gestalt principles [48, 49] are often discussed in visualization literature as factors that shape users' ability to understand complex systems, with a myriad of work studying the impacts of these principles on user performances on diagram-related tasks [23, 40, 51].

Applied to formal modeling, Danas et al. [12] used crowd-sourced experiments to investigate Alloy model-finding outputs, showing that Alloy's visualizations can at times mislead users, and our interview study [32] probed the strengths and limitations of existing visualization techniques for expert formal modeling stakeholders.

Many recently proposed improvements to visualizations have similarly been put to empirical scrutiny. For example, domain-specific visualizations are often cited as crucial by stakeholders [32], and domain-specific visualization tools such as Sterling [16] (via Forge [37]), Penlloy [32], and Cope-and-Drag [42] incorporated varying levels of empirical evaluations, from lightweight expert feedbacks to full-scale classroom studies.

2.7.2 Lack of evaluations for consistency. On the contrary, visual consistency is one principle that is frequently invoked by formal modeling stakeholders but rarely examined in depth. For example, participants in our interview study [32] actively expressed the need for improved consistency. While interview studies are valuable for surfacing users' perceptions of their needs, there is sometimes mismatch between user perceptions and reality [9, 45]. Sometimes, features widely assumed to be indispensable turned out not to yield the expected benefits once tested rigorously [17, 20]. Perceptual studies like Rensink [44] and Simons and Levin [47] concluded that, in general, the lack of consistency hampers change detection; additional empirical studies like Archambault and Purchase [2] concluded that consistency helps with graph-orientation tasks. But these studies did not evaluate consistency in the context of formal modeling, nor did they explore its broader roles in helping stakeholders understand complex underlying concepts behind the diagrams. Indeed, some contrary evidence from literature suggested that strict consistency may obscure or even mislead

formal modeling stakeholders (e.g., Sec. 4.2 of [37]). Despite its rhetorical prominence, there is still no rigorous evidence showing whether and how visual consistency actually helps modelers.

This work addresses that gap through a series of human-subject experiments. Our studies not only measure the concrete effects of visual consistency on model specifier' performance, but also shed light on how stakeholders actually make use of it in practice. The findings will help separate assumptions from reality, providing a firmer empirical foundation for the design of visualization tools.

3 Study Design

Through a series of human-subject experiments, we aimed to understand how visual consistency actually helps model specifiers understand and debug formal models, both important uses of visualizations [32]. Succinctly, participants of these experiments used visualizations equipped with different levels of consistency to diagnose (debug) two ill-specified models across eight tasks.

The research questions (RQs) for this study were:

RQ1 How does the incorporation of fully-consistent visualizations, as opposed to the Alloy default visualizations, affect users' speed and accuracy in diagnosing model bugs?

RQ1a How does it affect performances in *identifying* model bugs from visualizations?

RQ1b How does it affect performances in *localizing* model bugs within the model code?

RQ2 How does user performance in diagnosing model bugs differ when using *fully-consistent* versus *partially-consistent* visualizations?

RQ3 What roles does visual consistency play in users' model understanding and debugging from a *subjective* perspective?

3.1 Participant Recruitment and Selection

We recruited participants who had recent experience with the Alloy modeling language. These experiences could come from coursework or projects involving Alloy. Potential participants were drawn from student rosters of formal methods courses at our institution (which teaches Alloy as part of its curriculum), as well as from other institutions that offer Alloy-oriented courses. We also recruited a small number of researchers known to have prior experience with Alloy. We chose this population because they had recent familiarity with Alloy – including concepts such as models, instances, and visualizations – and because students were the group most likely to need support in understanding and debugging formal models.

Eligible potential participants were contacted via email and invited to participate voluntarily. Each potential participant completed a pre-experiment survey to confirm eligibility. These surveys confirmed that participants are above 18 years of age, and collected information on the students' prior experiences with Alloy. Students from the aforementioned formal methods courses (at our institution or elsewhere) were considered automatically eligible, as enrollment in those courses verified their recent Alloy experience. Researchers who had previously worked with Alloy were also automatically confirmed as eligible. For all respondents, the survey included screening questions asking them to either confirm the Alloy-related course(s) they have taken, or describe their recent Alloy-related activities (project or research use, etc.). Participants who could either provide concrete and verifiable course information or document a concrete and recent Alloy experience were invited to complete the full study. We did not collect additional demographics information except confirming that the participants are above 18 years of age.

A total of 32 potential participants completed the pre-experiment survey, and a total of 19 participants completed the study. Out of them, 16 took Alloy-related course(s): one took them in or before the Fall 2022 semester; two took them in the Fall 2023 semester; and 13 took them in the Fall 2024 semester. Four of them worked on research projects that involved Alloy modeling, one of whom also took the course(s). Each participant who completed the study received a gift card of value USD 25, except for one who declined the reward.

3.2 Participant Tasks

Each participant worked on two Alloy models, each consisting of four model-debugging tasks, for a total of eight tasks. Each task involved an operation in the model whose Alloy specification was intentionally buggy. Succinctly, participants were shown a visualization of the execution of an operation that contained a bug (such as a missing or incorrect constraint affecting the intended pre- or post-conditions), and were asked to use these visualizations to debug the operation. See Sec. 3.4 for a discussion of the types of visualizations that we provided to the participants.

The participants were notified that the tasks all involved buggy operations whose executions they would see in the provided visualizations. For each task, we asked the participants to answer two questions (in this order):

Q1 (for RQ1a) Using the visualization we provide, identify the bug in the operation. What appears wrong in the visualization?

Q2 (for RQ1b) Using the visualization and the buggy Alloy code for the operation, localize the bug from the buggy Alloy source code.

When answering each question, we encouraged the participants to first think about their answer. Once they have their answers in mind, we asked them to write down their answers. We measured both the thinking time (T, in seconds), the writing time (W, in seconds), and the correctness (C, as a binary variable) of their responses to each question.

The order of the two models and the orders of the tasks within each model were fully randomized. Before the start of each experiment session, we provided the participants with a practice task so that they can get familiar with the experiment tasks.

3.3 Experiment Models and Bugs

The first model (Model 1) is the aforementioned *ring network* model, which models a network of nodes arranged into a single ring structure. Each node can optionally be associated with one or more pieces of data. The second model (Model 2) is a more-complex *social network* model, which models entities like users, photos, and comments, and relations like friendship, content-ownership, tagging, commenting, etc., within a social network. See Appendix A for mode details on the models and the tasks and bugs associated with each model.

3.4 RQ1, RQ2: Experiment Visualizations

For each task and its associated buggy operation, we generated an instance of that operation's execution using Alloy's "run" command. Each participant then saw a pair of visualizations presented side-by-side: The left-hand diagram depicted the state *before* the operation, and the right-hand diagram depicted the state *after*. Together, these diagrams illustrated the effects of the buggy operation. A brief textual reminder of the executed operation was also included, enabling participants to connect the visualizations to the corresponding operation. This explicit before/after comparison formed the core of the debugging task.

Each task used one of two top-level visualization conditions, assigned at random:

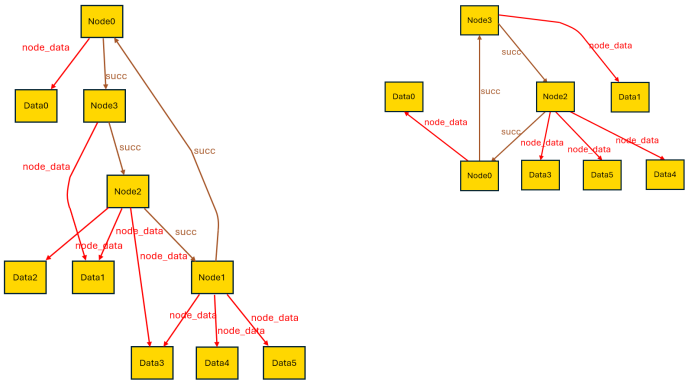
Default Visualization of Alloy We used Alloy's visual output, augmented with a few common "theming" options that make the diagrams easier to read (e.g., hiding entities present in the model but irrelevant to the ring or social network). Examples appear in Fig. 4a and Fig. 4b.

Fully-Consistent Visualization (FC) Starting from Alloy's default pre- and post-state diagrams f_1 and f_2, we created a modified post-state diagram f_2' that is visually consistent with f_1. For every node common to both states, we retained its position from f_1. Nodes present only in f_1 were removed, while nodes new to f_2 were placed near their connected neighbors in f_2' without introducing intersections. When possible, edges kept their original shapes; otherwise, they were rerouted to minimize crossings. This procedure mimicked Alloy's layout behavior for new nodes and changed edges. We then presented f_1 and f_2' as the fully consistent before/after pair, after applying the same set of common "theming" options as in the Default condition. An example is shown in Fig. 4c.

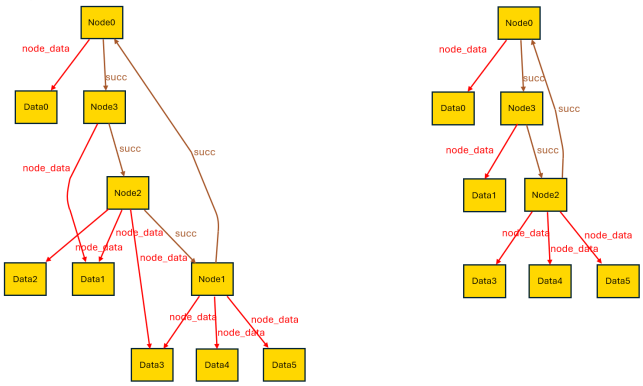
Because the conditions were assigned randomly for each participant and each task, we expected that half of the tasks each participant completed fell under the FC condition and the rest under the Default condition, making this study within-subject. The expected proportions matched the real distribution of the conditions: Across the participants, the mean proportion of FC tasks was .53 (σ = .09). Our analysis (Sec. 4.1.1) operates at the task level, with each completed task contributing one observation. Therefore, small variations in the number of tasks per condition per participant, due to random assignment, do not affect the validity of the comparison. We compared participant performance under the FC and Default conditions to answer RQ1.

In practice, Alloy's layout algorithm sometimes, by coincidence, produces diagrams that exhibit a degree of partial consistency. Although individual nodes may move, substantial substructures often preserve their shape and spatial relationships across frames. For every default visualization, we therefore manually sub-classified it as either "**Not-Consistent**" (**NC**, see Fig. 4a for example) or "**Partially-Consistent**" (**PC**, Fig. 4b). A visualization was labeled Partially-Consistent when key substructures (at least three interconnected nodes) maintain their overall shape and relative positioning across states, and together cover a majority of the nodes shared between frames. For example, Fig. 4b retains two major substructures – the cluster formed by Node0, Node2, Node3, Data0, and Data1, and the cluster formed by Data3, Data4, and Data5 – that remain stable across the pre/post diagrams. We emphasize that when a participant was assigned Alloy's default visualization for a task, its sub-classification of NC and PC is pre-determined not randomly, but solely by the task.

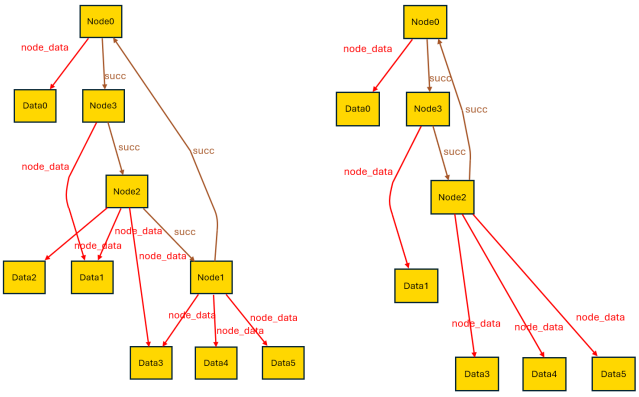
Out of the eight tasks, four had Default visualizations sub-classified as NC, and four sub-classified as PC. See Tables 5 and 6 in Appendix A on exactly which tasks are sub-classified as NC or as PC. Once again, because the top-level conditions (FC versus Default) were randomly assigned, we expected that half of the tasks each participant completed fall under the FC condition, 1/4 under the NC condition, and 1/4 under the PC condition. These expected proportions once again matched the real distribution of the experiment conditions: Across all participants, the mean proportion of FC tasks were .53 (σ = .09), the mean proportion of PC tasks were .24 (σ = .09), and the mean proportion of NC tasks were .23 (σ = .09). This naturally-occurring variation allowed us to compare participant performances on the FC and PC conditions, thereby answering RQ2.



(a) A mockup of the **Default Visualization**, the **Not-Consistent (NC)** variant. Observe that the position of every node in the diagram has drastically changed.



(b) The **Default Visualization**, the **Partially-Consistent (PC)** variant. Despite nodes repositioning, observe that significant substructures (one formed by Node0, Node2, Node3, Data0, and Data1; and one formed by Data3, Data4, and Data5) have remained consistent internally.



(c) The manually-constructed **Fully-Consistent (FC)** variant. Observe that the position of every node has remained unchanged.

Fig. 4. This example gallery illustrates the differences between Alloy's **Default Visualization** and our manually constructed **Fully-Consistent (FC)** variant, and the differences between the **Partially-Consistent (PC)** and **Not-Consistent (NC)** variants within the Default Visualization. We applied these notions of consistency onto an instance of the buggy operation `MergeWithSuccessor` in Model 1 to generate this example gallery. See Table 5 for a description of the bug.

3.5 RQ3: Exit Interview

To complement the quantitative results from the visualization tasks, we conducted a brief exit interview with each participant at the end of the study. The goal of this interview was to gather subjective insights into how participants perceived and experienced the different types of visualizations.

Participants were not initially informed that the study focused on different forms of visual consistency. At the start of the interview, we first asked participants about the perceived usefulness and limitations of the visualizations they encountered, without introducing the notion of consistency or any categorization of visualization types. Only after these initial, open-ended questions did we reveal that the experiment examined the effects of visual consistency. We provided a brief definition of our notion of visual consistency (that "corresponding nodes should not move between the states"). Following this reveal, we asked participants to reflect more explicitly on whether consistency, or the lack thereof, had influenced their performance. Example questions included:

- Which visualizations did you find easier or harder to work with, and why?
- How did the consistency (or the lack thereof) across frames influence your ability to track objects and reason about the system?
- In what ways, if any, did the visualization style impact your confidence in identifying the buggy operation?

Participants were encouraged to refer to specific tasks and visualizations they had encountered during the study, and many grounded their responses in concrete visualizations they saw.

These interviews allowed participants to reflect on their strategies, frustrations, and preferences beyond what could be inferred from task performance alone. Their comments revealed, most importantly, how it shaped their reasoning processes – for example, whether consistent layouts reduced cognitive load, enabled faster mapping of objects across states, or supported more systematic debugging strategies.

By analyzing the themes emerging from these interviews, we were able to address RQ3 (what roles does visual consistency play in model understanding and debugging from a subjective perspective). Specifically, the interviews provided evidence about the subjective experiences of participants, shedding light on the mechanisms behind the quantitative effects observed in RQ1 and RQ2, and offering a richer understanding of the broader value of consistency in visualization design.

3.6 Logistics

Each study session, including both the task portion and the exit interview, lasted at most two hours and was conducted remotely via Zoom. With participants' consent, Zoom sessions were recorded to support later analysis of the participant interactions during the tasks and interview responses. These recordings included audio, video, and screenshare. Two participants declined being recorded. For these two participants, timing and correctness data were not affected by the lack of recordings, because these data were recorded throughout the experiment sessions. However, for the exit interviews, analysis relied only on sporadic notes taken by the researchers during the interviews rather than on recordings or verbatim transcripts.

All tasks were completed within a pre-configured GitHub Codespace environment created specifically for the study. The environment was self-contained and included all task materials, such as written descriptions of the models, explanations of the operations under consideration, detailed task instructions, and the visualization file assigned to each participant. Task instructions were embedded directly in the Codespace, enabling participants to work independently once a task began. Therefore, the experimenter provided only minimal verbal guidance over Zoom, limited to directing

participants to the next tasks (e.g., "Please open Model 1, Task 4") and answering brief logistical questions (e.g., "Can I switch back-and-forth between the verbal description of the operation and the visualization?"). No additional explanations of the models or operations were provided beyond the written materials.

Before beginning the experimental tasks, participants completed a practice task designed to familiarize them with the Codespace environment, the visualization format, and the overall task workflow. During this practice task, the experimenter answered general questions to ensure that participants understood the study mechanics and were comfortable with the interface and procedure.

Participants recorded their responses by completing prepared files containing structured, fill-in-the-blank prompts corresponding to each task. These files were then pushed to the experiment repository, ensuring that responses were collected in a consistent and analyzable format.

In addition to collecting task responses, the experimenter occasionally recorded observational notes during the Zoom sessions when notable behaviors were observed – for example, participants repeatedly moving the mouse cursor between pre- and post-state visualizations while reasoning about a task. These observations provided contextual support for interpreting both quantitative results and interview responses.

4 Results

Tables 1 and 2 summarize the raw timing data and correctness data, respectively, across the different visualization conditions between RQ1 and RQ2.

4.1 RQ1: Effects of Full Consistency

To evaluate the impact of full consistency on participants' performance, we analyzed both the time taken and the correctness of their answers.

4.1.1 Analysis method. Each observation in our analysis corresponds to a single participant–task trial. For each such observation, we recorded the visualization type shown to the participant, the task identifier, the participant identifier, the associated time measures (thinking time T, writing time W, and total time $T + W$) and correctness measures (binary variable C) for both Q1 and Q2. We measured the effects of consistency on participant performance through *linear mixed-effects regression models* [18]. Linear mixed-effects regression models are commonly advocated for [28] and used [24] in recent HCI works for analyzing repeated-measures designs with participant-level and task-level random effects. These mixed-effects regression models are a strict superset of traditional ANOVA-based approaches and allow for more flexible modeling of both continuous and categorical response variables through generalized linear mixed models [4].

Each time measurement was modeled using a mixed-effects linear regression model of the form:[1]

$$\log(\text{time}) = \beta_0 + \beta_1(\text{visualization type}) + \alpha_{\text{participant}} + \alpha_{\text{task}} + \varepsilon.$$

We applied a log transformation to the timing measures to satisfy the model's assumption of homoskedasticity. The fixed-effect predictor "visualization type" is a binary indicator (Default vs. FC). Participants and tasks were represented as categorical grouping variables. The terms $\alpha_{\text{participant}}$ and α_{task} are random intercepts, with one intercept estimated for each participant and each task, respectively. These random intercepts capture systematic differences in baseline response speed attributable to individual participants (e.g., one participant might be more familiar with Alloy than

[1]The βs in the regression model equations are used conceptually to describe the model parameters, and do not directly correspond to the actual fitted βs (in Tables 3 and 4) which we use to discuss the regression results.

RQ	Task	Condition	N	Mean (s)	95% CI Lower	95% CI Upper
RQ1	Q1 (think)	Default	71	51.27	43.69	60.18
		FC	80	39.50	33.81	46.14
	Q1 (write)	Default	70	35.17	30.35	40.76
		FC	79	36.56	31.80	42.02
	Q1 (total)	Default	70	90.87	79.29	104.15
		FC	79	81.41	71.81	92.30
	Q2 (think)	Default	71	52.22	42.63	63.97
		FC	79	43.63	36.46	52.22
	Q2 (write)	Default	69	43.62	37.57	50.64
		FC	78	43.74	37.87	50.53
	Q2 (total)	Default	68	102.62	89.41	117.78
		FC	77	90.20	78.77	103.28
RQ2	Q1 (think)	FC	80	39.50	33.81	46.14
		PC	36	38.19	31.80	45.85
		NC	35	69.43	55.07	87.53
	Q1 (write)	FC	79	36.56	31.80	42.02
		PC	35	34.22	27.54	42.52
		NC	35	36.15	29.29	44.62
	Q1 (total)	FC	79	81.41	71.81	92.30
		PC	35	74.23	62.48	88.18
		NC	35	111.25	91.38	135.45
	Q2 (think)	FC	79	43.63	36.46	52.22
		PC	36	45.01	33.68	60.16
		NC	35	60.85	45.56	81.27
	Q2 (write)	FC	78	43.74	37.87	50.53
		PC	35	44.83	36.46	55.12
		NC	34	42.41	33.81	53.20
	Q2(total)	FC	77	90.20	78.77	103.28
		PC	35	97.82	79.22	120.77
		NC	33	107.97	89.69	129.97

Table 1. Tabular summary of the timing data (think, write, and total) we collected for RQ1 and RQ2. The column "**N**" stands for the number of observations. Because thinking times are positively skewed and the regression analyses were conducted on log-transformed times, we summarize raw data using geometric means. The geometric mean represents the central tendency on the same multiplicative scale assumed by the model. Confidence intervals for geometric means were computed by constructing intervals on the log-transformed scale and exponentiating the resulting bounds.

another) and to tasks (e.g., one task might be inherently easier than another), and are assumed to be drawn from mean-zero normal distributions. In plain terms, our model tests whether participants were faster when using consistent visualizations, while statistically controlling for repeated measurements from the same participant and for variation in task difficulty. We checked the residual plots and found no evidence of misfits.

Similarly, since correctnesses (C) of the answers to Q1 and Q2 are binary, we fitted mixed-effect logistic models of the form

$$\text{logit}(\Pr(\text{C} = 1)) = \beta_0 + \beta_1(\text{visualization type}) + \alpha_{\text{participant}} + \alpha_{\text{task}} + \varepsilon.$$

The model estimates the association between visualization type and the odds of producing a correct answer in our collected data, while once again accounting for participant- and task-level variability. We do not use the model to make

RQ	Task	Condition	N	Correct %	95% CI Lower	95% CI Upper
RQ1	Q1	Default	71	84.51	74.35	91.12
		FC	80	95.00	87.84	98.04
	Q2	Default	70	87.14	77.34	93.09
		FC	78	84.62	75.01	90.97
RQ2	Q1	FC	80	95.00	87.84	98.04
		PC	36	88.89	74.69	95.59
		NC	35	80.00	64.11	89.96
	Q2	FC	78	84.62	75.01	90.97
		PC	36	91.67	78.17	97.13
		NC	34	82.35	66.49	91.65

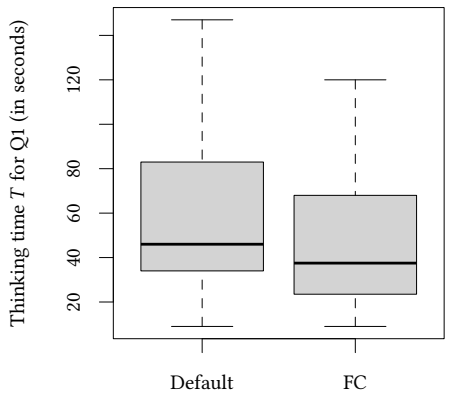
Table 2. Tabular summary of the correctness data. The confidence intervals for correctness were computed using Wilson's method [6].

predictions about future participants; rather, the fixed-effect coefficients quantify differences in correctness observed in this study. We performed likelihood-ratio tests [41, 53] and simulated-residual (disperson, zero-inflaction) tests [14, 22] on these regression models and found no evidence of misfits.

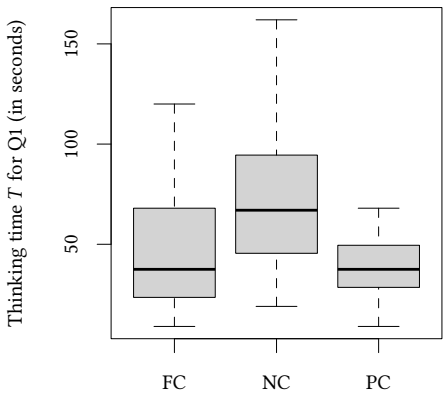
4.1.2 Analysis results. The results of these regression models are presented in Table 3. For identifying the bug from the visualizations (RQ1a, Q1), we found that consistent visualizations significantly reduced participants' thinking time ($\beta = -0.271$, $p = .001$) and improved correctness ($\beta = 1.470$, $p = .031$). Accounting for the log and logistic transformations, our regression models estimated that the thinking time reduced from 51.80 seconds to 39.51 seconds, a nearly 25% drop. This agrees with the raw data in Table 1, and is further illustrated by the boxplot in 5a. The estimated marginal probability of a correct response within our sample increased from .89 to .97, which is also similar to what the raw data shows in Table 2. Writing time was unaffected ($\beta = .080$, $p = .203$), and the effect on total time was marginal ($\beta = -0.102$, $p = .073$). Importantly, the direction and magnitude of the estimated effect on total time are consistent with the reduction observed in thinking time. It is therefore reasonable to hypothesize that full consistency also reduces overall task completion time. However, with only 19 participants and substantial between-participant variability in writing time – an activity not directly targeted by our treatments – we lacked sufficient statistical power to detect this effect with conventional significance thresholds. However, these results still suggest that consistency primarily helped participants reason more efficiently during the initial recognition phase and increased the likelihood of giving correct answers.

For localizing the bug in the Alloy code (RQ1b, Q2), we found no significant effects of consistency on thinking time ($\beta = -.113$, $p = .244$), writing time ($\beta = -.007$, $p = .915$), total time ($\beta = -0.104$, $p = .120$), or correctness ($\beta = -0.158$, $p = .752$). Thus, while consistency improved performance in the visualization-based task, we did not find any evidence of the knowledge transfer from bug identification to bug localization.

In summary, consistent visualizations helped participants more quickly and accurately identify bugs directly from the visual representation but did not provide measurable advantages when participants transitioned to working with the underlying Alloy code.



(a) A boxplot of the participant thinking times T (in seconds) for Q1, sliced by the visualization type (Default or FC) that participants saw. Full consistency, overall, decreased participants' thinking times.



(b) A boxplot of the participant thinking times T (in seconds) for Q1, sliced by the visualization type (FC, NC, and PC) that participants saw. While the non-consistent visualizations significantly increased the response time compared to fully-consistent visualizations, partially-consistent ones did not result in a discernible difference.

Fig. 5. Boxplots of participant thinking times T (in seconds), sliced into two visualization types for RQ1 and three visualization types for RQ2. These plots illustrate the effects of different variants of visual consistency described in Sec. 4.1 and Sec. 4.2. The boxplots are rendered after removing the outliers.

	Q1: Identify the Bug				**Q2: Localize the Bug**			
	$\log T$	$\log W$	$\log(T+W)$	logit$C$	$\log T$	$\log W$	$\log(T+W)$	logit$C$
Default Visualizations	3.947	3.555	4.515	2.091	3.903	3.788	4.617	2.115
Effects of FC (β)	$-.271$ ($p = .001^{**}$)	.080 ($p = .203$)	$-.102$ ($p = .073$)	1.470 ($p = .031^{*}$)	$-.113$ ($p = .244$)	$-.007$ ($p = .915$)	$-.104$ ($p = .120$)	$-.158$ ($p = .752$)

Table 3. Regression results for RQ1, comparing FC visualizations against the default visualizations. The phrase logit$C$ is short-hand for $\text{logit}(\Pr(C = 1))$. The "**Default Visualizations**" row contains values of the response variables ($\log T$, $\log W$, $\log(T+W)$, and logit$C$) when the Default visualizations are provided. The "**Effects of FC** (β)" row encodes the effects that Fully-Consistent visualizations make on the values of the response variables, compared to the Default visualizations. For example, for Q1, $\log T \approx 3.947$ with the Default visualizations; Fully-Consistent visualizations changes $\log T$ by approximately $\beta = -.271$, significantly with $p = .001$.

4.2 RQ2: Partial Consistency vs Full Consistency

As discussed in Sec. 3.4, we can manually categorize the "Default" variants of the visualizations into being either Partially-Consistent (PC) or Not-Consistent (NC), depending on whether or not there are significant recognizable portions of the diagram that remained consistent. Then, by setting "Fully-Consistent" as the base level and fitting a similar set of regression models as in RQ1, we can compare the effects of partial consistency against full consistency. The results of these regression models are presented in Table 4.

For identifying the bug from the visualization (Q1), compared to fully-consistent visualizations, we found that non-consistent visualizations significantly increased participants' thinking time ($\beta_1 = .426$, $p < .001$) and reduced correctness ($\beta_1 = -1.847$, $p = .017$) relative to fully-consistent visualizations. These results align with what we observed in RQ1, where consistency was shown to improve both efficiency and accuracy. In contrast, partially consistent visualizations did not differ significantly from fully consistent ones in either response time (all $p > .3$) or correctness ($\beta_2 = -1.005$,

	Q1: Identify the Bug				Q2: Localize the Bug			
	$\log T$	$\log W$	$\log(T+W)$	logit$C$	$\log T$	$\log W$	$\log(T+W)$	logit$C$
FC	3.549	3.635	4.415	3.549	3.790	3.779	4.512	1.918
Effects of NC (β_1)	.426 ($p < .001^{***}$)	−.140 (p=.107)	.156 ($p = .051$)	−1.847 ($p = .017^{*}$)	.143 ($p = .294$)	−.058 ($p = .540$)	.065 ($p = .495$)	−.020 ($p = .974$)
Effects of PC (β_2)	.1114 ($p = .336$)	−.019 ($p = .835$)	.045 ($p = .586$)	−1.005 ($p = .222$)	.082 ($p = .553$)	.075 ($p = .437$)	.144 ($p = .130$)	.435 ($p = .566$)

Table 4. Regression results for RQ2, comparing PC and NC visualizations against FC visualizations. The row "**FC**" contains values of the response variables under the Fully-Consistent (FC) condition, while the rows "**Effects of NC** (β_1)" and "**Effects of PC** (β_2)" encode the effects of the NC and PC conditions, respectively, on the response variables.

$p = .222$). See the boxplot in Fig. 5b which illustrates this result for thinking time. In other words, participants performed just as well with partial consistency as they did with full consistency.

This finding is encouraging. First, it reaffirms the detrimental impact of non-consistency: Participants need more time and are less accurate when confronted with inconsistent visualizations. Second, and more importantly, it shows that partial consistency can be just as effective as full consistency in supporting bug identification. This is a valuable insight because full consistency often comes at a cost: Enforcing global alignment may produce visualizations that look less natural, more cluttered, harder to read, and more misleading (see RQ3 later and Sec. 4.2 of [37]). The fact that partial consistency suffices means that we do not need to pursue full consistency in every case. Instead, ensuring that key structures remain recognizably stable across states may strike the right balance between usability and aesthetics.

Thus, these results point toward a practical design principle: Visualizations may only need to guarantee partial consistency to preserve user performance, avoiding the drawbacks of enforcing full global consistency.

4.3 RQ3: Subjective Results

We performed manual transcription, axial coding, and thematic analysis on the exit interviews to understand exactly how consistency affects model understanding and debugging. For participants who declined being recorded, we could only work with sporadic notes we took during the interviews without any concrete quotes. We uncovered some interesting insights from our participants, but acknowledge that they may not generalize to all Alloy users and all Alloy bug-finding and model-understanding tasks.

4.3.1 Participants look for differences in the diagrams. One of the most overarching themes is that participants compare the diagrams representing the pre-state and the post-state. One participant (P14) noted that this is reflected through their mouse cursors constantly moving between correspondding nodes in the two diagrams. Upon seeing the two diagrams during the post-experiment interview, nine different participants (P1, P3, P4, P6, P9, P10, P11, P14, P19) spontaneously mentioned comparing differences between two states as their primary strategy for finding errors in the diagram. For example, P1 stated that instead of fully comprehending the two diagrams individually, they found it "*much easier ... to compare what's the difference between the two diagrams*" (P1).

Comparing the two diagrams allowed participants to identify what has changed in the state as a result of the operation. Participants did this methodically, by going through each node in the pre-state and locate their corresponding node in the post-state:

> "*My first step is trying to localize the elements [nodes] that I'm trying to work with, ... then I try to look at the relations and compare them between the two ... diagrams.*" (P9)

After identifying the changes, participants tried to confirm whether or not these changes are expected, based on their understanding of the specification of the operation.

> "*Based on my understanding of the operation, usually I would, like, check, ..., the new relationships which are added into the [pre-state], if the new relationship is right, and also check if any of the relationship is deleted, and also I will check if that follows the definition of the operation.*" (P4)
> "*I just look for the differences initially, and then, because ... I was given the description [of] what is supposed to happen, I check to see if that's actually happening between the two diagrams.*" (P6)

In other words, participants built mental images of what a correct implementation of the operation is supposed to do. They then tried to compare the post-state visualization against this mental image to identify what went wrong in the model.

4.3.2 The lack of consistency indeed makes comparison difficult. When participants used the default, non-consistent visualizations, a considerable number of participants (P5, P7, P9, P13, P14, P16, P19) found it cumbersome to identify both what has changed between the pre-state and the post-state, and how the actual post-state differs from the expected, correct post-state. This is an issue that multiple participants spontaneously brought up when asked about potential difficulties in using the visualizations to compare two states:

> "*I wouldn't say [the comparison] is super difficult, but, because in the final state [post-state] is often permuted, it's a little bit of trying to see where everything has gone.*" (P3)

For some participants, the lack of consistency created extra work that actively impeded problem solving, as participants had to mentally reorient between model debugging and locating corresponding nodes. P5 spontaneously complained about this difficulty, even before we revealed to them that the study was on the effects of consistency:

> "*The arrangement of the different nodes themselves always changes, so between diagrams one [pre-state] and two [post-state], things just move around, so that becomes a lot harder to figure out, because now you have to relocate the same node, figure out the same relations. That definitely becomes a lot more difficult, so you can't have ... more than a couple signatures [types of objects] in a diagram.*" (P5)

Some participants believed that this extra work resulted in wasted time that could have been spent for problem-solving:

> "*Because of the way Alloy generates the elements in a random order, they don't tend to match between diagram one and diagram two, so it just increases the amount of time to basically comprehend the differences. ... I was having to switch back to try to remember the relationships [between two states] multiple times, which ended up being quite a time waste.*" (P9)

4.3.3 Visual consistency lowers the cognitive overhead. Compared to the non-consistent diagrams, participants (P1, P3, P4, P5, P6, P7, P8, P9, P10, P11, P14, P16, P19) generally preferred visually consistent diagrams and found them easier to work with.

> "*I like the placement of all elements in the same position.*" (P1)
> "*For those diagrams where relative positions remain the same, it's definitely easier to identify [the differences].*" (P7)

P9, who before complained about having to waste time context-switching between model debugging and locating corresponding nodes, said that cognitively, visually consistent diagrams helped with the recollection of locations of corresponding nodes.

> "*It was much quicker for me to do those [tasks with consistency] compared to the ones [without consistency]. ... It's much easier to remember the relations, like the links, when the nodes are relatively in the same place across diagrams.*" (P9)
>
> "*If I have the understanding that Alloy is going to keep the positions fixed, then it would remove the mental overhead of making sure that I am looking at the right node.*" (P11)

Visually consistent diagrams were helpful by making it more natural to mentally "*over-impose the two diagrams*" (P11, P19) on top of each other, streamlining the comparison process.

4.3.4 Visual consistency is non-intrusive to user workflow. During the post-experiment interviews, we asked about whether or not participants realized that there were two variants of diagrams (Default and FC) with varying levels of consistency (FC, PC, and NC). A considerable number of participants (P1, P5, P9, P11, P13, P16, P19) did not notice the difference between the variants, until we specifically revealed that the experiment was on visual consistency.

> "*No I didn't, I didn't realize they are two different visualizations.*" (P1)
>
> "*When I look back at the fact that things moved around, in my memory, I cannot recollect in the other diagrams if they moved around or not.*" (P5)

Only after we revealed the nature of the experiment did these participants offer insights on their perceived benefits of consistency, as outlined in Sec. 4.3.3.

Some participants, due to their prior experiences with Alloy, somewhat expected the locations of the nodes to change (since the default Alloy visualization is non-consistent), so when the locations did not change, they thought it was a coincidence.

> "*I was not expecting the position[s] to be the same, I was not ... really looking for it, because I have a framework in mind where I would like to make sure if the positions are the same and I thought this was a coincidence.*" (P11)

The participants reflected that most of the time they didn't even register when diagram elements shifted – suggesting that visual consistency was non-intrusive and blended well with their usual debugging approach. Visual consistency, then, becomes a "background enabler:" Users establish a mental map and follow their workflow without having to consciously verify that elements remain fixed. This natural integration can reduce cognitive overhead, making it easier to focus on the substantive differences between diagrams.

4.3.5 Benefits of visual consistency can be limited. While most participants found visual consistency helpful in lowering the mental overhead, some participants (P4, P5, P6, P8, P10, P11, P13, P15, P17, P19) reflected that they may have had limited effectiveness in some cases.

> "*For the [ring network] example, if everything still remains unchanged, like the positions, but only the relationships [edges] are changing, it will make the graph maybe harder to understand ... especially when we are passing the data of the nodes ... So I think ... [benefits of consistency] depends on the scenario.*" (P4)

The quote above refers to Task 3 of Model 1, which concerns the ring network model and its operation `PassData` which passes every node's data to its successor. This operation does not create any new entities or relations; it merely

changes the existing edges betwen the nodes and data to point ot a new node from the same data. In this case, P4 finds that visual consistency makes the diagram more confusing.

One hypothesis is that when users expect an operation to dramatically change the locations of the nodes like `PassData` does, the lack of such changes may bring more confusion as the fact that the nodes remained unchanged did not align with their mental image of the operation. If the layout is consistent but the relational data or state of an element changes, users might overlook or misinterpret these modifications. In other words, the very stability that aids recognition in one scenario can become a blind spot when it masks meaningful differences between states.

Sometimes, the enforcement of visual consistency may lead to a "bad layout" when a good layout would necessitate some position changes.

> "*Let's say that [between the pre-state and the post-state], there's some new edge being added. That new edge may be intersecting with some existing nodes. I could see how these intersections ... would make [the diagram] slightly crowded.*" (P14)

In this case, a good layout may necessitate an edge being moved to create more space for the new edge to go through. This suggests that when it comes to visual aesthetics or interpretability (e.g., edges should not intersect with irrelevant nodes), visual consistency may need to come second. This further confirms the need for varying levels of visual consistency, such as "partial consistency" we explored in RQ2 and "soft consistency" as suggested in [32].

4.3.6 Other techniques may also help lower cognitive overhead. Visual consistency is not the end of the story. While visual consistency significantly reduces the mental overhead, some participants suggested that it can be complemented with other visual cues that more effectively highlight changes between states.

> "*Instead of just removing [a node directly], [I would want something like] like soft deleting, like, let's say for deleting a node or adding something, ... we show it in dotted form, like slightly faded form, ... the new change that has been introduced in the [post-state] diagram after the [pre-state] diagram, that will also help a lot.*" (P2)

In other words, positions of nodes are not the whole story – other visual cues like dotted lines, faded objects, node colors, etc. may provide more drastic means to display what has changed in the diagram structurally. This echos the results of Archambault et al. [3], which concluded that "difference maps" (using visual cues like colors to explicitly show changes between two graphs) also resulted in improved understanding.

5 Discussion

Our study set out to investigate how visual consistency in Alloy visualizations affects the efficiency, accuracy, and subjective experience of model understanding and debugging. The findings converge on a central theme: Visual consistency matters, but its value depends on the form it takes and the context in which it is applied.

5.1 The Fine Value of Full Consistency

For RQ1, we observed that **fully consistent visualizations significantly reduced participants' thinking time** (from 51.80 seconds to 39.51 seconds, or a speedup of around 25%) **and improved correctness** (from .89 to .97) **when identifying buggy behavior from diagrams** (RQ1a, Q1). This indicates that consistency lowers the cognitive costs of mapping objects across states, enabling participants to focus more directly on the semantic differences between the pre-state and post-state. Interview responses from RQ3 reinforced this point: Participants described non-consistent

layouts as "confusing" or "distracting," and noted that full consistency acted as an invisible helper that allowed them to focus immediately on meaningful changes rather than wasting effort relocating nodes.

While these benefits were observed in relatively small, controlled tasks, their implications are magnified in real modeling practice. Even modest gains in correctness can prevent downstream errors, which often cost disproportionate time and effort to uncover and fix. Similarly, reductions in cognitive overhead compound across many repeated reasoning steps in large modeling projects. As a result, the combined effect of faster reasoning and fewer mistakes could translate into substantial efficiency gains in practice.

Interestingly, these benefits did not appear to extend to the task of localizing bugs in the Alloy code (RQ1b, Q2). These findings indicate that visualization improvements alone may not necessarily bridge the gap between bug finding and code understanding. At first glance, this lack of evidence for knowledge transfer may seem disappointing, but it is not surprising given the nature of the tasks. Localizing bugs in Alloy code requires more than visual cognition: It requires inference about the intended operation behavior, deeper understanding of Alloy's syntax and semantics, and careful inspection of textual constructs manifested through Alloy code. As such, the connection between the visualization and bug localization in code is inherently weaker than the connection between visualization and bug identification. Moreover, knowledge transfer typically requires substantial practice and reflection. Given that each experiment session lasted less than two hours, it would be unrealistic to expect strong cross-modal transfer to emerge in this limited context. Future visualization efforts should consider focusing on bridging this gap between bug identification and program comprehension. The absence of measurable Q2 effects, nonetheless, should not be interpreted as a failure of consistency, but rather as a reflection of fundamentally different cognitive demands of diagram reading versus code-level debugging.

5.2 The Practical Value of Partial Consistency

For RQ2, our results showed that **partial consistency performs on par with full consistency**: Participants were just as accurate and efficient with partially consistent layouts as with fully consistent ones. In contrast, non-consistent layouts significantly hindered performance, both slowing participants down and reducing correctness. These findings imply that **while inconsistency is harmful, enforcing full consistency is not strictly necessary**.

This has important implications for visualization design. As prior literature (e.g., [32], [43], and Sec. 4.2 of [37]) argued, maintaining full consistency may come at the expense of layout clarity, as fixed positions may lead to diagrams that are unnatural, cluttered, and potentially misleading. The same sentiments were expressed by participants of the post-experiment interview study for RQ3. This is especially true when the complexity (of the underlying models and of the changes introduced to the instances by the operations) increases. Small or localized changes benefit strongly from fixed layouts, whereas large structural changes may exceed the point where preserving positions supports comprehension. Identifying this transition point – how much structural or relational change warrants relaxing full consistency in favor of partial or non-consistency – remains an open research question.

Nonetheless, the fact that partial consistency suffices suggests that **visualization tools should instead prioritize recognizable consistency – anchoring key structures across states while allowing other parts of the layout to adapt for readability**.

5.3 Beyond Consistency

During the post-experiment interview (RQ3), several participants pointed to **other visualization techniques that could further enhance model debugging**. Suggestions included "soft deletion" (using faded / dotted representations for removed elements), highlighting newly added nodes or edges with distinctive visual markers, and using different

colors to emphasize changed relationships between states (e.g., as in Archambault et al. [3]). Such techniques would allow changes to stand out more directly, complementing consistency by reducing the need for manual comparison across states.

These ideas highlight an important future direction: Consistency is not the only lever for improving visualizations. Alternative or complementary cues could provide more explicit scaffolding for reasoning about dynamic behavior. Again in the spirit of Krishnamurthi et al. [30], **future work could systematically evaluate how such cues interact with consistency**: whether they amplify its benefits, or risk introducing distractions when overused.

5.4 Threats to Validity

As with any empirical study, several factors limit the interpretation and generalizability of our results.

5.4.1 Internal validity. One concern is that the **differences in participants' Alloy experience** might have influenced their task performance. All participants had recent exposure to Alloy. But some only used it in a course setting one or two semesters prior to participation, while others worked on extensive research projects with Alloy. Similarly, the **differences in difficulties of the tasks** might have influenced the performances on these tasks. The tasks, for example, represent operations of differing complexities, whose visualizations may have differing number of nodes and edges. We mitigated this threat by **modeling the variability of the participants and task difficulties as random effect intercepts** in our regression models.

Another potential threat lies in the **manual construction of our fully-consistent visualizations**. Although our goal was to preserve the default layout while stabilizing node positions, it is possible that these consistent diagrams also differed in other ways. For example, out of all possible fully-consistent diagrams for an instance, we may have unintentionally constructed a variant that looked the "nicest" based on our subjective opinions. Hence it is possible that the measured benefits of full consistency might not translate to all fully-consistent diagrams. As a mitigation, we used an **explicit set of guidelines to construct the fully-consistent visualizations in a way that mimics Alloy's default behavior** in the placement of new nodes and differing edges (see Sec. 3.4). Indeed, some participants reflected (see Sec. 4.3.5) that even our constructed fully-consistent diagrams were far from ideal in terms of layout quality. Thus, while our manual construction of the fully-consistent layout might have had some marginal effect, the central finding that consistency aids comprehension remains well-supported.

Finally, another potential internal validity concern lies in **our classification of the default Alloy visualizations into Partially-Consistent and Not-Consistent categories**. While we based these classifications on clear visual criteria (e.g., whether salient portions of the layout appeared preserved across states), the process was ultimately performed manually by the researchers and thus introduces some subjectivity. It is possible that a different researcher might have categorized borderline cases differently, leading to small variations in how the data was analyzed. As a mitigation, as outlined in Sec. 3.4, we applied a set of explicit guidelines for identifying partial consistency, and the classifications were cross-checked by multiple authors to reduce individual bias.

5.4.2 External Validity. Our participant pool consisted of students with recent Alloy experience due to the ease of recruitment. While it is representative of learners and novice modelers – an important group for evaluating debugging support – **this population may not fully reflect expert practices**. While some participants had additional formal modeling experience working in research labs with substantial Alloy modeling requirements, or had studied the Alloy language and analyzer in depth, our overall participant perspectives may still differ from those of expert practitioners. Certainly, further studies with professional practitioners would strengthen the generalizability of the results. At the

same time, **the use of a student population is not inherently problematic**. As Mook argued in "*In defense of external invalidity*" [36], the value of a study lies in how well it addresses the research question, not in whether the sample directly mirrors the target population. Our goal was to evaluate the cognitive impact of visualization principles in a controlled setting, for which students are an appropriate and informative sample. The mechanisms by which visual consistency aids or hinders comprehension – such as reducing cognitive overhead when comparing states – are cognitive processes that plausibly generalize beyond the immediate participant pool.

Similarly, the fact that our study only consisted of 19 participants, a small sample size, has hindered us from making the more powerful conclusions that we initially hypothesized. An example is that, because of the variability in writing time of the participants, we could not observe a statistically significant effect of consistency on total time participants took to answer Q1. Nonetheless, this sample still allowed us to make significant conclusions on thinking time and correctness, and the interview did provide valuable insight into how consistency influences the reasoning process.

Furthermore, the **experiment tasks were not fully reflective of the tasks professional practitioners would see in larger-scale industrial models**. Our tasks focused on two specific Alloy models, both of which reason about operations expressed as logical constraints between states. Although these models and tasks capture both simple and more complex structures and bugs, they **do not span the full range of formal models and debugging tasks**. In particular, Alloy supports a wide variety of modeling idioms, from purely static structural constraints to more elaborate infinite-state temporal behaviors [7], and our study does not cover this full spectrum, only modeling two states at once. Moreover, the chosen models involved relatively small numbers of entities and relations, which allowed participants to understand model specifications and complete tasks within a controlled experimental session. In practice, however, formal models can grow considerably in size and complexity, with dozens of signatures, operations, and invariants interacting in subtle ways. Although the benefits of consistency seen in smaller models may propagate into larger benefits, the impact of visual consistency on navigating such large-scale models remains an open question. Similarly, domain-specific models – such as those in avionics, distributed systems, or access control – may introduce specialized structures and semantics that could influence how users perceive consistency. The impacts of consistency on larger or more domain-specific models remain to be studied.

As a mitigation, **we deliberately chose models that reflect common modeling patterns** – such as defining invariants to enforce structural properties and operations to describe state transitions – that generalize beyond the specific tasks studied. Furthermore, our interview findings (see Sec. 4.3.1) suggest that participants' reasoning processes were not necessarily tied to the idiosyncrasies of the chosen models, but rather to the visual stability of the diagrams, lending confidence that our observations will extend, at least in part, to other modeling contexts.

5.4.3 Construct Validity. We measured performance primarily with **task time and correctness**. These metrics capture efficiency and accuracy but do not measure other aspects such as confidence. Our exit interviews (RQ3) provide some hints on these other measures, but interview results -– and in general subjective reflections — may have been influenced by the immediacy of the tasks and the short study duration. Longer-term studies, such as full-fledged studies observing participants use consistent or non-consistent visualizations in real-life modeling tasks, might surface different perspectives on the roles of consistency.

Furthermore, the **lack of evidence for knowledge transfer** from diagram-based reasoning (Q1) to bug localization in code (Q2) should be interpreted with caution. As discussed before, knowledge transfer across modalities typically requires more time and practice than was available in a two-hour experiment session. The absence of transfer effects

here is consistent with those constraints and should not be taken as evidence that such transfer is not present in normal modeling conditions.

5.5 Future Work

While our findings established the value of visual consistency, there are still many open questions that motivate future research.

Other consistency metrics. Our study focused on two levels of consistency – full and partial. We did not examine the more finer-grained distinctions which have been discussed in prior work, such as hard vs. soft consistency or positional vs. relative consistency [32]. These more nuanced notions could affect comprehension differently, especially in scenarios when full consistency may conflict with layout quality and interpretability. Exploring such distinctions represents an important next step.

Experience effect analysis. We note that while all participants had a baseline level of knowledge of Alloy from taking a previous formal methods class, participants had differing levels of experience. While our pre-screening survey logs some information regarding participant experience, we did not extensively log experiences of participants before the experiment. A post-hoc analysis of how experience affected bug-finding efficiency and accuracy is still an open question, and is an avenue for future work.

Beyond visual consistency. Visual consistency is but one of many visual principles that may enhance the usability of formal modeling. Out exit-interview results (RQ3) and prior works such as [3] hinted at other potential ways to assist with difference-finding, such as visual cues like dotted lines, different coloring, etc. Aside from difference-finding, participants of our prior interview study [32] also hinted at other visual principles to make formal modeling visualizations more interpretable, such as the "hierarchy" principle which states that visualizations should intuitively reflect the hierarchical nature of the underlying models. Evaluating these other principles alone, or the interplay between consistency and these other principles, may reveal richer strategies for improving visualization design in formal modeling tools.

Translation to code comprehension. While consistency had a notable effect in discovering bugs within a visualization, we did not find evidences for knowledge transfer from this identification of bugs into comprehending and localizing bugs within the model source code. Future studies should be conducted to more rigorously evaluate this specific translation, and future visualization efforts should focus more on identifying techniques whose effects can transfer into better code comprehension.

Larger-scale study. Our experiment only tested 19 participants, which may limit the generalizability of the findings. At the same time, while our study used two representative models and eight associated tasks, extending this evaluation to more complex, domain-specific, and larger-scale systems under real-life modeling conditions with more participants will help us validate the generalizability of our findings. These considerations highlight the need for a broader research agenda on empirically grounding visualization principles for supporting model comprehension and debugging, spanning both controlled studies and in-the-wild evaluations.

By situating visual consistency within this broader landscape, our study contributes to the growing empirical foundation for human-centered formal methods. Ultimately, the goal is not only to prove the correctness of systems, but also to make the modeling process itself more accessible, efficient, and reliable.

6 Conclusion

This paper presented the first empirical investigation of visual consistency in formal modeling. Through controlled user studies with Alloy visualizations, we evaluated whether consistent visualizations improve the effectiveness and efficiency of model debugging tasks. Our findings show that consistent visualizations significantly enhance both the speed and correctness of bug identification tasks. These improvements are not only statistically significant but also practically meaningful: In real-world modeling workflows, where mistakes often cascade into wasted effort, even modest gains in correctness and efficiency can compound into substantial time savings.

The study also highlights a nuanced role for consistency. While fully consistent visualizations were generally preferred and provided measurable benefits, participants also recognized the practical value of partial consistency, particularly when achieving full consistency is infeasible or unnecessary. In addition, participants suggested other visualization features that could further support comprehension, offering promising directions for future research.

Visualizations are an important part of formal modeling, with some even exclaiming,

> "*Without visualizations, Alloy is useless and I would have gone with a different tool.*" (P6 of [32])

Improved visualizations can increase the adoption of formal modeling, which may lead to more reliable systems. This paper advances this goal by understanding the roles of consistency in creating more effective visualizations.

References

[1] Daniel Archambault, Tamara Munzner, and David Auber. 2011. Tugging Graphs Faster: Efficiently Modifying Path-Preserving Hierarchies for Browsing Paths. *IEEE Transactions on Visualization and Computer Graphics* 17, 3 (2011), 276–289. doi:10.1109/TVCG.2010.60

[2] Daniel Archambault and Helen C. Purchase. 2013. Mental Map Preservation Helps User Orientation in Dynamic Graphs. In *Graph Drawing*, Walter Didimo and Maurizio Patrignani (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 475–486.

[3] Daniel Archambault, Helen C Purchase, and Bruno Pinaud. 2010. Difference map readability for dynamic graphs. In *International Symposium on Graph Drawing*. Springer, 50–61. doi:10.5555/1964371.1964377

[4] Benjamin M Bolker. 2015. Linear and generalized linear mixed models. *Ecological statistics: contemporary theory and application* 2015 (2015), 309–333.

[5] Stina S. Bridgeman and Roberto Tamassia. 2000. A User Study in Similarity Measures for Graph Drawing. In *Proceedings of the 8th International Symposium on Graph Drawing (GD '00)*. Springer-Verlag, Berlin, Heidelberg, 19–30.

[6] Lawrence D Brown, T Tony Cai, and Anirban DasGupta. 2001. Interval estimation for a binomial proportion. *Statistical science* 16, 2 (2001), 101–133.

[7] Julien Brunel, David Chemouil, Alcino Cunha, and Nuno Macedo. 2018. The Electrum Analyzer: Model Checking Relational First-Order Temporal Specifications. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 884–887. doi:10.1145/3238147.3240475

[8] Shardul Chiplunkar and Clément Pit-Claudel. 2023. Diagrammatic notations for interactive theorem proving. In *The International Workshop on Human Aspects of Types and Reasoning Assistants. doi*, Vol. 10. doi:10.5075/epfl-SYSTEMF-305144

[9] Thomas D Cook, Donald Thomas Campbell, and William Shadish. 2002. *Experimental and quasi-experimental designs for generalized causal inference*. Vol. 1195. Houghton Mifflin Boston, MA.

[10] Rui Couto, José C Campos, Nuno Macedo, and Alcino Cunha. 2018. Improving the visualization of Alloy instances. *arXiv preprint arXiv:1811.10817* (2018).

[11] Alcino Cunha, Nuno Macedo, and Eunsuk Kang. 2023. Task Model Design and Analysis with Alloy. In *Rigorous State-Based Methods*, Uwe Glässer, Jose Creissac Campos, Dominique Méry, and Philippe Palanque (Eds.). Springer Nature Switzerland, Cham, 303–320. doi:10.1007/978-3-031-33163-3_23

[12] Natasha Danas, Tim Nelson, Lane Harrison, Shriram Krishnamurthi, and Daniel J Dougherty. 2017. User studies of principled model finder output. In *International Conference on Software Engineering and Formal Methods*. Springer, 168–184. doi:10.1007/978-3-319-66197-1_11

[13] N. Dulac, T. Viguier, N. Leveson, and M.-A. Storey. 2002. On the use of visualization in formal requirements specification. In *Proceedings IEEE Joint International Conference on Requirements Engineering*. 71–80. doi:10.1109/ICRE.2002.1048507

[14] Peter K Dunn and Gordon K Smyth. 1996. Randomized quantile residuals. *Journal of Computational and graphical statistics* 5, 3 (1996), 236–244.

[15] Tim Dwyer, Kim Marriott, and Michael Wybrow. 2009. *Topology Preserving Constrained Graph Layout*. Springer-Verlag, Berlin, Heidelberg, 230–241. https://doi.org/10.1007/978-3-642-00219-9_22

[16] Tristan Dyer and John Baugh. 2021. Sterling: A Web-Based Visualizer for Relational Modeling Languages. In *Rigorous State-Based Methods*, Alexander Raschke and Dominique Méry (Eds.). Springer International Publishing, Cham, 99–104. doi:10.1007/978-3-030-77543-8_7

[17] Davide Falessi, Muhammad Ali Babar, Giovanni Cantone, and Philippe Kruchten. 2010. Applying empirical software engineering to software architecture: challenges and lessons learned. *Empirical Software Engineering* 15, 3 (2010), 250–276. doi:10.1007/s10664-009-9121-0

[18] Andrzej Gałecki and Tomasz Burzykowski. 2013. *Linear Mixed-Effects Model.* Springer New York, New York, NY, 245–273. doi:10.1007/978-1-4614-3900-4_13

[19] Loïc Gammaitoni and Pierre Kelsen. 2014. Domain-Specific Visualization of Alloy Instances. In *Abstract State Machines, Alloy, B, TLA, VDM, and Z*, Yamine Ait Ameur and Klaus-Dieter Schewe (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 324–327. doi:10.1007/978-3-662-43652-3_33

[20] Robert L Glass. 2002. *Facts and fallacies of software engineering.* Addison-Wesley Professional.

[21] Ben Greenman, Sam Saarinen, Tim Nelson, and Shriram Krishnamurthi. 2022. Little Tricky Logic: Misconceptions in the Understanding of LTL. *The Art, Science, and Engineering of Programming* 7, 2 (Oct. 2022). doi:10.22152/programming-journal.org/2023/7/7

[22] Florian Hartig. 2016. DHARMa: residual diagnostics for hierarchical (multi-level/mixed) regression models. *CRAN: Contributed Packages* (2016).

[23] Christopher Healey and James Enns. 2011. Attention and visual memory in visualization and computer graphics. *IEEE transactions on visualization and computer graphics* 18, 7 (2011), 1170–1188. doi:10.1109/TVCG.2011.127

[24] Run Huang, Anna Katherine Zhao, Zeinabsadat Saghi, Sadra Sabouri, and Souti Chattopadhyay. 2025. Beyond the Page: Enriching Academic Paper Reading with Social Media Discussions. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*. 1–25.

[25] Daniel Jackson. 2002. Alloy: a lightweight object modelling notation. *ACM Transactions on software engineering and methodology (TOSEM)* 11, 2 (2002), 256–290. doi:10.1145/505145.505149

[26] Daniel Jackson. 2016. *Software Abstractions, revised edition: Logic, Language, and Analysis.* MIT Press.

[27] Daniel Jackson. 2019. Alloy: a language and tool for exploring software designs. *Commun. ACM* 62, 9 (2019), 66–76. doi:10.1145/3338843

[28] Maurits Kaptein. 2016. *Using Generalized Linear (Mixed) Models in HCI.* Springer International Publishing, Cham, 251–274. doi:10.1007/978-3-319-26633-6_11

[29] Gerwin Klein, June Andronick, Matthew Fernandez, Ihor Kuz, Toby Murray, and Gernot Heiser. 2018. Formally verified software in the real world. *Commun. ACM* 61, 10 (Sept. 2018), 68–77. doi:10.1145/3230627

[30] Shriram Krishnamurthi and Tim Nelson. 2019. The Human in Formal Methods – *The Next 30 Years*, Maurice H. ter Beek, Annabelle McIver, and José N. Oliveira (Eds.). Springer International Publishing, Cham, 3–10. doi:10.1007/978-3-030-30942-8_1

[31] Jill H. Larkin and Herbert A. Simon. 1987. Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science* 11, 1 (1987), 65–100. doi:10.1016/S0364-0213(87)80026-5

[32] Yiliang Liang, Avinash Palliyil, Eunsuk Kang, and Joshua Sunshine. 2025. Towards Better Formal Methods Visualizations. *PLATEAU Workshop 2025* (8 2025). doi:10.1184/R1/29086949.v1

[33] Dor Ma'Ayan and Shahar Maoz. 2023. Using reactive synthesis: An end-to-end exploratory case study. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 742–754. doi:10.1109/ICSE48619.2023.00071

[34] Niloofar Mansoor, Hamid Bagheri, Eunsuk Kang, and Bonita Sharif. 2023. An empirical study assessing software modeling in alloy. In *2023 IEEE/ACM 11th International Conference on Formal Methods in Software Engineering (FormaliSE)*. doi:10.1109/FormaliSE58978.2023.00013

[35] Shahar Maoz and Jan Oliver Ringert. 2021. Spectra: a specification language for reactive systems. *Software and Systems Modeling* 20, 5 (2021), 1553–1586. doi:10.1007/s10270-021-00868-z

[36] Douglas G. Mook. 1983. In Defense of External Invalidity. *American Psychologist* 38, 4 (1983), 379–387. doi:10.1037/0003-066X.38.4.379

[37] Tim Nelson, Ben Greenman, Siddhartha Prasad, Tristan Dyer, Ethan Bove, Qianfan Chen, Charles Cutting, Thomas Del Vecchio, Sidney LeVine, Julianne Rudner, Ben Ryjikov, Alexander Varga, Andrew Wagner, Luke West, and Shriram Krishnamurthi. 2024. Forge: A Tool and Language for Teaching Formal Methods. *Proc. ACM Program. Lang.* 8, OOPSLA1, Article 116 (April 2024), 29 pages. doi:10.1145/3649833

[38] Chris Newcombe. 2014. Why Amazon Chose TLA+. In *International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z*. Springer, 25–39. doi:10.1007/978-3-662-43652-3_3

[39] Gerard O'Regan. 2023. *Overview of Formal Methods.* Springer Nature Switzerland, Cham, 255–276. doi:10.1007/978-3-031-26212-8_16

[40] Stephen E Palmer. 1992. Common region: A new principle of perceptual grouping. *Cognitive psychology* 24, 3 (1992), 436–447. doi:10.1016/0010-0285(92)90014-S

[41] José C Pinheiro and Douglas M Bates. 2000. *Mixed-effects models in S and S-PLUS.* Springer.

[42] Siddhartha Prasad, Ben Greenman, Tim Nelson, and Shriram Krishnamurthi. 2024. Grounded Language Design for Lightweight Diagramming for Formal Methods. arXiv:2412.03310 [cs.CL] https://arxiv.org/abs/2412.03310

[43] Zening Qu and Jessica Hullman. 2018. Keeping Multiple Views Consistent: Constraints, Validations, and Exceptions in Visualization Authoring . *IEEE Transactions on Visualization & Computer Graphics* 24, 01 (Jan. 2018), 468–477. doi:10.1109/TVCG.2017.2744198

[44] Ronald A Rensink. 2002. Change detection. *Annual review of psychology* 53, 1 (2002), 245–277. doi:10.1146/annurev.psych.53.100901.135125

[45] Robert Rosenthal and Ralph L Rosnow. 2008. *Essentials of behavioral research: Methods and data analysis.* McGraw-Hill.

[46] Ilia Shevrin and Shahar Maoz. 2025. What Properties Affect Boolean Formula Comprehension in Formal Specifications? *ACM Transactions on Software Engineering and Methodology* (2025). doi:10.1145/3744557

[47] Daniel J Simons and Daniel T Levin. 1997. Change blindness. *Trends in cognitive sciences* 1, 7 (1997), 261–267. doi:10.1016/S1364-6613(97)01080-2

[48] D. Todorovic. 2008. Gestalt principles. *Scholarpedia* 3, 12 (2008), 5345. doi:10.4249/scholarpedia.5345 revision #91314.

[49] Edward R. Tufte. 1983. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, Connecticut.

[50] Benjamin Tyler. 2023. *Formal Methods Adoption in Industry: An Experience Report.* Springer International Publishing, Cham, 152–161. doi:10.1007/978-3-031-43678-9_5

[51] Colin Ware and Robert Bobrow. 2004. Motion to support rapid interactive queries on node–link diagrams. *ACM Transactions on Applied Perception (TAP)* 1, 1 (2004), 3–18. doi:10.1145/1008722.1008724
[52] Michelle Werth and Michael Leuschel. 2020. VisB: A Lightweight Tool to Visualize Formal Models with SVG Graphics. In *Rigorous State-Based Methods*, Alexander Raschke, Dominique Méry, and Frank Houdek (Eds.). Springer International Publishing, Cham, 260–265. doi:10.1007/978-3-030-48077-6_21
[53] Samuel S Wilks. 1938. The large-sample distribution of the likelihood ratio for testing composite hypotheses. *The annals of mathematical statistics* 9, 1 (1938), 60–62.
[54] Pamela Zave. 2012. Using lightweight modeling to understand chord. *SIGCOMM Comput. Commun. Rev.* 42, 2 (March 2012), 49–57. doi:10.1145/2185376.2185383
[55] Pamela Zave. 2015. A practical comparison of Alloy and Spin. *Formal Aspects of Computing* 27, 2 (2015), 239–253. doi:10.1007/s00165-014-0302-2
[56] Pamela Zave. 2017. Reasoning About Identifier Spaces: How to Make Chord Correct. *IEEE Transactions on Software Engineering* 43, 12 (2017), 1144–1156. doi:10.1109/TSE.2017.2655056

A Descriptions of the Models, Bugs, and Tasks

The experiment involves two Alloy models, the ring network model and the social network model. This section describes the models, the operations associated with the models, and the bugs associated with the operations. Throughout the experiments, participants saw a version of the descriptions of the models and the operations, but did not see the descriptions of the bugs.

A.1 Ring Network Model

We define a "ring network" as a collection of nodes arranged into a single ring, where each node can be associated with one or more pieces of data. A valid "ring network" satisfies the following invariants:

- Dangling data are disallowed: Each piece of data in a network state must be associated with at least one node.
- All nodes in the network state must be arranged into one and only one ring.

The descriptions of the tasks, operations, the bugs, and the associated sub-classifications (PC or NC) of the Default visualizations are presented in Table 5.

Operations of Model 1 ("ring network model")

Task	Operation	Desired Behavior	Buggy Behavior	Default Visualization Behavior
1	AddNodeAfter(n_0, n_1)	Add a new node n_1 as the successor of an exising node n_0 in the network.	The new node n_1 is inserted in a way that breaks the ring structure.	Not-Consistent
2	RemoveNode(n)	Remove an existing node n from the network, deleting every data d associated with n if d is only associated with n.	Any data d associated with n is removed even if it is associated with a different node in the network.	Not-Consistent
3	PassData	Pass each node's data to its successor node. Formally, for each node n and its successor node n', the data associated with n' in the post-operation state should equal the data associated with n in the pre-operation state.	Each node's data is passed to its predecessor instead of successor.	Not-Consistent
4	MergeWithSuccessor(n)	Remove the successor node n' of n, and merge all data associated with n' into n.	Node n loses its original data before the operation.	Partially-Consistent

Table 5. The tasks for Model 1 ("ring network model"). Three of the tasks have default visualizations that fall into the NC (Not-Consistent) category; one falls into the PC (Partially-Consistent) category.

A.2 Social Network Model

The "social network" model simulates a simple social network consisting of users, photos, comments, and relations such as friendship, content-ownership, tagging, commenting, etc. Specifically,

- A **social network** consists of two entities: **users** and **contents**, which can further be splited into **photos** and **comments**.
- Users can be **friends** with other users.
- Users can **own** contents.
- Contents can have **tags** to users.
- A user can place a **comment** under a piece of content. Since comments themselves are contents, nested comments are allowed.

The invariants of the model are:

- **Friendship invariants** –
  - Friendship is symmetric – if u_1 is a friend of u_2 then u_2 is a friend of u_1.
  - No self-friendship – u cannot be a friend of u.
- **Content invariant** – Every content (photo or comment) is owned by exactly one user.
- **Tagging invariant** – If a content c owned by user u_1 contains a tag to user u_2, then u_1 and u_2 must be friends.
- **Commenting invariants** –
  - Every comment must be under exactly one content.
  - No comment-loops are allowed. That is, two comments cannot be comments under each other.
  - If a content c_1 has a comment c_2 under it, then the owner of c_1 must be either the same as, or a friend of, the owner of c_2. In other words, only the owner or owner's friends can comment on a content.

The descriptions of the tasks, operations, the bugs, and the associated sub-classifications (PC or NC) of the Default visualizations are presented in Table 6.

Operations of Model 2 ("social network model")

Task	Operation	Desired Behavior	Buggy Behavior	Default Visualization Behavior
1	AddFriend(u_0, u_1)	Add the friendship between u_0 and u_1 and vice versa.	Only add the friendship relation from u_0 to u_1 but not vice versa.	Partially-Consistent
2	RemoveUser(u)	Remove an existing user u from the social network, recursively removing each of u's associated content c and comments under those c, etc.	While u and c are removed, comments under c are not removed, leaving a dangling comment not associated with any existing content.	Partially-Consistent
3	AddTag(c, u)	Add a tag from content c to user u, only if u and the owner of c are friends.	A tag from c to u is added without checking if u and the owner of c are friends.	Partially-Consistent
4	RemoveFriends(u_0, u_1)	Remove the friendship between u_0 and u_1, also removing existing relations that are no longer allowed because u_0 and u_1 are no longer friends.	The comments left by u_0 under contents made by u_1 are not removed.	Not-Consistent

Table 6. The tasks for Model 2 ("social network model"). One of the tasks has default visualizations that fall into the NC (Not-Consistent) category; three fall into the PC (Partially-Consistent) category.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009
Manuscript submitted to ACM