



# “Too Theoretical and Nowhere Near Interesting”: Using a Tool to Increase Student Motivation for Formal Methods

Katherine Braught

braught2@illinois.edu

University of Illinois Urbana-Champaign  
Urbana, IL, USA

Katherine Driggs-Campbell

krdc@illinois.edu

University of Illinois Urbana-Champaign  
Urbana, IL, USA

Yangge Li

li213@illinois.edu

University of Illinois Urbana-Champaign  
Urbana, IL, USA

Sayan Mitra

mitras@illinois.edu

University of Illinois Urbana-Champaign  
Urbana, IL, USA

## Abstract

Using formal methods to evaluate software and hardware enhances system reliability, which is crucial for safety-critical applications such as airplanes and autonomous vehicles. Formal methods are mathematical modeling techniques that can be used to verify the safety of systems. The use of formal methods is limited in industry due to a shortage of trained engineers. Educators in formal methods often report that many students do not see the benefit of formal methods and perceive the involved math as not worth the effort for their future careers as software engineers. This study aims to understand the current state of student beliefs and how using a formal verification tool affects student motivation to learn about formal methods. We used an Expectancy Value Cost Lite survey to measure student motivation. Students completed this survey multiple times while designing algorithms to control vehicles in different scenarios, both with and without a formal verification tool. We found that students in an autonomy class are motivated to use formal methods. Although the findings are not statistically significant, we observed a slight increase in motivation after using the tool. Additionally, using a formal verification tool solely for modeling may contribute to increased motivation. These results suggest that incorporating tools into coursework may be a useful step in motivating more students to study formal methods and enter the workforce with these skills.

## CCS Concepts

• **Social and professional topics** → **Software engineering education**; • **Theory of computation** → *Verification by model checking*.

Authors' Contact Information: Katherine Braught, braught2@illinois.edu, University of Illinois Urbana-Champaign, Urbana, IL, USA; Yangge Li, li213@illinois.edu, University of Illinois Urbana-Champaign, Urbana, IL, USA; Katherine Driggs-Campbell, krdc@illinois.edu, University of Illinois Urbana-Champaign, Urbana, IL, USA; Sayan Mitra, mitras@illinois.edu, University of Illinois Urbana-Champaign, Urbana, IL, USA.



This work is licensed under a Creative Commons Attribution 4.0 International License.  
© 2025 Copyright held by the owner/author(s).  
ACM 2831-3194/2025/8-ART111  
<https://doi.org/10.1145/3724363.3729116>

## Keywords

Computer Science, Formal Methods, Motivation, Formal Verification

### ACM Reference Format:

Katherine Braught, Yangge Li, Katherine Driggs-Campbell, and Sayan Mitra. 2025. “Too Theoretical and Nowhere Near Interesting”: Using a Tool to Increase Student Motivation for Formal Methods. *ACM/IMS J. Data Sci.* 37, 4, Article 111 (August 2025), 7 pages. <https://doi.org/10.1145/3724363.3729116>

## 1 Introduction

Formal methods are a set of mathematical modeling techniques that can be used to model the behavior of computer systems and verify that the systems satisfy safety, functional, and design properties [23]. Their use is important for ensuring software safety and quality. Despite past recommendations by the Association for Computer Machinery (ACM), the Institute of Electronic and Electrical Engineering (IEEE) and The British Computer Society (BCS) that computer science programs should incorporate formal methods [23], the lack of engineers with enough training to use formal methods is a limiting factor for their adoption in industry [6, 11].

Formal methods are not extensively taught as part of the core computer science curriculum, and students generally lack interest in the elective courses. Students find formal methods challenging because advanced math and logic are often required, and the subject is rarely taught in a beginner-friendly manner [6]. They also tend to think that formal methods will not be useful in their career as software engineers [1]. A student in our study even stated that formal methods are “too theoretical and nowhere near interesting.” Increasing student motivation toward formal methods is a crucial first step in promoting their broader adoption in industry.

Our study tried to understand the current state of students' beliefs about formal methods. It also tests the use of tools to alleviate students' concerns. Students' lack of motivation to study formal methods can be explained through Expectancy Value Theory [10], which breaks student motivation into three key questions: “Can I do the task?” (expectancy), “Do I want to do the task?” (value), and “How much time does it require?” (cost). We hypothesize that tools can reduce the perceived cost of learning formal methods and enhance their value to students, as tools are often proposed to help students see the benefits of formal methods [22]. In this study, we will evaluate the effectiveness of a homework assignment that uses a tool called Verse to increase student motivation. Verse [16] is a

tool that allows engineers to prove that their vehicle controllers are safe, which means that the algorithm that controls the vehicle will not cause it to do something unsafe. The tool is a Python library that does not require students to have advanced mathematical modeling experience.

In our experiment, we provide students with two vehicle scenarios for which they will design a safe controller. Students model their scenario within Verse. Students will solve one problem only testing their controller, and the other problem using the formal verification features of Verse. Between solving the problems, students will take the Expectancy Value Cost (EVC) Lite Survey [22]. We seek to answer the following research questions:

**RQ1:** To what extent are students in an upper-level engineering course motivated to use formal methods?

**RQ2:** How does using a formal methods tool change each component (expectancy, value, and cost) of students' motivation to study formal methods?

**RQ3:** To what extent are students motivated by using a formal verification tool compared to completing the assignment without a formal verification tool?

Our study finds that students who study autonomy have some motivation for studying formal methods. The mean EVC Index was 0.7 out of 1. This result differs from educators' experience of low motivation [26]; however, our study was performed with students in a safe autonomy class who are more likely to be motivated. General computer science students are likely less motivated. We also find that giving students assignments that use formal verification can increase their motivation. While we were unable to find significant results, this study is a step toward changing students' minds about the importance of learning formal methods.

## 2 Background

### 2.1 Formal Methods

Formal methods are mathematical modeling techniques that are used to model the behavior of systems and verify that the system satisfies the desired requirements [23]. These techniques span from light-weight static analysis to interactive theorem proving with the shared goal of addressing the "need for precision and rigor in modeling and analyzing computer systems and software" [24]. Formal methods are useful for ensuring software quality, reducing costs, and fully understanding the behavior of systems [6, 13, 17]. Learning formal methods is also beneficial for students because it teaches abstraction and supports algorithmic problem solving [17, 24].

Formal methods have achieved widespread acceptance in the context of safety-critical systems because these settings require high standards of safety [9, 11, 24]. However, formal methods apply to many other areas of computer science. At least 17 of ACM's computer science knowledge areas would benefit from connections to formal methods such as algorithmic foundations, architecture, artificial intelligence, software engineering, and cybersecurity [24]. As society grows more reliant on computing, formal methods are more relevant than ever in these application areas [11, 17]. Educating students about the practical applications of formal methods is an effective way to increase their use in industry [17].

Despite the importance of formal methods and their application to many areas in computer science, the curriculum rarely extensively covers formal methods [6]. Many experts believe that formal methods are not given enough attention at the university level, and a lack of training in formal methods by engineers limits their use in industry [11]. We see several key challenges that may contribute to the lack of education in formal methods.

Skepticism by students is one of the biggest challenges when teaching formal methods [26]. Many studies find that students are concerned about the usefulness of formal methods and struggle to see their practical applications [1, 5, 12, 18, 19, 25]. This perception may be caused by formal methods not being introduced in early classes [13].

Students tend to perceive formal methods as difficult because they require advanced mathematics and lack beginner-friendly tools [1, 6]. Formal reasoning is often not introduced until later classes and only in pure logic courses [13]. Formal methods can be taught in beginner-friendly ways. Jaume and Laurent proposed a curriculum for teaching formal methods alongside discrete math [13]. Additionally, running a tool and seeing concrete results may make formal methods more accessible [6], and using a general language like Python can further simplify the material [26].

Although experts believe that tools may improve students' perception of formal methods, to our knowledge, few studies formally survey students about their perceptions and often evaluate tools after the courses end. This study seeks to understand the state of students' current motivation and whether a tool, written in Python and intended to be easy to use, can increase their motivation.

### 2.2 Student Motivation

Motivation, specifically how much value someone finds in an activity, is strongly linked to the action of approaching or avoiding the activity [10]. Therefore, this study tries to increase motivation to encourage students to study formal methods. One of the many cognitive theories of motivation, Expectancy Value Cost (EVC) theory [2, 15] explains that motivation is made up of three factors: expectancy, "Can I do this task?", value, "Do I want to do this task", and cost, "How much effort and time does this task require?". These three factors impact students' motivation or the amount of effort they put into an activity.

We chose EVC Theory to model student motivation because students' concerns with formal methods mirror the three components of the theory. Instructors describe students as perceiving formal methods with low expectancy and value and high cost. We will use the EVC Lite Survey [22] to evaluate student motivation because we ask the participants to take the survey multiple times. EVC Lite has fewer questions than earlier EVC surveys while still being able to assess motivation. The survey was also developed for students to take several times during a course, and the participants will take the survey multiple times.

### 2.3 Increasing Interest in Formal Methods

Several methods have been proposed to increase student interest in formal methods. cs4fn (Computer Science for Fun) is a program that finds unique ways to excite school-aged children about computer science and advanced computing topics like formal methods [8].

cs4n uses magic tricks to introduce children to the theory of formal methods. In contrast, our goal is to show students the value and applications of formal methods.

Other methods have been proposed to improve the teaching of formal methods at the university level. FormalZ [19] allows students to practice writing formal specifications in a deeply gamified environment with leaderboards and badges. Students in a Software Testing and Verification course at Utrecht University were on average positive or neutral about enjoying the game. Puzzles have also been proposed as a way to teach the uses and limitations of modeling checking [21], and a visualization tool called KalkulierbaR [14] has been proposed to increase student engagement with proof calculus as a serious game.

Another approach to increasing student motivation in formal methods is to avoid isolating it from practical applications and introductory programming. Blanco et al. developed an entire programming course centered around a formal methods approach [4]. They found that students often wrote well-structured programs in later courses, but the average student became too mechanical in their reasoning about programs.

Maude [7], a rewriting logic tool, is proposed as an effective tool for teaching formal methods because it allows students to use a familiar functional programming style of modeling [27]. The course using this tool was not evaluated outside of standard university-wide evaluations; however, most students rated the course as “Very good” and called Maude exercises “entertaining”. Verse, the tool we use in this study, has similar benefits to Maude. Additionally, students do not need to learn a new modeling language as Verse is a Python library, and our study directly evaluates student perceptions of the tool.

### 3 Verse - A Tool for Formal Verification

Verse [16] is a tool that was built to be a low-cost solution for formal verification. Some formal verification tools require learning a new modeling language or understanding advanced logic. Verse is written as a Python library and was created for engineers without formal methods experience. The tool is used to model vehicle scenarios such as cars in traffic or drones flying around each other.

Formal verification is a formal method that mathematically ensures that system designs precisely meet a functional requirement [3]. They require (1) a correct model of the system, (2) a description of the environment the system operates in, and (3) a property the system must fulfill or a requirement. This property is often a safety requirement. Without formal methods, checking that the requirement is met can be done with testing for violations; however, testing offers no guarantee because cases may be missed. One formal verification method is called reachability analysis, where every possible state of the system and environment is checked for requirement violations.

Controllers in Verse are Python functions that implement the algorithm that decides when a vehicle should brake, accelerate, turn, etc. In formal verification, the controller is the model of the system intended to meet the specified requirements. In Verse, a user only needs to write a Python function that takes in the current state of the vehicle and makes a decision for the next move. Requirements are specified in Python using the `assert` keyword within the controller

Degree Program	Count
BS Electrical and Computer Engineering	28
BS Computer Science	6
BS Physics	2
BS Aerospace Engineering	1
Graduate Students	20

**Table 1: Degree Makeup of Participants**

source code. The user supplies the `assert` statement with a boolean expression. The expression evaluates whether the property the user is verifying is true given the state of the vehicle.

Scenarios in Verse are used to define the environment in which the vehicle is operating. For example, a scenario for a vehicle in traffic would include the controller for the vehicle, controllers for all other vehicles, a map of the road, and a set of starting states for all the vehicles. To define a scenario in Verse, users import controllers and maps and define starting states for each controller.

Verse’s most powerful feature is its reachability function. The reachability function can prove that the vehicle will never violate the requirement or find a counterexample. Reachability provides more guarantees than testing because it shows that every starting state cannot eventually violate the property. In addition to reachability, Verse can also simulate the scenario using the controller and test the controller from a finite set of starting states.

## 4 Experiment

### 4.1 Setting

Participants in this study were students in an upper-level engineering elective course, Principles of Safe Autonomy. The course teaches algorithms for autonomous systems and uses software tools and a full-sized car to test student implementations of the algorithms. The first assignment in Principles of Safe Autonomy is a short answer and coding assignment where students must design two controllers for two different scenarios using Verse. Verse is the first tool students worked with. The students had lectures on safety and perception before the assignment was due.

Our study added four additional surveys to the assignment to assess student motivation to use formal methods throughout the assignment. 57 students consented to the study and completed all surveys. 28 participants were undergraduates in Electrical and Computer and Engineering. The remaining majors are shown in Table 1. Most students identified as having extensive experience with software engineering. In contrast, many students said they had no experience with formal methods. Table 2 shows that over 85% of students have no experience in formal methods or only learned about them as a part of a course with other topics.

The instructor for this course is not involved in the research team, and students were assured that their grades would not be impacted by their survey responses. Members of the research team attended class to give a demonstration of Verse. One teaching assistant for Principles of Safe Autonomy was on the research team but did not have access to individual survey responses.

Experience	Count
None	30
Learned about formal methods in a non-FM course	23
Taken a formal methods class	2
Used formal methods for a project	2

**Table 2: Formal Methods Experience Level for Participants**

## 4.2 Assignment

The assignment included two design problems where students are provided a scenario, safety requirements, and an unsafe controller written in Verse. Students were required to design a safe controller that meets the requirements for each scenario. For example, the first scenario is a car driving down a road while a pedestrian crosses the street. The safety requirement is that the vehicle should never hit the pedestrian. Students get points for making their vehicle controller safe with larger starting states. Students also earn points by making their vehicles move as fast as possible while maintaining safety.

The second scenario required students to design a controller for a vehicle approaching a stoplight. Students earned points for getting through the stoplight quickly without running a red light.

Students used testing or verification features of Verse to help check if their controller is safe. Students could also simulate their controller in the scenario. Using these features, students could debug their design and check if it worked. Finally, students were asked to show evidence that their design was safe.

## 4.3 Treatment Groups

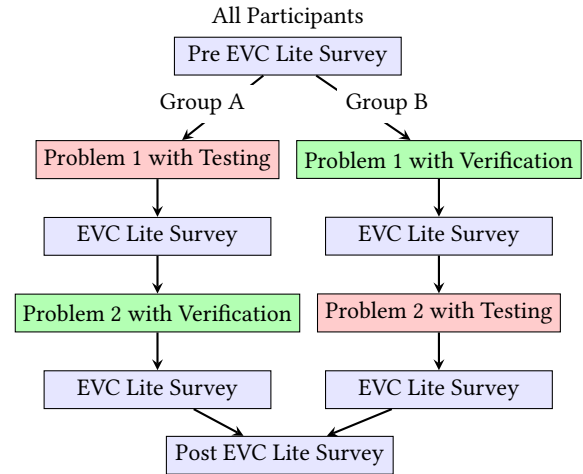
Students were randomly assigned to the A and B groups, with 30 participants in Group A and 27 in Group B. To solve the design problems described in Section 4.2, groups were given only a subsection of the functionality of Verse. Group A could use only testing features of Verse to check if their first design would result in unsafety. For their second design problem, they could also use the formal verification method within Verse, reachability analysis. Group B completed the same design problems, but using verification for the first problem and testing for the second problem. The flow of the assignment and surveys of both groups is shown in Figure 1.

Because students completed two design problems of approximately the same difficulty with each condition, we considered this study fair for students completing the assignment for a grade. This study was approved by the University of Illinois’s IRB.

## 5 Method

### 5.1 Data Sources

Survey data was collected multiple times throughout the assignment. Participants were surveyed before the assignment, after completing their first design problem, after completing their second design problem, and after completing the assignment, where they were asked to reflect on their overall experience. In each survey, participants were asked questions from the Expectancy Value Cost Lite Survey [22] on a 5-point Likert Scale (Strongly Disagree to Strongly Agree). The survey was modified to ask about formal verification.

**Figure 1: Order of Assignment and Surveys for Treatment Group A and Group B**

**Q1:** I can learn how to use formal verification and have success using it in design.

**Q2:** I can understand and learn formal verification.

**Q3:** Formal verification is useful during program/code design.

**Q4:** I understand the importance of formal methods during program/code design.

**Q5:** Formal methods require a lot of time and I have to give up doing other activities.

**Q6:** I don't have time to use formal methods.

**Open Ended:** Is there anything else you would like to share about your experiences with formal methods?

For this study, we focus mainly on the motivation questions. The surveys contained other questions about the usability of the tool, demographics, and open-ended questions about their experience.

### 5.2 Data Analysis

Using the survey items from Section 5.1, we computed the Expectancy Index (EI), Value Index (VI), Cost Index (CI), and Expectancy Value Cost Index (EVC Index) as follows, as described in by Schoeffel et al. [22]:

$$EI = \frac{Q1 + Q2 - 2}{8} \quad (1)$$

$$VI = \frac{Q3 + Q4 - 2}{8} \quad (2)$$

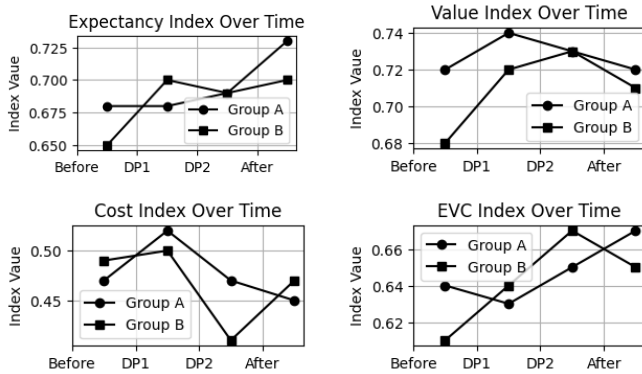
$$CI = \frac{Q5 + Q6 - 2}{8} \quad (3)$$

$$EVCIndex = \frac{EI + VI - CI + 1}{3} \quad (4)$$

These equations convert student Likert responses for each category of EVC to an index between 0 and 1. When calculating the EVC Index, we subtract CI because a lower CI is associated with higher motivation. Therefore, high values of EI, VI, and EVC Index correspond with high student motivation, while a high CI corresponds to low student motivation.

For each group, we use a Wilcoxon Signed-Rank test for paired data [20] to evaluate the statistical significance of differences between: (1) EVC Indices before and after the whole assignment (both problems), (2) EVC Indices before and after the test problem, (3) EVC Indices before and after the verification problem, and (4) the difference in EVC indices before and after testing versus verification.

## 6 Results



**Figure 2: Expectancy Value Cost Indices throughout the assignment. Each point represents the mean index value for each group before the assignment, after the first and second problems, and after the assignment. Group A used verification in design problem (DP) 2 and Group B used it in DP1.**

Category	Before DPs	After DPs	Change	p-value
Expectancy Index	0.669	0.717	0.048	0.116
Value Index	0.697	0.713	0.015	0.459
Cost Index	0.478	0.458	-0.02	0.712
EVC Index	0.629	0.657	0.028	0.167

**Table 3: Student motivation before and after completing the design problems**

As shown in Table 3, before completing the assignment, students had an average EVC Index of 0.629. They had an average EI of 0.669, VI of 0.697, and CI of 0.478. Figure 2 shows that after the assignment, EI, VI, and EVC Index increased or stayed the same, and the CI decreased. As shown in Table 3, these changes are not significant. However, between the two design problems, VI decreased for both groups. For group A, the EVC Index decreased after the first problem, which they solved without formal verification. While for group B, the EVC Index decreased after the final reflection survey. The CI increased for both groups after solving the first problem and decreased when solving the second. Group B had an increase in CI in the final reflection survey.

Table 4 compares the change in motivation for both groups after completing a design problem with and without formal verification.

Category	Group	Test Change	FV Change	Difference	p-value
Expectancy Index	A	-0.008	0.012	0.021	0.829
	B	-0.014	0.046	0.06	0.415
Value Index	A	0.025	-0.008	-0.033	0.576
	B	0.014	0.042	0.028	0.529
Cost Index	A	0.058	-0.058	-0.117	0.202
	B	-0.088	0.005	0.093	0.403
EVC Index	A	-0.014	0.021	0.035	0.556
	B	0.029	0.028	-0.002	0.878

**Table 4: Difference in change of motivation for both groups after problems with formal verification(FV change) and without (Test change).**

P-values for the change under both conditions were computed but are not shown because they are all insignificant. While the difference between the changes is also insignificant, we note that for Group A, the change in EVC Index after completing the design problem with formal verification was greater than without. Group A’s change in EI increased, their change in CI decreased, and their change in VI decreased. In contrast, Group B had a small decrease in the change of EVC Index, due to an increase in the change CI after completing the problem with formal verification. However, they did have a small increase in change EI and VI.

## 7 Discussion

### 7.1 RQ1: Student Motivation Levels

We evaluate student motivation level using the EVC Index. The mean EVC Index of participants in this study was 0.669. The index is on a scale from 0 to 1 and represents how motivated students are to study formal methods. A zero on the scale means a student disagreed with every expectancy and value item and agreed with every cost item in the survey in Section 5.1, while a score of 1 means the opposite. We can interpret 0.669 as meaning that generally students agree or are neutral that they can learn formal methods, it is valuable, and they have time to learn it. The mean EI (0.669), VI (0.697), and CI (0.478) all indicate that most students agree or are neutral about the specific categories, although more participants are neutral about having time to learn formal methods. This result appears to contradict educators’ claims that students are skeptical of formal methods and do not see their worth. However, given that the participants were in a Safe Autonomy elective course, we would expect them to have more positive feelings about formal methods than the average student. Therefore, it is likely that students in other computer science classes are less motivated than our participants.

The score of 0.669 is an exciting result that shows that when students are in courses with clear connections to formal methods, they have positive feelings towards formal methods. These results indicate that students in these courses would be receptive to more uses of formal methods. Advanced elective courses should consider incorporating the use of more formal methods that students are likely to have the motivation to learn. To the best of our knowledge,

this is one of the first studies to evaluate the feelings of students about formal methods before any treatment, and we have found that many students can be motivated to use formal methods.

## 7.2 RQ2: Change in EVC Indices

To answer RQ2, we examine how Expectancy, Value, and Cost Indices change during the assignment in Figure 2 and Table 3. Our results show that student motivation after completing the entire assignment increases insignificantly ( $0.028$ ;  $p = 0.167$ ). The change appears to be driven by an increase in EI ( $0.048$ ;  $p = 0.116$ ). In contrast, VI only increased by  $0.015$ , and CI only decreased by  $0.02$ . In Figure 2, we see that participants reported higher EIs after reflecting on the whole assignment, but lower VIs. It is likely that students may have reflected that they completed the assignment, raising their EI; however, they may have forgotten the value because they were no longer struggling to write a safe controller. One reason that we did not see major changes in motivation could be that students in this course were already motivated, so using a tool was not as impactful. We also note that prior research discusses that many formal methods tools are not user-friendly or are difficult to understand, which can deter students from employing formal methods. Likely, Verse is not user-friendly enough for students to receive the full benefits of working with a tool. Student feedback about the assignment included that it was difficult to debug code within Verse. Therefore, students' Expectancy and Value Indices may be higher, and their Cost Indices may be lower with a more usable tool.

While these results show minimal changes after the assignment, we are encouraged that student motivation did not decrease. The student who called formal methods “too theoretical” at the start of the assignment reflected that reachability analysis was helpful for solving the problems. Therefore, we still consider using a tool a good option for motivating students and showing them applications of formal methods.

## 7.3 RQ3: Testing Versus Verification

To answer RQ3, we compare how student motivation changed before and after the control, using Verse only as a testing and simulation tool, and treatment, using Verse as a formal methods tool. We believe that the order of the treatment versus the control does matter; therefore, we analyze Group A and Group B separately.

In general, for both Group A and B, changes in motivation after testing and verification were small. The differences in these changes were not significant. This is likely because students could not distinguish between what parts of the tool were and were not formal methods. However, this also indicates that students do not need to use heavyweight formal methods to see their benefits. Even showing students basic modeling may help motivate them to use formal methods, which is much easier to incorporate into introductory classes. Therefore, instructors in computer science should consider discussing modeling and its relationship to formal methods to encourage the widespread acceptance of formal methods.

Still, we see small differences in the change in motivation when testing versus verifying their designs. For example, when using testing, both groups' EI decreased, but increased when using formal verification. Using a formal methods tool increases students'

expectancy because they see that they can use formal methods. The change in VI increases for Group B and decreases for Group A by a small amount between testing and verification. An unusable tool is a barrier for students seeing the value in formal methods, so it is possible that the tool was not usable enough to impact this factor.

For cost, we see that Group A, which used testing first had a decrease in cost only after using formal verification. Likely, these participants completed the first design problem and understood how tedious showing safety can be. Then, when they got to use the formal verification tool, they realized how little time it took in comparison to manually showing safety. In contrast, Group B used formal verification first and were likely discouraged by the time it took, resulting in an increase in the CI. However, once these participants had to show safety without formal verification, they realized that it is a low-cost solution. These results are shown in Figure 2 as the cost increased after the first design problem and decreased after the second. These changes may be related to participants becoming familiar with the tool; however, a similar result does not appear for EI or VI. Related work claims that students usually do not think formal methods are worth their time, despite it being known by experts as a cost-effective way to increase software quality. When compared to showing safety without a tool, we see that students can learn that formal methods are low-cost. It may be useful to demonstrate the time formal methods can save with practical examples.

## 7.4 Treats to Validity

While this study is an important first step to understanding how using tools can impact student motivation, it has several limitations. First, the participants in the study are already taking a course in safe autonomy, so they may be more receptive to trying formal methods. We also had a small sample size due to many students not completing every survey. We had little control over when students actually took the surveys because they were contained within a one-part homework assignment we asked students to complete in a specific order. Most importantly, students may need more than one assignment to see the full benefits of formal methods. Evaluating students who use a tool for a whole semester or a large project may be more enlightening.

## 8 Conclusion

We have shown that students studying autonomy are motivated to learn formal methods. When surveying students as they completed a homework assignment, we saw small increases in motivation after using a formal verification tool. Future work should survey the motivation of students in introductory classes to understand the beliefs of most students in computer science and study the impact of using other tools, particularly tools already used in industry.

## Acknowledgments

This research was funded in part by NASA University Leadership Initiative grant (ULI) (80NSSC22M0070) for the AVIATE Project—Robust and Resilient Autonomy for Advanced Air Mobility. Thank you to Dr. Kathryn Cunningham for her suggestions and support at the start of this project and to Daniel Zhang for helping distribute this assignment to the students.

## References

- [1] Mehrmoosh Askarpour and Marcello M. Bersani. 2020. Teaching Formal Methods: An Experience Report. In *Frontiers in Software Engineering Education (Lecture Notes in Computer Science)*, Jean-Michel Bruel, Alfredo Capozucca, Manuel Maz-zara, Bertrand Meyer, Alexandr Naumchev, and Andrey Sadovykh (Eds.). Springer International Publishing, Cham, 3–18. doi:10.1007/978-3-030-57663-9\_1
- [2] Kenneth Barron and Chris Hulleman. 2014. *Expectancy-Value-Cost Model of Motivation*. pp. 503–509 (Vol. 8). doi:10.1016/B978-0-08-097086-8.26099-6
- [3] Per Bjesse. 2005. What is formal verification? *ACM Sigda Newsletter* 35, 24 (2005), 1–es.
- [4] Javier Blanco, Leticia Losano, Nazareno Aguirre, María Marta Novaira, Sonia Permigliani, and Gastón Scilingo. 2009. An introductory course on programming based on formal specification and program calculation. *ACM SIGCSE Bulletin* 41, 2 (2009), 31–37.
- [5] Hugo Brakman, Vincent Driessen, Joseph Kavuma, B Nij Bijvank, and Sander Vermolen. 2006. Supporting Formal Methods Teaching with Real-Life Protocols. (2006).
- [6] Antonio Cerone, Markus Roggenbach, James Davenport, Casey Denner, Marie Farrell, Magne Haveraaen, Faron Moller, Philipp Körner, Sebastian Krings, Peter Csaba Ölveczky, Bernd-Holger Schlingloff, Nikolay Shilov, and Rustam Zhumagambetov. 2021. Rooting Formal Methods Within Higher Education Curricula for Computer Science and Software Engineering — A White Paper —. In *Formal Methods – Fun for Everybody (Communications in Computer and Information Science)*, Antonio Cerone and Markus Roggenbach (Eds.). Springer International Publishing, Cham, 1–26. doi:10.1007/978-3-030-71374-4\_1
- [7] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. 2007. All about maude-A high-performance logical framework how to specify, program and verify systems in rewriting logic. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 4350. 1–819.
- [8] Paul Curzon and Peter W McOwan. 2013. Teaching formal methods using magic tricks. In *Fun with Formal Methods: Workshop at the 25th International Conference on Computer Aided Verification*. Citeseer.
- [9] James H Davenport and Tom Crick. 2021. Cybersecurity education and formal methods. In *Formal Methods–Fun for Everybody: First International Workshop, FM-Fun 2019, Bergen, Norway, December 2–3, 2019, Revised Selected Papers 1*. Springer, 159–172.
- [10] Norman T. Feather (Ed.). 2021. *Expectations and Actions: Expectancy-Value Models in Psychology*. Routledge, London. doi:10.4324/9781003150879
- [11] Hubert Garavel, Maurice H. Ter Beek, and Jaco Van De Pol. 2020. The 2020 Expert Survey on Formal Methods. In *Formal Methods for Industrial Critical Systems*, Maurice H. Ter Beek and Dejan Ničković (Eds.). Vol. 12327. Springer International Publishing, Cham, 3–69. doi:10.1007/978-3-030-58298-2\_1 Series Title: Lecture Notes in Computer Science.
- [12] Ganesh Gopalakrishnan. 2012. Formal methods for surviving the jungle of heterogeneous parallelism. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*. IEEE, 1321–1324.
- [13] Mathieu Jaume and Théo Laurent. 2014. Teaching Formal Methods and Discrete Mathematics. *Electronic Proceedings in Theoretical Computer Science* 149 (April 2014), 30–43. doi:10.4204/EPTCS.149.4 arXiv:1404.6604 [cs].
- [14] Eduard Kamburjan and Lukas Grätz. 2021. Increasing Engagement with Interactive Visualization: Formal Methods as Serious Games. In *Formal Methods Teaching*, João F. Ferreira, Alexandra Mendes, and Claudio Menghi (Eds.). Springer International Publishing, Cham, 43–59. doi:10.1007/978-3-030-91550-6\_4
- [15] Jeff J. Kosovich, Chris S. Hulleman, Kenneth E. Barron, and Steve Getty. 2015. A Practical Measure of Student Motivation: Establishing Validity Evidence for the Expectancy-Value-Cost Scale in Middle School. *The Journal of Early Adolescence* 35, 5–6 (June 2015), 790–816. doi:10.1177/0272431614556890 Publisher: SAGE Publications Inc.
- [16] Yangge Li, Haoqing Zhu, Katherine Braught, Keyi Shen, and Sayan Mitra. 2023. Verse: A Python Library for Reasoning About Multi-agent Hybrid System Scenarios. In *Computer Aided Verification*, Constantin Enea and Akash Lal (Eds.). Springer Nature Switzerland, Cham, 351–364.
- [17] Shaoying Liu, Kazuhiro Takahashi, Toshinori Hayashi, and Toshihiro Nakayama. 2009. Teaching formal methods in the context of software engineering. *ACM SIGCSE Bulletin* 41, 2 (June 2009), 17–23. doi:10.1145/1595453.1595457
- [18] Peter Csaba Ölveczky. 2009. Teaching formal methods based on rewriting logic and Maude. In *International Conference on Technical Formal Methods*. Springer, 20–38.
- [19] Wishnu Prasetya, Craig Leek, Orestis Melkonian, Joris ten Tusscher, Jan van Bergen, Jasper Everink, Thomas van der Klis, Rick Meijerink, Roan Oosenbrug, Jelle Oostveen, et al. 2019. Having fun in learning formal specifications. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 192–196.
- [20] Bernard Rosner, Robert J Glynn, and Mei-Ling T Lee. 2006. The Wilcoxon signed rank test for paired comparisons of clustered data. *Biometrics* 62, 1 (2006), 185–192.
- [21] Bernd-Holger Schlingloff. 2021. Teaching model checking via games and puzzles. In *Formal Methods–Fun for Everybody: First International Workshop, FMFun 2019, Bergen, Norway, December 2–3, 2019, Revised Selected Papers 1*. Springer, 143–158.
- [22] Pablo Schoeffel, Vinicius F. C. Ramos, Cristian Cechinel, and Raul Sidnei Wazlawick. 2021. The Expectancy-Value-Cost Light Scale to Measure Motivation of Students in Computing Courses. *Informatics in Education* (June 2021). doi:10.15388/infedu.2022.04
- [23] Anna Sotiriadou and Petros Kefalas. 2000. Teaching Formal Methods in Computer Science Undergraduates. (Jan. 2000).
- [24] Maurice ter Beek, Manfred Broy, and Brijesh Dongol. 2024. The role of formal methods in computer science education. *ACM Inroads* 15, 4 (2024), 58–66.
- [25] Mark Utting and Steve Reeves. 2001. Teaching formal methods lite via testing. *Software Testing, Verification and Reliability* 11, 3 (2001), 181–195.
- [26] Rustam Zhumagambetov. 2021. Teaching formal methods in academia: a systematic literature review. In *Formal Methods–Fun for Everybody: First International Workshop, FMFun 2019, Bergen, Norway, December 2–3, 2019, Revised Selected Papers 1*. Springer, 218–226.
- [27] Peter Csaba Ölveczky. 2021. Teaching Formal Methods for Fun Using Maude. In *Formal Methods – Fun for Everybody (Communications in Computer and Information Science)*, Antonio Cerone and Markus Roggenbach (Eds.). Springer International Publishing, Cham, 58–91. doi:10.1007/978-3-030-71374-4\_3