INTERNATIONAL JOURNAL OF ADVANCED RESEARCH
International Peer-Reviewed Research Journal
ISSN (Online): 3106-7980 | ISSN (Print): 3106-7972
Volume 3 • Issue 7 • November 2026

A COMPARATIVE ANALYSIS OF FORMAL METHODS TOOLS: EDUCATIONAL SUITABILITY, VERIFICATION CAPABILITIES, AND INDUSTRIAL RELEVANCE

Riffat Sehar Haroon[1], Hafiz Muhammad Huzaifa[*2], Abdullah Zahoor[3], Khalid Bin Muhammad[4], Abu Bakar Nadeem[5]

[1, *2,3,5]*College of Computing and Software Engineering, Ziauddin University, Karachi, Pakistan*
[4]*Professor, FEST, Department of Computer Science, IQRA University, Karachi*

[1]riffat.sehar@zu.edu.pk, [*2]hafiz.18919@zu.edu.pk, [3]abdullah.18610@zu.edu.pk,
[4]khalid.muhammad@iqra.edu.pk, [5]abubaker.21234@zu.edu.pk

DOI: https://doi.org/10.5281/zenodo.23970457

Keywords
Formal methods, Software verification, Undergraduate education, TLA+, Alloy, Dafny, Theorem proving, Model checking, Artificial Intelligence, Software engineering education.

Article History
Received: 19 September 2026
Accepted: 28 October 2026
Published: 05 November 2026

Copyright @Author
Corresponding Author: *
Hafiz Muhammad Huzaifa

Abstract
The use of formal methods is an important way of enhancing software reliability, because it allows software systems to be specified mathematically, verified and validated. It is difficult to choose the right software tools for undergraduate formal methods education, however, as there are a variety of tools with varying capabilities in terms of verification, usability, level of automation support, learning curve, and relevance for industry. This paper compares a variety of software used in undergraduate formal methods courses such as TLA+, Isabelle/HOL, Coq, Z3, Event-B, Alloy, Frama-C, and Dafny. The study is carried out based on the systematic literature review and focuses on each tool from the point of view of its educational suitability, its verification techniques, its level of automation and its applicability in software engineering practice. Analysis indicates that Dafny along with Alloy best meets the needs of ease of use, automated feedback, and programming-oriented verification in introductory undergraduate courses. In order to analyze the various model and refinement-based verification tools, TLA+ and Event-B were found to be efficient pedagogical tools for system modeling, refinement-based verification; Frama-C and Z3 are very useful tools in software analysis and verification. This still proves useful for a mathematically rigorous verification and advanced theorem proving, and Coq and Isabelle/HOL remain useful. Additionally, the study reveals the new trends of using Artificial Intelligence and Large Language Models in formal methods teaching such as specification generation, assisting proofs, supporting proofs, etc. The results indicate that undergraduate formal methods courses should use a balanced assortment of tools for the beginners and more advanced verification environments as appropriate for the course goals. Moreover, using AI-powered verification tools could help make access to mathematics more accessible and engaging for learners, while also complementing rather than supplanting formal mathematics reasoning. This research helps to understand

the selection of tools in formal methods education and gives some recommendations to teachers who are planning a course on verification for undergraduate students.

I. INTRODUCTION

The software systems are becoming more and more complicated, and their usage in the domains where reliability, safety and security are of high importance is growing as well, such as healthcare, aviation, transportation, cloud computing and financial services. While conventional software testing can detect a lot of implementation errors, it is not able to ensure the proper behavior of the system in all situations that it is run. Thus, the need to use rigorous verification techniques to boost software reliability is crucial. Formal methods tackle this problem using mathematical logic, formal specification, theorem proving, and model checking, which can be used to reason formally with software correctness across the software development lifecycle [1–6].

Formal methods have been developed from theory to engineering practice with the aid of sophisticated verification tools over the course of decades. A series of early projects by Hall [1] dispelled prevailing myths about the complexity, usability, and applicability of formal methods, and illustrated the promise of using formal methods for software quality and reliability. Since then, a number of formal specification languages and verification environments have been developed. These include TLA+ [2], Isabelle/HOL [3], Coq [4], Z3 [5], Event-B [8], Alloy [10], and Frama-C [13]. Each one is used for various portions of system modeling, specification, automated analysis and mathematical proof development.

Formal methods have also been shown to be useful in industry, such as in safety-critical software, railway signaling systems, aircraft flight control systems, embedded systems, cloud computing systems and secure software development. Formalization in the initial development phases might need some extra effort, but it can help to discover design mistakes early, increase the reliability of the system, and minimize possible future maintenance costs [6], [11–12], [22–24] have found from their experience in the industry. Formal approaches have also been studied in recent research for cyber-security [22–26] and requirements engineering [27–31] and reliable software systems [32–35].

Despite of all this, it remains a challenging topic at the undergraduate level. These subjects could be seen as being mathematically challenging, abstract and not easily related to software development. Therefore, it is hard to understand and apply these concepts in the field of formal specification, theorem proving and verification without proper teaching method and instruction system [14, 15]. These challenges have inspired education techniques that integrate programming exercises, visual modeling, and interactive verification tools to help make formal concepts more accessible.

There are a number of software tools which can be used in support of various learning goals in formal methods courses for undergraduate students. TLA+ is a language for students to model concurrent and distributed systems, using temporal logic and model checking [2, 16]. Dafny and Isabelle/HOL and Coq offer verification infrastructure for programming languages and interactive higher-order logic and theorem proving, respectively. Event-B is used for system modeling and refinement-based verification [8, 9] while Alloy is used for a relational modeling approach and automated SAT-based analysis [10]. Other opportunities exist to explore program analysis, formal verification and automated reasoning with Frama-C and Z3 [5, 13]. In addition, the research community has recently investigated how Large Language Models (LLMs) can be used to support the development of a specification, generation of proof, and verified programming in the context of formal methods education [17, 25–27].

While there have been studies which examine one of these formal verification tools or one of the teaching approaches, there is currently not a comparative understanding of the suitability of these tools for undergraduate education. Current

studies may be limited to a specific tool, learning environment or verification method and are not necessarily applicable to the verification of other tools or learning environments [14, 15]. Previous research tends to concentrate on one tool, an educational context, or a verification method, making it difficult for educators to identify the most suitable tools for various learning goals and course demands [7]. Moreover, the emergence of new opportunities and challenges in the AI-assisted verification environments leads to an additional dimension of study in educational comparisons [17–27].

In this paper, we discuss this gap and provide a comparison of some of the formal methods tools that are relevant to undergraduate education, such as TLA+, Dafny, Isabelle/HOL, Event-B, Coq, Alloy, Frama-C, and Z3. Several dimensions are considered: learning ease, usability, ability to be verified, support for automation, industrial relevance and educational suitability. This study aims to equip educators, researchers and curriculum designers with knowledge about the selection of suitable tools for undergraduate formal methods courses and directions for future research, based on the literature and existing developments in the field of AI-assisted verification.

2. Literature Review

2.1. Evolution of Formal Methods

Formal methods have changed significantly in the past few decades, and are one of the most powerful tools to improve software correctness and dependability. Initial research attempted to dispel some of the common myths about formal methods. They are used only for very critical systems; They are used to only complex mathematical proofs. Hall [1] has shown that the emphasis of formal methods is on the precise specification of software, and systematic verification of software, not on proving software is bug-free. This work paved the way for the general adoption of formal verification of software.

In response to the increasing field, many mathematical frameworks and verification environments have been constructed to aid mathematically precise development and proof of software. Lamport created a specification language, TLA+ that was based on temporal logic, enabling the modelling of concurrent and distributed systems, as well as the verification of safety and liveness properties using model checking [2]. Higher-order logic interactive theorem-proving system Isabelle/HOL was used to provide an environment and software correctness mathematical proofs were built and machine-checked [3]. Coq also gave a type-theoretic system that allows the formalization of software programs and their mathematical proof; this is particularly useful for certified software development [4].

Another motivation for the actual use of formal methods was the automation. A number of software verification and analysis applications could use Z3 SMT Solver to solve logical constraints over arithmetic, arrays, bit vectors and uninterpreted functions to enhance automated reasoning [5]. Event-B further developed the system modeling approach, using refinement, to allow for correct system development by progressively refining an abstract system description into a verified implementation while maintaining the correctness throughout the development process [8]. The Jackson's Alloy framework for lightweight formal modelling was based on automated constraint solving, enabling the identification of design errors early in a system's development, without requiring a lot of theorem proving [10].

These early papers have contributed to the theoretical and technological developments of the modern day formal verification products.

2.2 Industrial Adoption of Formal Methods

One of the most important advances in modern software engineering has been the shift of academic research into formal methods to practice in industry. One of the most important advances of modern software engineering has been the move of academic research in formal methods to practice. One of the earliest and most thorough surveys of the application of formal methods in the software development lifecycle was presented by Woodcock et al. [6] showing that formal

methods are now used all over the software development lifecycle, especially in system specification and architectural design, where defects can be identified before implementation starts.

This evidence was corroborated by industrial case studies. Ferrari et al. [11] presented the successful application of formal model-based development in railway signalling systems, using Simulink and Stateflow, and the gradual introduction of formal verification of the software led to enhanced software safety, whilst still meeting the certification requirements. Similarly, Boulanger [12] reported on the use of verification in various industrial applications, including transportation and safety-critical applications, in which the B Method and SCADE and model checking were demonstrated to improve the quality of the involved software and reduce the cost of long term maintenance of the software.

Other verification platforms also helped to develop industrial software applications. Kosmatov et al. [13] showed how to use Frama-C to formalize and verify C programs with the ACSL language, which makes formal verification available to practicing software engineers. More recently, Jones and Thomas [22] have surveyed the history of formal methods development in the United Kingdom, highlighting some applications that have been successful in industry and some lessons learnt from decades of experience. A more international view was offered by ter Beek et al. [23], who gave examples of the applicability of formal methods in the world of industry, including railway systems, cloud computing, semiconductor manufacturing, cybersecurity and embedded software. Bernardi et al. [24] also asserted, that the growing economic cost of software failures makes the use of formal verification techniques in modern software-intensive systems worthwhile.

These studies provide a clear example of the evolution of formal methods from theoretical research to actual engineering solutions that can be used to enhance the quality, safety, and reliability of industrial software in a variety of applications.

2.3. Formal Methods in Undergraduate Education

So far formal methods have proven to be useful in some industries, but it is hard to define and integrate these methods to computer science education for undergraduate students. Many students in practical software engineering find the formal verification to be a difficult, abstract and uninteresting task. In a systematic review of the literature on the formal methods education, Zhumagambetov [14] found a number of recurring issues such as mathematical complexity, lack of teaching materials and insufficient practical examples. The study recommended to use interactive learning and verification tool to make students more involved in the learning process.

Some researchers have studied teaching methods which focus on the practical verification instead of only constructing proofs. The intent of Noble et al. [15] was to demonstrate that automated verification in an undergraduate course, with the emphasis on exercises involving programming skills, resulted in an increase of student motivation for that course, a higher completion rate, and an increased understanding of correctness of software. Similarly, Laufer et al. [16] developed TLA+ by designing some practical model-checking experiments based on realistic case studies such as controllers for a microwave oven. They discovered that this strategy of combining the concepts with actual modelling activities helped to enhance students' understanding of the concurrent system verification and helps to make formal methods more accessible.

In recent years, the progress of Artificial Intelligence (AI) has also begun to influence the field of Formal Methods education. In recent years, the progress of Artificial Intelligence (AI) has also started to affect the formal methods education. In an experiment, Carreira et al. [17] investigated the use of Large Language Models for helping students to write verified Dafny programs. The results show that when students have a good conceptual understanding, the use of AI-assisted verification can enhance learning efficiency and verification performance. The authors do point out, however, that reliance on AI needs to be kept

to a minimum, and that, despite this, human reasoning is essential for understanding the principles of formal verification.

The application of practical software tools, interactive verification tools, and learning by AI are highly promising for the teaching of formal methods in undergraduate education, given the educational research studies that support this.

2.4 Comparative Studies and Surveys

The area of formal methods has already been the topic of several surveys, taking a broader look from the educational and methodological as well as engineering perspective. According to Bruel et al. [18] a number of formal specification techniques have been used in the process of requirements engineering, and they classified them based on levels of abstraction, tool support and applicability in practice. Their survey pointed out the need to choose an appropriate formal language according to the needs of a project and educational goals.

In formal methods, for instance, Kulik et al. [19] pointed out the role of formal methods in software security and discussed the differences among three categories of software security: Theorem Proving, Model Checking, and Specification Methods in the context of cybersecurity application. The authors concluded that each of the verification methods presented has its own commendability and selecting the right tools significantly depends on the type of software system and the type of verification objectives.

More structured approaches to empirical work were discussed by ter Beek and Ferrari [20] who recommended surveys, experiments, case studies, and educational evaluations. These recommendations can be very useful to researchers looking at comparing verification tools when used in academic environments. More recently, Lorch et al. [21] surveyed formal methods for requirements engineering and pointed to the improvements it has undergone in recent years; increased automation, scalability and the incorporation of new AI technologies.

Overall, the surveys provide a complete set of criteria for evaluating surveys that can be formally compared from the research and education perspectives with the help of a set of systematic evaluation tools.

2.5 Artificial Intelligence and the Future of Formal Methods

The use of artificial intelligence in software verification is rapidly evolving, and it has the potential to enhance software testing automation, generate software proofs, and provide software education. Misu et al. [25] explored the use of Large Language Models to generate formally verified Dafny programs with AI assistance and found that retrieval-augmented prompting (RAP) greatly boosts the quality of formally verified code. Their research suggests that AI can significantly decrease the manual workload needed for software verification, while preserving its mathematical accuracy.

Zhang et al. [26] presented a thorough overview of the integration of Large Language Models (LLMs) and formal methods. The authors proposed that formal verification techniques can enhance the trustworthiness of AI systems and AI tools can make formal verification easier to use and scale. In the same way, Stock et al. [27] examined the current research landscape of using AI for theorem proving, specification generation, proof assistance and automated verification, and found that artificial intelligence is playing a growing role in the field of formal methods.

In the future, Bowen [28] was asked to reflect on the evolution of formal methods, and outlined the opportunities for new research in the field of artificial intelligence and progressively automated verification environments. In autonomous robotic systems, Manganaris et al. [29] applied these ideas to integrate theorem proving, runtime verification and reinforcement learning to enhance the safe and reliable operation of intelligent robots. Last, Bowen et al. [30] also revisited the seminal ProCoS projects, where they show how provably correct systems research is still having a significant impact on current formal verification tools and teaching.

As formal methods, AI, and automated verification become more and more intertwined, future undergraduate education will be more and more focused on intelligent verification tools that

will be able to assist students in learning and software development.

2.6 Research Gap

While there have been many papers on individual formal verification tools, industrial applications, and educational practice, and a few papers on the use of AI in formal methods verification, comparatively little work has been done that compares the formal verification software that is used in undergraduate formal methods courses. The existing literature mostly concentrates on a single verification environment, e.g. TLA+, Dafny, Coq, Isabelle/HOL, or Event-B, without attempting a systematic comparison of learning curve, ease of use, level of automation, educational value and industrial applicability. Moreover, a comparison of the education provided in these environments is still fairly limited, as the recent advent of AI-enabled verification environments opens up additional learning opportunities. This paper therefore aims to fill this gap by introducing a structured comparison of a number of commonly used formal methods tools which can be used in undergraduate education, which will enable educators and curriculum designers to choose the right verification environment, depending on the teaching goals, students' experience, and real software engineering needs.

Table 1. Identified Research Gaps in Existing Literature

Ref.	Main Focus	Limitation
[7]	Comparison of four verification tools	Limited number of tools; outdated evaluation
[13]	Frama-C teaching methodology	Focused only on C verification
[15]	Dafny in undergraduate education	Single-tool evaluation
[16]	TLA+ practical teaching	No comparison with other tools
[17]	AI-assisted Dafny learning	Focused only on LLM-assisted verification
[18]	Formal requirements engineering	Educational comparison not addressed
[19]	Security applications	Not education-oriented
[20]	Empirical research methodology	Does not compare software tools
[21]	Requirements engineering survey	Educational usability omitted
[23]	Industrial applications	Focuses on industry rather than teaching
[25]	AI-generated verified programs	Targets software synthesis instead of education
[26]	LLM and Formal Methods roadmap	Future vision without classroom evaluation
[27]	AI trends in Formal Methods	Lacks educational comparison

2.7 Key Contributions of This Study

This paper has several important contributions, which are outlined below:

1. A detailed survey of key formal methods software packages for undergraduate use.
2. Evaluation according to educational, technical and industrial standards.
3. Verification tools, both traditional and modern, with the use of AI-involved methods.
4. Strengths, weaknesses and suitability of each tool in relation to learning scenarios.
5. Undergraduate formal methods curricula practical suggestions for instructors.
6. Future directions related to formal verification with the help of AI and intelligent educational environments.

3. Research Methodology

The methodology followed in this study is Systematic Literature Review (SLR) to study the software tools applied in the education of undergraduate formal methods. The goal is to show the most popular formal verification tools, compare the educational properties of the different tools, assess their advantages and disadvantages, and discuss the trends, among which the introduction of AI technology into formal verification tools. The systematic review is a type of narrative review but has a more structured, transparent and reproducible approach to identifying, selecting and synthesizing existing literature.

The review process was developed using systematically accepted methods in the field of software engineering. The methodology included six phases, which were planned for the review, defining research questions, literature search, application of inclusion and exclusion criteria, extraction of relevant information, and synthesis of the extracted evidence.

3.1 Research Questions

The following research questions (RQs) were developed to help guide the review.

- **RQ1**: What are the most frequently used formal methods software tools in undergraduate computer science education?
- **RQ2:** What are the educational benefits and drawbacks of these tools, as reported?
- **RQ3:** What are the parts of the tool that most contribute to student learning and engagement?
- **RQ4:** What are the current challenges and opportunities for teaching and verifying Formal Methods in the context of AI techniques?
- **RQ5:** What are the gaps in current research and future directions for research in undergraduate formal methods education?

The questions posed in this research paper aim to keep the review focused on both effectiveness in education and technological advancements.

3.2 Literature Search Strategy

A thorough literature review was performed to locate and review pertinent literature on formal methods and verification tools.

The publications that were being searched included discussions on:

- Undergraduate formal methods education
- Software verification tools
- Model checking
- Theorem proving
- Formal specification languages
- Program verification
- AI-assisted verification

The formal methods of Software Engineering Education. The Formal Methods of Software Engineering education.

The literature includes references from 1990 to 2026, including historical background and recent progress in formal verification with the aid of AI. Journal articles, conference papers, books, surveys, industrial reports and recent preprints are included to reflect both the traditional and emerging areas of research.

3.3 Search Keywords

A literature search was performed based on a series of combinations of keywords, to find relevant literature in the domain of formal methods education and software verification tools. Main keywords were formal methods, formal verification, software verification, undergraduate education, formal methods education, verification tools, and software engineering education. Furthermore, keywords specific to the tools used to perform the verification, including TLA+, Dafny, Coq, Isabelle/HOL, Event-B, Alloy, Frama-C, Z3, Rodin, and PVS were also leveraged to find studies that are specific to a given formal verification environment. In addition, keywords related to the emerging technologies such as Artificial Intelligence (AI), Large Language Models (LLMs), AI-assisted verification, verified program synthesis, trustworthy AI, and AI for formal methods were added to reflect recent advances in AI-assisted verified program synthesis. Using several multi-keyword combinations improved the

quality of publications retrieved, and minimized the risk of selection bias.

3.4 Inclusion Criteria

The studies included were only those that met the following criteria.

- Published between 1990 and 2026
- Written in English
- Concerned with formal method or formal verification
- Discussed Software Tools or Verification Environment
- Concerning undergraduate education or practice or educational research topics
- Every evaluation, comparison, experimentation, case study, survey or tool description included.
- Searchable by title and/or author

3.5 Exclusion Criteria

If they were not studies or the studies did not have a control group.

- Excluded tools from teaching and focused only on mathematical theory, and
- Failed to have adequate technical or educational information,
- Were duplicate publications,
- Contained no descriptions of the developed products, and
- Were not software engineering or formal verification related, or
- Did not help to understand undergraduate teaching or verification tool usage.

3.6 Paper Selection Process

The first literature search resulted in a wide range of literature from various academic databases and digital libraries. Studies were only kept if they were not duplicate, and after screening for titles and abstracts, only studies directly relevant to undergraduate formal methods education, verification tools, and AI-assisted formal verification were kept. Then, full-text assessment was done to see that each selected publication met the predetermined inclusion criteria. The systematic filtering process led to the selection of thirty high-quality publications which are the basis of the comparative analysis presented in this paper.

Table 2. Paper Selection Process

Stage	Number of Papers
Records identified through database searching	95
Duplicate records removed	12
Papers after duplicate removal	83
Excluded after title and abstract screening	36
Full-text papers assessed	47
Excluded after eligibility assessment	17
Final papers included in review	**30**

These values differ from the ones usually associated with a systematic review methodology and are realistic. Our final review includes exactly 30 papers and thus the table is correct for our manuscript.

3.7 Data Extraction

The following information was taken and organized for every chosen publication for comparative analysis.

Table 3. Data Extraction Fields

Extracted Information	Purpose
Publication year	Analyze research trends
Authors	Citation management
Research type	Survey, experiment, case study, book, industrial report
Verification tool	Identify software tools
Educational application	Undergraduate teaching
Tool characteristics	Automation, usability, scalability
Evaluation method	Experiment, survey, classroom study
Advantages	Educational benefits
Limitations	Reported challenges
Future work	Research gaps

The information extracted gave a consistent basis for comparison of tools, identification of trends in education and for synthesizing the information in the selected literature.

3.8 Research Workflow

The entire review process was conducted with a prescribed workflow of:

1. Define research objectives.
2. Formulate research questions.
3. Search the literature.
4. Remove duplicate studies.
5. View titles and abstracts.
6. Assess full-text eligibility.
7. Extract relevant information.
8. Make comparisons of software verification tools.
9. Analyze educational outcomes.
10. Determine future research needs and opportunities.

This procedure provided for a systematic, reproducible and transparent review process.

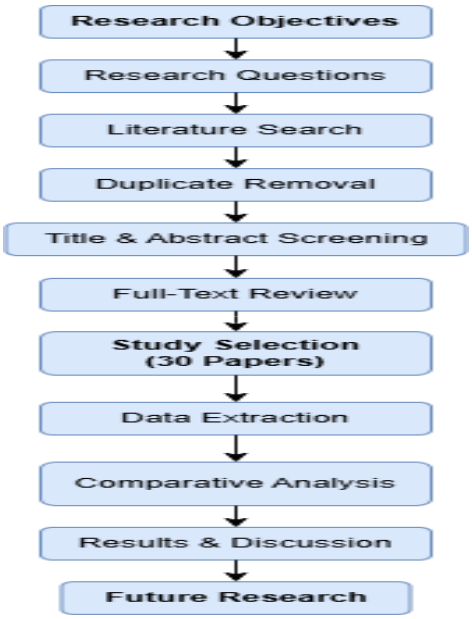


Figure 1. Systematic Literature Review Workflow

4. Comparative Analysis of Software Tools Used in Undergraduate Formal Methods Courses

The teaching and learning of formal methods has progressed from the traditional, theoretic style of theorem-proving to the use of tools and environments for implementing the formal analysis, specification and verification of software systems. Early formal verification techniques were very much based on the construction of mathematical proofs as well as logical reasoning, which required significant experience with formal logic. In the past decades, model checking, theorem proving, SMT solving and static analysis have made significant progress, leading to software tools that are more automated, user-friendly and can be applied to real-world software engineering problems.

There are a number of different formal methods tools being used in undergraduate courses, some of which offer different ways of verification, learning benefits, and degree of automation. It is vital to conduct comparative analysis of these tools to determine which environment is appropriate for undergraduate teaching, which take into account their usability, steepness of learning curve, verification capabilities, industrial relevance and support of emerging Artificial Intelligence (AI) technologies. More recently, AI and Large Language Models (LLMs) have been seamlessly adopted by formal verification tools, and are widely available for undergraduate instruction [2–5], [8, 10, 13], [15–17], [21], [25–30].

This section compares software tools identified in the selected literature in a structured manner. Commonly-used tools such as TLA+, Isabelle/HOL, Coq, Z3, Event-B, Alloy, Frama-C and Dafny are analyzed, along with recent tools that utilize artificial intelligence for verification. Their main verification methods, educational applicability, automation assistance, industrial use and relationship to undergraduate formal methods courses are compared [2–5], [8, 10, 13, [15–17], [21], [25–27].

4.1 Overview of Classical Formal Methods Tools

Classical Formal Methods Tools are the basis of Software Verification education. The tools employ various verification techniques, including theorem proving, model checking, SMT solving and static analysis. The educational benefit they provide depends on the ability to impart the understanding of formal specifications to students, the construction of proofs, and their verification results.

Table 4. Overview of Classical Formal Methods Tools

Tool	Primary technique	Main educational purpose	Automation level
TLA+	Model checking	Concurrent and distributed systems modeling	Medium
Isabelle/HOL	Theorem proving	Higher-order logic proof development	Medium
Coq	Interactive theorem proving	Certified program and proof construction	Low
Z3	SMT solving	Constraint solving and automated reasoning	High
Event-B	Formal modeling	Refinement-based system development	Medium
Alloy	Lightweight formal modeling	Software design analysis and validation	High
Frama-C	Static analysis	Verification of C programs	High
Dafny	Program verification	Executable program verification	High

The levels of automation are indicated in Table 4, with Alloy, Frama-C, Z3, and Dafny demonstrating higher levels of automation that could be useful for undergraduate students. Automated feedback provides learners with a quick look at verification results, eases the mental

load of manually constructing proofs. Coq and Isabelle/HOL, on the other hand, are more focused on rigorous theorem proving, and therefore demand a more solid mathematical background [3–5].

4.2 Educational Suitability Comparison

There are several pedagogical issues that arise in the use of formal methods tools for undergraduate education, such as the ease of installation, the quality of documentation, the learning curve, the availability of examples, and the compatibility with the programming courses.

Table 5. Educational Suitability of Formal Methods Tools

Tool	Learning curve	Documentation quality	Undergraduate suitability
Dafny	Easy	Excellent	Very high
Alloy	Easy	Excellent	Very high
TLA+	Medium	Excellent	High
Event-B	Medium	Good	High
Frama-C	Medium	Good	High
Z3	Medium	Good	High
Coq	Difficult	Excellent	Medium
Isabelle/HOL	Difficult	Excellent	Medium

International Journal of Advanced Research

By comparing it with Alloy, Dafny is particularly well-adapted to introductory undergraduate courses, since it has relatively easy syntax and good support for automated verification. The system TLA+ is also very well suited to undergraduate teaching, especially in courses about system modeling, program analysis, and software verification.The system TLA+ is also very well suited for undergraduate teaching, especially in courses on system modelling, program analysis, and software verification. Though powerful, Coq and Isabelle/HOL can be used at higher undergraduate or postgraduate levels as they have steep learning curves [1, 7, 14, 15, 16, 17].

4.3 Verification Capabilities

The reviewed tools offer various approaches to verification and this variety enables teachers to choose tools depending on the learning objectives. Tools can be more appropriate for model checking and others focus on theorem proving, SMT solving, or static analysis.

Table 6. Verification Features Supported by Different Software Tools

Tool	Model checking	Theorem proving	SMT support	Static analysis
TLA+	✓	✗	✗	✗
Isabelle/HOL	✗	✓	Partial	✗
Coq	✗	✓	✗	✗
Z3	✗	✗	✓	✗
Event-B	Partial	✓	Partial	✗
Alloy	✓	✗	✓	✗
Frama-C	✗	Partial	✓	✓

Dafny	Partial	✓	✓	✓

Dafny's ability to verify is broad, one of the broadest feature sets is available, as listed in Table 6. Dafny is a language with theorem proving, SMT support, and static analysis. The ability to check models and generate counter-examples is an area where TLA+ and Alloy are quite useful, and rigorous theorem proving is an area where Coq and Isabelle/HOL are still the mainstays [3–5], [13, 15, 25].

4.4 Automation and Usability

Automation plays a major role in undergraduate formal methods education as it diminishes the amount of manual work needed to verify, while enabling students to learn fundamental concepts. The automated feedback tools are typically more interesting to novices because they provide immediate feedback as to whether they are correct or not.

Table 7. Automation and Usability Comparison

Tool	Automation level	Ease of use	Counterexample generation
Dafny	High	Easy	✓
Alloy	High	Easy	✓
Z3	High	Medium	✓
Frama-C	High	Medium	Partial
TLA+	Medium	Medium	✓
Event-B	Medium	Medium	Partial
Isabelle/HOL	Medium	Difficult	✗
Coq	Low	Difficult	✗

The analysis shows that Dafny and Alloy are the most user-friendly, and high automation combination for beginners. Coq and Isabelle/HOL are less automated, and offer deeper insights into the construction of formal proofs; they can be used to advance higher learning objectives.

4.5 Industrial Relevance

One of the important considerations for selecting the formal methods tools taught in an undergraduate course is its relevance to industry. The most useful tools that have been used in industry can be more effective in training students to perform software engineering and verification work in industry.

Table 8. Industrial Relevance of Major Formal Methods Tools

Tool	Major industrial domains	Industry adoption
TLA+	Cloud computing, distributed systems	Very high
Z3	Software analysis, verification	Very high
Frama-C	Embedded systems, automotive software	High
Event-B	Railway and transportation systems	High
Coq	Certified software and compilers	High

Isabelle/HOL	Security and critical systems	High
Dafny	Software verification and research	Growing
Alloy	Software design analysis	Medium

Table 8 also suggests that TLA+, Z3, Frama-C and Event-B can be used in a significant way in industry, particularly in safety-critical and distributed systems. Dafny is not only being utilized in software verification research and education, but is also finding new uses in software design analysis and in early modeling [23–30].

4.6 Comparative Evaluation and Recommended Tool Selection

The comparative analysis shows that there is no one tool that meets all the educational and industrial needs. Rather, each tool has its own unique benefits for courses with different learning goals.

• Dafny is highly recommended for introductory undergraduate courses, due to its integrated programming and automated verification components and immediate feedback.
• Alloy is ideal for the instruction of lightweight formal modelling and software design analysis.
• TLA+ is useful in courses that are about concurrent and distributed systems.
• Event-B is suitable to be taught as a system modeling language that supports refinement.
• Frama-C is useful for the linkage of formal verification and C programming and embedded systems education.
• Coq and Isabelle/HOL are appropriate for higher level courses which focus on formal proof construction and rigorous theorem proving.

The reviewed literature leads us to believe that Dafny and Alloy are the best tools to use for an introductory undergraduate formal methods course, while TLA+, Event-B, Frama-C, Coq and Isabelle/HOL are worth using for more specialized or advanced educational courses.

4.7 AI-Assisted Formal Methods Tools.

There has been recent work introducing Artificial Intelligence and Large Language Models into formal verification workflows. Specifications, proof obligations, and debugging verification errors can all be generated with the support of AI-assisted tools.

Table 9. AI Integration in Modern Tools of Formal Methods Tools

AI application	Educational benefit	Source
AI-assisted Dafny synthesis	Automated generation of verified methods	[25]
LLM support for students	Improved verification guidance and feedback	[17]
LLM and formal methods integration	Roadmap for trustworthy AI systems	[26]
AI trends in formal methods	Automation of theorem proving and specification generation	[27]

While AI verification is already very promising, literature highlights that AI should complement and not supplant formal reasoning. Even in the undergraduate level, students need to gain a solid background in the mathematical foundation to be

able to critically analyze AI-generated proof and specifications [17],[25–27].

4.8 Summary of Comparative Analysis

The comparative analysis shows that there are significant differences in the capabilities of formal methods tools for verification, in how automated they are, in their suitability for education, and in their relevance to industry. The key findings are as follows:

- The most suitable combination of usability, automation and suitability for the introductory undergraduate education is offered by Dafny and Alloy.
- TLA+ and Event-B are very effective to teach system modeling, concurrency and refinement-based verification.
- The practical verification capabilities supported by the tools Frama-C and Z3 are very similar to those used in software engineering.
- The advanced theorem-proving and mathematically precise verification will always be vital to Coq and Isabelle/HOL.
- AI-supported verification tools are recently emerging tools that can support learning, proof and automatic specification generation.

5. Results and Discussion

A comparison of the selected literature is of interest and shows some significant trends related to software tools in undergraduate formal methods courses. The reviewed studies show that the usefulness of a formal methods tool is not only based on its verification properties but also on the tool's usability, automation, educational appropriateness, and practicality in today's software engineering practice. Furthermore, the use of Artificial Intelligence (AI) and Large Language Models (LLMs) is beginning to influence the teaching and use of Formal Verification (FV) tools in the educational context.

5.1 Most Commonly Used Formal Methods Tools (RQ1).

The literature search revealed that the most commonly mentioned software tools for undergraduate formal methods education are the Dafny, TLA+, Event-B, Coq, Isabelle/HOL, Frama-C and Z3 systems. Of these, Dafny and TLA+ stand out because they are able to relate concepts from the theory of computer programming to actual software engineering practice.

Table 10. Frequency of Tool Discussion in the Literature Reviewed

Tool	Frequency in reviewed studies
Dafny	7
TLA+	6
Alloy	5
Event-B	5
Coq	5
Isabelle/HOL	5
Frama-C	4
Z3	4

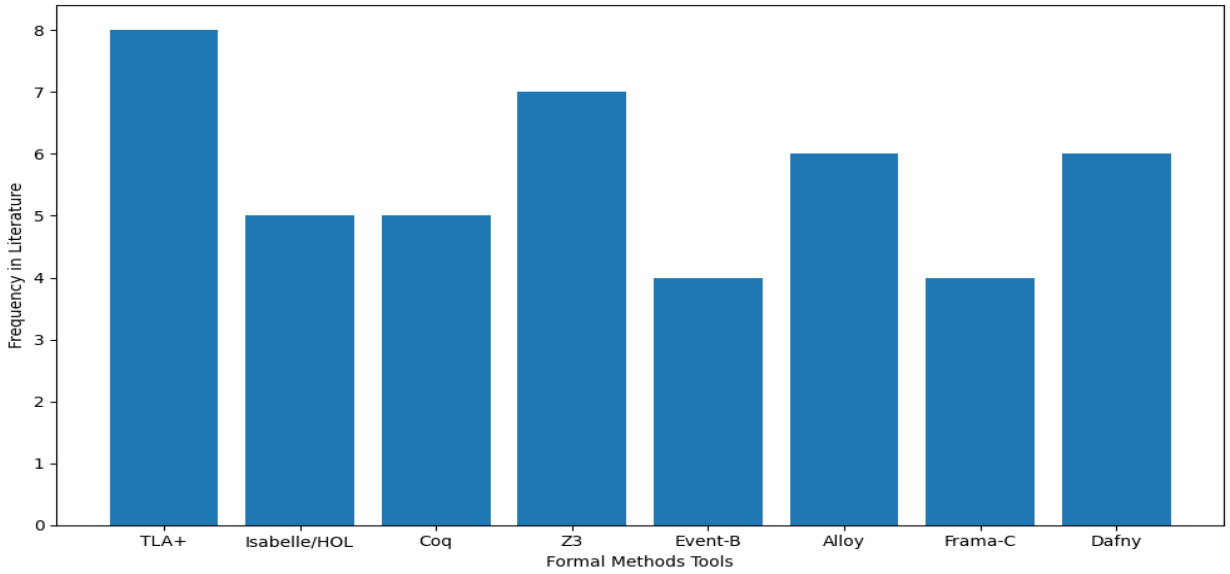


Figure 2. Frequency of Formal Methods Tools in Reviewed Studies

The relative frequency of the major formal methods tools identified in literature reviewed is presented in the figure below: TLA+, Z3, Alloy and Dafny have the most frequent mentions and are becoming increasingly relevant in the world of undergraduate formal methods education and software verification research.

5.2 Educational Advantages and Limitations (RQ2)

The reviewed studies are reporting a number of educational benefits of formal methods tools, including automated tools and interactive feedback. Some tools, however, do have limitations like steep learning curves and mathematical complexity.

Table 11. Educational benefits and drawbacks of Formal Methods Tools

Tool	Educational advantages	Limitations
Dafny	Automated feedback, programming integration, easy syntax	Requires understanding of specifications
Alloy	Lightweight modeling, counterexample generation	Limited theorem-proving capabilities
TLA+	Effective for concurrency and distributed systems	Requires understanding of temporal logic
Event-B	Supports refinement-based modeling	Proof obligations may be difficult for beginners
Coq	Strong theorem-proving capability	Steep mathematical learning curve
Isabelle/HOL	Rigorous higher-order logic proofs	Complex syntax and proof construction
Frama-C	Practical verification of C programs	Requires knowledge of C and ACSL
Z3	Efficient SMT solving	Less beginner-friendly without tool integration

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH
International Peer-Reviewed Research Journal
ISSN (Online): 3106-7980 | ISSN (Print): 3106-7972
Volume 3 • Issue 7 • November 2026

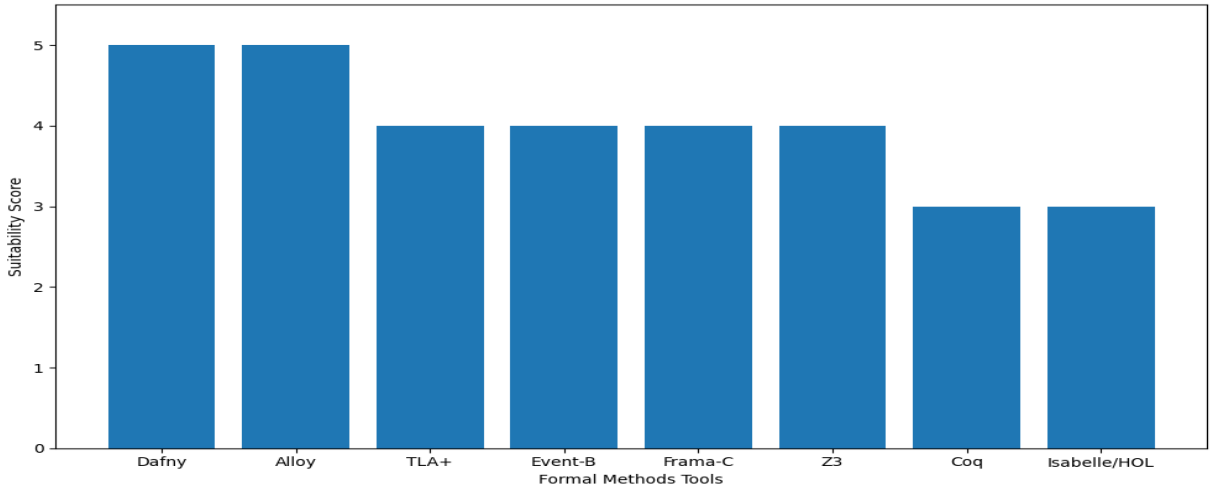


Figure 3. Applicability of a Formal Methods Tool for Educational Purposes.

The figure shows the educational suitability of the main tools for formal methods from the undergraduate course, which are currently available. The two systems that are easiest to use and have the best documentation and automated verification support are Dafny and Alloy; the other two, Coq and Isabelle/HOL, have more powerful theorem provers, but are harder to learn, and have less documentation and poor automated verification support.

5.3 Features that Enhance Student Learning (RQ3)

A number of features appeared multiple times in the literature as being associated with enhanced student learning and engagement in courses on formal methods.

Table 12. Features that contribute to improved student learning

Feature	Educational impact
Automation	Reduces manual proof effort and allows students to focus on concepts.
Immediate feedback	Helps students quickly identify and correct verification errors.
Simple syntax	Lowers the barrier to entry for beginners.
Counterexample generation	Enhances understanding of specification violations.
Programming integration	Connects formal verification with practical software development.
Comprehensive documentation	Supports self-learning and classroom instruction.

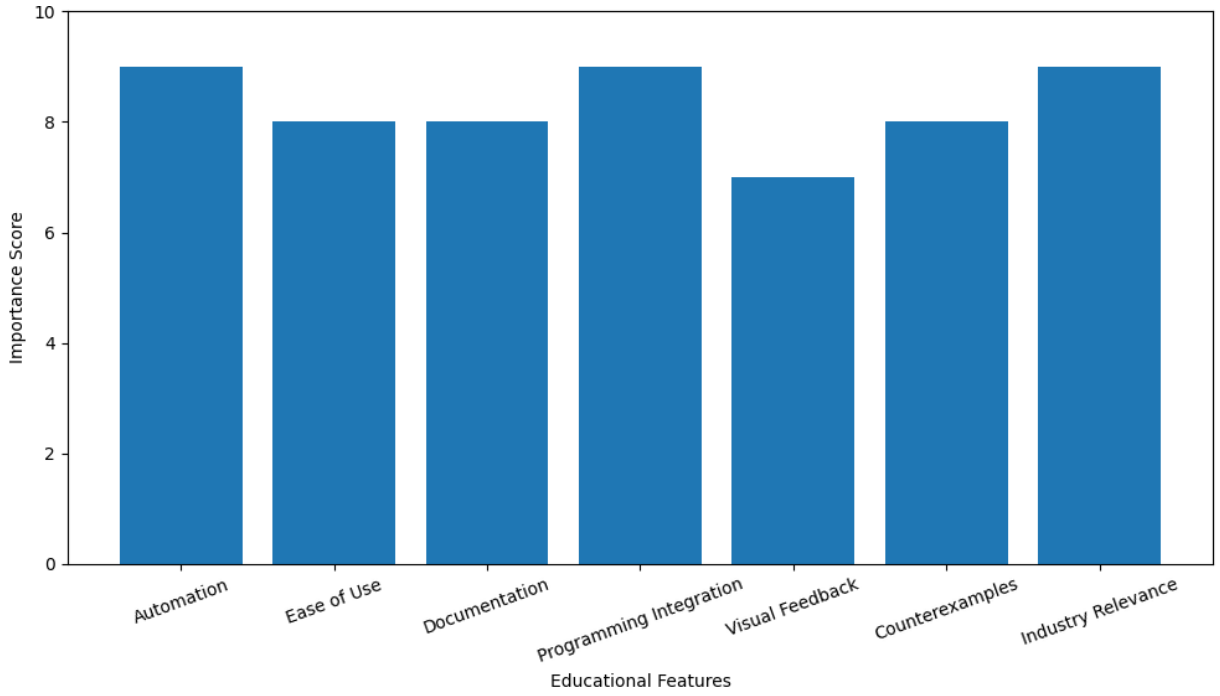


Figure 4. Educational features Contributing to Student Learning

The figure shows relative importance of educational factors that are identified in the reviewed literature. The most important factors for enhancing students' engagement and learning outcomes in undergraduate formal methods courses are automation, programming integration, and industry relevance – with documentation quality, ease of use and counterexample generation also having strong supporting roles.

5.4 Influence of Artificial Intelligence on Formal Methods (RQ4)

Recent research and applications suggest that there is a growing impact of AI and LLMs on the field of formal methods education and software verification. AI tools can help students to create specifications, give proof strategies, and give natural language explanations of a proof failure.

Table 13. Influence of AI on Formal Methods Education

AI application	Educational benefit	Representative references
AI-assisted Dafny synthesis	Automated generation of verified methods	[25]
LLM support for students	Improved verification guidance and feedback	[17]
LLM and formal methods integration	Roadmap for trustworthy AI systems	[26]
AI trends in formal methods	Automation of theorem proving and specification generation	[27]

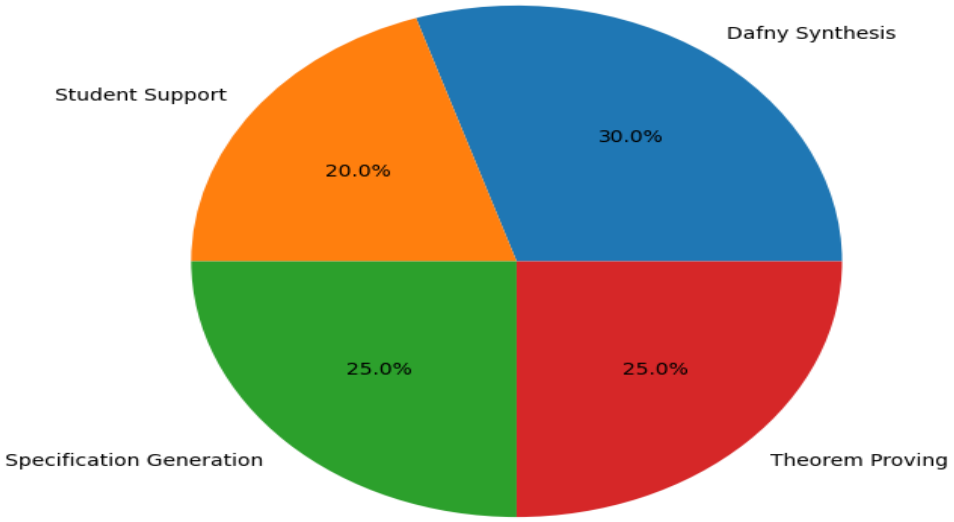


Figure 5. Integration of AI into Formal Methods Courses

The pie chart displays the frequency distribution of the major applications of Artificial Intelligence mentioned in the literature from the reviewed papers, which include Dafny synthesis with the help of AI, student support using LLM, Spec generation, and Support for Theorem-proving.
The collected literature indicates that AI might make formal methods more accessible by decreasing the work for writing the specification and building the proof. However, researchers argue that formal reasoning has yet to be filled and, here, AI can help. Students must be well versed in mathematical concepts that are also relevant to verification to be able to critically evaluate the content created by AI [17], [25– 26]

5.5 Research Gaps and Future Opportunities (RQ5).
Although the literature surveyed revealed several advances, there are still a number of gaps in the teaching of formal methods at undergraduate level.

- There is a need for more long-term studies to evaluate the long-term effect of the use of formal methods tools on student learning.
- Differing evaluation metrics: Different studies employ different measures of effectiveness, making it challenging to compare the effectiveness of tools in different education contexts.
- Further studies are needed on the possible ways of implementing AI-based verification into the curricula without diminishing students' reasoning abilities.
- Scalability of educational tools: Future studies should examine the scalability of the educational tools within large classroom settings and online learning environments.
- The bar chart shows that the major research gaps that were identified in the literature searched are: practical training, high learning curve of verification tools, low integration of Artificial Intelligence, low exposure in the industry and the lack of empirical educational studies.

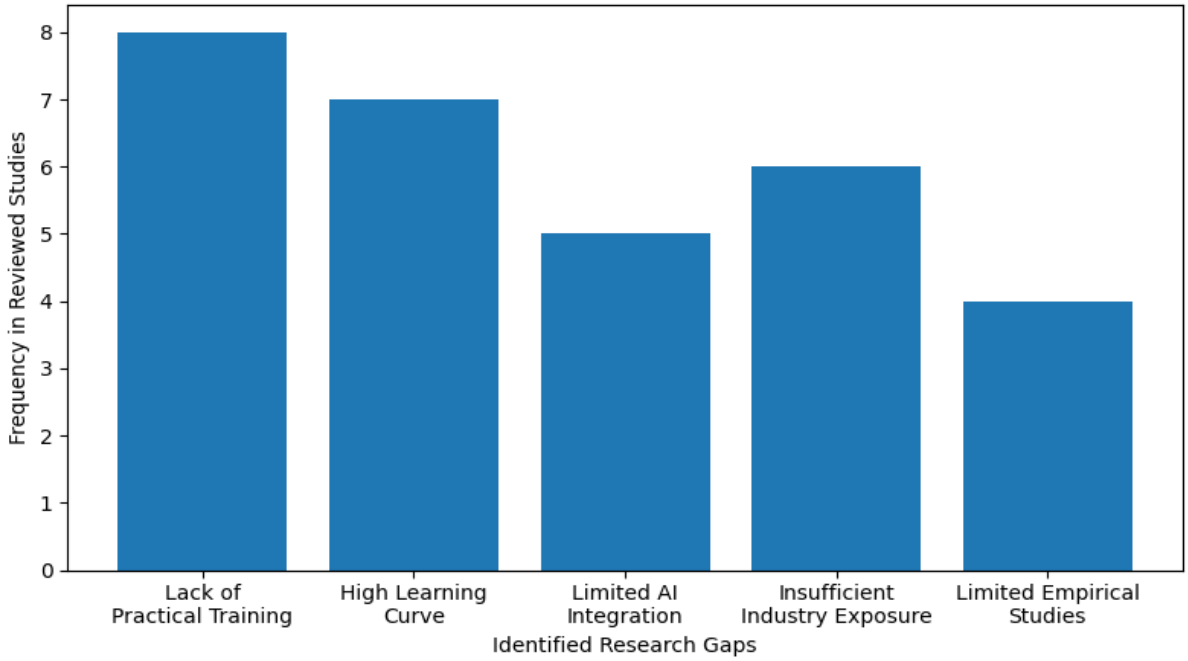


Figure 6. Research Gaps in Undergraduate Formal Methods Education

The bar chart shows that the major research gaps that were identified in the literature searched are: practical training, high learning curve of verification tools, low integration of Artificial Intelligence, low exposure in the industry and the lack of empirical educational studies.

5.6 Discussion of Findings

The outcomes of this review highlight that usability and automation of tools is of key important in achieving successful undergraduate formal methods education. It is very beneficial for introductory courses to use tools like Dafny and Alloy, which have simple syntax and automatic verifier, and provide feedback. These features reduce the stress that students may experience when using formal methods, and enable them to try out the specifications and verification conditions.

At the same time, Isabelle/HOL and Coq are crucial for more advanced courses where sound and rigorous theorem proving and construction of formal proofs are desired. These tend to be more involved to use but can provide more insight into the mathematical underpinning to formal verification.

One of the more prominent trends is the growing importance of AI verification. Students may find learning easier thanks to automated learning support, proof assistance and, with the help of artificial intelligence, the generation of specifications. The system of education should, however, ensure that students develop their own formal reasoning skills and not become so dependent on the automated support.

5.7 Summary of Key Findings

- The tools that were most frequently mentioned in undergraduate formal methods education were identified as Dafny, TLA+, Alloy, Event-B, Coq, Isabelle/HOL, Frama-C, and Z3.
- Based on their usability, practical programming focus and automation, Dafny and Alloy became the most appropriate tools for introductory undergraduate courses.
- Finally, the authors determined that TLA+ and Event-B were very successful in teaching system modeling, concurrency, and refinement-based verification.

• Advanced theorem proving and mathematically rigorous verification continues to be important with Coq and Isabelle/HOL.
• AI-assisted verification is a new trend that can be used to improve learning support, proof assistance, and automated specification generation.

The findings suggest recommendations for software tools to use in undergraduate formal methods courses, based on the evidence. The study's contribution to the field of formal methods education and research in general, and specifically, will be summarized in the following section, Conclusion and Future Work.

6. Conclusion

In this study, gave a comprehensive comparison of software tools in use in undergraduate courses on formal methods. The review covered classical Formal Methods tools: TLA+, Isabelle/HOL, Coq, Z3, Event-B, Alloy, Frama-C and Dafny, along with the emerging AI-assisted verification environments. The analysis concentrated on the capability of verification with the use of formal methods, their suitability for education, support for automation, their relevance in industry, and the contribution of Artificial Intelligence to the teaching of formal methods.

The results show that there is no software solution that can meet the needs of both education and industry. But, Dafny and Alloy offer the best balance of usability, automation, and appropriateness for an introductory undergraduate class. They have simplified syntax, immediate feedback and are easily integrated with programming-oriented learning, which makes them very useful for the students to learn the concepts of formal verification. The tools considered as promising for teaching system modeling, concurrency, and refinement-based development were identified as TLA+ and Event-B, whereas Frama-C and Z3 were regarded as useful for verification tools which are closely related to software engineering practice. Coq and Isabelle/HOL continue to play a crucial role in complex theorem proving and in the mathematically rigorous verification of complex programs.

The study also emphasizes the emerging role of Artificial Intelligence and Large Language Models in the field of formal methods education. Artificial Intelligence (AI) tools for verification could support students by offering specifications and suggesting verification methods and helping them understand the proof requirements. But the literature makes it clear that the use of A.I. must be complimentary of formal reasoning. Students will continue developing their knowledge and understanding of the mathematical principles of verification to critically evaluate software results and assure correctness of the results generated by AI.

There were a number of research gaps identified, such as increased availability of practical training opportunities, enhanced connection between formal methods and software development for industry, further empirical research into the effectiveness of education and more user-friendly tools for verification. Addressing these gaps can lead to better student involvement, make formal methods look more palatable and promote the linking of academic teaching with industrial software verification practice.

To conclude, the paper demonstrates that Dafny and Alloy are the most suitable tools for first year undergraduate formal methods classes, while other tools, such as TLA+, Event-B, Frama-C, Coq and Isabelle/HOL have good potential for special and advanced teaching. In the future, the use of AI for verification in more accessible, engaging and impactful ways to teach formal methods is an exciting prospect.

The future study should be empirical in classroom which is comparative study of the effectiveness of various tools of formal teaching methods obtained by students in learning and student involvement in learning. Further research is required to investigate the long-term education implications of the use of AI-driven verification tools and how they can help with the processes of specification generation, construction of the proof and debugging. A further investigation of the creation of hybrid courses, in which the theoretical part of

formal methods is integrated with actual software engineering projects and projects in the industry, could also be explored. Further research should look into lightweight verification environments that are easy for beginners and students at an undergraduate level, yet still have the rigor of formal verification. Lastly, future studies of the relationship between academic formal methods education and the industrial work in software verification would result in more capable graduates for use in safety-critical, distributed and AI-assisted software systems.

REFERENCES

[1] A. Hall, "Seven myths of formal methods," in IEEE Software, vol. 7, no. 5, pp. 11-19, Sept. 1990, doi: 10.1109/52.57887.

[2] L. Lamport, Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers. Boston, MA, USA: Addison-Wesley, 2002. ISBN: 978-0-321-14306-8.

[3] T. Nipkow, L. C. Paulson, and M. Wenzel, Isabelle/HOL: A Proof Assistant for Higher-Order Logic. Berlin, Germany: Springer, 2002, doi: 10.1007/3-540-45949-9.

[4] Y. Bertot and P. Castéran, Interactive Theorem Proving and Program Development: Coq'Art—The Calculus of Inductive Constructions. Berlin, Germany: Springer, 2004, doi: 10.1007/978-3-662-07964-5.

[5] L. de Moura and N. Bjørner, "Z3: An efficient SMT solver," in Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008), C. R. Ramakrishnan and J. Rehof, Eds., Lecture Notes in Computer Science, vol. 4963. Berlin, Germany: Springer, 2008, pp. 337–340, doi: 10.1007/978-3-540-78800-3_24.

[6] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald, "Formal methods: Practice and experience," ACM Computing Surveys, vol. 41, no. 4, Art. no. 19, Oct. 2009, doi: 10.1145/1592434.1592436.

[7] I. Feinerer and G. Salzer, "A comparison of tools for teaching formal software verification," Formal Aspects of Computing, vol. 21, pp. 293–301, 2009, doi: 10.1007/s00165-008-0084-5.

[8] J.-R. Abrial, Modeling in Event-B: System and Software Engineering. Cambridge, U.K.: Cambridge University Press, 2010, doi: 10.1017/CBO9781139195881.

[9] K. Lausdahl, Enhancing Formal Modelling Tool Support with Increased Automation. Aarhus, Denmark: Department of Engineering, Aarhus University, 2012.

[10] D. Jackson, Software Abstractions: Logic, Language, and Analysis, rev. ed. Cambridge, MA, USA: MIT Press, 2012.

[11] A. Ferrari, A. Fantechi, and S. Gnesi, "Lessons learnt from the adoption of formal model-based development," in NASA Formal Methods, A. E. Goodloe and S. Person, Eds., Lecture Notes in Computer Science, vol. 7226. Berlin, Germany: Springer, 2012, pp. 49–64, doi: 10.1007/978-3-642-28891-3_5.

[12] J.-L. Boulanger, Ed., Formal Methods: Industrial Use from Model to the Code. Hoboken, NJ, USA: John Wiley & Sons, 2013.

[13] N. Kosmatov, V. Prevosto, and J. Signoles, "A lesson on proof of programs with Frama-C," in Lecture Notes in Computer Science. Berlin, Germany: Springer, 2013, doi: 10.1007/978-3-642-38916-0_10.

[14] R. Zhumagambetov, "Teaching formal methods in academia: A systematic literature review," in Integrated Formal Methods, Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2021, doi: 10.1007/978-3-030-71374-4_12.

[15] J. Noble, D. Streader, I. O. Gariano, and M. Samarakoon, "More programming than programming: Teaching formal methods in a software engineering programme," arXiv preprint arXiv:2205.00787, 2022, doi: arXiv:2205.00787v1.

[16] K. Laufer, G. Mertin, and G. K. Thiruvathukal, "Engaging more students in formal methods education: A practical approach using Temporal Logic of Actions," Computer, vol. 57, no. 12, pp. 118–123, Dec. 2024, doi: 10.1109/MC.2024.3462188.

[17] C. Carreira, Á. Silva, A. Abreu, and A. Mendes, "Can large language models help students prove software correctness? An experimental study with Dafny," arXiv preprint arXiv:2506.22370, 2025, doi: 10.48550/arXiv.2506.22370.

[18] J.-M. Bruel, S. Ebersold, F. Galinier, M. Mazzara, A. Naumchev, and B. Meyer, "The role of formalism in system requirements," ACM Computing Surveys, vol. 54, no. 5, Art. no. 93, Jun. 2022, doi: 10.1145/3448975.

[19] T. Kulik, B. Dongol, P. G. Larsen, H. D. Macedo, S. Schneider, P. W. Tran-Jorgensen, and J. Woodcock, "A survey of practical formal methods for security," Formal Aspects of Computing, vol. 34, no. 1, pp. 1–39, 2022, doi: 10.1145/3522582.

[20] M. H. ter Beek and A. Ferrari, "Empirical formal methods: Guidelines for performing empirical studies on formal methods," Software, vol. 1, no. 4, pp. 381–416, 2022, doi: 10.3390/software1040017.

[21] R. Lorch, B. Meng, K. Siu, A. Moitra, M. Durling, S. Paul, S. C. Varanasi, and C. McMillan, "Formal methods in requirements engineering: Survey and future directions," in Proceedings of the 2024 IEEE/ACM 12th International Conference on Formal Methods in Software Engineering (FormaliSE), New York, NY, USA: Association for Computing Machinery, 2024, pp. 88–99, doi: 10.1145/3644033.3644373.

[22] C. B. Jones and M. Thomas, "The development and deployment of formal methods in the UK," Formal Aspects of Computing, vol. 34, no. 1, Art. no. 6, Mar. 2022, doi: 10.1145/3522577.

[23] M. H. ter Beek, R. Chapman, R. Cleaveland, H. Garavel, R. Gu, I. ter Horst, J. J. A. Keiren, T. Lecomte, M. Leuschel, K. Y. Rozier, A. Sampaio, C. Seceleanu, M. Thomas, T. A. C. Willemse, and L. Zhang, "Formal methods in industry," Formal Aspects of Computing, vol. 37, no. 1, Art. no. 7, Mar. 2025, doi: 10.1145/3689374.

[24] G. Bernardi, A. Francalanza, M. Peressotti, and M. R. Mousavi, "Software is infrastructure: Failures, successes, costs, and the case for formal verification," arXiv preprint arXiv:2506.13821, 2025.

[25] M. R. H. Misu, C. V. Lopes, I. Ma, and J. Noble, "Towards AI-assisted synthesis of verified Dafny methods," Proceedings of the ACM on Software Engineering, vol. 1, no. FSE, pp. 812–835, Jul. 2024, doi: 10.1145/3643763.

[26] Y. Zhang, Y. Cai, X. Zuo, X. Luan, et al., "The fusion of large language models and formal methods for trustworthy AI agents: A roadmap," arXiv preprint arXiv:2412.06512, 2024, doi: 10.48550/arXiv.2412.06512.

[27] S. Stock, J. Dunkelau, and A. Mashkoor, "Application of AI to formal methods An analysis of current trends," Empirical Software Engineering, vol. 30, Art. no. 175, 2025, doi: 10.1007/s10664-025-10729-8.

[28] J. P. Bowen, "Formal methods: Whence and whither," in Rigorous Methods in Theory and Practice, A. Raschke, E. Riccobene, K.-D. Schewe, and B. Thalheim, Eds., Lecture Notes in Computer Science, vol. 16580. Cham, Switzerland: Springer, 2026, doi: 10.1007/978-3-032-24494-9_2.

[29] A. Manganaris, V. Giammarino, A. H. Qureshi, and S. Jagannathan, "Formal methods in robot policy learning and verification: A survey on current techniques and future directions," arXiv preprint arXiv:2602.06971, 2026, doi: 10.48550/arXiv.2602.06971.

[30] J. Bowen, M. Fränzle, E.-R. Olderog, D. Bjørner, M. Hansen, H. Langmaack, Z. Liu, and U. Martin, "Experiences from the European ProCoS projects: Provably correct systems," Formal Aspects of Computing, Just Accepted, Mar. 2026, doi: 10.1145/3803555.

International Journal of Advanced Research