An Educational Module for Temporal Features in Alloy 6

Luca Padalino[0009−0009−7357−7058], Francesca Pia Panaccione[0009−0005−8007−963X], Francesco Santambrogio[0009−0001−9677−7142] (✉), Elisabetta Di Nitto[0000−0003−3422−5171], and Matteo Rossi[0000−0002−9193−9560]

Politecnico di Milano, Milano, Italy
{luca.padalino,fracescapia.panaccione,francesco2.santambrogio}@mail.polimi.it
{elisabetta.dinitto,matteo.rossi}@polimi.it

Abstract. As software systems have become increasingly important, teaching Software Engineering students how to develop high-quality software is essential. In this regard, formal modeling and verification are important educational tools that help students in getting an in-depth understanding of software. Nonetheless, formal languages are not straightforward to teach and, therefore, carefully designed materials are needed to convey them. In this paper we focus on Alloy, which is an easy-to-learn formal language equipped with a usable analyser, and we present a complete teaching module that can be used by teachers to support students in learning the temporal constructs defined in its latest version, Alloy 6. The module is designed exploiting active learning methods and is supported by multimedia content. It is openly available and can be reused and tailored to the need of specific courses.

Keywords: Alloy 6, formal methods, modeling language, active learning, teaching module

1 Introduction

With the increasing importance of software systems in our daily lives, Software Engineering students must be provided with the knowledge and skills they need to develop high-quality software, which increasingly relies on formal methods for verifying design, safety, and functionality. In particular, formal specification languages are essential for accurate modeling and verification and, thus, to guide the definition of requirements and to facilitate smooth design, implementation, and testing processes.

Among formal specification languages, Alloy [9] stands out as having one of the simplest syntaxes to read and write, yet with considerable expressive power [15]. In Alloy, a system can be represented using a collection of types defined through *signatures*, each having different fields, governed by rules and

constraints defined as *facts*. Alloy has been widely applied in the literature,[1] and it is supported by a mature tool, the Alloy Analyzer, which enables fully automated system analysis and can reveal weaknesses early on and promote incremental development. These features make Alloy an appealing language to introduce students to the problem of formally specifying real-life systems. Alloy has recently introduced the sixth version[2] where linear temporal logic constructs have been added to the language. This has enriched Alloy's expressive power, but, at the same time, has introduced new teaching and learning challenges.

The contribution of this paper is the development of a teaching module designed to present the temporal features of Alloy 6 to students who have been already introduced to the basic features of Alloy. The module includes slides, Alloy models, videos, and guidelines for instructors. The material is meant to be used within a traditional introductory lecture, a flipped class, an exercise session, and a challenge to encourage students to explore the practical application of the approach. Moreover, the material is organized in several parts that can be used and—possibly—tailored independently from the others. The material is openly available on our Github repository.[3] So far, it has been used, in a shortened format, in the Software Engineering 2 course at Politecnico di Milano in the Academic Year 2023-2024. As a preliminary evaluation, we report on this experience commenting on the performance of students during the exam both in terms of achieved score and in terms of types of errors made. This analysis shows that the module is potentially effective.

The paper is structured as follows: Section 2 presents the state of the art on the preparation of teaching modules and on Alloy. Section 3 presents an overview of the temporal features of Alloy 6. Section 4 presents the proposed teaching module and Section 5 evaluates it. Finally, Section 6 presents a critical discussion and concludes the paper.

2 State of the art

Teaching modules A teaching module is a significant, highly homogeneous and unified part of a planned disciplinary program. One of the main characteristics of a teaching module is having well-defined and verifiable learning objectives which not only enhance student engagement and motivation [18], but also assist instructors in designing assessments and selecting the content of the lectures effectively.[4] To achieve such goals, educators must consider how to help students take advantage of the contents by identifying effective teaching materials and strategies.

With the advent of educational technology, universities are increasingly integrating instructional multimedia into course delivery as part of the teaching materials, such as slides, videos, or online quizzes, providing several advantages

[1] alloytools.org/applications.html
[2] alloytools.org/alloy6.html
[3] github.com/lucapada/ResearchProjectAlloy6
[4] cteresources.bc.edu/documentation/learning-objectives

including increased access to content, personalized learning opportunities, and greater student engagement [19]. Multimedia-enhanced learning environments can foster student motivation and facilitate problem-solving skills through self-exploration and collaboration [12]. In the context of teaching modules, teaching strategies aim to overcome the limitations of traditional classroom teaching, where students often passively consume information, and leverage active learning instead, which fosters students' engagement and encourages them to take responsibility for their learning [11].

Howell [8] discussed student experiences and perceptions of an interdisciplinary social science course, concluding that over 90% of respondents agreed that in-class active learning exercises made the classes more engaging and the material more memorable than usual. Active learning includes projects, problem-solving tasks, and team assignments; it offers numerous benefits, such as receiving immediate feedback, building confidence, and promoting cognitive development [7]. Accordingly, it is suitable for disciplines with practical aspects like software engineering and specification languages in particular. The rest of this section reviews some of the most useful active learning strategies explored and tested in the literature.

One of the most used and simplest active learning strategies is questioning, which consists in having teachers pose proper questions to learners. They encourage discussion, argumentation, and the expression of opinions and alternative views. When used effectively, questioning provides immediate feedback about students' understanding, supports informal assessment, and evaluates teaching strategies' impact [5]. Questioning can be seen as a form of feedback for the students who can measure their knowledge against the asked questions and required answers. In general, feedback is an active learning strategy that informs a student or a teacher about their performance and is acknowledged as an essential element for improving the students' learning process [6]. Feedback redirects or refocuses students' actions so that they can align effort and activity toward a clear outcome [5].

Besides teaching theoretical aspects and involving students with questioning and feedback, examples may help them understand better the concepts of the lectures. Worked examples, in particular, aid initial cognitive skill acquisition by presenting formulated problems, solution steps, and final solutions [17]. Studying worked examples is effective for teaching complex problem-solving skills as it provides expert mental models to novices and reduces cognitive load, facilitating skill acquisition [20]. Once worked-examples are assimilated, students can push toward more complex real-world problems. Problem-based learning is the instructional approach where learning occurs through the process of solving such problems [2]. Allen et al. concluded that such a method enhances the affective domain of student learning, improves student performance on complex tasks, and fosters better retention of knowledge [1]. This strategy exploits projects that should be as complete as real-world instances so that students can experience the whole process that is likely to be seen in future working environments, including cooperation with other peers.

In general, as stated by Laal and Ghodsi [10], there are several benefits brought by collaborative learning, like enhancing interactions, learner autonomy, teamwork, and problem-solving skills within a group, as members exchange ideas and collectively build shared understanding. Integrating communication, interaction, and cooperation skills among team members is crucial for successful software development. This innovative approach prepares students for future industrial settings by emphasizing the significance of social aspects in software development.

In the development of our teaching module we have used and adapted to the specific context of Alloy learning all the strategies presented in this section. We discuss the implementation of these features in Section 4.

Teaching material for Alloy 6 Alloy has been extensively integrated into undergraduate and graduate courses worldwide. Examples of courses adopting it can be found in the Formal Methods Europe (FME) database[5] of formal methods courses. From an analysis of the syllabus and, where possible, the material, it results that most of the listed courses do not focus on the temporal features offered by Alloy 6. Notable exceptions are the Formal Methods for Software Engineering course,[6] taught at University of Minho (Portugal) by the group that contributed to the development of Alloy 6 (and the Alloy4Fun web application [13]), the Logic for Systems course at the Brown University,[7] and the University of Iowa's Formal Methods in Software Engineering course.[8].

In all the cases we could analyse, the offered teaching material was not fully self-contained and organized to be reused in other classes. Also, the presented examples were limited in number and relatively simple. As such, we concluded that a comprehensive teaching module addressing the complexities of Alloy's temporal features can be useful to aid Software Engineering teachers in incorporating such features in their courses and to help students in mastering their application in significant cases.

3 Overview of Alloy's temporal features

Alloy 6 introduces the concept of *mutable* signatures and fields, enabling users to express their temporal evolution using operators derived from Linear Temporal Logic (LTL, [16]). In a nutshell, a mutable signature captures a set whose members can change over (discrete) time; similarly, if a signature (not necessarily a mutable one) includes a mutable field, then an instance of the signature could be such that the value of the field changes over time. Mutable signatures and fields are identified through the `var` keyword. Alloy uses a discrete notion of time, so for each time instant $i \in \mathbb{N}$ the instance of a mutable element can be different

[5] fme-teaching.github.io/courses
[6] haslab.github.io/MFES
[7] csci1710.github.io/2024
[8] homepage.cs.uiowa.edu/∼tinelli/classes/181/Fall23/index.shtml

(signatures and fields that are not marked as `var`, instead, have the same value for each time instant).

Alloy includes the classic LTL temporal operators (including their past counterparts), such as `always`, `eventually`, `until`. The LTL "next" operator is called `after` in Alloy, and `before` is its past counterpart. For example, the following declarations define that signature `S` is mutable, initially it does not contain any elements, but at some point it will contain 3.

```
var sig S{}
fact { #S = 0 and eventually #S = 3 }
```

Alloy 6 can perform both *bounded* and *unbounded* model checking of temporal specifications (the latter through the nuXmv formal verification tool [4]). If a bounded approach is used [3], the time horizon (i.e., the maximum length of the traces to be explored in search for a loop) must be provided through the `steps` keyword.

Alloy 6 also features an improved visualizer tool which displays traces in a user-friendly way. More precisely, the visualization pane is split in two parts showing the model instance in consecutive states. The visualizer allows users to explore traces (move along the current trace, explore a new one, etc.) through suitable commands.

4 Organization of an Alloy 6 Teaching Module

The developed teaching module incorporates various teaching strategies (see Section 2) and focuses on the temporal features introduced in Alloy 6. It assumes that students are already familiar with the core notions of the Alloy modeling language (signatures, facts, predicates, etc.), but not with its temporal features. It includes theoretical and exercise lectures, within which a timeline was defined for each macro-topic to be presented. To experiment with alternatives to classic lectures, additional activities (self-assessment quizzes, a flipped classroom, a challenge) were added, which function as tools to assess comprehension related to learning objectives. Additionally, a comprehensive guide with an accompanying video was developed to assist instructors in understanding how to use the module effectively. The organization of the teaching module is presented in Figure 1.

First theoretical lecture In the first theoretical lecture, the focus is on comparing how time-varying systems are handled in Alloy 5 vs. Alloy 6. The aim is to demonstrate the enhanced effectiveness of Alloy 6 in this regard, thanks to the introduction of new logic operators and keywords. By the end of this lesson, students should grasp how Alloy 5 addresses dynamic modeling, recognize its limitations and understand why the new features in Alloy 6 are necessary. The lecture begins with an overview of the learning objectives, setting the stage for what students will gain from the session. Then, the distinction between the static and dynamic world in Alloy is explored, providing clarity on the significance of

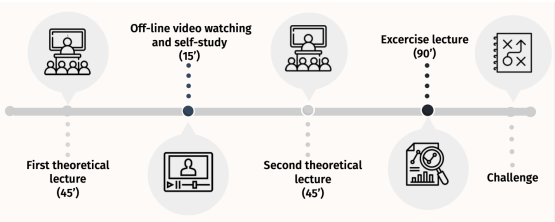


Fig. 1: Temporal organization of the teaching module.

these terms within the context of modeling. The bulk of the lecture focuses on examining how dynamic models are represented in Alloy 5, which implies the presentation of the Alloy ordering package and of the `Time` signature and its limitations. Through examples and explanations, students gain insights into the challenges faced when modeling dynamic systems using Alloy 5. Then, the focus shifts to Alloy 6, introducing the `var` keyword and LTL concepts and operators. These additions enhance Alloy's capabilities in handling temporal aspects and introduce mutability, addressing some of the shortcomings observed in Alloy 5. Throughout the lecture, the emphasis is on providing clear explanations and practical examples to aid understanding.

Video watching and self-studying Video watching and self-studying help students learn the topics related to temporal operators introduced in Alloy 6. Students should have attended the first lecture, which introduces the features used in the video. The teacher, in the recorded video, unidirectionally explains the concepts, leaving the opportunity to raise doubts and gather opinions to the next lesson, through feedback and questioning. The recording focuses on temporal connectives—both future and past—that a user can exploit to define predicates, facts, and assertions. All connectives are presented in the same way, always following the same pattern: syntactic definition, semantics, and application example. As teaching material, the teacher provides the video and the slide set containing the whole content of the video.

Second theoretical lecture In the second lesson, the objectives are two-fold: firstly, through the flipped class approach, to assess students' understanding of the material covered in the first lesson and in the video; secondly, to continue exploring the new temporal features introduced in Alloy. Students will have the opportunity to evaluate their comprehension and the effectiveness of their study methods through quizzes administered partly during class time and partly as take-home assignments. By the end of the class, students will gain a comprehensive understanding of Alloy's temporal features and capabilities. They will be able to use temporal connectives and address related arguments.

Exercise lecture During the exercise lecture, students will engage in hands-on activities designed to reinforce their understanding of Alloy's temporal concepts. Through a worked-example approach, the lecturer will lead students through practical coding exercises aimed at tangibly applying theoretical knowledge. This interactive session will provide a valuable opportunity for students to deepen their comprehension and develop essential problem-solving skills. To complete the exercises, students must have attended the lectures covering Alloy's new features, watched the related video on connectives, and installed the latest version of the Alloy tool. The exercise session comprises the following three exercises.

Exercise 1: Concurrent Communications in Distributed Systems This exercise delves into the concept of parallel system operations, focusing on modeling communication in distributed systems. Students will have to define mutable signatures and establish facts defining the conditions that are perpetually true as well as the ones that describe the way the system evolves over time. By modeling scenarios such as message transmission and reception, students will gain insight into handling dynamic systems within the Alloy framework.

Exercise 2: Travel (Interrail) In this exercise, students will tackle the modeling of travel scenarios using Alloy. They will define signatures representing entities involved in travel, such as cities and travelers, and create static and dynamic models of travel itineraries. By modeling a person's journey between cities and defining completion criteria, students will gain proficiency in modeling dynamic real-world processes.

Exercise 3: Mailbox The final exercise focuses on modeling a mailbox system with dynamic behavior. Students will define a model for messages that can be deleted and restored from a recycle bin, with their locations changing over time.

Throughout the exercise session, students will actively engage with the material, applying theoretical concepts to practical scenarios. By completing these exercises, students will enhance their understanding of Alloy's capabilities and develop the skills necessary for effective modeling and analysis.

Challenge The extra activity offered by the module is a challenge where students can apply what they have learned in solving a particular specification within a two-week time frame. To tackle the challenge and solve the exercise, students must have previously (i) attended the previous lectures, (ii) watched the video related to connectives, which are extensively used when defining predicates, facts, and assertions, (iii) attended the exercise session and (iv) installed the latest version of the Alloy tool. The challenge presents students with a real and current problem that may not have a single solution. Problem-solving, a skill greatly developed by the challenge, encourages each individual to propose a personal solution based on what they have learned. In this proposed challenge, the strategy of collaborative learning is employed, with students working together in small groups (two or three students) to develop skills related to teamwork and problem-solving. The proposed theme for the challenge is Software-Defined Networks, inspired by [14], which are increasingly important in cloud computing

Module's component	**Teaching strategy**
First theoretical lecture Second theoretical lecture	Feedback Questioning Learning goals setting
Video watching and Self-study	Multimedia teaching material usage
Exercise lecture	Questioning Feedback Worked example Problem-based learning
Challenge	Problem-based learning Collaborative learning

Table 1: The various components of a teaching module and the associated teaching strategies and goals within the module.

and network architectures to facilitate their administration, configuration and monitoring, and to improve their performance.

For what concerns the teaching material, the teacher provides the slide set containing the outline of the challenge and how to deliver it. At the end of the challenge—thus, at the end of the proposed module—students will be able to face situations in which they are required to model complex systems by working in a group in a very practical and professional setting.

Each component of the teaching module was carefully selected to maximize its effectiveness, through the active learning strategies described in Section 2. Table 1 outlines the various components of the module, detailing the specific teaching strategies and goals associated with each part.

5 Evaluation

In this section, we present a preliminary evaluation based on the module's adaptation and usage within the Software Engineering 2 (SE2) course at Politecnico di Milano. The course is mandatory for students enrolled in the Master in Computer Science and Engineering. It is offered during the Fall semester (September-December) to more than 600 students divided in three classes. Most of the students enrolled in the SE2 course have a background (typically a Bachelor's degree) in computer science and engineering, though there are also students of other engineering disciplines such as Telecommunications Engineering and even Mechanical Engineering.

Alloy was presented as a specification language for model definition and verification at the requirement analysis and design levels, and its sixth version was introduced in the course during the 2023-2024 academic year. Due to time constraints, the module was condensed and adapted to fit in a 1-hour lecture, plus a half-hour exercise session. Hence, a full comparison of the differences between the new temporal features and the approaches previously available in Alloy (see Section 3) was not possible. The condensed lecture, instead, focused on how

the new temporal features of Alloy can capture, in a natural way, the change in the state of a system (i.e., in its mutable parts) when certain operations are performed. In addition, it provided, through examples, an overview of the most commonly used temporal operators (`after`, `aways`, etc.), and showed how they can be used to describe constraints on the evolution of mutable features.

The exercise session provided further examples of definitions of temporal properties (including assertions to be checked) through the development of a complete, albeit small, specification of a message handling system, a variation of the mailbox exercise of the teaching module (see Section 4). In a way, the exercise session replaced the challenge, which was created to drive students to solve (possibly through group work) a particularly long and complex problem. The introduction of this new material did not result in an increase of the number of hours dedicated to Alloy in the course. To make optimal use of the available class time, we showed how the new temporal features in Alloy 6 allowed for a clearer description of the relations between states before and after the execution of operations with respect to Alloy 5 (where such relations are described through the introduction of specific atoms capturing different instances of the state).

In the context of the SE2 course, the negative impact, teaching-wise, of dropping the challenge was lessened by how assessments are carried out. A significant portion of the students enrolled in the course elect to acquire part of the course credits through the development of a project focusing on the requirements analysis and design activities related to a given application. As part of these activities, students are asked to create a working Alloy specification, which is another form of challenge. Finally, the vast majority of students enrolled in the SE2 course complete the course credits by taking a written exam, which includes an Alloy-specific question requiring students to define a small number of signatures and properties.

To evaluate the effectiveness of teaching the temporal aspects of Alloy using the material derived from our module, we have analyzed only the performance of students during the written exam. This choice was driven by the fact that written exams are individual, thus more objective and more comparable across different course editions. We performed two types of analyses. First, we collected the scores achieved by students in the Alloy question and compared them with those obtained in the previous 4 years (in which temporal features were not presented). Then, we selected a subset of 50 exams from this year and we manually examined them to identify the most common errors. This allowed us to highlight the aspects where students are weaker and require more support. For each academic year, scores were manually assigned by the same two course instructors (who each has been teaching the SE2 course for the past 7 years). Each instructor graded roughly half of the exams. The analysis carried out for this work does not include feedback provided by students, which we plan to systematically collect in future editions of the SE2 course.

Analysis of students' scores We considered the scores of the Alloy question of the exams taken by students in the academic years from 2019-2020 to 2022-2023.

	19-20	**20-21**	**21-22**	**22-23**	**23-24**	**no time op.**	**time op.**
#stud	130	143	153	125	142	551	142
Average	72%	60%	73%	72%	68%	69%	68%

Table 2: Number of students taking the exam (February call only) and average score expressed in terms of percentage of correctness.

We compared them with the scores of the exams taken in the current academic year, in which the Alloy question explicitly required to use the new temporal features. Our hypothesis is that the grade distribution is the same across years, even after introducing the new Alloy 6 features, thanks to the adoption and tailoring of the teaching module.

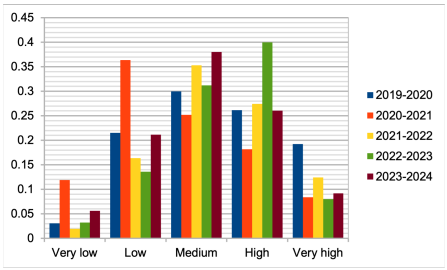
Table 2 provides an overview of the number of students considered in the analysis and the corresponding average scores. Notice that, at Politecnico di Milano, for each course there are 5 exam calls spread across the academic year (2 in January-February, 2 in June-July, and 1 in September), and students can take the exam—possibly multiple times—in any of the 5 calls. The analysis was carried out focusing on 2 of the 3 classes (totalling more than 400 students) in which students are divided. For each considered academic year, only the February call was taken into account; for the SE2 course this is typically the second-most attended call (after the January one), since the course is taught in the Fall semester.

The first columns of Table 2 focus on the five academic years under consideration and show that the number of students taking the exam every year is statistically relevant and relatively stable between 125 and 153 and that the average score, computed in terms of percentage of correctness, is between 60% and 73%, with the lowest score obtained in the year 2020-2021. The sixth column provides the cumulative results obtained in the first four considered academic years and shows that the average score obtained this year (68%) is not significantly different from that of previous years (69%). In Figure 2a scores are organized in the six categories defined in Figure 2b. The distribution of scores in Figure 2a highlights the bad performance obtained in the year 2020-2021, with 22% of low scores; also, it shows that this year's performance is in line with that of previous years, with most scores in the "medium" category.

In conclusion, our hypothesis that this year's grade distribution is similar to that of previous years can be considered confirmed, even though the observation of future academic years is needed to consolidate it.

Analysis of students' answers We evaluated the exam held in February 2024. The text of the question had two parts, as shown below:

Consider a system to monitor the accesses of vehicles to the center of a city (think "Area C" in Milano). The system detects vehicles entering the City Center Area (CCA for short) using "gates" installed on the streets through which vehicles can access the CCA. Each time a vehicle goes through a gate, the gate reads the license plate of the vehicle and provides the system with the corresponding



(a) Scores classification.

Category	From	To
Very low	0%	30%
Low	31%	59%
Medium	60%	75%
High	76%	90%
Very High	91%	100%

(b) Definition of categories.

Fig. 2: Scores per year, organized by categories.

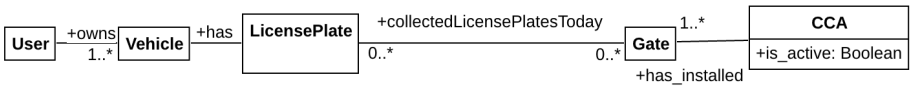


Fig. 3: UML class diagram provided in the exercise.

information (license plate number and time of passage of the vehicle). The CCA is active only until a certain time of the day (e.g., until 7pm). You are asked to use the features of Alloy 6 to capture some features of the system, focusing on the handling of the accesses for a single day. In particular, consider the (simplified) domain model for the system shown in Figure 3 and represented through a UML Class Diagram.

(**Q1**): *Define suitable signatures and constraints to capture the domain model shown above. In particular, identify the elements of the model that are mutable.*

(**Q2**): *Define a fact* `activeCCAdef` *that states that the CCA is initially active, then at some point it must become inactive; after becoming inactive, the CAA cannot become active again.*

The proposed solution to the two questions is shown in Listing 1.1. To assess the correctness of the solutions, the following elements were considered: the correspondence between the Alloy model and the UML class diagram, the appropriate identification of mutable elements, and the correct definition of the `activeCCAdef` fact.

Listing 1.1: Proposed solution to the exam questions.

```
sig User { owns : some Vehicle }
sig Vehicle {
    has : one LicensePlate
}{ one u : User | this in u.owns }
sig LicensePlate {}{ one v : Vehicle | v.has = this }
sig Gate {
    var collectedLicensePlatesDay : set LicensePlate
}{ one cca : CCA | this in cca.has_installed }
```

```
sig CCA {
    has_installed : some Gate
    var is_active : Boolean
}
fact activeCCAdef {
    all cca : CCA | cca.is_active = True
        and eventually cca.is_active = False
        and always (cca.is_active = False implies
            always cca.is_active = False)
}
```

We manually analyzed the answers produced by 50 students. This sample was selected from the batch of 142 exams graded by the course instructors, using the following procedure: (i) the exams of those students who obtained a score lower than or equal to 50% of the total were eliminated, as the nature of the errors in these cases certainly went beyond simply the usage of temporal operators; (ii) for each score between 50% (excluded) and 100% (with increments of 10%), a roughly equal number of exams per score was randomly extracted. The answers were finally made anonymous to allow for the sharing of the exams with people other than the course instructors.

We analyzed the exams to identify the most common errors and the corresponding topics in the teaching material, we investigated the reasons why students could have missed those concepts, and we checked whether the proposed teaching module offers material in that regard and could help. The analysis evidenced the following three main categories of errors.

Mutable signature ***`var`***. Fifty percent of the students (25 over 50) did not consider that the set of license plate numbers read when passing through the gate (attribute `collectedLicensePlatesDay` in Listing1.1) and the status of the CCA (attribute `is_active`) should be defined as mutable using the keyword `var`. This is an error that the majority of students made in only one of the two attributes to be defined as mutable. Moreover, this is the only kind of error made by many of the students who correctly defined the fact, thus suggesting that the main cause for the error was a lack of focus. However, this also highlighted that the teaching material should stress the importance of the `var` keyword, which is the starting point from time-dependent specifications. The number of examples and remarks made during theoretical lectures could be increased, especially when presenting temporal connectives. In this respect, the original module is more complete compared to the tailored one, as it presents multiple examples that focus, as a first step, on the importance of identifying what is mutable and what is not, and only after this define facts that use temporal connectives.

`Time` *signature definition.* Half of the students (25 over 50) created an ad-hoc `Time` signature as the basis for time-dependent properties. This is similar to how such properties are introduced in Alloy 5, a method made obsolete by the new features of Alloy 6. The following is an example found in one exam:

```
sig Time {hour: one Int}{ hour > 0 and hour < 24 }
```

```
sig Information {  plate: one LicensePlate,
                   time: one Time  }
```

Most likely, students who followed this approach took the course in the previous academic year and did not take care to study the new aspects introduced this year. Indeed, the tailored material does not include a history of the evolution of the representation of time-dependent features from the previous versions of Alloy (5 and before) to the latest one; this is instead available in the complete module, which explains how defining a `Time` signature is no longer necessary in Alloy 6 thanks to the introduction of mutable features.

Usage of temporal connectives and definition of the fact. Seventy-eight percent of students (39 over 50) made conceptual, syntactical, and logical errors regarding the definition of facts. More than half used an incorrect temporal connective. Given the heterogeneity of errors, it was particularly complex to find a correlation among all the errors and a single reason. Not considering syntactic errors—which typically originate from a superficial study of the language syntax—we separated conceptual errors concerning the usage of first-order logic constructs from errors inherent to the usage of temporal connectives. For example, the errors in the following definition of fact `activeCCAdef` are mostly first-order logic-related and show that the student did not grasp the semantics of the `all` and `some` quantifiers. Having misunderstood these basic aspects, the student wrote a fact that is also incorrect from the temporal viewpoint.

```
fact activeCCAdef {
  all c: CCA | c.is_active
  some c: CCA | not c.is_active
  all c: CCA, g: Gate | not c.is_active implies
                        c'.has_installed = g and c'.is_active }
```

In the following example, the student used the temporal operators in a correct way, but missed the quantification on variable `c`:

```
fact activeCCAdef {
  always(c: CCA | c.is_active in True implies
                              eventually(c.is_active in False))
  and always (c: CCA | c.is_active in False implies
                                always(c.is_active in False)) }
```

Students might have misunderstood the meaning of each temporal connective and this may have impacted on the definition of the fact. In this regard, the module offers examples and quizzes for each temporal connective, with exercises focusing on why a certain connective is used and others should not. The examples presented in the theoretical lectures also help to better integrate the novelty of the temporal connectives with the existing logical constructs.

Threats to validity The experiments and the analysis presented in this paper are susceptible to the following threats to validity, which we plan to overcome in the future.

The evaluation of the teaching module from the instructors' perspective has been performed only internally, within the same group that has originated the module. A more in-depth evaluation involving instructors from multiple different academic institutions will be targeted for the next academic years.

The evaluation of the teaching module from the students' perspective is indirect, since the class that was the object of the evaluation of Section 5 was not directly exposed to the complete teaching module, but only to its tailoring (even though the whole module was available to them). This threat is mitigated by the fact that, since the complete module is more detailed (especially from the practical viewpoint) compared to the tailored one, we expect the students' learning experience to actually improve through exposure to the complete module.

Finally, the analysis has considered a group of students who attend the same study course, hence have a homogeneous background. It could be beneficial to carry out experiments with a more diverse student population, though this might be difficult to achieve.

6 Discussion and Conclusion

For software engineering students, understanding formal specification languages for requirements modeling is very important, though far from trivial. Alloy is an educationally accessible modeling language that has recently introduced important new features that revolve around an integrated concept of time and do not require the use of external modules; they include new keywords for marking objects and their properties as mutable, new operators for expressing properties related to past and future time instants, and a new visualizer tool.

We proposed a module to effectively teach students the new Alloy features by combining different teaching strategies, such as frontal lectures and flipped classrooms, to stimulate students' interest and help them better understand the language.

Additional research is needed to further validate the proposed approach. Experiments could be carried out to test the effectiveness of the teaching module objectively. In particular, we plan to compare different teaching methods by dividing a class into two groups: one group testing the module, while the other uses a different teaching approach.

Quizzes, exercises, and the challenge can be used to build an up-to-date picture of the state of student learning. Thanks to them, one could observe the improvement achieved by the class over time in terms of: (i) timeliness of students' responses, (ii) obtained scores, and (iii) quality of answers in relation to the used teaching method. In particular, the first two factors can be evaluated through quizzes and quick exercises, while the third is assessed through activities such as drills, challenges and written exams. The quality of answers, in particular, concerns how well students manage to tackle a complex problem from scratch, without teacher guidance.

As discussed in Section 5, to increase the objectivity of the analysis, we plan to carry out experiments in classes taught by different teachers and taken by students with different backgrounds.

We believe that our approach to teaching Alloy can be an important starting point for improving the accessibility of formal specification languages in software engineering; however, further research is needed to confirm the validity of our solution. Depending on the results of the validation, the proposed teaching module can be modified and fixed to be as effective as possible.

Acknowledgements We are very grateful to all students involved in the SE2 course and to Alessandra Viale, who helped us with the preparation of the exams for the detailed analysis.

References

1. Allen, D. E., Donham, R. S., and Bernhardt, S. A. Problem-based learning. *New directions for teaching and learning 2011*, 128 (2011), 21–29.
2. Barrows, H. The essentials of problem-based learning. *Journal of Dental Education 62*, 9 (1998), 630–633.
3. Biere, A., Cimatti, A., Clarke, E., and Zhu, Y. Symbolic Model Checking without BDDs. In *Tools and Algorithms for the Construction and Analysis of Systems*, vol. 1579 of *Lecture Notes in Computer Science*. 1999, pp. 193–207.
4. Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., and Tonetta, S. The nuXmv Symbolic Model Checker. In *Proc. of CAV* (2014), vol. 8559 of *LNCS*, pp. 334–342.
5. Dept. of Education and Training, State of Victoria. High impact teaching strategies: excellence in teaching and learning. ISBN: 978-0-7594-0820-3, 2020.
6. Ferguson, P. Student perceptions of quality feedback in teacher education. *Assessment & evaluation in higher education 36*, 1 (2011), 51–62.
7. Ghilay, Y., and Ghilay, R. Tbal: Technology-based active learning in higher education. *Journal of Education and Learning 4* (9 2015).
8. Howell, R. A. Engaging students in education for sustainable development: The benefits of active learning, reflective practices and flipped classroom pedagogies. *Journal of Cleaner Production 325* (11 2021).
9. Jackson, D. *Software Abstractions: logic, language, and analysis*. MIT press, 2012.
10. Laal, M., and Ghodsi, S. M. Benefits of collaborative learning. *Procedia-social and behavioral sciences 31* (2012), 486–490.
11. Li, Y. W. Transforming conventional teaching classroom to learner-centred teaching classroom using multimedia-mediated learning module. *International Journal of Information and Education Technology 6* (2016), 105–112.
12. Liu, M., Horton, L., Lee, J., Kang, J., Rosenblum, J., O'Hair, M., and Lu, C.-W. Creating a multimedia enhanced problem-based learning environment for middle school science: Voices from the developers. *Interdisciplinary Journal of Problem-Based Learning 8* (3 2014).
13. Macedo, N., Cunha, A., Pereira, J., Carvalho, R., Silva, R., Paiva, A. C., Sozinho Ramalho, M., and Silva, D. Experiences on teaching alloy with an automated assessment platform. *Science of Computer Programming 211* (2021), 102690.

14. María-Del-Mar Gallardo, L. P. Modelling and specifying software systems with alloy * (tutorial).
15. Moreira, R. M., and Paiva, A. C. A novel approach using alloy in domain-specific language engineering. In *2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)* (2015), IEEE, pp. 157–164.
16. Pnueli, A. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)* (1977), IEEE, pp. 46–57.
17. Renkl, A. *The Worked-Out Examples principle in Multimedia Learning.* 01 2005, pp. 229–245.
18. Seidel, T., Rimmele, R., and Prenzel, M. Clarity and coherence of lesson goals as a scaffold for student learning. *Learning and Instruction - LEARN INSTR 15* (12 2005), 539–556.
19. Smith, A. R., Cavanaugh, C., and Moore, W. A. Instructional multimedia: An investigation of student and instructor attitudes and student study behavior. *BMC Medical Education 11* (2011).
20. van Merrienboer, J. *Training complex cognitive skills: A four-component instructional design model for technical training.* Educational Technology Publications, 1997.