An Educational Module for Alloy 6

Luca Padalino
Computer Science and Engineering
Politecnico di Milano
luca.padalino@mail.polimi.it

Francesca Pia Panaccione
Computer Science and Engineering
Politecnico di Milano
francescapia.panaccione@mail.polimi.it

Francesco Santambrogio
Computer Science and Engineering
Politecnico di Milano
francesco2.santambrogio@mail.polimi.it

***Abstract*—As software systems have become increasingly important, teaching Software Engineering students how to develop high-quality software is essential. In particular, one of the main goals of software engineers is verifying that developed software meets specific requirements. In this regard, a precise and simple modeling language like Alloy can be useful, especially with its temporal features introduced in the sixth version. Nonetheless, such languages concern students due to their abstract concepts, therefore a comprehensive and effective teaching method is pivotal for them. Accordingly, we present our work on a teaching module for Alloy 6 that promotes an active learning environment leading to specific learning goals with the aid of multimedia.**

***Index Terms*—Alloy 6, formal methods, modeling language, specification language, software requirements, educational strategy, teaching module**

I. Introduction

With the increasing importance of software systems in our daily lives, it is essential that Software Engineering students are provided with the knowledge and skills they need to develop high-quality software. In this regard, formal methods can be useful since they are exploited to verify that a software system satisfies design, safety, and functional properties. As a matter of fact, one of the main goals of Software Engineering is to develop good software that can be delivered only within specific requirements. For this purpose, specification languages are essential since they enable software engineers to formally specify the requirements and constraints of a software system accurately, making it easier to design, implement, and test.

Among the several specification languages, Alloy [1] stands out as one of the most precise and concise [2]. A system can be represented in Alloy using a collection of types called signatures, each having a number of fields. To state the requirements, constraints could be added as facts. The Alloy Analyzer, a tool that enables fully automatic system analysis, is what defines Alloy most importantly. It can reveal weaknesses early on and promote incremental development. Two types of analyses can be performed using the Alloy Analyzer: searching for an instance, obtained by filling in signatures with elements while satisfying a predicate, or seeking a counterexample of an assertion. The Alloy model includes both assertions and predicates, and both kinds of analyses rely on the small-scope hypothesis [3]: the premise that if there is an instance or a counterexample, there is one of small size, limits the search to a finite subspace. In addition to these features, Alloy has recently introduced the sixth version [4] where the linear temporal logic is implemented within the language. Such a novelty has significantly empowered and enriched Alloy with new keywords, making it trickier as well for Software Engineering students who already find specification languages challenging to learn. This is mainly due to their formal nature and complex syntax that involve abstract concepts such as predicates, relations, and logical operators. For this reason, effective teaching and learning methods are necessary to overcome the challenges associated with learning this type of languages and in particular Alloy 6. An alternative method to the traditional chalk-and-talk one promoted in literature is the teaching module.

A teaching module is a significant, highly homogeneous, and unified part of a planned disciplinary program [5]. The first distinctive hallmark of a module is the clarity in the definition of learning goals which set student expectations about the following lectures and guide their learning processes [6]. A study [7] showed a positive effect of classes with high goal clarity and coherence on the students' reports on supportive learning conditions, self-determined learning motivation, and organizing learning activities. The description of learning goals can be thought of as a "map" for students to have a clear understanding of where the course is leading them and what is expected to be successful, By returning to the learning objectives throughout their study, students are able to guide and oversee their learning. Instead of focusing on what the teacher will do during the course, the objectives should be student-centered and outline what the students are supposed to learn. On the other hand, the definition of learning goals is helpful to the instructors as well to guide the design of fair course assessment plans and the selection of content and activities and make sure all critical course components are aligned to support student learning [8]. As a matter of fact, learning objectives are the foundation for instructional alignment, meaning that learning objectives, assessment tools, and instructional methods mutually support the same educational outcome [9].

After defining the lesson objectives, the teacher has to think about how to have his students achieve them. The means of teaching have changed a lot in the last decades and education has evolved as a result. Nowadays, it is common to see universities use instructional multimedia in course delivery as this is considered part of educational technology, and the didactic module is an example of this. It is not strange anymore to involve lecturers and students using multimedia technology in the classrooms due to its several benefits. [10] posited

that "Advantages of instructional multimedia include increased availability and repetition of instructional content, improved ability of students to learn at their own pace, increased student control of material, less demand on instructor time, and the provision of an alternative approach to describe complex topics or three-dimensional relationships". Multimedia technology can attract students' attention easily and consequently they are motivated in a way to continue exploring the presentation. When this process goes on, the students are engaged in the learning process. Hence, multimedia technology is able to create some initiatives for the students to keep learning. A study on using multimedia-enhanced problem-based learning environments to teach Science has shown that this learning environment is not boring and students are motivated to learn Science [11]. In a technology-driven, constructivist learning environment, multimedia serves as an effective tool for facilitating problem-solving through self-exploration, collaboration, and active participation. It is an essential tool for mastering basic skills by means of learning by doing, understanding abstract concepts, providing access to remote locations, and enabling individualized and cooperative learning experiences. The incorporation of simulations, models, and a variety of media-rich learning materials, including still and animated graphics, video, and audio, is vital in making the acquisition of new knowledge more efficient. With its interactive nature, multimedia is flexible enough to accommodate different learning styles, allowing both educators and learners to work together in an informal setting. Additionally, it stimulates peer learning and individual creativity and innovation.

Once the students have been acquainted with the lesson objectives, the teacher, also taking advantage of multimedia, involves them in actively participating in the lectures. In this regard, didactic modules try to overcome the limitations of traditional and deeply rooted classroom teaching where participants are put up in a classroom mode where they just listen to the lectures and go through some presentations [12]. This kind of approach places participants in a passive role as they try to memorize and reproduce the facts in order to clear the examinations. The courses traditionally enforce a reactive rather than proactive approach. i.e., the participants are expected to react to a solution presented instead of being proactive about the problem. Conversely, active learning refers to a process through which teachers and students construct a shared community wherein teachers facilitate learning and students become more responsible for their own learning [12]. This process often incorporates out-of-classroom projects, problem-based cases, team assignments, or applied research. For most learners, it is more motivating to be active than passive since a task that one has done himself or herself or even better as part of a group is more highly valued. Students who work together on active learning tasks learn to work with other people of different backgrounds and attitudes and learn strategies for learning by observing others. McKeachie and Svinicki [13] claim that active learning has a lot of other benefits for students. They are more likely to access their own prior knowledge and find personally meaningful problem solutions or interpretations, receive more frequent and immediate feedback, increase their self-confidence and self-reliance, and the student conceptions of knowledge change, which in turn have implications for cognitive development.

Accordingly, the didactic module is an effective teaching method that is suitable especially for disciplines with practical aspects with a strong need for active thinking, exactly like Software Engineering and specification languages. For this reason, a teaching module for specification languages is prominent for Software Engineering students. In fact, a teaching module can help students to overcome the challenges associated with learning these languages by covering their syntax, semantics, and practical applications, making it easier for students to understand and use them effectively. To have a valuable module, teaching techniques that address the main characteristics of the module are necessary. Therefore, in this work, we present teaching strategies for a didactic module to teach Alloy 6 to Software Engineering students.

II. Background State of the Art

A. Teaching strategies

Identifying effective teaching strategies to be adopted by the teacher is pivotal in education. In literature, numerous strategies are explored and tested, and hereafter some of the most useful are reviewed.

1) Collaborative Learning: Collaborative learning is a technique that involves groups of learners working together to solve a problem or complete a task. Loads of review articles outline the benefits of learning in collaboration starting from the definitions of the term and continuing by using collaborative methods [14]. Collaborative learning refers to an instruction method where knowledge is constructed by means of interactions among learners. For that reason, the learner is the major focus of instruction. Through interacting with each other toward a common goal, learners exchange ideas and elaborate their knowledge. Also, learners try to arrive at shared understanding by providing meaningful conversations about the problem and elaborated explanations. Collaboration is similar to cooperation; however, in cooperative learning, each learner is assigned a portion of the task. Collaborative learning; on the other hand, involves learners working in groups on the same task in order to search for understanding. [15]. Furthermore, the encountered challenges can turn beneficial if the students manage to regulate their learning activities with relevant strategic activities [16]. Collaborative learning is one of the best approaches with the advantage to enhance interactions, learner autonomy, teamworking, and problem-solving skills in a group. Members of a group exchange ideas and create understanding which help them with better achievement.

2) Problem-Based Learning: Problem-Based Learning is a learner-oriented instructional strategy. Barrows et al. define PBL as the following: "Problem-based learning is the learning that results from the process of working toward the understanding or resolution of a problem. The problem is encountered first in the learning process" [17] [18]. PBL has been widely

employed since learners need to solve real-world problems by themselves through cooperative working in groups, and so cooperative learning previously mentioned is a related and required technique. The main characteristics of PBL are the use of real-world problems, encouragement of students' active participation, integration of different viewpoints, encouragement of self-directed learning, encouragement of team collaboration, and enhancement of education quality [19]. Due to the limitations in a classroom environment, most software engineering projects assigned to students are smaller than the actual size of projects used in companies. The nature of projects, however, still 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. Students always have to take an agreed decision and handle all situations in team projects. Each student is encouraged to strongly participate as a team member, which is essential to the successful completion of the assignment. A student may also have a different viewpoint or may take important decisions to solve issues, so it's important to find the proper communication channel in order to effectively explain the problem to the other teammates. An instructor, as a mentor, is supposed to help students manage problems without deeply being involved in team activities. Students combine the knowledge they have learned and apply it to a real-world problem. It is desirable to integrate human-based aspects into software engineering classrooms so that students can be equipped with communication, interaction, and cooperation among team members, which is essential for successful software development. The innovative adaption aims to help students understand the significance of social aspects in software development and improve teamwork skills. This approach is expected to provide a systematic framework to assist learners in preparing for rigorous industrial settings in the future.

3) Worked-Example: Another technique is the worked-example, which "is a step-by-step demonstration of how to perform a task or how to solve a problem" [20]. Worked examples are designed to support the initial acquisition of cognitive skills by introducing a formulated problem, solution steps, and the final solution [21]. Studying worked examples is an effective instructional strategy to teach complex problem-solving skills [22]. This is because example-based instruction provides expert mental models to explain the steps of a solution for novices. It's a scaffolded learning approach that reduces a learner's cognitive load, so skill acquisition can become easier. After the teacher has presented each step of a worked example, students can use the worked examples during independent practice and review and embed new knowledge. [23] It can be used in hands-on-lab sessions where a teacher shows the solution of the problem linking the mental approach with the lines of code actually written, thus being able to explain the syntax and semantics of the language and ways to build a project from scratch.

4) Questioning: Questioning is a powerful tool and effective teachers deploy it regularly for many purposes. It engages students, stimulates interest and curiosity in learning, and makes links to students' lives. It unfolds opportunities for students to talk together, discuss, argue, and express opinions and alternative views. If used effectively, questioning yields immediate feedback on students' understanding, supports informal and formative assessment, and captures feedback on the impact of teaching strategies. [23] Effective teachers regularly use questioning as an interactive means to engage and challenge students, and use it as a tool to check student understanding and evaluate the effectiveness of their teaching. There are some methods that can be useful to remember when questioning should be used. Some of these come from Lemov's book, and they are "cold calling" (all students should be involved in engaging with the teacher-student dialogue with time to think, and not be allowed to hide, dominate, or be overlooked) and "no opt-out" (Students should feel safe in answering when unsure but, if they don't know or get things wrong, they should be given the opportunity to gain confidence by consolidating correct or secure answers. Also, students should not be allowed to opt-out by saying 'I don't know') [24]. Other questioning methods are:

- "check for understanding": teachers should not assume that knowledge aired and shared in the public space of the classroom has been absorbed; or learned by any individual;
- "probing": in order to explore a student's schema in any depth, you need to ask them several questions; asking several students one question each provides shallow responses compared to when each student has to provide multiple responses;
- "whole-class response": sometimes is useful or even essential to get a response from every single student at the same time. This provides quick feedback to the teacher about the success of the relevant teaching and learning exchanges, identifies individuals who need further input, and can help direct subsequent questions or exercises as you respond to the feedback you gain;

[25]

5) Feedback: Feedback informs a student or a teacher about the student's performance relative to learning goals. Its purpose is to improve the student's learning. Feedback redirects or refocuses the actions of teacher and student so the student can align effort and activity with a clear outcome that leads to achieving a learning goal [23]. Feedback can be conducted physically in the classroom, or through anonymous online surveys.

6) Explicit-Teaching: When teachers adopt explicit teaching practices they clearly show students what to do and how to do it. Students are not left to construct this information for themselves. The teacher decides on learning intentions and success criteria, makes them transparent to students, and demonstrates them by modeling. In addition, the teacher checks for understanding, and at the end of each lesson revisits what the lesson has covered and ties it all together [26] [23]. Explicit teaching is systematic and sequential. It directly supports guided practice using a series of steps. First, teachers are explicit about the learning goals and the success criteria.

Teachers then demonstrate how to achieve them by modeling and providing examples. The final step is to provide students with opportunities to practice and demonstrate their grasp of new learning.

7) Flipped Classroom: The flipped classroom is a set of pedagogical approaches that move most information-transmission teaching out of class using class time for learning activities that are active and social and require students to complete pre-class activities to fully benefit from in-class work [27]. The purpose of the flipped classroom is to promote "active engagement with the content, the instructor, and other students, rather than passive reception of the content transmitted by the instructor" [28]. As indicated by Talbert (2014) [29], for a flipped classroom experience to be effective, it ought to incorporate the following:

1) Very organized pre-class assignments which are equipped towards presenting the students with new theoretical notions;
2) Tools for responsibility to guarantee that students will finish the required pre-class assignments and out-of-class work;
3) Activities should be well planned and designed, and attractive for the students to engage with during lecture time;
4) The lines of correspondence all throughout the course should be open, so the students can communicate freely with their professor [30];

Jensen et al. [31] identify two phases of teaching: the "content attainment phase" when students acquire conceptual understanding, and the "concept application phase" during which they evaluate concepts and/or apply them to novel problems/situations. In contrast to traditional lecture-based teaching, where class time is primarily focused on content attainment, the flipped classroom model places this beforehand, so that class time can be used for concept application. While students are engaged in learning activities requiring "higher order" thinking skills, such as analyzing evaluating, or creating material [32], they, therefore, have support from the lecturer and other students. Although videos are often used for pre-class instruction, it may be desirable to include readings as preparatory work (e.g. so that students become familiar with how to write academic texts, or to reduce the lecturer's course design workload), otherwise, students' concentration may fail [33]. Non-completion of flipped classroom pre-class work negatively affects both under-prepared and well-prepared students and can lead to complaints [34] [35]. Giving credit for low-stakes assessment of understanding of the preparatory material, such as quizzes, can motivate pre-class preparation [35]. Meta-analyses [36] [37] [38] found that including quizzes in the flipped classroom model has a positive effect on student performance. At the "content attainment" stage, being able to pause and re-watch recorded lectures enables students to learn at their own pace [39] [40], and may help manage cognitive load [27]. At the "concept application" stage, flipped classroom promotes teacher-student interaction in the next activities. In the traditional way of learning, students try to catch what is being said by the professor at the very moment when he teaches. They cannot stop it, rewind it, or listen to it again, nor reflect upon what is being said, and they may miss valuable parts of the material because they are trying to write down the professor's words. On the contrary, the concept of flipped learning is to provide students lectures in a video format and other supportive materials to review as their homework, get the maximum of it, and then, use the next class time for in-class activities and problem-solving exercises [30] [2].

B. An Evaluation of Techniques

An internationally renowned New Zealand education academic, John Hattie, has assembled the world's largest evidence-based study of factors that improve student learning in his book [41], involving more than 80 million students worldwide and bringing together more than 50,000 smaller studies. The effectiveness index of 1.0 is typically associated with a one-year advance in student achievement, or a 50 percent improvement in the rate of learning; a correlation between certain variables (e.g., the amount of homework) and achievement of about 0.50; and a two-grade jump in GCSE, for example, from a grade 4 to a grade 6. It is possible to consider that individual activities are penalized: their effect size is under 0.40. An important activity is the feedback, well considered by the students, which has an effect size of 0.74.

C. Alloy 6

Alloy has been widely taught in undergraduate and graduate courses for many years. At Brown University, Logic for Systems course uses Alloy for modeling and analysis of system designs. Because the Alloy language is very close to a pure relational logic, it has also been popular in the teaching of discrete mathematics, for example, in a course at Michigan Technological University. In the Formal Methods in Software Engineering course at the University of Iowa, Alloy is taught along with the new features introduced in the sixth version to make dynamic modeling [42]. At the University of Minho in Portugal, Alcino Cunha, the creator of the Electrum [43] extension for Alloy that includes some of the most important features introduced in Alloy 6, teaches an annual course on formal methods using Alloy, and has developed a Web interface to present students with Alloy exercises, which are then automatically checked, named Alloy4Fun [44]. In literature, some works propose to integrate the Alloy language in Computer Science courses, such as Software Modeling course [45], or Formal Methods courses [46]. Other works propose some exercises using Alloy for discrete structure [47] or discrete mathematics course [48]. Yet little has been done about the sixth version of Alloy since it is still recent, except for some works presenting the new features [49]. What seems to lack is a comprehensive teaching module about Alloy, especially its sixth version, in order to tackle its difficulties in understanding and turn it into a useful tool for Software Engineering students and not only.

III. Design of an Alloy 6 Module

This work proposes a didactic model that incorporates various teaching strategies to teach Alloy 6. In particular, the model includes theoretical and exercise lectures, both lasting two academic hours, within which a timeline was defined for each macro-topic to be presented. In order to experiment with modalities alternative to the lectures, additional activities, such as self-assessment quizzes, flipped classroom, and a challenge, were added, which serve as tools to test understanding related to the learning objectives.

A. Theoretical Lectures

Two lessons of 45 minutes each are designed to give a full explanation of the features introduced by Alloy 6. A flipped classroom is provided between the two classroom lessons, described in detail in the following section. In order to be able to deliver these lectures, students are required to be familiar with first-order logic and notation, and specification methods of Alloy5 in particular. The main strategy to follow in the theoretical part of the module is structuring lessons with the aid of succinct slides to call attention, especially to the main concepts. At the beginning of the theoretical part, the setting goals strategy is chosen to explain exactly what students are supposed to understand and what they must be able to do. The first part of the two lessons must link to the previous students' learning, while the end of the second reviews the main ideas of the theoretical part of the module. The second lesson also furnishes opportunities for self-assessment, peer feedback, and teacher feedback through quizzes that also signal transitions between lesson parts. In order to indicate the students' level of content understanding, traffic light questioning and feedback from students must be present during both lessons. Both the first and second lessons are to be explained with the support of a set of slides created with Microsoft PowerPoint and attached to this guide. In addition, quizzes to test comprehension of the topics and the effectiveness of self-study are to be given at the beginning of the second lesson in class, as will be detailed later

B. First theoretical lecture

The aim of the first lesson is to make a comparison of how variant time systems are represented in Alloy 5 and Alloy 6, showing how the latest version of the language is more effective for this purpose due to the introduction of a new logic and new keywords. At the end of this first lesson, students should be able to understand how Alloy 5 deals with dynamic modeling, its limitations, and the need for the new version of Alloy.

Table 1, shown below, details the topics and the time required to explain each one.

Argument	Duration
Overview of learning objectives	10 min.
Static vs. Dynamic World in Alloy (meaning)	5 min.
How to represent dynamic models in Alloy 5? The ordering method and its limitations; the use of the `Time` signature and limitations.	20 min.
Alloy 6: Introduction to the use of Linear Temporal Logic and use of `var` keyword.	10 min.

C. Flipped lecture

The flipped classroom method is used to help students to learn the part of the topics related to temporal connectors introduced in Alloy 6. Since the pattern used for their explanation is fairly repetitive, students are ready to understand all the connectives, with the possibility to dwell on the more complex ones as they see fit. Students should have attended the first lecture where the features of Alloy 6 related to the temporal logic and the mutable signatures were presented. The strategy used by the teacher is that of explicit teaching. The teacher, in the recorded video, unidirectionally explains the concepts leaving the opportunity to raise doubts and gather opinions in the next lesson, using feedback and questioning later. The focus of the flipped classroom is on temporal connectives, both present and future, which the user can use after understanding the concept of LTL to define assertions, facts, and predicates. 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. At the end of the flipped classroom, students are ready to use consciously all temporal connectives provided by Alloy 6 and they are able to face the following arguments. With this method, students develop the ability to study independently with a video, without a guide.

D. Second theoretical lecture

The second lesson has two main objectives: the first is to check, as mentioned above, the student's understanding relative to the first lesson and the flipped classroom, while the second is to carry on the explanation of the new features introduced by Alloy 6. At the end of the class, students will be able to evaluate themselves and comprehend whether their study method for Alloy 6 was effective or not, based on the results of quizzes taken during class time. Finally, they will have a comprehensive overview of the innovations and expressive power of Alloy 6.

Table 2, shown below, provides the schedule of the lecture.

Argument	Duration
Quizzes about the previous lecture and the flipped one	20 min.
Alloy 6 (Time Horizon, New visualizer, Concurrency)	10 min.

E. Exercise lecture

The exercise lecture is an optimal time to fix the concepts seen in class. Two paradigmatic exercises allow the lecturer to

exhaustively review what was explained during the theoretical lecture and to show some of their possible application through the Alloy Analyzer. The total duration of the exercise is 90 minutes. To tackle the in-class exercise and solve the home exercise, students must have attended the lectures in which the features of Alloy 6 were presented, have watched the video related to connectives, which will be extensively used when defining predicates, facts, and assertions, and have successfully installed the latest version of the Alloy tool. The strategy to be adopted by the teacher in the exercise is the worked-example, along with questioning and feedback. In detail, the lecturer starts the exercise by writing the code together with the students using the exercise prompts as an opportunity to review and demonstrate how to get to the definition of facts, predicates, and assertions with a practical and easy-to-understand example. In order to have a learner-centered lecture, the lecturer leaves space and time for students by interacting with them and having them come up with possible solutions for the exercises and ask when in doubt. The teacher can use the set of slides created with the texts and codes from the exercises. The slides also contain animations that graphically render what the code means all the way through. In addition to the slides, there are .als source codes of all the exercises, already tested, which can be run in the classroom and left to the students. At the end of the exercise, students will be more aware of the potential of Alloy and some of its applications. They will be able to model realistic scenarios thanks to the examples seen and the strategy adopted by the teacher, and thus ready for the extra activity explored in depth in the following sections.

Table 3 shows the time partition of the exercises proposed during the lecture.

Exercise	**Duration**
Concurrent Communication in Distributed System	15 min.
Travel (Interrail)	30 min.

F. 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. The challenge aims to model a particularly complex scenario using the approach seen in the tutorial and all the background provided by the lectures. To tackle the challenge exercise and solve the exercise, students must have attended the lectures in which the features of Alloy 6 were presented. have watched the video related to connectives, which will be extensively used when defining predicates, facts, and assertions, have attended to the exercise have installed the latest version of the Alloy tool, and have understood the delivery method. The teacher uses the strategy of collaborative-learning, having students work together in small groups (2 or 3 students) and allowing them to develop skills related to teamworking and problem-solving. Since the effectiveness of the teaching module is to be observed all the way through, the lecturer should give directions only when strictly necessary. The proposed theme for the challenge is Software-Defined Network, inspired by [50]: a very current approach in cloud computing and network architectures that facilitates their administration and configuration in order to improve performance and facilitate their monitoring. It is required to model signatures, then static behavior, and eventually dynamic behavior. 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 and thus at the end of the module, students will be able to face situations in which they are required to model something very complex by working in a group in a very current, practical, and professional setting. What students are going to model is a real and current problem that does not necessarily have only one solution. Problem-solving, which is the skill that the challenge greatly develops, wants each person to propose a personal solution based on what they have seen in class, but not enough to fulfill the request

IV. Alloy 5 vs. Alloy 6

It is possible to better realize the importance of the innovations brought by Alloy 6 through a comparison with the previous version, in particular, by identifying the limitations that have been overcome by this new version of the language. Until Alloy 6, there was no intrinsic notion of time and transitions between states, but it was still possible to represent the dynamic evolution of a system explicitly and with various artifices. In particular, two main strategies were possible: the first consists of the use of the util/ordering module exploited to make a certain signature ordered, while the other one involves the use of the signature `Time` that applies the util/ordering module for the simulation of time [51]. A deeper analysis of the two methods follows below :

1) For the explanation of the ordering method, reference will be made to a generic signature `S`. To place an ordering on `S` means that its instances are representative of the ”same physical element” but at different time instants. In this way, it is possible to visualize `S` as a state machine where an initial and final state are identified and properties and constraints valid for individual temporal states can be defined. To do this, it is necessary to import the `util/ordering` package with the following syntax:

```
open util/ordering[S]
sig S{}
```

The code above is equivalent to sorting exclusively the signature that is placed between the square brackets, in this case, `S`. The functions provided by the package that are useful for this purpose are: `first`, `next`, `last`, and `prev`. They can all be invoked exclusively with reference to `S` (e.g. `S.first`, `S.last`, ...).
To construct a valid evolution of the system through these functions, a trace is needed, namely a fact with a

set of constraints defining how the system evolves. The syntax to be used is as follows:

```
open util/ordering[S]
...
fact traces {
    S/first = S1
    all s: ord-ord/last |
    let s' = s.next |
    op1[s, s'] or ... or opN[s, s']
}
```

In this case, the initial state has been defined, and all states after the first one, except the last, are represented as "next". Any operation, denoted in the code as `opI` (where i is an integer from 1 to n, so as not to impair generality) can be defined so that it can establish the transition between states. Despite it seeming intuitive and easy to use, this first method has some limitations. In fact, it allows sorting only one signature at a time, that is the one specified in the square brackets. Consequently, it is not possible to create order on several signatures at once but it is necessary to rewrite the code as many times as the number of signatures to be sorted. When there are many time-varying properties and signatures, such a method lacks of efficiency.

2) The second method turns out to overcome the limitations of the previous one. Instead of placing the ordering on individual signatures that change over time, the order is placed on a single signature called `Time`, with the following syntax:

```
open util/ordering[Time]
sig Time {}
```

Each variable signature must be related to the signature `Time` as in the following piece of code:

```
sig Time{}
sig S {
    relation: S1 lone-> Time
}
sig S1 {}
```

This is equivalent to constructing ternary relationships of the type `<S, S1, T0>`, where the relationship between `S` and `S1` is true at time `T0`. Signature `Time` can also be used by establishing another type of relationship between a field and directly a set of signature times such as

```
sig S2 {
    Relation2: set Time
}
```

The difference is that the relationship that changes over time in this case is boolean, whereas before it was arbitrary. By relating a signature to a set of times, it is possible to establish the existence of a set of times when the relation is true.

Even with the Time signature expedient, the dynamic modeling until Alloy 6 has some limitations. For example, the concept of a trace does not tell if a system is deadlocked, so if there is no valid next state as part of the trace. If the trace is encoded as a fact, as seen above, the entire model is discarded; if the trace is encoded as a predicate, Alloy will provide any model that does not match the trace as a counterexample. In addition, Alloy cannot verify that a certain property is guaranteed in an infinite time (liveness). However, the most significant limitation of Alloy 5 is that the notion of time is not built-in into the language, but it is a module that must be imported, as seen before. Conversely, Alloy 6 has an inherent notion of discrete time. This implies that it is possible to model the evolution of the system without importing packages or introducing new signatures, but simply by exploiting the enriched syntax of the language. The following chapters will present the major innovations that make it possible to achieve such a goal.

V. Alloy 6 Main Concepts

A. Linear Temporal Logic (LTL)

In Alloy 6 Linear Time Logic is introduced, defined by professors Hashari and Habib [52] as an infinite sequence of states in which each point in time has a unique successor, based on a linear time perspective. Like most specification languages, Alloy divides time into a series of discrete steps, where *i* is denoted as the current state, *i+* the future states, and *i-* the past states. In Alloy 6, the instances become traces, namely infinite sequences of states, and traces are not facts anymore but they are implicit within a mutable model. A state is in turn a valuation for signatures and fields. Since the finite memory of a computer cannot consider an infinite sequence of states, the considered traces are represented as lasso traces, that is, a sequence of a finite number of states featuring a loop from the last state back to a former state.

B. Mutable Signatures and Fields

[53] One of the most important innovations in Alloy 6 is the keyword `var`, used to make a signature or a field mutable. The absence of this keyword, instead, indicates that the signature or field is static and is assumed constant over time. The valuation of a mutable signature or field is likely to vary from state to state in a given trace, while static ones remain unchanged in a given trace.

C. Temporal Connectives

The temporal connectives can be divided into future and past connectives: as can be seen in the following table, there is a 1-to-1 correspondence between most of them, except for the binary past operator ";" which is equivalent to "and after." There are both unary and binary operators.

Future	Past
`always`	`historically`
`eventually`	`once`

after	before
until	since
releases	triggereed
;	- no dual -

- The expression *after F* is true in state i iff *F* is true in state $i + 1$;
- The expression *always F* is true in state i iff *F* is true in every state $>= i$.
- The expression *eventually F* is true in state i iff *F* is true in some state $>= i$.
- The expression *F until G* is true in state i iff *G* is true in some state $j >= i$ and *F* is true in every state k such that $i <= k < j$.
- The expression *F releases G* is true in state i iff *G* is true in every state $>= i$ up to and including a state k in which *F* is true, or there is no such k in which case *G* holds in any state $>= i$.
- The expression *F ; G* is true in state i iff *F* is true in state i and *G* is true in state $i + 1$.
- The expression *before F* is true in state $i > 0$ iff *F* is true in state $i - 1$. By convention, *before F* is false in state 0.
- The expression *historically F* is true in state i iff *F* is true in every state $<= i$.
- The expression *once F* is true in state i iff *F* is true in some state $<= i$.
- The expression *F since G* is true in state i iff *G* is true in some state $j <= i$ and *F* is true in every state k such that $j < k <= i$.
- The expression *F triggered G* is true in state i iff *F* is true in some state $j <= i$ and *G* is true in every state $j < k <= i$, or *F* if false in every state $<= i$ in which case *G* is true in every state $<= i$.

D. Time Horizon

[53] In addition to placing bounds on the size of signature sets, the scope specification may constrain the time horizon, that is, the possible number of transitions of lasso traces to explore. To do so, Alloy features a reserved `steps` keyword:

- If no time horizon is given, this implicitly equals 10 steps.
- If the time horizon takes the form for N steps, this is equivalent to for 1 .. N steps.
- If the time horizon takes the form for M .. N steps, only lasso traces with at least M transitions and at most N ones, including the looping transition starting in the last state, will be explored.
- If the time horizon takes the form for 1.. Steps, the number of steps is greater or equal to 1..
- If the time horizon takes the form for 1.. Steps, the number of steps is greater or equal to 1. Unlike the first three expressions, the time horizon is unbounded. In this case, the solver must support the complete model checking that allows checking over all possible traces and this is possible since the state space is finite

E. New Visualizer

[53] Alloy 6 also features a Visualizer enhanced to display traces in a user-friendly way. The visualization pane is split into two contiguous panes which show two consecutive states in a trace where variable fields and signatures are distinguished by dashed lines. The lasso trace represented above shows the displayed step-pair of states in white, while the other states of the trace that are currently not displayed are in grey. Finally, the Visualizer features a sophisticated way to explore alternative instances of a specification:

- *new config* yields a trace where the configuration (that is, the valuation of static parts) changed;
- *new trace* yields a new trace under the same configuration;
- *new init* yields a trace with a different initial state, under the same configuration;
- *new fork* yields a new trace which is similar to the current one until the currently-displayed state but differs afterward.

VI. Conclusion

Formal specification languages in requirements modeling are as important as hard to understand for Software Engineering students. In particular, Alloy is an educationally accessible modeling language that recently has introduced important new features all revolving around an integrated concept of time, without involving external modules, but bringing in new keywords for making properties of objects or objects themselves mutable, new operators for expressing bindings to the past and future, and a new Visualizer.

We proposed the use of a teaching module to effectively teach students Alloy by combining different teaching strategies, such as frontal lecture and flipped-classroom, in order to stimulate students' interest and help them better understand the language.

Further research is needed to validate the proposed approach. Experiments could be conducted to first test the effectiveness of the teaching module objectively and then by collecting feedback from students through additional questionnaires. In objective key testing, it is interesting to compare different teaching methods by dividing an entire course class into two groups: one group testing the module, while the other using another teaching approach.

Quizzes, exercises, and the challenge can be exploited to build an ongoing picture of the state of student learning. As time, and thus activities, change, since quizzes will be taken before the exercises and before the challenge, some factors can be taken into account to test the module validation:

- timeliness of students' responses
- obtained scores
- quality of answers in relation to the used teaching method

In particular, the first two factors can be evaluated through quizzes and fast exercises, while the latter factor gains more relevance in the evaluation of activities such as drills and challenges. The quality of answers, in particular, is intended as

how well students manage to tackle a complex problem from scratch without teacher guidance.

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

REFERENCES

[1] Daniel Jackson. Alloy: A language and tool for exploring software designs. *Communications of the ACM*, 62:66–76, 9 2019.

[2] Rachel A. Howell. 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.

[3] Daniel Jackson. *Software Abstractions: logic, language, and analysis*. MIT press, 2012.

[4] http://alloytools.org/alloy6.html,.

[5] Dr Kandarp Sejpal. Modular method of teaching, 2013.

[6] https://medicaleducation.weill.cornell.edu/medical-education/instructional-design-services/importance-learning-objectives.

[7] Tina Seidel, Rolf Rimmele, and Manfred Prenzel. Clarity and coherence of lesson goals as a scaffold for student learning. *Learning and Instruction - LEARN INSTR*, 15:539–556, 12 2005.

[8] https://cteresources.bc.edu/documentation/learning-objectives/.

[9] Debnath Chatterjee and Janet Corral. How to write well-defined learning objectives. *The journal of education in perioperative medicine : JEPM*, 19:1–4, 10 2017.

[10] A. Russell Smith, Cathy Cavanaugh, and W. Allen Moore. Instructional multimedia: An investigation of student and instructor attitudes and student study behavior. *BMC Medical Education*, 11, 2011.

[11] Min Liu, Lucas Horton, Jaejin Lee, Jina Kang, Jason Rosenblum, Matthew O'Hair, and Chu-Wei Lu. 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.

[12] Yap Wei Li. Transforming conventional teaching classroom to learner-centred teaching classroom using multimedia-mediated learning module. *International Journal of Information and Education Technology*, 6:105–112, 2016.

[13] Yaron Ghilay and Ruth Ghilay. Tbal: Technology-based active learning in higher education. *Journal of Education and Learning*, 4, 9 2015.

[14] Marjan Laal and Mozhgan Laal. Collaborative learning: what is it? *Procedia - Social and Behavioral Sciences*, 31:491–495, 2012. World Conference on Learning, Teaching Administration - 2011.

[15] Nihat Salma. Collaborative learning: An effective approach to promote language development. 7:5, 06 2020.

[16] Lisa Linnenbrink-Garcia, Toni Kempler Rogat, and Kristin L.K. Koskey. Affect and engagement during small group instruction. *Contemporary Educational Psychology*, 36:13–24, 1 2011.

[17] Charlie Shim, Mina Choi, and Jung Kim. Promoting collaborative learning in software engineering by adapting the pbl strategy. 01 2009.

[18] HS Barrows. The essentials of problem-based learning. *Journal of Dental Education*, 62(9):630–633, 1998.

[19] Robert Delisle. *How to use problem-based learning in the classroom*. Ascd, 1997.

[20] Ruth Clark, Frank Nguyen, John Sweller, and Melissa Baddeley. Efficiency in learning: Evidence-based guidelines to manage cognitive load. *Performance Improvement*, 45, 10 2006.

[21] Alexander Renkl. *The Worked-Out Examples principle in Multimedia Learning*, pages 229–245. 01 2005.

[22] J.J.G. van Merrienboer. *Training complex cognitive skills: A four-component instructional design model for technical training*. Educational Technology Publications, 1997.

[23] Victoria. Department of Education and Training. *High impact teaching strategies : excellence in teaching and learning.*

[24] https://teachlikeachampion.org/books/teach-like-champion-2-0.

[25] https://teacherhead.com/2018/08/24/great-teaching-the-power-of-questioning/.

[26] John Hattie. *Visible Learning: A Synthesis of Over 800 Meta-Analyses Relating to Achievement*. 01 2009.

[27] Lakmal Abeysekera and Phillip Dawson. Motivation and cognitive load in the flipped classroom: definition, rationale and a call for research. *Higher Education Research & Development*, 34(1):1–14, 2015.

[28] Li Cheng, Albert D. Ritzhaupt, and Pavlo D. Antonenko. Effects of the flipped classroom instructional strategy on students' learning outcomes: a meta-analysis. *Educational Technology Research and Development*, 67:793–824, 2018.

[29] Robert Talbert. Inverting the linear algebra classroom. *PRIMUS*, 24(5):361–374, 2014.

[30] Blerta Prevalla and Hüseyin Uzunboylu. Flipped learning in engineering education. pages 656–661, 05 2019.

[31] Jamie L Jensen, Tyler A Kummer, and Patricia D D M Godoy. Improvements from a flipped classroom may simply be the fruits of active learning. 2015.

[32] Lorin W. Anderson, David R. Krathwohl, and Benjamin Samuel Bloom. A taxonomy for learning, teaching, and assessing: A revision of bloom's taxonomy of educational objectives. 2000.

[33] Ngoc Thuy Thi Thai, Bram De Wever, and Martin Valcke. The impact of a flipped classroom design on learning performance in higher education: Looking for the best "blend" of lectures and guiding questions with feedback. *Computers Education*, 107:113–126, 2017.

[34] Adam Butt. Student views on the use of a flipped classroom approach: Evidence from australia. *Business Education and Accreditation*, 6:33–44, 2014.

[35] Wenliang He, Amanda Holton, George Farkas, and Mark Warschauer. The effects of flipped instruction on out-of-class study time, exam performance, and student perceptions. *Learning and Instruction*, 45:61–71, 10 2016.

[36] Khe Hew and Chung Kwan Lo. Flipped classroom improves student learning in health professions education: A meta-analysis. *BMC Medical Education*, 18, 03 2018.

[37] Chung Kwan Lo, Khe Foon Timothy Hew, and Gaowei Chen. Toward a set of design principles for mathematics flipped classrooms: A synthesis of research in mathematics education. *Educational Research Review*, 22:50–73, 2017.

[38] David C.D. van Alten, Chris Phielix, Jeroen Janssen, and Liesbeth Kester. Effects of flipping the classroom on learning outcomes and satisfaction: A meta-analysis. *Educational Research Review*, 28:100281, 2019.

[39] Travis Roach. Student perceptions toward flipped learning: New methods to increase interaction and active learning in economics. *International Review of Economics Education*, 17:74–84, 2014.

[40] Darryl Yong, Rachel Levy, and Nancy Lape. Why no difference? a controlled flipped classroom study for an introductory differential equations course. *PRIMUS*, 25(9-10):907–921, 2015.

[41] John Hattie. *Visible learning*. Routledge, 2008.

[42] https://homepage.cs.uiowa.edu/ tinelli/classes/181/fall22/notes/06-dynamic-models.

[43] Nuno Macedo, Julien Brunel, David Chemouil, Alcino Cunha, and Denis Kuperberg. Lightweight specification and analysis of dynamic systems with rich configurations. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2016, page 373–383, New York, NY, USA, 2016. Association for Computing Machinery.

[44] Nuno Macedo, Alcino Cunha, José Pereira, Renato Carvalho, Ricardo Silva, Ana C. R. Paiva, Miguel S. Ramalho, and Daniel Silva. Sharing and learning alloy on the web. 7 2019.

[45] Zoltàn Istenes.

[46] Zhenhua Duan Shaoying Liu.

[47] Charles Wallace. Learning discrete structures interactively with alloy: (abstract only). In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, page 1051, New York, NY, USA, 2018. Association for Computing Machinery.

[48] Leo C. Ureel and Charles Wallace. Discrete mathematics for computing students: A programming oriented approach with alloy. volume 2016-November. Institute of Electrical and Electronics Engineers Inc., 11 2016.

[49] https://www.hillelwayne.com/post/alloy6/.

[50] Laura Panizo María-Del-Mar Gallardo. Modelling and specifying software systems with alloy * (tutorial).

[51] https://haslab.github.io/formal-software-design/teaching-alloy/index.html.

[52] Rehab Ashari and Sahar Habib. Linear temporal logic (ltl).
[53] https://alloytools.org/.