Adapting to Different Types of Target Audience in Teaching Formal Methods*

Antonio Cerone[1] and Karl Reiner Lermer[2]

[1] Department of Computer Science, Nazarbayev University, Nur-Sultan, Kazakhstan
antonio.cerone@nu.edu.kz
[2] Department of Computer Science, ZHAW Zurich University of Applied Sciences, Winterthur, Switzerland
lrka@zhaw.ch

Abstract. Formal methods can be considered as the area of computer science that most effectively bridges the gap between mathematics and computer science. They are potentially a great educational tool for fostering mathematical reasoning skills and problem-solving abilities in a very wide audience of potential learners from university, industry, school and research. But, unfortunately, formal methods are taught only in a limited number of computer science university programmes, mainly at postgraduate level, and are usually presented as such a difficult topic that university students keep away from them and the industry, in general, does not consider them as a worthy research and development investment.

In this paper we draw upon our experience in teaching formal methods to the heterogeneous audience of potential learners. We report on how teaching methods and materials must be adapted to the specific type of target audience to effectively produce learning outcomes. We observe that motivation, fun and practice are essential dimensions of such an adaptive approach.

Keywords: Formal methods · Teaching approach · Target audience

1 Introduction

There is a widespread misconception that mathematics and computer science are two independent, fully distinct disciplines, with the fact that computers can be used to perform complex mathematical calculations being the only perceived connection between the two disciplines. This misconception determined a large gap between computer science and mathematics, mainly in educational and industrial contexts, and partly also in the context of scientific research.

The source of this misconception is that computer science is normally identified with programming, and programming is seen more like a kind of art rather than an applied science. This widespread perception of computer science has its

* Work partly funded by Seed Funding Grant, Project SFG 1447 "Formal Analysis and Verification of Accidents", University of Geneva, Switzerland.

roots in people's common beliefs as well as in school education. Programmers are normally considered like weird people, fully immersed in their work activities and in some sense detached from the real world, just like artists. Even those who see programmers somehow as scientists, actually identify them with "crazy scientists', in fact, with 'gits', namely unpleasant or contemptible people. This stereotype is so widely accepted that the platform that collects most of nowadays open source software projects, to which programmers contribute in their free time, almost addictively and mostly without been paid, is called *GitHub*.

The gap between computer science and mathematics led to a debate on the centrality of mathematics and logic in computer science curricula: on the one side the claim that rigorous mathematical knowledge is not necessary for computer science practitioners [15] and, on the other side, the belief [24, 25] and the empirical evidences [21, 23] that learning rigorous discrete mathematics and formal methods has an important impact on problem-solving and programming skills and is perceived by students as useful in practical problems and helpful in improving their mental processes [26].

In the last two decades computers have been heavily introduced in schools. In many schools computer science has even been introduced as a new, stand-alone subject. However, this has been normally done without connecting computer science with mathematics but, instead, by seeing computer science as a "service subject", whose only scope is that of providing tools that facilitate the students in carrying out their homework and class projects [14]. The teaching of computer science to school pupils tends, therefore, to focus on using office-oriented tools to write documents, prepare presentations and organise data in spreadsheets.

1.1 Formal Methods and its Potential Audience

Formal methods is one of the most challenging areas of computer science. It has at least four distinctive aspects that make it unique in several respects. We believe that these distinctive aspects originates from the fact that formal methods is the area that most effectively bridges the gap between mathematics and computer science. From a theoretical point of you we can say that a *formal model* is a mathematical representation of a computer program, namely a mathematical object that we can manipulate with potentially infinite mathematical tools. Thus, moving from computer programs to formal models through an *abstraction process* allows us to exploit the power of mathematics in order to understand what the program does, namely its *semantics*, reason about it, and analyse it statically in a precise way.

Therefore, formal methods, on the one hand, foster those mathematical reasoning skills that are essential in producing correct, effective software and, on the other hand, make computer science, in some way independent of computers. A third distinctive aspect of formal methods, one with very practical consequences, is its potential to provide an effective way to analyse a large number of critical, non-functional properties of software, including safety, security, reliability and usability. In trying to deal with such critical properties software engineering principles, guidelines and methodologies have always been struggling and never managed to fully provide assurance. Meeting these properties is necessary for the most critical and innovative technology in use today and represents a present and future challenge for the exponential increase in system complexity determined by ubiquitous computing and the internet of things. Finally, the fourth distinctive aspect is that formal methods can be applied, beyond computer science and technology, to several disciplines, including physics, chemistry, biology, ecology, psychology, cognitive science and economics.

We can thus claim that formal methods have the potential to address a very wide audience, which comprises

university students in computer science, who need to develop abstraction and reasoning skills needed to produce, understand and analyse software;

school pupils in order to allow them to establish the mathematical problem-solving bases that can enable them to succeed in scientific and technology-oriented university programmes;

research being formal methods applicable to a wide range of domains, especially to innovative technologies, they must adapt to continuoulsly evolving technologies and to the heterogenuous needs of interdisciplinary research teams.

industry not only in the software area but also in a number of technology sectors, either safety-critical, such as transportation, avionics, aerospace, chemical plants, nuclear power plants, medical devices, or security-citical, such as e-commerce and defence, and to complex systems encountered in chemistry, biology, ecology, psychology and economics;

Given such a large potential audience, why then aren't formal methods widely taught in universities and schools as well as in industrial training? Why aren't they widely accepted and used in industrial contexts and fully recognised in research areas such as software engineering and human-computer interaction?

1.2 Structure of the Paper

In this paper, after describing our backgrounds and the settings in which we carried out teaching, student supervision, research and collaboration with industry (Section 2), we give an account of our experience in teaching formal methods and identify and discuss a number of dimensions that drove the development of our engagement strategy through the years (Section 3.1). Section 4 illustrates how engagement strategy may be adapted to various kinds of audience. Section 5 concludes the paper.

2 Authors' Background and Formal-methods-related Activities

The authors of this paper have backgrounds in computer science and mathematics, respectively. They have several years of experience in teaching and supervising students in their respective areas, both at the undergraduate and post-graduate level. They also carry out research in the area of formal methods, especially in terms of applications to safety-critical systems and in the modelling and analysis of complex systems within the domains of biology, ecology and psychology.

Both authors worked several years at the Software Verification Research Centre (SVRC), a special research centre of the Australian Research Council, which was active during 1993–2003 in the areas of formal methods tool development and formal verification of industrial software. The authors were employed in a 50-50 research and technology transfer programme. In addition to carry out research in the area of formal methods, as SVRC employees, they provided consultancy services to several organisations and companies, including DSTO (now DST – Defence Science and Technology), British Aerospace, Foxboro, Computer Science Corporation and Santos, as well as training in form of short intensive courses.

The first author then moved to the International Institute for Software Technology of the United Nations University (UNU-IIST), which was located in Macao SAR, China, where for almost 10 years he continued to carry out research in the area of formal methods and was particularly involved in the diffusion of software technology and formal methods in developping countries through the delivery of short intensive courses and supervision of graduate fellows and PhD students. He has also taught Master and PhD courses at UNU-IIST, the University of Pisa, IMT Lucca and Nazarbayev University.

The second author moved to Zurich where he took up a lecturer position at the Zurich University of Applied Sciences (ZHAW). As a member of the University's Safety Critical Systems Research Lab he is doing active research and technology transfer in formal methods in various industrial projects. Under a recently funded University project he developed E-Learning materials to enrich and improve the Bachelor education in mathematics.

3 A Multi-dimensional Engagement Strategy in Formal-methods Education

This section discusses the lessons learned during the authors' teaching, supervision and consultancy activities in the area of formal methods. Since formal methods are not well received by both the academic and industrial audience, the main challenge was to develop a strategy to reverse this trend and keep the learner continuously motivated and engaged in order to retain any acquired form of interest. In the process, we identified a number of dimensions of this engagement strategy, which we illustrate in Sect. 3.1–3.5.

3.1 Motivations

Given the widespread reluctance to learn formal methods, the simple strategy of providing potential learners with a list of reasons for being interested in this area would not be effective. We experienced that a better strategy is, instead, that of enabling learners to build themselves their intrinsic and extrinsic motivations. We have identified a number of tools to achieve this objective:

Start with the general context rather than the foundations
A typical mistake in teaching or even just advertising a challenging subject is that of starting from theoretical foundations and basic technical aspects. Such an approach appears dry and non motivating to potential learners. The result is that the least skilled potential learners will get scared and run away and the most skilled ones will get bored and find little interest in the subject. In our view, based on our experience, motivations can be enabled through an initial, broad presentation of the general context in which formal methods are successfully applied, but without actually putting any emphasis on formal methods themselves and leaving out all technical details. Moreover, when technical details are introduced, in a soft, incremental way, they must always be referred to this motivational context and possibly contribute to extend it. We will present some examples of this approach in Sect. 4.

Present specific success stories and showcases
In spite of industry's general reluctance in accepting formal methods, some companies have actually used them in research projects or in the verification of their software products or deployed systems. There is a number of success stories that could be presented to potential formal methods learners. Preference should be given to popular companies and the success stories should be presented using high-level descriptions, normally available in newspapers, magazine, short communications or internet resources rather than technical journal papers. However, the success story should not be an unrealiastic celebration of a panacea approach but, to be credible, should describe a global positive outcome that includes both pros and cons. A good example in this sense could be the use of formal methods at Amazon Web Services, which is reported as big success but with some remaining caveats [20].

Consider and incorporate current trends
In terms of extrinsic motivation it is important to connect formal methods to the most trendy areas of the moment, which are seen as a must in the job market, a hot topic in research and an essential tool in industrial production, thus appealing to the entire potential audience from students to researcher and industry. A today's example is represented by the hot area of *data science* and its subdisciplines. Providing the intuition on how formal methods connect to data science, using examples on current work [2, 11] as well as ideas for future research, is necessary to boost strong intrinsic motivations.

Start education in formal methods early enough
The absence of the appropriate mathematical background is the biggest barrier for potential formal methods learners. A common belief is that formal methods are very far from people's normal way of thinking and reasoning. But the opposite is actually true. The same problem-solving and reasoning skills needed in real-life can be used to solve problems in the area of formal methods. The only difference is that the reasoning object is not a concrete

fact, but an abstract model. *Abstraction skills* are what enables us to move from the reality to its models, to which formal methods can be applied.

Unfortunately, the current status of mathematics teaching around the world is not addressing abstraction skills [14]. In fact, mathematics should be taught using a *mathematical problem-solving* approach since the early school years [4, 16, 22], in which it is already possible to introduce formal methods [14], and continuing with such an approach during the university years. The opposite seems, instead, to happen in the last years, with schools emphasising on calculation rather than reasoning abilities or on repeated pattern recognition problems that are never finalised to a successful abstraction process [14]. Even at the university level, a fundamental mathematics subject like calculus, which was recognised in the past as the best tool to develop abstraction and reasoning skills through the development of proofs in the real spirit of mathematical analysis, is now restricted to the teaching of mere calculation techniques. Similarly, the formal semantics of programming languages is no longer taught in the early programming courses, which nowadays just focus on syntax.

In such an unfavorable situation, in addition to try to propose innovative approaches to be carried out globally, starting from the early school years, we claim the importance of introducing formal methods already at undergraduate level, both in core subjects, such as programming, software engineering and operating systems as well as in elective subjects, such as human-computer interaction, information security and project-based electives. We will discuss these proposals in Sect. 4.1–4.2.

3.2 Fun

One important aspect of formal methods is the possibility of combining notations that support problem specification with powerful tools that, given the specification as an input, provide problem solutions almost automatically. The authors are always impressed by the combined feeling of surprise, happiness and sense of achievement externalised by learners when they realise that their specifications actually “work” with the tool.

Moreover, formal methods can be applied to a large range of problems, basically any problem, well beyond the domain of computer science. In fact, in addition to classical computer science problems, such as the dining philosopher, communication and cryptographic protocols and distributed algorithms, we have a huge range of candidates among classical mathematical puzzles as well as popular games and even video games.

Mathematical puzzles that can be visually represented, such as the “river crossing puzzle” [3], present an easy way to approach formal methods; they allow learners to gradually move from the graphical representation to a mathematical notation. This kind of problems, which have a clear visual representation, are particularly suitable in school and early undergraduate courses [13].

Other more complex mathematical puzzles and popular games, such as sudoku and card games, have the potential to strongly engage learners [12]. At the end of a midtem examination that consisted in the modelling and formal analysis of a card game, one of the authors had the nice experience to hear a student saying: "This examination was real fun!"

The authors often provide learners with examples of formal methods descriptions of video games and invite them to create formal models of their favourite video games. This kind of tasks appeared to be very engaging and can be successfully carried out even in high school and early undergraduate courses. In this context, it is particularly important to blur the distinction between learner and instructor by letting the learners drive the choice of exercises and use their creativity to identify and specify potential problems and invent new games.

In general, we can claim that, if "motivation" is the dimension that allows learners to build up interest in formal methods, "fun" is actually the essential dimension to keep learners continuously engaged, thus assuring the retention and possibly increase of their interest over the time [6, 10].

3.3 Which Formal Methods?

In our teaching and supervision activities our students have been exposed to a large variety of formal methods including the Z specification language, the refinement calculus, Petri nets, process algebras, and several logic systems, from rewriting logic to temporal logic. And this has been done both theoretically and practically using theorem-proving and model-checking tools.

From our experience we observed that the choice of which formal methods to present to the learners mostly depends on a three parameters:

1. the age, level and background of the learners;
2. the application domain, which may be identified with the taught subject in the case of university students;
3. the availability and features of software tools.

Some specific discussion on Parameter 1 will be presented in Sect. 3.4 and 4. Parameter 3 will be discussed in Sect. 3.5.

Unfortunately, we cannot establish general rules for using such parameters to drive our choice, which actually depends not only on the characteristics of the learners, but also on the preference, background and skills of the instructor. Therefore, we limit the discussion in this section to the account of the first author's experience in teaching several variants of a postgraduate course in formal methods to a variety of audiences at postgraduate level

- in several developing countries as part of the United Nations University training programme;
- in PhD courses in Macao SAR (at UNU-IIST) and Italy (at the University of Pisa and at the IMT School for Advanced Studies Lucca); and
- in Master Courses at Nazarbayev University, Kazakstan.

Teaching to university lecturers and postgraduate students in developing countries was very challenging. In addition to the logistic and infrastructural problems of such in-house courses, the biggest challenge was the limited mathematical background of the learners, but still with a large variability, which could not predicted *a priori*. The strategy for dealing with this challenge was an on-the-fly adaptability of such courses, which definitely contributed to the development of the proposals and approaches illustrated in this paper. Being these teaching contexts very specific, a detailed account on such experiences is beyond the scope of this paper.

Three different formal methods approaches used in the course were

- RAISE (Rigorous Approach to Industrial Software Engineering) and its specification language (RSL) and associated tools;
- the CSP (Communicating Sequential Processes) process algebra, initially with the support of the CWB-NC (Concurrency Workbench of the New Century) tool, later replaced by PAT (Process Analysis Toolkit);
- rewriting logic and the Maude language and model checker.

The use of RAISE was soon abandoned due to the difficulties encountered by the students in producing consistent specifications and to the poor usability of the associated tools. Therefore, we only compare the process algebra and rewriting logic approaches.

In the PhD course taught at IMT during the academic year 2014–2015, both formal methods approaches were introduced and specifically applied to the modelling of interactive systems. The translations of a description language for human behaviour tasks to both formal methods were presented during the course. However, the way that course was conducted does not reflect the approach proposed in this paper. In fact, the first part of the course was devoted to the theoretical presentation of the two formal methods approaches and to pen and paper modelling exercises. Only in the second part of the course the PAT and Maude tools were introduced. At the end of the first part of the course, students were asked three questions:

1. “In which of the two approaches did you find easier to get the model right?”
2. “Which of the two translations is more elegant?”
3. “In which of the two approaches the resultant behaviour is easier to guess?”

The PhD students unanimously answered “the rewriting logic approach” to Questions 1 and 3, and “the process algebra approach” to Question 2. It is interesting to observe that, in spite of finding the process algebra approach more difficult, the student unanimously agreed that it is more elegant. These answers, as well as further remarks and opinions that emerged in an open discussion that followed, are an indicator that students have a strong interest for solutions that are concise, elegant and abstract, and that they are happy to tackle challenging problems in order to look for elegant rather than easy, but somehow messy solutions. In this specific case, the “elegant challenge” was the use of concurrency in modelling the system in a compositional way, whereas the “easy but messy solution” was the monolithic modelling of the global system using rewrite rules. Given the small number of students and the absence of research design we cannot draw empirical conclusions from the students’ answers and remarks, although these appear to be in line with the results of previous research [26].

With respect to Parameter 2 above, the students' answers seem to suggest that rewriting logic is more suitable than process algebra and other approaches based on parallel composition to model human behaviour. As we will see in Sect 3.5 this cannot be a definite conclusion.

3.4 Textual versus Visual Notations

In the case of school children simple, visual notations, such as Petri nets and finite state machines are obviously the best choices for introducing formal methods. Several formal methods concepts, such as refinement, abstraction and concurrency, can be directly identified on the visual representation, in most cases with no recourse to formal mathematical representations. The important thing for school children is the discovery and internalisation of such concepts rather then their representations in some dry textual notation.

Visual notations also help a lot in the case of undergraduate students, but need to be finalised to the "discovery" of the formal semantics and its possible representations in mathematical notations. For example, Petri nets can be first introduced visually together with an informal presentation of their semantics, or actually their possible semantics. Then the students can be guided to represent such semantics in a mathematical way that can be used to calculate the future behaviour of the system. In the case of Petri nets students may visually identify and represent the semantics by

1. decomposing the net to represent each arc in terms of its sources and target;
2. decomposing the net to represent each transition in terms of its sources and targets;
3. drawing a table transition × places for the entire net;
4. analising for each transition all markings enabling it.

Students would also observe that place may have their token produced by different transitions and that different place may cooperate to the firing of the same transition. Thus it does not make sense to represent a place in terms of the transitions that separately produce token in it and the transition that may separately consume tokens from it. The three representations above correspond respectively to

1. a flow relation, which can be expressed as two boolean functions on incoming and outgoing arcs;
2. the pre-set of post-set of each transition;
3. an incidence matrix;
4. a partial function from a marking to a set of markings, one for each enabled transition.

As the next step, depending on which the four representations above they have worked out, the students can be guided to discover the way to calculate the future behaviour, that is, the formal semantics of the Petri net. Finally, the equivalence of the various representations can be discussed.

3.5 Practice and Tools

In Sect. 3.3 we have reported students' opinions in the comparison of parallel composition and rewriting logic in modelling interactive systems, specifically human behaviour tasks. These opinions were collected after introducing the theory but before introducing the tools and starting using them. However, at the end of the course, after using both PAT and Maude, the opinions of the students were substantially unchanged.

More recently, at Nazarbayev University, the two approaches were used in the same postgraduate course on formal methods and applications as well as separately in two distinct instances of the undergraduate course on human-computer interaction. In these cases the approaches have been introduced together with the usage of the tools. In fact, the tools were used to introduce the language constructs and their semantics. Although a complete comparison cannot be carried out for the undergraduate courses since each students was exposed to only one of the two approaches, a better performance was achieved by the students exposed to the process algebra approach. Moreover, the postgraduate students found easier to use the process algebra approach than the rewriting logic approach.

The general lessons learned from these experiences are that [10]

1. instead of tediously going through the semantics of each construct in a formal language, students should be allowed to experiment with an appropriate tool to discover the semantics by themselves;
2. tools for simulation visualisation are essential to allow students to understand the behaviour associated with their models.

In fact, the introduction and usage of tools appear beneficial only if done early enough. If tools start to be used only after introducing the theory, it is difficult to reverse students' negative opinions and feelings.

Moreover, in general, in order to be beneficial to formal methods learners, tools should not require additional learning time and should, instead, facilitate the learning process. Thus they have to be easy to learn, at least in their basic features, documented in a concise but well-organised way and equipped with visual interfaces.

MAUDE and PAT are somehow complementary in terms of presentation of results, also due to the different characteristics of the formal methods on which they are based. MAUDE does not support any form of graphical representation but supports a form of under-approximation [17], by filtering the output through additional rewrite rules, and allows the designer to easily track which rewrite rule is applied and check the content of all data structures, thus tracking the behaviour back to the architectural view of the designer. PAT facilitates the visual representations of the global behaviour in terms of finite state machines, but the form of under-approximation introduced by the CSP hiding operator is not very effective due to the possible introduction of nondeterminism, while the represented behaviour does not reflect the structure, in terms of concurrent components and synchronisations, from which the global behaviour has been attained. However, the use of both these tools in our course has allowed students to make use of all needed presentation features, visualisation from PAT, underapproximation and behaviour tracking from MAUDE. Moreover, in our class discussions, students showed the perception that the fact that the two tools are based on two distinct modelling paradigms contributed to stimulate and develop their abstraction and problem solving skills.

MAUDE documentation is well-written and presented at three levels: a primer, which allows a user to be able to effectively use the tool within a short time, a textbook specifically designed for an undergraduate course, but also appropriate for postgraduate courses, and a comprehensive manual for consultation and for acquiring a more advanced level of expertise. PAT documentation is unsatisfactory, especially for a novice. It consists of a poorly organised manual, several research papers, several talk presentations, materials on experiments and a couple of advertising videos. None of this material is really suitable for a learner.

4 Types of Target Audience

4.1 University Students

University students require a good balance between intrinsic and extrinsic motivations. We have discussed in Sect. 3.1 that motivations can be enabled through an initial, broad presentation of the general context in which formal methods are successfully applied. For university students this can be done at two levels:

1. sparsely in core subjects, such as programming, software engineering and operating systems, and in elective subjects, such as human-computer interaction and information security.
2. in a focussed way, within a specific formal-methods-related subject.

Level 1 is the most effective in boosting early intrinsic motivations, which can play a decisive role later at the time of choosing elective subjects and thesis topics. In fact at this level, the presentation of the context in which to apply formal methods should be very broad, e.g. the integration of formal verification within the software life-cycle, in a software engineering course, or the simulation and formal analysis of human behaviour, in a human-computer interaction course. Showing the "pleasant aspects" of using formal methods is the actual objective at this level. For example, students of a software engineering undergraduate core subject are more likely to enjoy building a concise formal specification, with which they can also play using a tool, rather than writing a long, verbose and certainly boring specification document. Students of a human-computer interaction undergraduate elective subject taught by the first author enjoyed the formal modelling and analysis of a variety of small human tasks, including classical ones such as the interaction with an ATM (automatic teller machine) and the general car driver's behaviour, but also a number of fun, in some respect even hilarious, examples such as failing a driving test and baking a cake.

Of course, the feasibility, simplicity and fun of the tasks proposed by the teacher or, better, agreed between students and teacher, are essential for determining a pleasant rather than frustrating experience. In fact, although formal methods tend to be time consuming when applied to large systems, for several classes of small examples (e.g. the ones that can be systematically decomposed or are naturally recursive) they are indeed effective in saving time and reducing the workload, thus also boosting extrinsic motivations. However, in our experience among the motivations generated at this level, intrinsic motivation are probably going to be more long-lasting, especially for junior students. Future references to these contexts within the same subjects or even in other subjects would normally bring back intrinsic motivations. Senior students, who are often already looking for a job, are instead also very much affected by extrinsic motivations.

When dealing with specific formal-methods subject (level 2 above), the presentation of the context in which to apply formal methods should be more focussed on the learning objectives of the course. In the Master course on formal methods and applications at Nazarbayev University, formal methods were presented first in the general context of the software life cycle, then in terms of more specific domain-oriented development, specifically their application to interactive systems, and finally in the light of synergetic approaches with trendy areas such as data science.

As a final remark concerning the use of tools, we would like to add that, from the perspective of a university student, it is important to see simulation and model-checking results directly on the low-level semantic structures underlying high-level domain structures. This is, in fact, an effective way for the student to understand and internalise the semantics of the language. Moreover, in the students' perspective, the presentation of results must aim at highlighting relations between behaviour and semantics and using under-approximation [17]. In fact, such a capability to output only relevant states and/or events is beneficial in stimulating and developping students' abstraction and problem solving skills.

4.2 School Pupils

For primary school pupils only intrinsic motivations make sense. In intermediate and high school, instead, also extrinsic motivation start to play a significant role.

Although in teaching to school pupils our experience is limited to research projects and practice with our own children, we agree with Gibson [14] about the importance of using formal methods to allow school pupils to establish early enough the mathematical problem-solving bases that can enable them to succeed in scientific and technology-oriented university programmes. Of course this has to be done at the right level and with the appropriate learning objectives.

The main challenge in teaching computer-science-related skills to school pupils is to make them aware that such skills have a general value, which is independent of the use of computers. For this reason we support an "unplugged approach" to teaching formal methods to school pupils, using activities that foster children reasoning and do not require the use of a computer [4]. In fact, this should be done in a multidisciplinary context; teaching formal methods should build on all school subjects, which, in the world of the school pupil, represent the most natural reality to be modelled formally. Obviously, mathematics should be the first subject to provide materials to manipulate in a formal fashion. However, all other subjects have also plenty of materials on which students may carry out modelling and analysis.

Moreover, as we discussed in Sect. 3.4, the emphasis should be on the discovery of concept through the use of visual representations and the visual manipulation of such concepts aiming at their internalisation rather than their translation into some dry textual notation. If fact, we note that it would be pointless to just provide children with the definitions of new notions, concepts and processes, such as algorithms, and hope they understand them, remember them and are then able to apply them to practical situations. Children learn best if they are actively involved in the process through problem-solving [22].

4.3 Industry

The most natural use of formal methods should be in the area of industrial software verification. However, given the high cost required by the use of formal methods, in terms of human and economic resources as well as time, the software development industry partly accepts the use of formal methods only for safety-critical systems. This partial acceptance is often not even a choice but the legal need to comply with the standards (e.g. IEC 61508 [18]) that suggest the use of formal methods for the most dependable software integrity level (SIL). Although no standard prescribes the use of formal methods as mandatory, the appeal to standards' recommendations is an effective incentive to offer practical courses on formal methods to industries working in the area of safety-critical systems. Additional enablers for extrinsic motivations for industry are success stories and showcases as we described in Sect. 3.1.

Moreover, rather than proposing or, even worse, imposing formal methods as a new approach, a better strategy is to present formal methods as integrated, or integrable, within a context which industry is familiar with. For example, ZHAW provides training and consultancy to safety-critical systems industry, such as railway, process industry, pharmaceutical industry, nuclear power plants and transportation, using STAMP (System-Theoretic Accident Model and Processes), a model-based approach centred on system theory to analyse accidents [19]. STAMP has been developed by Nancy Leveson as a model for safety engineering along with the System Theoretic Process Analysis (STPA) method, a hazard analysis method for finding inadequate design. This powerful top-down hazard analysis methodology has proven to be applicable in various industrial applications, including automotive, avionics, health care, power plant, railway, and many others [1]. It has also been successfully applied in the planning and technical system development and on already existing systems. The ZHAW Safety Critical Systems Research Lab experiences a growing demand for STPA analysis in a widespread variety of industrial areas, such as power plants, railway, health care and automotive.

In STPA, safety is viewed as a control problem, which is actually the natural way nuclear power and transportation engineers use to describe problems, and is managed by a control structure embedded in an adaptive socio-technical system and acting as constraints on the system. Therefore, STAMP models, being at the socio-technical system level can be clearly understood by engineers and other domain and safety experts.

The project "Formal Analysis and Verification of Accidents", a collaboration between the ZHAW Zurich University of Applied Sciences, Winterthur, Switzerland, and Nazarbayev University, Nur-Sultan, Kazakhstan, aims at the combination of a cognitive architecture for the formal analysis of human-computer interaction [7] and its associated description language, the Behaviour and Reasoning Description Language (BRDL) [8], with the STAMP approach. The cognitive architecture has been implemented using a formal methods approach based on Maude [5, 9], which supports formal verification using model checking. The basic idea is to identify the steps in the STPA method that are suitable for formalisation, develop analysis engines based on model-checking and encapsulate them within tools equipped with high-level editors and interfaces appropriate for the usage by engineers and domain experts. The final objective of the project is to allow our industrial partners to use STPA-based tools, which also provide, in an unintrusive way, automated formal verification capabilities.

Finally, in terms of tools, we observe that the perspective of industry practitioner is very different from that of university students: they prefer tools that hide the formal semantic structures underlying domain structures.

4.4 Interdisciplinary Research Teams

Formal methods methodologies and tools can potentially be applied to several disciplines, not just computer science and the area of sofware-critical systems. Formal modelling and analysis can be potentially exploited as effective research tools in many disciplines such as physics, chemistry, biology, ecology, psychology, cognitive science and economics. Unfortunately, formal methods experts are often so much focussed on the investigation of theoretical aspects of formal notations rather than on their applications to real problems, that they often neglect the needs of practitioners from applicative domains. The result is that methodologies and tools have limited usability for non computer scientists.

This problem has become very actual nowadays, with the launching of large interdisciplinary research projects, especially in areas such as biology, ecology and cognitive science. Different categories of experts within the same interdisciplinary research team may experience difficulties in understanding each other and share thoughts, due to both the different technical languages they use and their different way of reasoning. In research projects involving the application of formal methods to systems biology, ecosystem modelling and analysis of human behaviour and human errors, it is normally the formal methods expert who make the effort to understand the application domain, whereas the domain experts tend to act just as data providers or, in the best case, as consultants. And too often this leads to the development of unrealistic models and analysis tools with limited scope.

There is a need to change this situation. The most effective effort from the formal methods expert should actually be a technology transfer to domain experts in a form suitable to them. The embedding of formal methods within domain specific languages [8] and domain specific tools [9] represents an essential step in this direction. In fact, with this kind of audience it is necessary to use methodologies and tools that support domain specific notations, human-oriented proof and checking techniques, and domain-related formulation of properties. In this sense we could speak of *human-oriented* formal methods [6].

5 Conclusion and Future Work

In this paper we gave an account of our experience in teaching formal methods to various kinds of audience and identified and discussed a number of dimensions that drove the development of our engagement strategy. We observed that, on the one hand, these dimensions apply to such various kinds of audience differently. For example, the use of tools is not recommended for school pupils, is essential for university students in understanding and internalising semantic aspects, requires the hiding of the formal semantic structures underlying domain structures for industry practitioners and applicative domain researcher and, for the latter, also requires domain-specific notations and domain-related formulation of properties. On the other hand, we observed that, for any kind of audience, motivation is the dimension that allows learners to build up interest in formal methods, while fun is actually the essential dimension to keep learners continuously engaged, thus assuring the retention and possibly increase of their interest over the time.

In terms of future work we plan to develop teaching-oriented formal methods tools appropriate to difference audiences and, within the "Formal Analysis and Verification of Accidents" project the embedding of formal methods within methodologies that are widely accepted in industrial contexts (e.g. STPA).

References

1. Partnership for systems approaches to safety and security (PSASS). Web page. http://psas.scripts.mit.edu/home/materials/.
2. A. Aibassova, A. Cerone, and M. Tashkenbayev. An instrumented mobile language learning application for the analysis of usability and learning. In *FM'19 Collocated Workshops (DataMod)*, Lecture Notes in Computer Science. Springer, 2019. In press.
3. M. Ascher. "a river-crossing problem in cross-cultural perspective". *Mathematics Magazine*, 63(1):26–29, February 1990.
4. T. Bell. A low-cost high-impact computer science show for family audiences. In *23rd Australian Computer Science Conference*, pages 10–16. ACM, 2000.
5. A. Cerone. A cognitive framework based on rewriting logic for the analysis of interactive systems. In *Software Engineering and Formal Methods (SEFM 2016)*, number 9763 in Lecture Notes in Computer Science, pages 287–303. Springer, 2016.
6. A. Cerone. Human-oriented formal modelling of human-computer interaction. In *STAF 2016 Collocated Workshops (HOFM)*, volume 9946 of *Lecture Notes in Computer Science*, pages 232–241. Springer, 2016.
7. A. Cerone. Towards a cognitive architecture for the formal analysis of human behaviour and learning. In *STAF collocated Workshops 2018 (FMIS)*, number 11176 in Lecture Notes in Computer Science, pages 216–232. Springer, 2018.

8. A. Cerone. Behaviour and reasoning description language (BRDL). In *SEFM 2019 Collocated Workshops. In press.*, Lecture Notes in Computer Science. Springer, 2019.
9. A. Cerone and P. C. Ölveczky. Modelling human reasoning in practical behavioural contexts using real-time maude. In *FM'19 Collocated Workshops (FMIS)*, Lecture Notes in Computer Science. Springer, 2019. In press.
10. A. Cerone, M. Roggenbach, B.-H. Schlingloff, G. Schneider, and S. Shaikh. Teaching formal methods for software engineering — ten principles. *Informatica Didactica*, 9, 2015.
11. A. Cerone and A. Zhexenbayeva. Using formal methods to validate research hypotheses: The duolingo case study. In *STAF 2018 Collocated Workshops (DataMod)*, volume 11176 of *Lecture Notes in Computer Science*, pages 163–170. Springer, 2018.
12. J. F. Ferreira and A. Mendes. The magic of algorithm design and analysis: teaching algorithmic skills using magic card tricks. In *Proc. of ITiCSE 2014*. ACM, 2014.
13. J. F. Ferreira and A. Mendes. Open and interactive learning resources for algorithmic problem solving. In *Proc. of FM'19 Collocated Workshops (OpenCERT)*, Lecture Notes in Computer Science. Springer, 2019. In press.
14. J. P. Gibson. Formal methods: Never too young to start. In *FORMED 2008*, pages 151–160, Budapest, Hungary, March 2008.
15. R. L. Glass. A new answer to "how important is mathematics to the software practitioner?". *IEEE Software*, 17(6):136–136, 2000.
16. P. Hilton. The mathematical component of a good education. *Miscellanea Mathematica*, pages 145–154, 1991.
17. A. Idani and N. Stouls. When a formal model rhymes with a graphical notation. In *SEFM 2014 Collocated Workshops (HOFM)*, volume 8938 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 2014.
18. IEC 61508-1. *Functional safety of electrical/electronic/programmable electronic safety-related systems — Part 1: General requirements*, 2.0 edition, April 2010.
19. N. Leveson. A new accident model for engineering safer systems. *Safety Science*, (42):2, April 2004.
20. C. Newcombe, T. Rath, F. Zhang, B. Munteanu, M. Brooker, and M. Deardeuff. How amazon web services uses formal methods. *Communications of the ACM*, 58(4):66–73, 2015.
21. R. L. Page. Software in discrete mathematics. In *Proc. of ICFP 2003*, volume 38 of *ACM Sigplan Notices*, pages 79–86. ACM, 2003.
22. A. H. Schoenfeld. *Mathematical Problem solving.* Academic Press, 1985.
23. A. E. K. Sobel and M. R. Clarkson. Formal methods application: an empirical tale of software development. *IEEE Trans. Softw. Eng.*, 28(3):308–320, 2002.
24. J. M. Wing. Teaching matematics to software engineers. In *Proc. of AMAST 1995*, volume 936 of *Lecture Notes in Computer Science*, pages 18–40. Springer, 1995.
25. J. M. Wing. Weaving formal methods into the undergraduate computer science curriculum. In *Proc. of AMAST 2000*, volume 1816 of *Lecture Notes in Computer Science*, pages 2–9. Springer, 2000.
26. A. Zamansky and E. Farchi. Exploring the role of logic and formal methods in information systems education. In *SEFM 2015 Collocated Workshops (HOFM)*, volume 9509 of *Lecture Notes in Computer Science*, pages 68–74. Springer, 2015.