# The 2020 Expert Survey on Formal Methods

Hubert Garavel[1], Maurice H. ter Beek[2], and Jaco van de Pol[3,4]

[1] Univ. Grenoble Alpes, INRIA, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France
[2] ISTI–CNR, Pisa, Italy
[3] Aarhus University, Denmark
[4] University of Twente, The Netherlands
hubert.garavel@inria.fr, maurice.terbeek@isti.cnr.it, jaco@cs.au.dk

**Abstract.** Organised to celebrate the 25th anniversary of the FMICS international conference, the present survey addresses 30 questions on the past, present, and future of formal methods in research, industry, and education. Not less than 130 high-profile experts in formal methods (among whom three Turing award winners and many recipients of other prizes and distinctions) accepted to participate in this survey. We analyse their answers and comments, and present a collection of 111 position statements provided by these experts. The survey is both an exercise in collective thinking and a family picture of key actors in formal methods.

**Keywords:** cybersecurity · education · formal method · modelling · safety · software engineering · software tool · specification · survey · technology transfer · verification

## 1 Introduction

FMICS, the international conference on Formal Methods for Industrial Critical Systems, is celebrating its 25th anniversary. The FMICS community recognised the revolutionary potential of verification technology quite early on. Its members devoted their energy to evolve this technology, and to apply it to the verification of complex industrial critical systems. These 25 years have brought numerous highlights, like better specification languages, more efficient verification algorithms, landmark tools, and academic recognition in the form of awards. But also many successful industrial applications, the rise of "verification engineer" as a new job title, and the advent of industrial laboratories that focus on formal verification technology.

After decades of glory, formal methods seem at a turning point. In industry, many engineers with expertise in formal methods are assigned new priorities, especially in artificial intelligence. At the same time, the formal verification landscape in higher education is scattered. At many universities, formal methods courses are shrinking, likely because they are deemed too difficult. The transmission of our knowledge to the next generation is not guaranteed. So we cannot lean back.

As part of the celebration, and in order to address this turning point, we have conducted a survey among a selection of internationally renowned scientists who have played a big role in formal methods, either within the FMICS conference series, or outside of it. We report on their collective vision on the past, present, and future

of formal methods with respect to research, industry, and education. What did we achieve? What did we miss? Where should we go?

**Related Work.** Early introductions to the application of formal methods are those by Wing [22] and Rushby [19]. The 1996 survey by Clarke and Wing [8] illustrates many case studies in specification, theorem proving, and model checking. Other classical texts that reflect on the industrial application of formal methods use the metaphors of seven myths [15] or ten commandments [6].

We list a few more recent historical overviews of formal methods. A 2009 survey [23] reports about the application of formal methods in 62 industrial projects; that paper also provides an interesting overview of 20 earlier surveys on formal methods in industry from 1990 to 2009. The handbook [14] published by the FMICS community in 2012 presents applications of formal methods in various domains of industrial critical systems. An 2013 study [11] provides a synthetic account of the diverse research in formal methods, including a list of 30 carefully selected, well-documented case studies that illustrate the progress in formal methods during the period 1982–2011. A history of 40 years of formal methods [5] includes an analysis of some obstacles to their application, while [17] focusses on their history in the UK.

Other papers examine the adoption and industrial strength of formal methods. Three recent surveys with stakeholders [4] investigate what are the most prominent formal methods styles used in the railway domain and the expectations railway practitioners have from formal tools [3]. In a follow-up experimental study [10], a panel of experts judges the suitability of nine formal methods for the specification and verification of (sub)systems in that domain. Barriers to the adoption of formal methods in aerospace are considered in a survey [9] among 31 individuals from nine organisations: the top three barriers stem from education, software tools, and the industrial environment. Multiple contributions have been made for lifting these respective barriers: [7] proposes a coherent formal methods curriculum in higher education; [20,12] reflect on the development of software tools to make it more efficient and relevant, while software competitions [2] help to enhance the quality and visibility of tools; [18] provides economical evidence by demonstrating the benefits of the application of formal methods to industrial-strength problems. Finally, a recent position paper [16] discusses some obstacles and enablers for the application of formal methods, and translates them to actionable recommendations for industry, academia, and policy makers, to improve the situation.

**Outline.** The present report is organised as follows. Section 2 exposes the methodology used for our survey. The next five sections present and discuss the responses, which are organised in five themes: assessment of formal methods (Section 3), formal methods in research (Section 4), formal methods in industry (Section 5), formal methods in education (Section 6), and the future of formal methods (Section 7). Finally, Section 8 gives concluding remarks and Section 9 presents the 111 position statements collected during the survey.

## 2 Survey Methodology

This section presents the main decisions concerning the organisation of the survey.

### 2.1 Participants

Initially, the plan was to centre our survey around FMICS, from its origins to our times, by asking all FMICS working group chairs, all FMICS programme committee chairs, and all FMICS invited speakers to participate in the survey. This gave a list of 94 names, much longer than that of the 1996 survey on formal methods [8], which involved 27 participants. But it became clear that our survey would benefit from an even larger panel of experts. We then started adding further names of key players in the field, based upon personal knowledge, discussions with colleagues, and taking the extensive 92-page bibliography of [11] as a source of inspiration. This resulted in a list of 230 names, which, unfortunately, was too long, since we wanted to offer each participant the possibility to write a 10-line position statement, but had only a limited number of pages in the present LNCS volume. We then devised a thorough selection procedure, based on individual scores and other criteria, in order to retain only 170 names from the list of 230. Doing so, we tried to achieve a good coverage of academia and industry, hardware and software, global corporations and technology startups, etc., as well as a fair balance between the various styles of formal methods and a suitable geographical diversity, making sure to invite experts from most countries with a notable activity in formal methods. As the three survey organisers, we decided to exclude ourselves from the list of participants.

### 2.2 Questions

Through a long iterative process, we progressively elaborated a set of 30 questions for our survey. These questions are divided into 5 themes: *assessment* of formal methods (5 questions), formal methods in *research* (6 questions), *industry* (9 questions), and *education* (5 questions), and the *future* of formal methods (5 questions).

Concerning the content, most of the questions derived from our own professional experience in developing software tools, collaborating with various industries, and teaching formal methods at several universities. We also drew inspiration from other sources, among which [11,20,12,16]. For each question, we proposed a set of predefined, Likert-scale[5] answers and, whenever possible, we added an *Others* alternative in case these answers would not be found relevant. We deemed many of these questions to be difficult, in the sense that we had no obvious answers for them; instead, we were curious to see all the answers given by our colleagues to figure out what was the opinion of the formal methods community on such matters. Some questions were even intentionally provocative, in order to push reflections out of the comfort zones.

[5] https://en.wikipedia.org/wiki/Likert_scale

Concerning the form, we chose to use the open-source LimeSurvey[6] software, an instance of which was already installed and freely available on an INRIA server, because this choice provided the best privacy guarantees for the experts. We thus implemented our 30 questions as an online LimeSurvey questionnaire to be filled in by the experts. For each question, we used the most appropriate LimeSurvey template, depending on whether the question had mutually exclusive answers (represented with round buttons) or multiple-choice answers (represented with square buttons). In the latter case, we often imposed a higher bound on the number of answers that experts could select, thereby forcing them to exclude approximately 33% (at least) of the proposed answers and keep only the most relevant ones. Also, whenever possible, the lists of answers were proposed in random order to eliminate option order bias (i.e. the tendency to pick the first or last answer option).

We prepared four successive beta-versions of the questionnaire and had it pretested by nine reviewers from four different countries. Their feedback helped us to improve the questionnaire through successive iterations.

### 2.3 Survey

To ease the practical management of the survey, we split the list of 170 experts into two groups of 100 and 70 people, respectively. Both groups were invited to fill in the LimeSurvey questionnaire within two successive time frames (June 3–14 and June 17–28, 2020). Each expert received one invitation and, possibly, two reminders by e-mail. In addition, intensive e-mail exchanges took place between the three survey organisers and certain experts, to provide them with more information about the survey, adapt their position statements to formatting constraints, and/or recover from technical issues with LimeSurvey (eventually, no input data was lost),

We received 130 responses after sending 170 invitations. Such a response ratio of 76% seems particularly high for an online survey. A few experts declined participation in the survey, while others remained silent. Some experts initially promised to participate in the survey, but eventually did not because they were too busy with students or peer reviews. After expiration of the deadline, in July, we also received, from a few experts, offers to participate, which we unfortunately had to decline.

In spite of the 40 missing responses, we are most happy to count, among the high-profile participants to our survey, three Turing Award winners: Hoare (1980), Emerson (2007), and Sifakis (2007); all the three recipients of an FME Fellowship Award: Jones (2015), Broy (2018), and Meseguer (2019); thirteen CAV Award winners: Alur (2008), Dill (2008), Rajamani (2011), Rushby (2012), Larsen (2013), Wang Yi (2013), Godefroid (2014), Peled (2014), Valmari (2014), Grumberg (2015), Abdulla (2017), Biere (2018), and Cimatti (2018); as well as the recipients of many other awards and distinctions that we do not list here exhaustively.

### 2.4 Answers

In total, 130 experts replied to our 30 questions. Most of them also answered a 31st additional question, which was a request to (optionally) provide a short (10-line)

[6] https://en.wikipedia.org/wiki/LimeSurvey

position statement (cf. Section 9). The statistics recorded by LimeSurvey indicate that the mean time spent by each expert on the questionnaire was 90 minutes (and 36 seconds), while the median value was 56 minutes (and 24 seconds). Actually, the real durations are probably longer, for at least three reasons: (i) due to LimeSurvey problems, a few experts had to restart their questionnaire from scratch, and their initial attempts are not counted; (ii) many experts chose to complete their 30 answers first, and write their position statement offline to send it later by e-mail; (iii) there have been iterations with many experts to finalise their position statements. In any case, the aforementioned timing statistics represent an important collective effort from the formal methods community.

Using the LimeSurvey features, the answers of all experts were aggregated to produce, for each question, cumulative statistics, which are presented in Sections 3–7. Because it was specified that all answers to the 30 questions would remain anonymous, we considered each question in isolation and made no attempt at tracking or correlating the answers of a given expert across different questions. For the same reason, we did not try to analyse the answers using personal information about the respondents, such as country, place of work, hardware or software background, teaching activities (if any), etc.; in particular, our questionnaire did not ask for any information about the profile of participants.

### 2.5 Comments

For most questions, the questionnaire proposed a comment field in which the experts could input some text to express their opinions in more detail. Our idea was that such comments would be extremely valuable, and we intended to use them as a basis for discussing the findings of the survey, thus avoiding the pitfall of presenting statistical results only.

Such a possibility was greatly appreciated by the experts, and we received a large volume of comments (namely, 5000+ lines of 80 characters, corresponding to 111 pages of text in LNCS format) that exceeded our expectations by far. Given that all these comments could not be quoted in the present report, we had to make a selection, which raised a triage problem. A careful examination of comments led us to dispatch them into several categories:

- A *critical comment* expresses the dissatisfaction of the expert with the question and/or its proposed Likert-scale answers. For instance: “just a weird question”.
- An *explanatory comment* gives the justification for the particular answer chosen by the expert. For instance: “too much irrelevant ‘nice’ theory”.
- A *restrictive comment* defines the conditions in which the proposed answer is valid. For instance: “depends on the industry”.
- An *alternative comment* provides an alternative answer (typically associated with the *Other* answer) and/or justifies this choice. For instance: “governments/states (through regulations)” to answer a question asking who can best drive the adoption of formal methods in industry.
- A *redundant comment* does not provide new information. Example: the answer “yes” accompanied by the comment “there is no doubt about this”.

– A *conflicting comment* introduces a contradiction with the answer it accompanies. For instance: “I chose ‘probably not’ but I have no opinion in fact”. In such cases, we kept the answer as it was and discarded the conflicting comment. Such situations were rare and, thus, statistically negligible.
– A *misplaced comment* does not address the current question, but another question discussed elsewhere in the survey. Most often, “elsewhere” means “later”, i.e. the respondent has anticipated on a question yet to come. In such cases, we either discarded the comment or moved it to the most appropriate question.

Such a classification was not always easy, especially for long comments (e.g. from 5 to 10 lines of text) that contained different ideas. But we did our best to process all comments and quote many of them in Sections 3–7. Some contents are ironic, or even sarcastic; mentioning them does not mean that we necessarily endorse their point of view.

The analysis of comments revealed an issue that we had not anticipated. Most questions of the survey deal with general topics such as past, present, and future of formal methods, as well as human factors, economical considerations, impact on industry and society, etc. The answers to such questions cannot be fully formal; instead, they are subjective opinions, reflected in the proposed Likert-scale options (“definitely”, “probably”, “probably not”, “to a limited extent”, etc.). Moreover, to keep the survey short and knowing that the invited experts are busy people, we tried to provide concise questions, without a lengthy set of preliminary definitions, taking as granted a number of common expressions. After submitting the questionnaire, we got some negative reactions, as the imprecision of our questions was antithetic to the culture, based on mathematical rigour, of formal methods experts. In particular, the first two questions, which we expected to be easy, made certain experts unsure and raised criticisms due to missing definitions (“what is the meaning of ‘trustworthy’?”; “how do you interpret ‘quality’?”; “what is the exact difference between ‘partial failure’ and ‘partial success’?”; etc.). We believe that these questions discouraged a few experts to further consider the questionnaire.

### 2.6 Terminology

The term *formal methods* progressively evolved over time, starting from a narrow initial definition to a broader meaning that covers a plethora of methods and tools applied all along the design life cycle, from the elicitation of requirements and early design phases to the deployment, configuration, and run-time monitoring of actual systems. At present, formal methods encompass multiple, diverse artefacts, such as the description of the environment in which the system operates, the requirements and properties that the system should satisfy, the models of the system used during the various design steps, the (hardware or software) code embedded in the final implementation, etc. Formal methods can be used to specify these artefacts and express conformance relations between them.

Being aware of this evolution, we gave a definition of formal methods on the welcome page of the LimeSurvey questionnaire, to make sure that all respondents

would agree on a common definition before answering the 30 questions of the survey. We adopted a modern, inclusive point of view by defining formal methods as "mathematics-based techniques for the specification, development, and (manual or automated) verification of software and hardware systems". However, when analysing the comments received (this is also manifest when reading some of the position statements in Section 9), we observed at least four different interpretations of the perimeter and scope of formal methods:

- The *extensive mathematical interpretation* assumes that any use of mathematics in computer science is part of formal methods. To us, this definition is too wide; for instance, the use of linear algebra in computer graphics is usually not considered to be formal methods.
- The *extensive theoretical interpretation* considers as "standard basic formal methods" all the concepts of formal languages, grammars, finite-state machines and automata, lexer and parser generators, etc. To us, this definition is also too wide, even if formal methods borrow many ideas from the (pre-existing) language and automata theories; for instance, the construction of a "traditional" compiler cannot be called formal methods.
- The *lightweight interpretation* considers as formal methods all those language features introduced for defensive programming (type checking, library interfaces, program assertions, loop invariants, pre- and post-conditions, etc.), as well as all related verifications, from simple compiler checks to advanced static analyses. Even if some concepts predate formal methods (e.g. types were already present in Algol-60), we agree that such "lightweight" techniques, which are increasingly successful in industry, are indeed part of formal methods.
- The *heavyweight interpretation* recognises as formal methods only those approaches that are fully mathematical and based on proofs. We consider that such a definition is too restrictive, both for design needs (in the early phases of system design, the requirements are rarely fully formal) and for practical use ("heavyweight" techniques have a clear potential, but their success stories are isolated).

Although such diverging interpretations might have affected several answers to our questionnaire, we do not see them as a serious threat to validity, given the large number of participants in the survey. But this is an important problem for the community, as it is more difficult to promote formal methods if experts do not agree on their definition. The same issue occurs at various places, e.g. in the arXiv classification[7] where formal methods must fit either under "cs.LO" (logics in computer science) or "cs.SE" (software engineering); yet, many aspects of formal methods (e.g. executable specification languages, concurrency theory, or hybrid systems) cannot easily be reduced to logics, while the numerous applications of formal methods in hardware design do not belong to software engineering. We thus call for a standalone category of formal methods, whose perimeter should be considered inclusively. As one comment wisely pointed out: "we should act as a community".

[7] https://arxiv.org/corr/subjectclasses

# 3 Assessment of Formal Methods

## 3.1 System Design

With this first question, we wanted to query the experts about the necessity of formal methods for system design, i.e. whether or not they are dispensable or replaceable by alternative approaches.

| *Is it possible to design trustworthy (hardware or software) systems without using formal methods?* | | | | |
|---|---|---|---|---|
| Definitely: 16.2% | Probably: 21.5% | Probably not: 33.1% | Definitely not: 29.2% | N/A: 0.0% |

The answers are scattered, with no clear majority. Only the analysis of the 90 comments received may provide better insight.

Several comments display some criticism, since the answer depends on the definition/scope of formal methods (cf. Section 2.6), the complexity of the system, and the definition of *trustworthiness*. The latter is a valid point: many comments mention that a system is trustworthy only if there is an objective justification of its reliability. This interpretation introduces a substantial overlap with the next question (quality assessment). Most comments seem to agree that the question is about real systems, which are complex. We note that the answers *probably not* (for complex systems) and *probably* (for simple systems) actually express the same opinion. Five comments contradict the selected answer (maybe due to the implicit negation in *without*). In hindsight, a better formulation would have been: *is using formal methods necessary to design well-functioning complex (hardware or software) systems?*

The comments that explain that designing trustworthy systems is (definitely or probably) impossible, fall into two broad classes. The first class (14 comments) explains that formal methods are necessary to handle the inherent system complexity: “it is the size and the complexity that matter”, and, consequently, that informal methods are incomplete: “it is so easy to make bugs with informal methods and extensive testing is so difficult, that adequate formal methods do help a lot in the end”. The other class (14 comments) explains that trustworthy systems require some form of objective argumentation, involving unambiguous requirements. This was actually the topic of the next question. One argument was by analogy with (general) engineering. The following comment summarises these positions nicely: “The answer depends on the size, nature and complexity of software, and on the notion of ‘trustworthy’ you are interested in. Certainly, it is not possible to trust complex, safety critical software, built without recurring to any formalisation of its functions”.

Several comments indicate that not using formal methods is possible, but infeasible or costly. “There is a very important trade off between costs, time to delivery, quality”. The comments that explain that designing trustworthy systems is (definitely or probably) possible, fall into two categories: 15 comments mention counterexamples of systems that we generally trust, but that did not use formal methods in their design, such as airplanes, while four comments even apply this to the majority of systems: “there are many examples of systems such as airplanes that are produced without the use of formal methods and in general these are still considered ‘trustworthy’”. Another 16 comments claim that it is possible to build trustworthy systems by

alternative methods, such as simulation and testing, or building in redundancy, but seven comments state this is the case only for simple or non-critical systems: "properties of systems can be fully verified by exhaustive simulation if they are sufficiently small", and provided that our expectations on their reliability are sufficiently low.

### 3.2 Quality Assessment

This question also aimed to query the experts on the necessity of formal methods but, this time, for assessing the quality of complex systems.

| *Is it possible to assess the quality of complex (hardware or software) systems without using formal methods?* | | | | |
|---|---|---|---|---|
| Definitely: 15.4% | Probably: 26.9% | Probably not: 36.9% | Definitely not: 20.0% | N/A: 0.8% |

A majority of 56.9% deemed the use of formal methods important for quality assessment.

This question received 73 comments. Eight of them state that *quality* is a too broad notion that possibly includes performance, usability, process, etc., for which formal methods are not the most appropriate tool. The comments also indicate that the position of experts depends on whether systematic testing is considered to be part of formal methods or not.

There are mainly two arguments in favour of using formal methods to assess system quality. The first is that, in order to be scalable, assessment requires the use of tools, which need to rely on proper semantics: "Complex systems need scalable methods, scalable methods need a degree of automation, and such automation cannot be trusted if there is no mathematical notion of 'quality' and a model of the system supporting it". The second, more frequent, argument is that an assessment of quality requires to demonstrate the conformance of the product to an unambiguous specification: "a complex system's quality needs to be checked against well-specified requirements, and this again involves formal methods". One comment indicates that such an argument could be phrased in natural language, in principle. Another comment states: "the only way of assessing quality is by examination of the code itself, which is best conducted by specialised software based on sound theory".

Twenty-five comments mention alternative methods that can, at least, increase the confidence in digital systems: testing, simulation, statistical fault analysis, quality metrics, user interviews, and analysis of system logs. On the other hand, several comments state that alternative methods would be incomplete for complex systems, or that applying them exhaustively would be very costly ("the testing costs would be huge!"). One comment indicates that assessing the quality of the process is insufficient, although "certification institutions base their opinion mainly on criteria concerning the development process". Some comments mention certain systems considered reliable, despite not being assessed by formal methods, e.g. "Linux" and "Isabelle/HOL". Some comments distinguish quality assessment of brand new versus long-existing systems: "some military applications do surprisingly well without using formal methods. However, these are almost exclusively new variants of previously

deployed systems. Assuring the behaviour of a brand new system without using formal methods would be, in my judgement, very challenging".

### 3.3 Expected Benefits

This question provided the experts with a list of promises often associated to formal methods, so as to query whether these promises are actually kept.

| *Do you believe that formal methods, together with the rigorous use of formal analysis tools, can deliver the promise of:* | Definitely | Probably | Probably not | Definitely not | N/A |
|---|---|---|---|---|---|
| Better software quality | 81.5% | 16.9% | 0.8% | 0.0% | 0.8% |
| Improved system safety | 92.3% | 7.7% | 0.0% | 0.0% | 0.0% |
| Enhanced cybersecurity | 65.4% | 31.5% | 0.8% | 0.0% | 2.3% |
| Higher performance systems | 27.7% | 46.2% | 19.2% | 0.0% | 6.9% |
| Cheaper software development | 19.2% | 40.8% | 30.0% | 5.4% | 4.6% |
| Reduced time to market | 19.2% | 37.7% | 31.5% | 4.6% | 6.9% |
| Easier certification | 61.5% | 35.4% | 2.3% | 0.0% | 0.8% |
| Easier long-term maintenance | 60.0% | 36.9% | 2.3% | 0.0% | 0.8% |

Quasi unanimously, the experts confirmed that formal methods deliver quality, safety, security, easier certification, and easier maintenance. With weaker, yet clear majorities, the experts estimated that formal methods lead to better performance (73.9%), lower costs (60%), and faster development (56.9%).

One critical comment expresses that the proposed scale (*definitely*, *probably*, etc.) was too coarse. We received no other comment for this question, presumably because it already asked for many inputs from the experts.

### 3.4 Relation to Cybersecurity

This question sought confirmation from the experts concerning the need for formal methods to properly address cybersecurity issues.

| *In your opinion, are formal methods an essential part of cybersecurity?* | | | |
|---|---|---|---|
| No: 0.8% | Marginally: 16.9% | Yes: 74.6% | N/A: 7.7% |

The large majority of experts recognised an important role for formal methods in cybersecurity.

This question attracted 57 comments. Several experts (including those with *no opinion*) indicated not to be cybersecurity experts. Note that, indeed, the question was addressed to an audience of, primarily, formal methods experts.

Among the 13 comments for *marginally*, eight indicate fundamental problems, half of them because one cannot foresee and formalise all possible threats, such as side channel attacks ("the problem is how to formally specify and analyse the huge variety of possible attacks"), others because cybersecurity is very broad, involving, for instance, social aspects. Five comments see this as an opportunity to apply formal

methods more widely, but similar arguments are also found quite often among the *yes* comments.

Many comments for the *yes* answer indicate opportunities in code analysis (e.g. avoiding memory leaks) and protocol analysis: "many cybersecurity issues involve code with memory issues, issues with access control and faulty security protocols. These would, I'd say, be typical issues that can be (and are being) addressed by formal methods". Other comments point to programming languages offering strong guarantees. Another opportunity is mentioned: "cybersecurity is particularly interesting because there are so many social factors, like social engineering, that can override verified algorithms. The challenge of how to model and verify, e.g. properties of social networks, represents an interesting frontier for formal methods".

Two comments indicate that there is much low-hanging fruit that should be harvested before applying formal methods, e.g. "programming language, architecture, development processes". There were relatively few concrete case studies mentioned, the most concrete one being the "network access restrictions [...] checked using formal methods in Azure [to] both ensure security (e.g. prevent configurations where SSH ports are opened) and avoid customer issues (detect and prevent common misconfigurations that block services)".

### 3.5 Missed Opportunities

To complete the assessment of formal methods, we wanted to know from the experts whether they believe academics have sufficiently applied formal methods.

| *Do you think the academic community has missed some opportunities to apply formal methods in industry, in other sciences, and/or in society at large?* | | | | |
|---|---|---|---|---|
| Definitely: 40.0% | Probably: 42.3% | Probably not: 10.8% | Definitely not: 0.0% | N/A: 6.9% |

Clearly, most experts (82.3%) believe that some opportunities must have been missed, although, when analysing the 73 comments received, very few concrete examples are given.

Many comments put the blame either on academic practice (in particular its publication culture and its focus on theoretical results), or on industrial practice. A few comments acknowledge that the required multidisciplinarity is difficult, since academia and industry have conflicting goals. One comment describes "a healthy tension" between "on the one hand, to do as much as we can to bring formal methods to industry; but on the other, to develop intrinsically better technologies". Another comment wonders about the apparent brakes on change in industry: "why is it that we still are fighting to get accepted as a mainstream (software engineering) discipline? Why is C still the most dominant implementation language in the world?"

The *probably not* answer is explained in most comments by the fact that applications have been in the focus of formal methods research from the beginning: "I think there have been many serious attempts to transfer formal methods ideas to industry". Therefore, 10 comments explicitly blame industry for underestimating formal methods, e.g.: "the choice not to use formal methods can be based on silly things, such as not having the IDE one is used to".

On the other hand, several comments state that formal methods have been oversold: "it may be the case that formal methods have been sold to industry while they were still immature". Many other reasons why we have *probably* or *definitely* missed out on opportunities were mentioned, such as the lack of standard notations, service providers, whole-system engineering approaches, support of design processes, and data-driven approaches.

Finally, only a few concrete missed opportunities are mentioned, like: "we have probably missed an opportunity to introduce formal methods in the design of medical devices"; "there are so many domains with domain-specific languages that could greatly benefit from the formal methods toolkit"; and "the formal methods community should have shown that formal methods can fit modern agile development".

## 4 Formal Methods in Research

### 4.1 Overall Evaluation

This first question polled the experts concerning the degree of success, from an academic perspective, of formal methods.

| *How would you evaluate the achievements of formal methods in academia?* | | |
|---|---|---|
| A failure: 0.8% | A partial failure: 6.9% | |
| A partial success: 62.3% | A success: 28.5% | N/A: 1.5% |

The experts almost unanimously agreed that formal methods are a *success* or a *partial success*; only 7.7% stated the contrary, while a tiny minority had *no opinion*.

Analysing the 73 comments received, the question was largely misunderstood and criticised. One reason for this was the imprecision of the term *academia* (two comments mention this explicitly, e.g. "I am not sure what is meant by 'in academia'"). When drafting the question, we were interested in the perceived success of formal methods in research, but some respondents considered a different scope: 19 comments evaluate the success as partial, because of the limited success of formal methods in education ("I refer here to education, not research") and/or their lack of impact in industry. Other comments consider a *partial failure* and a *partial success* to be indistinguishable options.

The few comments from experts who consider the outcome to be a failure can be summarised as follows: "nice theory", but a lack of impact in industry "to drive formal methods into actual design, analysis, and deployment processes", and even in curricula. Note that the impact of formal methods in industry and education is addressed later in this survey using specific questions.

Further criticism, explicit or implicit, concerns how to measure success. Such ambiguity created a lot of variability in the comments, especially those considering formal methods to be a *success*. The most frequently used measures of success, mentioned in 12 comments, are based on objective data, such as the size of the formal methods community (considered to be an active community), the number of conferences embracing formal methods (e.g. FM, FMICS, iFM, CAV, POPL), the number

of associations and working groups (e.g. FME and FMICS), the number of ERC grants, the number of formal methods researchers hired by industry, and (less objective) the sheer diversity of techniques and tools.

Eight comments attribute the success of formal methods to some of its specific sub-fields (e.g. formal verification or SMT solving) or to a few success stories. Another nine comments call for more success stories (especially on real-world systems) and improved visibility of the existing ones. Indeed, formal methods have a "nice corpus of theories and techniques, many good tools, a few impressive applications", "but many researchers program and verify as if they never heard of formal methods", and students often believe that "producing (buggy) code and then fishing for bugs is the 'best practice' and the grown up way to design and implement". But another comment recalls, in software and hardware design, the existence of "achievements nobody can question, everybody takes for granted, and we forget to be proud about". Finally, a few comments also mention geographic differences, with more success in Europe than in China and the US.

### 4.2 Foundational Nature

This question wanted to know from the experts whether they believe formal methods are one of the scientific backbones of computer science.

| *In your opinion, do formal methods provide mathematical foundations for many branches of computer science?* | | | |
|---|---|---|---|
| No: 0.8% | To a limited extent: 36.9% | Yes: 61.5% | N/A: 0.8% |

Nearly all the experts agreed that formal methods do form the foundation for many branches of computer science, but only *to a limited extent* for just over one-third of them. Only one expert answered *no* and another one had *no opinion*.

This question received 56 comments, all of which for the *yes* or *to a limited extent* answers. There was some criticism on the question, basically boiling down to what is to be understood by *formal methods*. Although we gave a preliminary definition in our survey (cf. Section 2.6), seven comments mention that their answer strongly depends on the chosen definition. As one comment states: "depending on what you mean by 'formal methods,' the question could be tautological in that mathematical foundations are formal methods". Three comments actually claim the reverse, i.e. many other "branches of computer science provide mathematical foundations for formal methods". A couple of comments go as far as stating that "it establishes computer science as a science" and "by definition". One comment contains a more pondered variant: "I do believe that formal methods, or rather mathematical logic, is as essential to computer science as mathematical analysis is to physics".

A few comments put forward that knowledge of formal methods provides one with a competitive edge in industry. The opinion is substantiated in 12 comments, which note that formal methods are fundamental for understanding software and hardware (typically citing programming languages and compiler design).

Finally, one comment (from an expert who answered *yes*) points out some exceptions: "soft subjects like human-computer interaction rely more on psychology

and sociology. Formal methods for artificial intelligence hardly exist". However, this might (need to) change, since another comment notes that "the 'explainable artificial intelligence' movement [...] cannot decently succeed without formal methods". Six comments of experts who answered *to a limited extent* also mention exceptions: "human-computer interaction", "speech recognition, computer graphics, computer vision", "data science", "machine learning", and "complexity theory"; yet, another comment specifically mentions complexity theory as one of the branches for which formal methods do provide foundations.

### 4.3 Main Criticisms

This question tried to weigh to what degree the experts agree with frequently heard criticism concerning misplaced efforts of academic researchers in formal methods.

| *Would you agree with the criticism that most academic researchers in formal methods are:* | |
|---|---|
| Not investing enough effort to develop software tools that are usable and robust? | 66.9% |
| Too much novelty-driven and not enough interested in the consolidation of existing results to make them available to a wider audience? | 60.8% |
| Too much focussed on the most difficult and challenging problems, while neglecting the development of broader approaches applicable to "real world" issues? | 53.8% |
| Other criticism | 33.1% |

*(multiple answers allowed; answers sorted by frequency)*

The three frequently heard types of criticism suggested by the proposed answers created consensus among a large number of experts, namely 87, 79, and 70 experts (in the order of frequency displayed in the table). One-third of the experts had (also) other criticism concerning academic researchers in formal methods.

This question generated the remarkable amount of 170 comments. The experts who answered *other criticism* had quite varying opinions, ranging from not agreeing with the proposed answers to criticising the question, typically because they believe it is difficult to generalise or because they believe neither of the suggestions belongs to the task of academic researchers. Most, however, share two general beliefs that also featured very frequently in the comments provided by those experts who did choose one of the proposed answers. Basically, the effort and interest to develop more usable and robust tools, to consolidate results and approaches and make them more widely applicable and available — clearly perceived by the experts as improving the transfer of technology to industry — is hindered by two current realities in academia: (i) a lack of academic recognition (criteria for publications and thus career promotions are based on novelty); and (ii) a lack of funding for industrial application (requiring tool advancement and maintenance). Several comments nicely summarise this belief. Some picks: "even though industry participation is sought, in essence academia creates its own bubble where criteria for success are mostly within the bubble"; "there is no business case for long term support of (academic) tools; industry needs stability and performance, academics need to innovate"; and "at the end of the day, researchers do not get much credit (nor funding) for building and

maintaining tools and high-quality software, despite the enormous effort involved; instead, publications are more rewarded and are often what counts". This opinion recurred in 67 comments.

Finally, it is worth mentioning that two comments are positive on artefact evaluations, which "have at least accomplished that reported results are reproducible, but this is still miles away from a tool that is mature enough to be used by industry". However, one comment is convinced of the contrary: "the current practice of 'artefact evaluation' is harmful as it rewards building prototypes that are not really used by anyone, but give the illusion of building tools".

### 4.4 Topic Relevance

With this question we simply wanted to know whether the experts still consider formal methods a hot topic.

| *Do you believe that formal methods are still a major topic today for academic research in computer science?* | | | | |
|---|---|---|---|---|
| Definitely: 71.5% | Probably: 20.0% | Probably not: 7.7% | Definitely not: 0.0% | N/A: 0.8% |

The vast majority of experts claimed that formal methods are indeed still a major research topic; only ten thought this is *probably not* the case, while one expert had *no opinion*. Interestingly, not a single expert thought this is *definitely not* the case.

This question attracted 67 comments. The seven experts who commented their choice for *probably not* constitute two groups of more or less equal size. One group believes that "the momentum has gone elsewhere", in one case attributing this to the fact that "industry has chosen a different direction". The other group actually seems to be convinced that formal methods are *(definitely) not* a major topic for research ("most computer science departments at major universities do not have anyone specialising in formal methods currently"), partly criticising the question: "still? It has been a marginal activity at most universities for a while".

Several of the 14 experts who commented their choice for *probably* mention that formal methods *should* still be a major research topic, but that it is currently "under pressure of other 'hot' topics such as artificial intelligence and machine learning". Half of the 93 experts who believe that formal methods *definitely* are still major research topic today added a comment, mostly explaining their choice: "though there are always certain hypes, formal methods are an important and solid basis for the development of safety-critical systems"; "in fact, formal method papers are appearing in major conferences, even outside the community. Look at the latest POPL and PLDI conferences"; and "as more and more aspects of human societies rely on some computing system, formal methods are more relevant than ever". But, there is room for improvement: "we have to learn to switch from an 'individual problem view' to a global view which exploits the power of the various methods, i.e. like going from 'post mortem verification' to 'correctness by design', which allows us to exploit the strength of many formal methods disciplines". Interestingly, one comment contradicts the opinion expressed in a previous comment: "in some public institutions the number of research teams dedicated to formal methods is relatively significant".

### 4.5 Research Priorities

Assuming that resources for supporting research in formal methods are limited, this question asked the experts to establish a ranking between various research topics.

| *Which should be the most urgent priorities of researchers working in formal methods?* | |
|---|---|
| Scalability: design more efficient verification algorithms | 70.0% |
| Applicability: develop more usable software tools | 68.5% |
| Acceptability: enhance integration into software engineering processes | 65.4% |
| Discovery: explore new classes of problems and application domains | 44.6% |
| Theory: search for the next fundamental breakthroughs | 35.4% |
| Languages: design more expressive and user-friendly notations | 31.5% |
| Other | 16.2% |

*(from 1 to 4 answers allowed; answers sorted by frequency)*

Analysing the three most selected answers, one observes a strong wish that formal methods are applied to real problems, especially industrial ones. In this respect, the importance of *scalability* can be explained as the desire to overcome major obstacles to practical applications. Also, the big difference between both extremes, namely *scalability* and *languages*, might lay in the perception that the former addresses hard, objective problems deeply rooted in complexity theory, whereas the latter deals with softer, subjective issues that are largely a matter of human conventions. Such an explanation perhaps ignores the fact that languages are a key factor for industrial acceptability, and that poorly-designed languages may significantly increase the cost of formal analyses.

This question received 19 comments. Six of them refuse to define priorities, pointing out that "all the above problems are important and should be addressed" or that "science should not be priority-driven"; instead, one "should encourage researchers to follow their inspiration" and "focus on the task they are best in". One comment on *scalability* stresses the importance of modularity, with "compositional and reusable verification of code fragments and libraries". Two comments on *acceptability* point out that the formal methods community should ensure explainability (i.e. provide "a justification for the diagnostic/result" computed by software tools) and "influence standards and regulations to make sure formal methods are required where it makes sense". Three comments on *languages* mention that they should be "co-developed" with verification tools and methodologies, suggesting to "design more deterministic and analysable languages (which will likely be less expressive)" and to build "good code generators" for specification languages, so as to enhance their "integration with existing programming languages". Five other comments propose alternative research priorities: validation of requirements, code synthesis, process mining, and connections to artificial intelligence, such as "artificial-intelligence-driven invariant discovery".

### 4.6 Software Development

This final question on formal methods in research tries to poll expert opinions on the role and responsibility of academia with respect to the delivery of professional tools.

| *Which one of these two statements do you like best?* | |
|---|---|
| Public research in formal methods should only develop prototype (proof-of-concept) tools, while leaving the development of professional tools to industry | 36.2% |
| Formal methods are too involved and their market is too small for most companies, so academia should invest effort to develop and consolidate usable tools | 38.5% |
| Other answer | 25.4% |

This question apparently divided the experts: while one-fourth did not like either of the two statements best, we note an almost perfect distribution of the remaining experts among the two statements. The outcome thus provides little guidance as to where the effort concerning professional tool development should come from.

This question received 80 comments, most of which are insightful. The 34 comments provided with *other answer* actually show a remarkable variety of opinions. Ten comments believe (to a certain extent) in both: "the tools we develop should be usable (and extensible) by researchers in our own community, and should therefore go beyond the proof-of-concept stage. However, we should not spend time on polishing the things that matter for acceptance in industry, such as user interfaces, have round-the-clock available help desks, liability, etc.". Ten comments (strongly) believe in neither of the two statements. Five comments believe in a "combination of the two statements": "neither fully, both partially". Another ten comments believe something similar, namely that developing formal methods tools should be a collaborative effort by academia and industry, but four of them note that academia should be leading the development, and five of them that academic prototypes "should go beyond the current state of proof-of-concept tools". A couple of comments, finally, mention that "effort should be devoted to open-source community efforts".

The 16 comments provided by those experts who best like the first statement are very much in line, mainly indicating two (related) reasons. First, eight comments claim that tool development is "not the role of academia" and "most academic institutions are not equipped to maintain professional level tools, even if they manage to develop a first version". Second, four comments claim there is a lack of "funding to develop industrial-strength tools".

The 30 comments provided by the experts preferring the second statement are less in line, but there are two recurring reasons in support of this statement. First, eight comments state that "good research groups tend to work on one tool, for decades, which brings about solid tools"; CADP, UPPAAL, and Z3 are explicitly mentioned as examples. Second, six comments state that "this is the only way to provide technological transfer to industry, [as] in most cases efficient implementation requires to know a bit about the underlying theory".

## 5 Formal Methods in Industry

### 5.1 Impact Evaluation

This first question asked from the experts to evaluate the degree of success of formal methods in industry.

| *How would you qualify the impact of formal methods on industrial software development practices?* | | |
|---|---|---|
| A failure: 2.3% | A partial failure: 29.2% | |
| A partial success: 63.8% | A success: 3.1% | N/A: 1.5% |

According to most answers, the impact is neither a complete success nor a complete failure, but in between, and clearly more of a success than a failure.

This is confirmed by the 79 comments, which are distributed as follows among the proposed answers: 0 for *failure*, 24 for *partial failure*, 51 for *partial success*, 3 for *success*, and 1 for *no opinion*. Eighteen comments mention "a few great achievements" of formal methods in CAD and "EDA tools for hardware design and embedded software", "in a few enlightened industries (aerospace, railway, nuclear)", "in some fields like avionics, distributed algorithms, and now security", and in "many of the most successful companies", which "develop and adopt formal methods for [their] own use" — with mentions of Airbus, AMD, ARM, ASML, AWS, Facebook, Google, Huawei, IBM, Intel, Microsoft, Philips Healthcare, and Siemens, "just to name a few". Building a global picture is difficult however, since "some of the work is being done by secretive companies who do not publish/highlight their successes": formal methods are often used "behind the scenes" and "unfortunately, once a formal-methods tool becomes successful, it [is] usually renamed to something else".

Twelve other comments list "ideas of formal methods [that] found their way into modelling and programming languages", e.g. "typed languages (in spite of the current Python frenzy)", "type checking" and "type inference", "interfaces for libraries", "assertions in programs", "pre- and post-conditions of functions", but also "techniques that improve the code production", e.g. "model checking", "automatic test case generation", "lightweight verification tools (runtime verification, 'linters', etc.)", "static analysis and other embedded analyses, [which] are accepted and included in industrial toolchains", and "now routinely used for systems software and open-source software", sometimes "without people even realising it".

On the critical side, forty comments express almost the same idea: "formal methods are [...] used only to a limited extent, and not where it would be required"; "in a few industries it has been a success but in the majority of industries not"; "there are some successes to celebrate, but they are at the moment too few to have impact". One comment underpins this general opinion: "there are some ways to measure this impact: the offers to hire professionals with formal-methods background, the investment of software licenses for formal-methods-based tools, the contracts with research institutions or companies to solve specific projects, etc. I do not see a big impact with these parameters".

Some comments justify the fact that many companies are not using formal methods either for financial reasons ("too little of a cost reduction in return for too great an investment of time and skill"; "company cultures [are] particularly hostile to things that give no immediate product, but merely add quality to a product"), or due to human factors ("the inertia of industrial software development practices is enormous"; "the somewhat heavy emphasis on having a rich math background [...] is not going to be mainstream"), or by historical after-effects ("formal methods may have gotten a 'bad reputation' [since they] in some cases have become associated with '1980s

style formal methods' such as VDM, Z, B method and the like; even though such approaches would be considered outdated today, they are still mentioned in applicable standards in some industries, and this in effect delays introduction of more modern formal methods").

Nonetheless, the majority of comments remains optimistic, as many "prefer to see the glass as half full rather than half empty: formal methods are making their way [...] maybe not as widely as we would like, and probably not in their most theoretical or full-blown strength, but they make an impact".

### 5.2 Technology Readiness

With this question we wanted to learn about the perceived readiness of formal methods for technology transfer.

| *Do you believe that formal methods are now ready to be used extensively in industry?* | | | |
|---|---|---|---|
| No: 3.8% | Only to a limited extent: 67.7% | Yes: 26.9% | N/A: 1.5% |

Two-third of the experts answered that formal methods are, *to a limited extent*, ready for industry, while another quarter expressed that formal methods can already be used extensively. Only a few experts indicated *no* or *no opinion*.

When analysing the comments, it appears that many *yes* answers are nuanced and should be interpreted as *yes, but*. The twelve most outspoken *yes* answers point to successful projects that have demonstrated that formal methods are ready and their application is beneficial: "there are plenty of academic case studies that appear to scale well enough for industrial application" and "formal methods are already widely used in industry in existing tools". Four of these comments explicitly mention hardware: "formal methods have been used extensively for quite a few years now in hardware design verification". The reasons why formal methods are only ready to a certain extent are often related to application domains, tool maturity, or people's skills and willingness.

Nineteen comments restrict the readiness of formal methods tools to certain *application domains*, in particular "domains with high standards for safety and cybersecurity, where requirements are well understood". Even in such cases, "industrial researchers need to do the work to fit this into existing development flows". For instance, "we need to show how formal methods can be used to explore system design alternatives much faster".

Concerning *software tools*, fourteen comments indicate that the maturity of the current tools is not acceptable for industry: "[formal methods] tools are in general much lower quality than programming language tools"; "the existing tools are, for the most part, too brittle and hard to use"; and "the industry should be involved in developing tools that meet industrial standards". Yet, thirteen comments point out that particular lightweight tools can be applied in continuous integration pipelines and, thus, readily deployed: "I think that we are getting close with tools like hybrid fuzzers (that combine fuzzing with symbolic execution), test-case generators, [and] bounded model check[ers]. I think that these would make a measurable difference

in productivity and quality". "They can be a useful bug finding tool. Ideally, they will be integrated into IDEs and compilers and operate in the background".

Fifteen comments note that formal methods are only ready to be applied by sufficiently *skilled and willing people*: "there is probably still a lack of trained engineers and of will" and "it also requires scientific skill and attitude". There are conflicting comments around "being modest". On the one hand: "it may still be too early for a wide-spread roll-out of formal methods in industry. We run the risk of over-promising", but, on the other hand: "how many whip lashes should you self-apply before you have the permission to venture out in the world?"

### 5.3 Return on Investment

This question asked the experts to make an informal cost-benefit analysis over time.

| *In your opinion, are formal methods profitable enough to outweigh their costs?* | | |
|---|---|---|
| No return on investment: 2.3% | Profitable in the long term only: 12.3% | |
| Immediately profitable: 15.4% | Profitable in medium and long terms: 58.5% | N/A: 11.5% |

A small majority judged that the application of formal methods is *profitable in medium and long terms*. Another 15% (resp. 12%) indicated that they pay off *immediately* (resp. *in the long term*). A few experts answered that formal methods do not pay off, while a relatively large group has *no opinion*.

This question received 73 comments. In the *no opinion* category, two comments criticise the question as ill-posed: "your scale is very unhelpful", or even: "your question is part of the problem". The other ten, however, indicated that the answer depends too much on the specific circumstances.

The comments justifying immediate pay-off are very diverse. Some see the pay-off in the added value, either "to explore and analyse design-time problems", or as an alternative to "more ad-hoc methods such as testing", or in "added security and safety". Others justify the pay-off by the huge costs of errors in critical software. Three comments condition an immediate pay-off on the proper alignment with software development processes, for instance: "the key is to align the formal methods [...] with incremental software development".

Ten comments explicitly mention that initial investment costs prevent an immediate return on investment: "as for any technology move, one needs to adapt methods and tools, to train and educate, to practice". One comment concludes that "the initial cost is really high, and a critical research focus should be on how we can provide lightweight formal methods that are more proportional in their effort/value ratio", while another expects that "if smoothly integrated into the development process, the extra cost will be amortised by the savings gained from better quality". Indeed, several comments point out that "the real savings [come] later with improved product quality and reduction of errors".

But "a clear problem is that the benefits cannot be quantified clearly", especially when "companies get away with the consequences of their bad development", "as long as states/governments do no enforce strict regulations with proper penalisation". Four further comments explain that the real cost savings appear only later,

with less and cheaper maintenance due to fewer failures. Another fifteen comments note that the return on investment "is really depending on the context, and the right choice of technique and problem".

Another argument justifies long-term-only benefit after considerable investment: "we should think of formal methods as a 'disruptive technology'. Such technologies have the potential to change the way things are done and generate a process of 'creative destruction' in Schumpeter's sense; but this of course generates resistance and requires investment, more than of money, investment on people".

### 5.4 Most Effective Framework

This question polled the experts to know in which companies formal methods can be most efficiently deployed.

| *Which kind of company is best suited for using formal methods?* | |
|---|---|
| Large companies, because they have the budget and time frame needed to experiment with formal methods | 23.8% |
| Small companies, because they are agile enough to prototype with non-standard languages and software tools | 6.9% |
| Any kind of company, whatever its size | 63.8% |
| N/A | 5.4% |

The majority of the experts (around 70%) did not select a clear advantage for either *large* or *small companies*, when it comes to the application of formal methods. The number of experts that expect a fruitful application from *large companies* was three times larger than the number of experts who expect this from *small companies*.

Looking at the 61 comments received, 20 of them indicate that the presence of skilled and enthusiastic people is more important than company size. Another 13 comments express that the application domain is more important than company size. These reasons can explain why many experts did not choose any of the extremes: "the size of the company does not matter. What matters is their implicit motivation (to be the best in the business), the ability of a local champion (to carry the torch, overcome internal hurdles, motivate other people), and an obvious business opportunity where the application of a formal technique is of paramount benefit".

The comments provide further insights in the perceived difference between small and large companies. On the one hand, eight comments indicate that *large companies* are more suited, as they can devote time and budget to formal methods: "large companies are typically willing to invest in pilot projects to study the feasibility of using formal methods. They have the financial means to do so". On the other hand, eight comments indicate that *small companies* are more agile to adopt formal methods quickly: "small companies can decide faster and are more dependent on quality"; "the actual killer case would be a startup company formed of people who are already highly trained in formal methods and have a killer app for which formal methods gives them overwhelming advantage". Three other comments discuss examples of formal methods deployed in small or big companies.

### 5.5 Limiting Factors

This question asked the experts to rank a large number of potential barriers and obstacles that may prevent formal methods from being accepted in industry.

| *What are the limiting factors for a wider adoption of formal methods by industry?* | |
|---|---|
| Engineers lack proper training in formal methods | 71.5% |
| Academic tools have limitations and are not professionally maintained | 66.9% |
| Formal methods are not properly integrated in the industrial design life cycle | 66.9% |
| Formal methods have a steep learning curve | 63.8% |
| Developers are reluctant to change their way of working | 62.3% |
| Managers are not aware of formal methods | 57.7% |
| Many companies do not pay enough attention to software quality and security | 56.2% |
| Formal methods are not properly integrated in standard certification processes | 46.9% |
| Formal methods focus on relevant problems, but only on a small part of all problems | 36.9% |
| Benefits of formal methods are not marketed enough | 36.9% |
| There are too many formal methods, with a lack of independent comparison | 28.5% |
| Formal methods are too costly, with no perceived immediate added value | 26.9% |
| Formal methods are too slow to meet current time-to-market constraints | 17.7% |
| Professional tools are too expensive because of the small market for them | 14.6% |
| Other approaches to software quality outperform formal methods | 13.1% |
| Industrial software development practices change too often and too quickly | 8.5% |
| Formal methods focus on the wrong problems | 7.7% |
| Other | 13.8% |

*(from 1 to 12 answers allowed; answers sorted by frequency)*

Interestingly, obstacles arising from human factors predominate, as the 1st, 4th, 5th, and 6th most selected answers reflect educational problems, namely a lack of knowledge from managers and developers, and their difficulties to learn and deploy formal methods. Technical factors appear in the 2nd and 3rd answers, whereas financial factors underlie the 7th answer.

This question attracted 17 comments, most of which are attached to the *other* answer but actually correspond to answers already present in the above list (namely, the 1st, 2nd, 4th, 5th, and 8th most selected answers). For instance, five comments echo the 1st answer (*engineers lack proper training in formal methods*), one of them regrets that "education in formal methods frightens off students and puts them off for life rather than showing potential benefits". Two comments, somewhat related to the 5th answer (i.e. *developers are reluctant to change their way of working*), raise concerns about misguided applications of formal methods: "formal methods people are too stubborn; they advocate that everything should be formal", but "trying to apply formal methods everywhere is a non-sense", as "formal methods have to be sold [only] to people with problems". Two other comments reinforce the 8th answer (i.e. *formal methods are not properly integrated in standard certification processes*), regretting that "professional bodies do not encourage best practices, like they do in other disciplines" and that "regulation often focuses on process quality, not product

quality". Another comment draws a critical eye on those limiting factors: "it is like benefits that show in the long-term, they are trumped by short-term obligations".

### 5.6 Research-Industry Gap

This question tried to evaluate the distance, and its growth trend, between the problems actually faced by industry and the solutions brought by academic researchers.

| *Which one of these assertions do you consider to be the most appropriate?* | |
|---|---|
| There is no gap between academic research in formal methods and industry | 2.3% |
| There is currently a gap, but it is narrowing | 68.5% |
| There is currently a gap and it is growing | 20.0% |
| N/A | 9.2% |

Most experts agreed upon the existence of a gap between academic research and industry, and they are also positive that this gap is getting smaller.

This question received 54 comments distributed as follows among the proposed answers: 3 for *no gap*, 33 for *narrowing gap*, 13 for *growing gap*, and 5 for *no opinion*. One comment indicates that "a general response" to such "a very open question" is impossible. Three comments point out that "in hardware companies there is virtually no gap", as these "companies are long-time users of formal methods"; so, most of the discussion focuses on software and systems development. A large majority of comments consider that the gap is narrowing, since "technology transfer is visibly increasing", but five comments notice that "progress is slow" or "very slow", and "there is a huge work to be still done". Other comments make a clear distinction between "a few elite companies" (Amazon, Facebook, Google, Microsoft, and Thales Railways are cited in four comments as "examples of the gap narrowing"), certain application domains ("in hardware design, communication protocols, critical applications like avionics, and formal system testing, [the] gap seems to be narrowing"), and the rest, for which "the situation is heterogeneous", as "more industries get interested" in formal methods, "but few do more than experiments". Formal methods are also successful in domains such as "cryptocurrencies, [where] any bug can cause an enormous financial loss". The market size, in itself, does not seem to be a criterion: "the gap is narrowing in safety-critical robotics [but] growing in Android mobile-phone apps; both are multi-billion dollar industries".

The explanations given for a growing gap are threefold: complexity of industrial projects and agility of industrial processes ("industry is moving forward very fast, and academia has a hard time to catch up"), fragmentation of formal methods ("too many competing approaches with too little distinguishing impacts in practice", as well as increasingly complex "extensions of [...] computational models that are only relevant to increasingly smaller audiences"), and lack of properly trained personnel ("the education of software professionals seems to contain less and less hard topics such as logic"). This latter point is deemed crucial, as "the dismal lack of mathematical abilities of the iPhone generation and the dismantling of theoretical courses [...] even in top universities" prevents the gap from being filled ("if you always need the

academic doctor working in formal methods for a real industrial project, then something is wrong").

Six comments confirm that "there is a gap" but one "cannot tell whether it is narrowing or growing", because "while formal methods are becoming more mature and capable to handle larger problems, problems are also becoming more complicated". Finally, a comment suggests that the gap is perhaps different from what one would expect, as "Google, Facebook, and Amazon have stronger formal-methods research than most academic groups", whereas another comment recommends that academic research "should make progress regardless of industry, as long as there are realistic applications".

### 5.7 Design Life Cycle

It has often been stated that formal methods are best applied all the way, step by step, from the initial requirements to the final executable code. However, many publications report successful uses of particular formal methods in particular phases of the design life cycle. This question tries to explore and quantify the discrepancy between the ideal expectations and the practical achievements.

| *In which phases of the design life cycle are formal methods likely to be the most useful?* | |
|---|---|
| Generating test cases, especially for corner cases | 77.7% |
| Capturing and formalising requirements | 75.4% |
| Checking whether models are correct | 69.2% |
| Building models of the system | 64.6% |
| Validating the requirements | 53.8% |
| Generating code from models | 53.1% |
| Certifying correctness of the final code | 45.4% |
| Monitoring deployed software at run time | 43.1% |
| Maintaining consistency between models | 42.3% |
| Detecting mistakes in handwritten code | 39.2% |
| Evaluating the test results | 20.0% |
| Other | 10.8% |

*(from 1 to 8 answers allowed; answers sorted by frequency)*

The presence of test-case generation at the top of the list is significant, as it contradicts the ideal vision of a fully formal design flow, where refinement is used at each step to ensure that the final code satisfies the initial requirements. Indeed, in such a design flow, tests would be no longer necessary or, at least, their importance would decrease. Instead, the stated relevance of formal methods for test-case generation indicates that formal methods fit well with conventional design flows, in which testing efforts often represent more than a half of the total development costs. Cutting down such efforts (e.g. by generating tests automatically and/or by generating tests of a better quality) is thus a promising target for formal methods. The next answers in the list show that different methods can be beneficially used during the various

phases of the design life cycle. All in one, the answers suggest that formal methods can be evolutionary, rather than revolutionary.

This question received 13 comments, all associated with the *other* answer. A first group of comments stresses that formal methods should be used in all phases of the life cycle to maintain some consistency from requirements to code. A second group of comments suggests other specific uses of formal methods: "certified compilation", "deployment configuration", analysis of "legacy systems", assurance that "certain classes of bugs" are "absent [from] the final code", and development and verification of "concurrent and distributed systems" and "systems of systems".

### 5.8 Dissemination Players

The next question tried to determine who, in the stakeholder network that exists between academia and industry, can contribute most to the industrial deployment of formal methods.

| *Who could best drive a more widespread application of formal methods in industry?* | |
|---|---|
| Universities and engineering schools | 63.8% |
| Research and technology institutes | 63.8% |
| Large industrial companies | 50.0% |
| Tool-vendor companies | 46.2% |
| Dedicated service companies | 30.0% |
| Others | 14.6% |

*(from 1 to 4 answers allowed; answers sorted by frequency)*

The answers make it clear that all stakeholders have a role to play, perhaps at a different level and with a different impact factor. Somewhat paradoxically, the respondents show greater confidence in public (or non-for-profit) institutions than in private companies, although the goal is to trigger methodological changes in entities belonging to the private sector — a trend that is in line with former answers, such as those of Section 4.6.

This question attracted 131 comments distributed as follows among the proposed answers: 29 for *universities and engineering schools*, 27 for *research and technology institutes*, 24 for *large companies*, 20 for *tool-vendor companies*, 13 for *dedicated service companies*, and 18 for *others*.

Concerning *universities and engineering schools*, their most important mission is, according to 17 comments, to "create the necessary critical mass of talent-pool" by "delivering more graduates who know and like formal methods". Yet, five comments point out that "these institutions should be doing it better" and "definitely enhance their commitment in formal methods". Four comments list research-oriented missions, such as "illustrating novel ideas", "demonstrating the state-of-the-art via prototypes", and "develop[ing] and maintain[ing] formal methods tools".

Concerning *research and technology institutes*, four comments cite the examples of Fraunhofer (Germany), GTS (Denmark), INRIA (France), IRTs (France), MPI (Germany), MSR (worldwide), and SRI (California). Six comments point out that such

institutes “can play an important role in industrial take-up of formal methods” since “they are at the interface between researchers and industry” and, thus, “have more chance to be closer to the problem domain”. Ten comments expect them to “play a crucial role” in “devising user-friendly formal methods, designing efficient analysis methods, and developing robust tools”, and in “taking up larger challenges” to “demonstrate the value of formal methods on actual systems”. Two comments claim that such institutes “are better at long-term investment than individual universities”, but “they have to realise the missions set to them by their paymasters”.

Concerning *large companies*, seven comments mention Airbus, Amazon Web Services, Facebook, Google, Intel, Microsoft, and Thales, as well as “organisations in general that build critical software systems”. Eleven comments consider such companies and organisations as ideal hosts for formal methods: “they have the money, they have staff to spare, they have problems at scale, and they have the visibility that when they speak up, others listen”; this latter point references their capacity to “champion formal methods” and “boost the[ir] widespread application” by “commit[ting] their suppliers” and “provid[ing] a market for tool vendors”. However, two comments warn that large companies “do have the resources, but are often slow to react”, so that “we keep seeing companies on the brink of bankruptcy due to catastrophic errors that formal verification could catch”.

Concerning *tool-vendor companies*, two comments stress the importance of software: “without tools, no application of formal methods”. Nine comments state the missions expected from such companies: “transfer academic ideas and prototypes to industrially applicable software tools”, “sell and maintain [these] tools” and “make [them] appealing” by “working on usability issues”, “provide tutorials and courses”, and “offer consultancy” services. Two comments consider tool vendors as “companies that are quite successful”, while three other comments predict that such companies “have the heaviest resources and motivation to promote formal methods”, and that “tool vendors that open up the market can make a difference” and “will ultimately decide the acceptance of formal methods”.

Concerning *dedicated service companies*, five comments discuss the business model and genesis of such companies, which can be either “started up by academics” or spun off from larger companies that “prefer outsourcing this activity, at least temporarily”. Seven comments define such companies as arrays of “consultants”, who “concentrate a critical mass of expertise” and “specialised knowledge” to “help choosing the most appropriate approach” and deliver “formal methods as a service”.

Concerning *others*, five comments do not give a precise answer, while other comments suggest further stakeholders who could contribute to the industrial adoption of formal methods: governments/states (through regulations), certification authorities (through quality standards), funding agencies, alliances for open source and open APIs, non-for-profit associations, and communities of software developers.

### 5.9 Academic Policies

The last question of this group reviewed the concrete actions academia can do to improve the transfer of formal methods results to industry.

| *Which academic policies can contribute most to the adoption of formal methods in industry?* | |
|---|---|
| More collaborative projects between research and industry | 78.5% |
| Increased support for academic researchers developing tools | 68.5% |
| Construction of benchmarks and datasets for formal methods | 53.1% |
| Construction of learning resources for formal methods | 48.5% |
| Dedicated engineers to increase the quality and TRL[8] of academic tools | 45.4% |
| Databases of case studies showing the applicability of formal methods | 44.6% |
| Collaborative software platforms integrating tools from different institutions | 35.4% |
| Economic studies to estimate the return on investment of formal methods | 34.6% |
| Increased resources and scientific credits to software competitions | 26.9% |
| Others | 10.8% |

*(from 1 to 7 answers allowed; answers sorted by frequency)*

The analysis of the most selected answers shows three main lines of action for academia: (i) *collaborative projects* with industry, the number of which should be increased; (ii) *software tools*, for which academia should receive greater financial and human support — notice that such a confirmation of the manifest role of academia in tool development corroborates the prior results of Section 4.6; and (iii) *scientific data*, by producing benchmarks, datasets, case studies, and learning resources.

This question received 14 comments, most of which associated with the *others* answer. Four comments recommend to "invest in long-term collaboration with industry" (as opposed to the usual short-term projects supported by funding agencies), with "academic reward structure changes", "increased support and scientific credit for researchers involved in collaborative projects with industry", and "programs for PhD theses to be done in collaboration between academia and industry". However, another comment warns that "formal methods will [only] succeed in industry when a CEO decides it is a priority", a possible reminiscence of Bill Gates's famous memo on security [13]. Two other comments evoke the "inclusion of formal methods in regulatory regimes", with "standards and regulations that demand the kind of guarantees that only formal methods can provide". Finally, four other comments mention educational issues (specifically addressed in Section 6 below), with the suggestions of "updating curricula in ICT professionals at bachelor level" and "teaching students on a large scale", with a "compulsory formal methods module" and "better courses that speak to students' needs rather than professors' passions".

## 6 Formal Methods in Education

### 6.1 Course Level

Our first question concerning education was to ask the experts about the most suitable place for formal methods in an ideal teaching curriculum.

[8] Technology Readiness Level (http://en.wikipedia.org/wiki/Technology_readiness_level).

| *When and where should formal methods be taught?* | |
|---|---|
| In master courses at the university | 80.0% |
| In bachelor courses at the university | 79.2% |
| In professional (software) engineering schools | 70.8% |
| In continuing education for professionals | 70.0% |
| During doctorate studies | 31.5% |
| Others | 3.8% |

*(from 1 to 4 answers allowed; answers sorted by frequency)*

The main lesson is that formal methods should be taught early, in *bachelor* and *master* courses. Waiting until *doctorate* studies would be a mistake, as the PhD students would not have enough time to acquire a proficiency level in formal methods sufficient to survive on the international research scene.

There were only five comments on this question, all associated with the *others* answer. Most of them indicate that formal methods should be taught in all the proposed answers. Another comment points out that non-specialists should be taught "mathematical thinking and the capacity of abstraction, not formal methods per se".

### 6.2 Importance Level

This question asked the experts about the current situation of formal methods in computer science teaching. In order to avoid *no opinion* answers from respondents lacking a global overview of universities, we added a restriction to familiar schools.

| *What is your opinion on the level of importance currently attributed to teaching of formal methods at universities? (If you feel that the question is too general, restrict your answer to the universities you know best.)* | |
|---|---|
| Not enough attention | 50.0% |
| Sufficient attention, but scattered all over | 31.5% |
| Right level of attention | 6.9% |
| Too much attention relative to other skills | 1.5% |
| No opinion | 10.0% |

Exactly half of the experts indicated that formal methods do *not* receive *enough attention* in university curricula, while roughly one-third expressed it does, but in a *scattered* way. Only nine experts responded that universities attribute the *right level of attention* to teaching formal methods, while two experts answered it receives *too much attention*. Thirteen experts had *no opinion*.

This question received 47 comments. The four comments expressing *no opinion* mention that the answer varies too much "from country to country and institution to institution". Four of the six experts commenting on the *right level of attention* base their opinion only on their personal situation. These two types of comments are also common for the *not enough attention* and *sufficient attention, but scattered* answers.

Another recurring comment is that education in formal methods is often isolated. Five comments indicate that applications of formal methods should occur in other courses, like databases, algorithms, concurrency, distributed systems, operating systems, security, compilers, and programming languages. A few comments also

mention that the role of formal methods in the software development process and in actual engineering practice should be taught. Another comment explicitly mentions that formal methods should be given "the same relevance as programming".

Finally, the comments also point to some causes of the under-representation of formal methods in curricula: unawareness among staff and management, emergence of new hypes (e.g. heuristic and agile approaches), computer science curricula getting more and more crowded with other topics, and students of an increasingly variable entrance and abstraction level.

### 6.3 Course Format

This question investigated the target audience and the most appropriate contents for formal methods courses.

| *Which of the following course formats is preferable?* | |
|---|---|
| Intensive courses on formal methods, targeted to a small number of good students, so as to ensure that the research in formal methods remains strong | 6.9% |
| Non-specialist courses giving a flavour of formal methods combined with other topics (software engineering, distributed systems, embedded systems, etc.) | 5.4% |
| Both: specialist courses taught to a limited number of students, and gentle introduction to formal methods for a larger number of students | 83.8% |
| Other answer | 3.8% |

The answers show a clear consensus of the experts on the 3rd answer (*both*), with an overwhelming majority.

This question received 32 comments, distributed as follows among the proposed answers: 4 comments for *intensive courses*, 2 comments for *non-specialist courses*, 21 comments for *both*, and 5 comments for *other answer*. Three comments decline to answer the question, arguing that "it very much depends on the level and of the kind of students" and that "each instructor has to figure this out". Five comments state that "every bachelor in computer science/informatics should know about formal methods" and "be trained in applying [them]". Four comments stress that "intensive courses [taking] the matter seriously are the only way to truly educate people", whereas "overview courses (gentle introductions)" giving "just a flavour [are] more likely to lead to disappointment than to something good". Another comment warns, however, that "students (and practitioners) will avoid intensive courses as long as [...] they are too complex". Eleven comments support the two-level approach proposed by the 3rd answer (*both*), putting forward the need to educate a few specialists, who will design new methods and tools, as well as a majority of software engineers and future managers, who will adopt these methods and tools in their professional practice. Finally, one comment recommends to "spread the word" about the excitement of formal methods: "it is not a religion, but treating it as such may help".

### 6.4 Course Organisation

The next question asked about the best manner to set up formal methods courses.

| *How should formal-method courses be organised?* | |
|---|---|
| Top-down: primarily focused on theory, starting from theoretical results and possibly illustrating them with applications | 6.9% |
| Bottom-up: starting from concrete problems, and later introducing theory as a means to solve these problems | 40.8% |
| Alternative way, possibly with a combination of top-down and bottom-up (you can explain your vision in the comment field) | 44.6% |
| Other answer | 7.7% |

Only nine experts answered a *top-down* setup would be best, but the vast majority was divided between a *bottom-up* setup and an *alternative way*.

Fortunately, the analysis of the 58 comments provides more detailed information. Ten experts chose *other answer*, but six of them actually suggested an *alternative way* in their comments, often even a mixture of top-down and bottom-up. Besides, two experts who opted for *bottom-up* also added such comments. This means that just over half of the experts consider an *alternative way* to be the best choice for organising a formal methods course.

As could be expected from our suggestion in the option, a majority of 36 comments came from experts who opted for an *alternative way*. Nine comments indeed favour a combination of the two extremes, but 11 comments also mention that the answer depends on factors, such as the context, the lecturer, the course ("foundational courses [...] can be top-down; more applied courses [...] should be bottom-up"), and the students ("bottom-up at the BSc level, [as] young students want to solve problems [...], top-down at the MSc level, [as] more mature students like to learn new theories [...]"). A few comments suggest to take inspiration from how we teach mathematics and programming.

From the comments, one can distil a fundamental motivation for the *bottom-up* approach, namely: we need to teach students to understand the problem and the requirements before selecting a particular tool or solution. More than half of the 13 comments from experts opting for *bottom-up* agree that "starting from examples is important", since "a good theory always comes with a good practical motivation". Six of the experts opting for an *alternative way* also mention a need for appealing running examples and non-trivial applications. Ironically, the only comment received by an expert opting for *top-down* is: "no good tools without theory!"

### 6.5 Tool Usage

This question asked the experts whether, and to which extent, students should be exposed to software tools when being taught formal methods.

| *Which role should software tools play in formal-methods courses?* | |
|---|---|
| No role at all, as they obscure or divert from theoretical concepts | 0.0% |
| Marginal role: their existence should be mentioned to show that theoretical concepts can be implemented | 3.1% |
| Fair role: students should be told to learn by themselves about a few selected tools | 19.2% |
| Major role: lab exercises on concrete applications should be assigned to students | 75.4% |
| No opinion | 2.3% |

An overwhelming majority of answers judged the use of tools essential when teaching formal methods. Moreover, nobody supported the idea that tools should be kept away from formal-methods lectures; this is one of the very few questions where one of the proposed answers has been chosen by none of the respondents. This confirms a high consensus about the usefulness of tools in teaching.

This question received 55 comments distributed as follows: 0 for *no role at all*, 3 for *marginal role*, 10 for *fair role*, 39 for *major role*, and 3 for *no opinion*. Most comments claim that tools "need to be integral part of the courses", since "many students enjoy working with tools", "exercises on paper are not convincing", "without tools you will not be able to convince students that things are applicable, i.e. in their minds: worth studying", and "unless students are able to apply the concepts they learn to concrete examples, theory will not stick". Other comments put forward that "if you want people to use tools, you have to get students to use some", because "if students get a feel of formal methods tools, they are more likely to apply them in practice during their professional life". Thus, "hands-on courses are needed" and "not only lab exercises but also almost all homework should involve tools". One comment suggests that "another option is to have students implement tools".

A few comments express various reservations: (i) tools should be carefully chosen, because "if students negatively perceive a tool, this also reflects on the applicability and usefulness of the theory"; (ii) tools are only part of a larger problem, as "formulating a good model is a huge challenge in many cases" and students should "understand what the tools are saving them from having to do themselves"; (iii) there should be a correct balance between theory and practice, as "theoretical aspects [are] not always taught best with tools only" — a comment notices that "this is the same as the debate about teaching things that can be used immediately vs. teaching foundations that will be valid in 25 years from now".

## 7 Future of Formal Methods

### 7.1 Future Dissemination

This first question on the future of formal methods aimed to evaluate the long-term industrial uptake of formal methods.

| *Do you believe that formal methods will eventually spread more widely in industry?* | | | | |
|---|---|---|---|---|
| Definitely: 37.7% | Probably: 52.3% | Probably not: 6.9% | Definitely not: 0.0% | N/A: 3.1% |

A huge majority of 90% thinks the use of formal methods will likely become more widespread in industry, while only nine experts doubt this and four have *no opinion*.

This question received 57 comments. From the experts who doubted an increasing use of formal methods in industry, two comments base their expectation on what happened in the past; one comment thinks that "industry is just too conservative for that"; one comment urges the formal methods community to "radically change the way we 'sell' formal methods"; and one comment displays a general disappointment with society, concluding that "it does not look like that the humankind will be interested in software quality".

Fifteen comments justify an increase of the application of formal methods by an expectation for a growing demand, either because "the risk induced by (cyber-physical) systems [...] will be omnipresent", or because "with the increased advent of certification regulations, industry companies having their products certified will have a competitive advantage". Several comments mention the growing complexity of systems, which "get so complex and hardly predictable that they need all type of computer-aided support to assure safety and correct functioning". One comment identifies a new mission for formal methods: "because [unmanned, autonomous] systems cannot rely on a human operator to act when a serious problem occurs, developers of autonomous systems want high assurance that these systems behave safely and securely and that they are functionally correct".

A few comments believe that the tools will become easier to use in the future. Yet, successful technology transfers might remain confidential, since a comment reports that "once a formal method tool achieves success, it is usually given a new name, probably to avoid the stigma of being a 'formal method'". Finally, another comment expresses careful optimism: "the word is getting out! More companies are hiring formal methods engineers".

### 7.2 Future Users

This question aimed at predicting the target audience for the future applications of formal methods.

| *Who are most likely to use formal methods in the future?* | |
|---|---|
| A large number of mainstream software engineers | 42.3% |
| A small number of skilled experts | 43.8% |
| Others | 13.8% |

The experts were quite divided on this question, since the first two proposed answers attracted nearly the same number of proponents (55 vs. 57 experts).

The analysis of the 65 comments received provides more insight. From the 18 experts who selected *others*, actually 12 indicate in the comments that they believe that the answer is *both*: "a large number will make small, rote usage of the tools, [while] a small number of skilled experts will be heavy users".

Two comments express the belief that *domain experts* will be the power users of formal methods. One comment wonders how hardware experts fit in the question, but another comment confirms their role: "within hardware development, it has been standard practice to perform Logical Equivalence Checks for some time; this is a specialised use of SAT to check equivalence of two circuits and it is used for translation validation and sometimes to check manual optimisations".

Many comments further distinguish between various kinds of formal methods. Mainstream software engineers are expected to use lightweight formal methods, particularly the automated ones, which are hidden in standard development tools. This idea is present in at least 22 comments. On the other hand, specialists will always be needed to advance tool development; several comments also claim that the explicit use of formal methods will require experts, whenever skills like modelling, specification, abstraction, and interactive proof generation are involved.

It is encouraging that many experts envision a wide audience of mainstream software engineers as future users of formal methods. The fact that this is still not happening is attributed by some to the low quality of (automated) tools: "until enough progress is made to make formal methods accessible to mainstream programmers, only trained experts will be able to use the tools". Others explain the issue by a lack of appropriate scientific/technical education.

### 7.3 Promising Applications

This question tried to list all domains in which formal methods may have impact.

| *Do you foresee promising upcoming applications of formal methods?* | |
|---|---|
| In other branches of computer science | 69.2% |
| In finance (digital currencies, smart contracts, etc.) | 61.5% |
| In other sciences (biology, etc.) | 58.5% |
| In politics (e-voting, e-government, etc.) | 43.8% |
| In other parts of society | 31.5% |

*(multiple answers allowed; answers sorted by frequency)*

The experts appeared rather optimistic (60–70%) concerning new applications of formal methods in hard sciences (including biology), but less convinced (30–40%, which is still important) by applications in social/human sciences and other parts of society. A pessimistic expert added one missing option: "I see them nowhere".

The question attracted 144 comments. Many of them explicitly suggest applications in *other branches of computer science*, among which (i) software engineering: program synthesis, legacy software, aspects, product lines, human interfaces, business process modelling, etc.; (ii) networking and distributed systems: internet of things, sensor networks, security, etc. (iii) safety-critical systems: embedded systems, robotics, cyber-physical systems, control software for infrastructural systems, etc.; (iv) data science, machine learning, and artificial intelligence; and (v) traditional fields such as compilers, databases, algorithms, numerical computing. As one comment observes: "all those communities realise their problems are too hard to solve just by brute force, and growing calls for reliability in these fields are forcing investigation into and adoption of formal methods".

Among *other sciences*, the most frequently cited ones are biology, epidemiology, surgery, and medicine: "I am excited by the work in model checking of biological systems! Maybe we can even help with drug development?" Chemistry and physics are also mentioned ("one can perfectly well see a hydrogen atom as a state machine"), as well as engineering disciplines such as automotive, transportation, traffic control, aerospace, power or energy control, and the verification of numerical simulations. Social sciences, including law, are also considered as potential application areas.

The comments are optimistic about applications in *finance*, in particular to make transactions more secure. One comment phrases the urgency as: "we sit on a financial bomb". This raises many interesting challenges: "highly complicated cryptography and contracts need formalism". Several comments point to recent successes in formalising blockchains and smart contracts. There are also grander visions, such as

developing “formal models of the entire financial service industry, banks, [and] stock exchanges”.

Several comments mention that more research on *e-voting* is required, since “trust is really required in that domain”. As a comment predicts, “formal methods will show that we have to be careful with e-voting”. More ambitious expectations consider a much larger scope: “Perhaps more formalisation of laws and regulations” is needed, and “I would like to see formalised notions of fairness, causality, justice, etc.”. Finally, the comments addressing *in other parts of society* largely overlap with the aforementioned ones.

### 7.4 Potential Competitors

This question polled the experts whether other rising approaches that are increasingly in competition with formal methods to get research funding, to capture industry interest, and to attract students might eventually overshadow formal methods.

| *Do you believe that alternative approaches (e.g. artificial intelligence or quantum computing) will eventually replace formal methods?* | | | | |
|---|---|---|---|---|
| Definitely: 0.0% | Probably: 3.1% | Probably not: 41.5% | Definitely not: 51.5% | N/A: 3.8% |

A vast majority (93%) of the experts stated that formal methods will *(probably or definitely) not* be replaced by alternative methods. No expert believed that this will *definitely* happen. Five experts indicated they have *no opinion* on this matter.

Analysing the 72 comments received, 26 of them indicate that formal methods are incomparable or complementary to the proposed alternatives. A few comments criticise the question for this reason. Interestingly, 38 comments stress that formal methods and artificial intelligence can strengthen each other. Only 10 such remarks were made for quantum computing. No comment mentions another alternative than the two proposed ones.

Several comments explain why neither artificial intelligence nor quantum computing can replace formal methods. Many of them argue that only formal methods can provide guarantees about correctness, e.g. “artificial intelligence is wrong in 10 to 25% of the cases and must be hand-tuned. What is formal there?” and “artificial intelligence will need to be certified. What methods will be used to certify it?”.

A quite different reason is provided in two comments that praise the crucial role of formal methods in requirements specification: “at the end of the day, both the ambiguous setting and the mapping to the unambiguous setting are characteristic of human activities. I have a hard time imagining that these creative aspects can be fully automated”.

Other comments see fruitful interactions between both fields. Formal methods may help understand artificial intelligence, generate explanations, assist the certification of machine-learned components, or complement them with safety supervisors. In turn, artificial intelligence can improve formal methods by providing heuristics for guiding proof search: “I have seen artificial-intelligence-guided first-order provers that can learn from manual interactive proofs”. Another comment adds: “we need approaches that combine model-driven and data-driven techniques”.

Although less frequent, similar comments appear for quantum computing: “of course, quantum computing could provide a big hammer”, but “there will still be a place for formal methods to study the computational models and perform reasoning about quantum computing” and “formal methods have also been investigated to show correctness of quantum programs”.

Finally, we mention a few diverging opinions. Two comments recall that certain parts of formal methods (in particular, symbolic reasoning) were originally a branch of artificial intelligence. Two other comments fear that, in the perception of the public, artificial intelligence could replace formal methods: “but in terms of ‘branding’, formal methods might disappear from the perception of users who may think of these things more as artificial intelligence. This will require active intervention”.

### 7.5 Major Breakthroughs

This last question on the future of formal methods wanted to know whether a scientific breakthrough can be expected any time soon.

| *Do you expect that a major breakthrough (“game changer") will happen in formal methods?* | | |
|---|---|---|
| Not really: 33.8% | Within 2 years: 0.0% | Within 5 years: 12.3% |
| Within 10 years: 17.7% | Within 25 years: 6.9% | N/A: 29.2% |

The answers listed in this table are better understood by examining the 46 comments received for this question. We first discuss those comments arguing that a breakthrough is *not really* expected. Ten comments foresee a more gradual, evolutionary progress of the techniques tending to their widespread adoption. Five comments point to external factors, e.g. “societal and economic factors” and “the cultural barrier to the use of formal methods”. Another comment shows some hope: “the only game changer I would see, if more and more standards, certifications, and regulations demand the kind of guarantees that only formal methods can provide”.

From the comments that, sooner or later, expect a breakthrough, three of them see it coming from killer applications, six others from a particular combination of methods, and four others from a single technical development that could be a game changer. Interestingly, the advances in SAT/SMT solvers are mentioned, in those three sets of comments, as the example of the most recent game changer.

An example of such a potentially groundbreaking combination is given: “just as tools (e.g. solvers) in formal methods have grown tremendously over the past few decades, so too have tools in other areas: most obviously in machine learning, but also in fields like stochastic optimisation. Putting together all these in meaningful ways may lead to dramatic improvements in all of them”.

Three examples of individual advances that could lead to a breakthrough are also given: “synthesis of correct-by-construction control components for critical systems”, “serious use of models instead of programs, coupled with automatic generation of code”, and applications of “big data” or “quantum computing”.

## 8 Conclusion

Formal methods are now more than 50 years old, and after half a century of sustained research, development of new techniques, and continuous enlargement of the perimeter of formal methods, it was high time to review the situation: the 25th anniversary of the FMICS conference was a suitable opportunity to do so.

The present survey is an unprecedented effort to gather the collective knowledge of the formal methods community. Not less than 130 internationally renowned experts agreed to participate in the survey and spent significant time to express their views, through answers to our questionnaire, through detailed comments accompanying these answers, and through position statements that deliver the personal opinions of these key actors in formal methods.

Many lessons can be learned from all these contributions, the collective ones are synthesised in Sections 3–7, while the individual ones can be found in Section 9. The general opinion is that formal methods achieve many technical successes, but are not yet mainstream to their full potential. There is still much to be done and, among all the pending tasks, we wish to highlight three action points more particularly:

- The results of the survey indicate a consensus about the essential role of *education* to give the next generations of students a sufficient background and practical experience in formal methods. Unfortunately, it appears that the current situation is very heterogeneous across universities, and many experts call for a standardisation of university curricula with respect to formal methods. A recent white paper [7] provides a good starting point for such an undertaking. Also, one should not neglect continuing education and make sure that industry professionals who did not attend university classes can learn about formal methods from alternative channels (online courses, tutorial videos, etc.).
- The results of the survey also make it clear that formal methods are no longer a paper-and-pencil activity: like other fields such as logic and computer algebra, formal methods have shifted their orientation, and their progress now closely relies on *software tools*. A majority of experts considers that universities and research institutes have a central role to play in the construction of such tools. However, software development is often underrated by standard academic evaluation, which primarily measures excellence in terms of publications in scientific conferences and journals. Thus, many experts call for a revision of the current academic reward system to better encourage long-term investment in the development of innovative, high-quality software. Researchers are also invited to join forces to build common platforms that can become part of mainstream development practices.
- Computing takes an ever-growing importance in modern societies but is still much less regulated than other sectors (transportation, real estate, healthcare, etc.), even though software or hardware bugs may have dire consequences in an increasingly connected digital world. The industrial dissemination of formal methods really progresses when companies that produce software or software-intensive systems decide to protect the safety, security, and privacy of their customers — thereby protecting their own assets and reputation at the same time.

The current incentives for such virtuous behaviour are probably not enough, and many experts call for a greater *regulation* of software quality (beyond the traditionally supervised sectors of aerospace, nuclear energy, and railways), with stricter standards that examine the final product rather than its development process, and a stronger promotion of best practices by professional bodies. Such measures (together with, e.g. finer risk assessment of software products by insurance companies) would turn formal methods into a profitable investment.

We hope that this survey will highlight the potential of formal methods to the numerous stakeholders (policy makers, regulators, company managers, research funding agencies, professionals, students, etc.) and encourage industry to use these methods more intensively.

## 9 Position Statements

Each expert who answered all 30 questions of the survey was then warmly invited (yet not required) to write a short statement (not exceeding 10 lines) about formal methods. Guidelines were given in the form of three questions, with the expectation that each position statement would address these questions, or a subset of them:

1. Personally, what do you consider to be the next challenges for formal methods?
2. How are you currently contributing to these efforts?
3. Which of your contributions could be most beneficially picked up and carried forward by the next generation?

It was stated that each position statement would be nominally attributed to its author, the intention being to confront individual visions from many high-profile experts — contrary to the 30 questions of the survey, whose answers would be anonymised to distil the collective opinion of the formal methods community.

Nearly 60 acronyms occur in the position statements. In below tables, we only expand those needed to understand ideas; we neither detail acronyms defined before being used in position statements, nor well-known acronyms (OS, PC), nor names of computer languages (UML, VDM), software tools (CADP, SPIN), universities (ECNU, RWTH), organisations (ISO, NASA), and conferences/workshops (FMICS, MARS).

| Acronym | Signification | Acronym | Signification |
|---|---|---|---|
| AI | Artificial Intelligence | IT | Information Technology |
| CPS | Cyber-Physical System | JSON | JavaScript Object Notation |
| CTL | Computation Tree Logic | LTL | Linear Temporal Logic |
| DSL | Domain-Specific Language | ML | Machine Learning |
| FM(s) | Formal Method(s) | PLC | Programmable Logic Controller |
| GPU | Graphics Processing Unit | PR | Public Relations |
| GUI | Graphical User Interface | REST | Representational State Transfer |
| HMI | Human-Machine Interface | SAT | Boolean Satisfiability Problem |
| HW | Hardware | SMT | Satisfiability Modulo Theories |
| IP | Intellectual Property | SW | Software |

Please note that each position statement reflects only its authors' views, and not necessarily the opinions of the authors of this report, nor those of their employers.

The scope of FMs extends beyond program and model verification. FMs are applied in Workflow and Business Process Management, and recently in data-driven approaches like process mining. This reconfirms the importance of concurrency theory, a subfield of FMs rooted in early work of Carl Adam Petri. Petri's guiding principle was "Concurrency should be a starting point for system design and analysis and not added as an afterthought (locality of actions)". Operational processes are inherently concurrent, and the availability of event data allows to discover such processes. I anticipate FMs and data science to converge. We need FMs to describe real-world phenomena, and the abundance of data offers a unique opportunity. Thus, the practical applicability of FMs will continue to increase. It is vital that students learn to abstract and structure, and FMs are the tool for this. Edsger Dijkstra once said "Beauty Is Our Business". I would like to add "...and business is good".

*Wil van der Aalst*

The pioneering works of Clarke, Dijkstra, Emerson, Hoare, Milner, Pnueli, Sifakis, Vardi, and many others set the stage for an exciting area that lays the foundations of computer science. In the early days, program semantics and verification were restricted to small idealized calculi, but there has been a rise in rigorous engineering methods to elucidate and analyze real-life problems of processor architecture, programming languages, computer networks, database systems, etc. Formal methods also play an essential role in education by teaching students abstract thinking, problem solving, and the ability to communicate new ideas in an articulate manner. Thus, integrating formal methods in under- and post-graduate education is vital to make students more prepared to their academic or professional careers regardless of their areas of specialization.

*Parosh Abdulla*

Being addicted to the beauty of math, I look back to two decades of joyful time to develop and connect mathematically rigorous approaches for the modeling and analysis of various types of systems. In the area of fast rising new technologies for autonomous and learning systems, the formal methods community enthusiastically adapts, extends and creates new ways to contribute to the development of correct and safe systems. But our achievements unfold their full potential only if there are people who are willing to as well as able to use the developed methodologies and tools. We should put the strongest possible weight on education, explaining elegant formal methods algorithms, which are perfectly suited to awake interest, train precise analytical thinking and to prepare to use formal methods tools.

*Erika Ábrahám*

Formal methods should be uniquely those which are concerned with *proofs*. This eliminates many activities pretending to be formal methods. Companies give formal methods a mixed reception: some industries (e.g., railways) are fervent adopters while others (e.g., aerospace) firmly resist, often because managers are reluctant to introduce new approaches that disrupt current work methodologies that took much effort to install. Such resistance is likely to stay for the next thirty years. Teaching formal methods in academia is fundamental. Unfortunately many teachers are reluctant because they do not want to put sufficient effort into competence in proving. I am currently contributing to these efforts by teaching, developing tools, and cooperating with Chinese universities like ECNU.

*Jean-Raymond Abrial*

Emerging machine-learning algorithms are enabling new generations of autonomous systems in robotics, medicine, and transportation. Two challenges are that (1) safety is of paramount importance in these applications, and (2) it is unclear how to integrate data-driven models generated by machine-learning algorithms in complex software systems. Their combination offers promising opportunities for formal methods, both in terms of academic research and integration in industrial practice. At the intersection of formal methods and machine learning, there are many research problems ranging from integrating formal specifications in training algorithms to formally verifying systems that include, say, a neural-network-based controller. These are all challenging problems, and progress is likely to be achieved by focusing on specific case studies first.

*Rajeev Alur*

The overwhelming complexity of systems critical to our society emanates not from complicated computations, but from latent combinatorics of the potential interactions of their parts. It is notoriously hard to carve a protocol out of such an exponentially large interaction space to correctly manifest the desired behavior of a system. It is even more difficult to ensure that such a protocol simultaneously excludes other unforeseen slices of that vast space that constitute undesired behaviors that, e.g., compromise privacy, safety, or security.
Reo is a language for compositional construction of protocols, based on a model of concurrency that treats interaction as its only first-class concept. In Reo, protocols become concrete software constructs independently specified and compiled, which one can separately verify and reuse, or compose into more complex protocols.

*Farhad Arbab*

In my experience, there is a lot of progress in formal languages to express data, basically types, such as JSON Schema. Engineers use these languages in practice, but struggle constructing more complex type definitions. Far too often I have seen “string” as a type with a comment that this string should contain a date or similar. This shows a lag between academia, for which this is no longer an issue, and industrial practice. By automatically generating test cases for such type definitions, we contribute to improved specifications and more reliable code. Expressing behaviour is even more of a struggle in practice. General languages to express such behaviour are not widely accepted. But embedded in patterns (like REST) they widely exist. Formal methods should use the semantics embedded in patterns to assist developers.

*Thomas Arts*

With a PhD in mathematics, I have devoted a large part of my scientific life to the mathematical underpinnings of computer science, thus contributing to the science of “computer science”. I am happy to have played a part in this. I view formal methods as the application of these mathematical underpinnings to the practice of computer science. I am not so satisfied with the achievements I have been able to make in formal methods, and hope others in the community can do better. I find that convincing IT companies to adopt formal methods is more difficult than convincing companies making other products (that are software-intensive). I find mechanical engineering students more interested in learning and using formal methods than computer science students. Therefore, maybe the best way forward is to work in systems engineering rather than software engineering.

*Jos Baeten*

For me, the main challenges are better formal specification and programming languages, simpler general verification assistants, more powerful automatic methods and software, and, above all, more widespread tools and education with respect to these subjects. I used to be quite active in academia and industry on the development of formal synchronous languages, which reconcile concurrency and determinism while supporting fully formal verification and compiling technologies: Esterel v5 for embedded software, and Esterel v7 for hardware microarchitecture, with semantics-based links between these languages and formal verification techniques. I did it first in academia with strong links to software and hardware industry, then as chief scientist of the Esterel Technologies company. I finally taught this, among other things, as a professor at Collège de France.

*Gérard Berry*

The biggest challenge is to put formal methods into a language and software process that is easy to access and similar to existing approaches. One has to make sure that such a new formal approach is combined with other important parts of system design, particularly testing, fuzzing, debugging, coverage etc. The next most important aspect is how to combine inductive and deductive reasoning for rigorous design of systems with a learning component. Third, there are still huge potentials for improving logical reasoning tools, which then will allow to tackle larger problems. This last challenge is of course the one on which I focus most, also because I have witnessed a steady increase in scalability of for instance SAT and SMT solvers, which led to a wider adoption of formal methods in academia and industry.

*Armin Biere*

In formal software development, one should first understand and describe the domain, then analyse and prescribe the requirements, before finally specifying and designing the software itself. Descriptions, prescriptions, and specifications should be mathematical entities, formulated in one or more languages (e.g. VDM, Z, B, Alloy, CASL, CafeOBJ, Maude, or RAISE), which have formal syntax, a mathematically expressed semantics, and one or more proof systems that (more or less) cover the entire language. Formal software development involves a series of iterated domain descriptions, requirements prescriptions, and software specifications, where mathematical properties can be expressed and reasoned upon for each of these phases, and where correct transformations between them can likewise be formally argued.

*Dines Bjørner*

There is no shortage of next challenges for formal methods. The emergence of large-scale cloud-based systems presents one such opportunity; and many challenges. They are the ultimate dream/nightmare of IT administrators obsessed with solutions to capture *intent*, driving systems to *goal states* in *day two* of deployments that integrate many interoperating pieces. It is an opportunity to build foundations for semantic interoperability, interfaces and tools to ensure security and reliability. It also presents opportunities to add formal methods value to systems driven by a continuous life-cycle involving measurement, synthesis, optimization and deployment. I contribute to this line by deploying the SMT solver Z3 and other formal tools in the Azure cloud.

*Nikolaj Bjørner*

Recently, NASA looked for a “formal methods engineer”. We should always present formal methods explicitly and consistently as an engineering discipline within computer science: “formal methods” alone is a too generic term. Underlining the engineering aspect in research and education clarifies their practical role. We should teach “FM engineering” as a specialisation of software engineering. FM engineering applies formal methods to improve software reliability, but also involves software development to support the application of formal methods. Underlining the engineering aspect in research would counter-balance the limited understanding of formal methods as a purely theoretical research topic. Hopefully, underlining the engineering aspect will help promote formal methods in academy and industry, and the term “formal methods (software) engineering” will broaden the understanding and appreciation of formal methods.

*Frank de Boer*

Given that my main occupation in the past ten years has been in university management, I prefer to formulate my position in terms of two general principles. First, the next challenge for formal methods is always to remain relevant, i.e. to develop methods and tools to design systems with desirable properties in the context of new computational paradigms. Current examples are machine learning, big data, probabilistic programming, quantum-inspired computing — I am sceptical about quantum computers, but very interested by novel hardware architectures faking them. Second, always try and combine good theories with real applications, and try to contribute to both. This entails developing and adapting theory, systematic tool development, and having in-depth knowledge of relevant application domains.

*Ed Brinksma*

Industrial critical software and cyber-physical systems need formal techniques to guarantee correctness, safety, security and long-term quality. We need scientific progress in techniques to model critical aspects, progress in practical methods and tools to deal with large, complex systems, and integration through experimentation. While we have seen a lot of progress, we are far from a satisfying situation. Scientific research and results in formal modelling and verification often do not address the most relevant issues from practice, just like interesting scientific approaches often are not really evaluated by practitioners and commercial tools do not integrate them. Reasons include insufficient education and insufficient understanding by managers. I would like to see much more intensive collaboration between science and practice, supporting also the development of effective and powerful tools.

*Manfred Broy*

Our society’s safety and well-being depend on our software infrastructure, and current inspiring perspectives for robotics and autonomous systems will increase such reliance. Industry has applied formal methods for software engineering successfully, but we need that to become standard practice. It is possible to develop useful systems without formal methods, but if software engineering is to be truly an engineering discipline, based on mathematics, then we have to cater for the theory that explains practice. Formal methods are the scientific backbone of software engineering and can be used to improve quality and to reduce costs. To maximise this potential impact, we have to deal with usability via palatable notations and effective tools. We have done a lot as a community. We have a lot of very exciting work still to do.

*Ana Cavalcanti*

A large part of research on automated symbolic verification has been focusing on developing dedicated engines for model checking. This has led to huge progress in scalability, thanks to SAT-based methods for the finite-state case, and to SMT-based methods, in combination with automated abstractions, for the infinite-state case. There are equally important problems that deserve attention, in order to increase the penetration of formal methods in the standard process for system design: requirements modeling and validation; provably correct contract-based design; safety assessment, i.e. methods to analyze a system's response under faults; design-space exploration of parametric models for the identification of configurations meeting the desired requirements.

*Alessandro Cimatti*

I believe the great future challenge for formal methods is for users to stop being pleasantly surprised when they work, as is typically the case currently, and instead to be irritated when they do not! Achieving this vision requires notational, methodological, technological and outreach-focused advances. Notations need to be standardized within domains and, together with analysis tooling, embedded in design processes. New technologies such as quantum and machine learning need formal support. Students and professionals also need exposure to formal methods. My contributions currently are focused on approaches for inferring system properties from observations of system behavior, to provide a bridge to formal methods for engineers who are not versed in them.

*Rance Cleaveland*

Expanded use of formal methods in aircraft software is crucial due to the high cost of verification and certification activities, as well as the extremely high potential cost (both financial and human) of design defects that escape into service. Our experience shows that formal methods can both reduce costs (through automation and reduced rework) and eliminate hard-to-find defects. However, we are not starting with a blank slate but face huge process inertia. We are working now to incorporate formal methods into model-based system engineering environments. This presents a great opportunity to use a common system design model throughout the life-cycle to drive safety/security analysis, system development, verification, infrastructure code generation, and certification evidence.

*Darren Cofer*

Within the formal methods spectrum, I mainly worked on fully automated verification, which either restricts itself to decidable cases, or requires sound (but sometimes incomplete) approximations. I specialized on the latter approach, working on theory (abstract interpretation), practice (static analysis), and education of students, designers, and end-users. This led to Astrée [1], a successful tool for proving the absence of runtime errors and invalid concurrent behaviour in safety-critical software written or generated in C. The next challenges for automated formal verification are threefold: *scope* (coping with a variety of specification and programming languages), *scalability* (analysing programs of millions of lines with reasonable resources and sufficient precision), and *applicability* (designing formal methods that can be inserted in industrial development methodologies at an acceptable cost).

*Patrick Cousot*

The challenge of the formal methods community is to have them properly included in the system development process. Today, most companies are not willing to adopt formal methods as they believe it is costly and time demanding. Since the software industry process is not properly regulated, companies bluntly cross the ethics boundaries. Thus, it is my belief that we need public policies that properly regulate software industry, forcing companies to include rigorous and mathematically based techniques and to properly document it in the different stages of the development process. Cost and time might be an issue at the beginning, but once formal methods are normally adopted they will have a positive impact in both aspects. For this, we need that governments really understand the consequences of a bad software product.

*Pedro R. D'Argenio*

I have believed for several decades that the primary barrier to widespread adoption of formal methods is not educating engineers, friendly notation, or better GUIs. It is the value and productivity of the methods in expert hands. We need to be able to develop demonstrably superior systems at a cost that is significantly less than the benefit of applying the methods. The only path I can see is co-development of formal methods, tools, system design methods (including languages, programming patterns, etc.) to maximize the benefit and minimize the cost of developing high-quality systems in particular application areas. It will require an iterative effort and a single-minded focus on optimizing the value of the method, with creativity and without dogma.

*David L. Dill*

Most of the work on formal methods so far has been contending against *stupidity*. It is so easy for humans to commit programming or design mistakes that the supply of stupidity is almost infinite and we will never run out of issues to work on. Although the poet and philosopher Friedrich Schiller wrote: "Against stupidity the gods themselves contend in vain", formal methods are valuable and effective enough to prevent or detect certain classes of mistakes. But more work on formal methods is now needed to contend against *malevolence*. It is much too easy for attackers to exploit software vulnerabilities, such as buffer overflows, dangling pointers, and race conditions. Our techniques can also address such issues, and the area of formal methods for cybersecurity should definitely receive more attention.

*E. Allen Emerson*

As most mainstream programming languages have not been designed to be analyzed formally, the adoption of formal methods in industry should be addressed by developing tools that operate at scale on legacy and existing programming environments. One should also pave the way to a better future by inventing languages with sound bases for formal analysis, and equip these languages with effective development tools, including comprehensive libraries and code profilers. Concurrency remains a major challenge, especially with the rise of general-purpose GPU computing, from data centers to mobile devices. The programming models of such heterogeneous systems may have underspecified concurrency aspects (such as scheduling fairness, which is the focus of my current research at Google) that formal methods help to clarify and verify.

*Hugues Evrard*

The next challenges for formal methods are related to the pervasive use of artificial intelligence (machine-learning engines) in safety-critical applications: formal methods will be called in to help certifying the safety of such systems, which is a very hard job due to the extremely vague notion of "correctness" that can be ascribed to this kind of systems. For the latter challenge of certifying the safety of such systems, which has very recently emerged as really important, I conjecture that the notion of a *safety envelop* for a moving object will need to be extended to objects moving in an uncertain environment. My own recent contributions on formal modelling and formal verification of various kinds of railway signalling systems can be fruitfully considered for such systems, and extended outside the railway domain.

*Alessandro Fantechi*

Traditionally, formal methods have focussed on functional correctness, but the Meltdown and Spectre attacks demonstrated that it is not enough: even in processor designs where each individual execution is correct, it may be possible for an attacker to obtain secret information by comparing multiple executions. Formal methods need to treat concerns like information-flow security and privacy (and other non-functional requirements, such as robustness and perspicuity) with the same level of rigour as functional correctness. In my opinion, a central role in this new class of formal methods will be played by hyperproperties, which generalize trace properties to relations between multiple traces and can express many non-functional properties of interest. We recently developed algorithms that monitor, verify, and even synthesize systems from hyperproperties: this shows great promise, but plenty of interesting questions are still open.

*Bernd Finkbeiner*

As computing technology becomes integrated into our physical and social fabric, formal methods can help create systems on which billions rely, from secure data networks to sustainable water supplies. Such projects require *collaborative* methods, tools that mesh with those of other disciplines, and researchers and practitioners who can create and use them. Our work at Newcastle University envisions collaborative modelling, co-simulation and co-verification. There are challenges not only in laying the foundations of these techniques, but in integrating robust methods and tools at the level of systems and systems-of-systems. Meeting these challenges requires a generation of formalists who can work across traditional divides between disciplines and departments — and that means a more open, intellectually and socially diverse formal methods community.

*John Fitzgerald*

At Eindhoven University of Technology we cooperate with Rijkswaterstaat, which is responsible for development and maintenance of infrastructure in the Netherlands. A wide range of companies make the control software of bridges, waterway locks, tunnels in a variety of ways, leading to software that cannot be easily maintained. We turn system requirements into PLC code automatically by means of supervisory control synthesis. We are now involved in the design process of several infrastructural systems. Recently, a bridge was operated in real life by software generated automatically from the requirements. PhD students on the projects have pushed the boundaries of supervisory control synthesis by developing novel specification and verification techniques that have been pivotal in scaling this method to real-life applications.

*Wan Fokkink*

My area of research is the verification of cyber-physical systems, where continuous variables evolve with time and interact with control software. The proliferation of artificial intelligence (AI) in perception and decision-making poses a formidable verification challenge, e.g., in robotics and autonomous vehicles. We investigate how formal methods can assist such that safety is guaranteed both during training and operation. We enhance the learning algorithms and add a supervisor (shield) that interferes in time to avoid critical behavior. Our work on efficient yet precise set propagation and abstraction can help to ensure that such checks are fast, while at the same time being accurate enough to provide the required guarantees without being overly cautious.

*Goran Frehse*

Software-intensive systems increasingly behave as autonomous entities living in the physical world, augmenting it, collaborating with humans, and offering new advanced functionalities. Novel opportunities and challenges arise for formal methods, to support both development and operation of such systems. Design often requires multi-disciplinary, domain-specific competences. In addition, design decisions must often be made in the presence of high levels of uncertainty about the embedding physical world. Formal modeling notations and validation methods must be revisited and engineered to effectively comply with these issues. Formal models and verification must also live at run-time to support dependable support to autonomy. In particular, they should allow software to self-adapt to detected changes in the environment's behavior and offer support to co-evolution with humans.

*Carlo Ghezzi*

Whole system assurance I see as a next challenge for formal methods: a coordinated verification effort covering the entire chain from hardware and operating systems to selected user applications. Proving liveness properties, saying that systems will do what they are meant to do, is a high priority for me. A lot of work has been done on safety and security, but often separated from checking functional correctness. I see good prospects to address functional correctness in combination with safety and security properties. To ensure the sustained application of formal methods, we need forums and repositories for showcasing the fruits of formal modelling and verification. I am proud to be one of the founders of the MARS workshop and repository, which focusses on the formal modelling of real systems.

*Rob van Glabbeek*

The next challenges for formal methods are in their pervasive application in the development of more and more sophisticated cyber-physical systems, such as for example autonomous systems, to assure their dependability, safety, and security. One aspect of these challenges concerns the early formalisation phase. In this area, I have recently focussed on methods and tools to remove defects in natural language requirements and to avoid possible misinterpretations. The design of critical systems can benefit from the paradigm of software product lines, which allows developing families of systems starting from the same initial model. In this field, we have provided a behavioural and logical framework for modelling and analysis of safety-critical systems, such as railway control systems.

*Stefania Gnesi*

Formal specification and verification methods have made tremendous progress over the last decades. Some have by now been adopted in many industrial software and hardware domains. In the software world, type checking is probably the most widespread use of formal verification. As another example (dear to my heart), our SAGE project at Microsoft has tested and verified memory safety of large parts of the Windows OS and of Office applications using formal program analysis techniques like symbolic execution, constraint generation and solving, by formalizing x86 semantics and leveraging SMT solvers; as a result, a billion users world-wide now enjoy their PCs more safely and securely. Of course, much is still to be accomplished to have even more impact.

*Patrice Godefroid*

With the end of Moore's law, we face extreme parallelism leading to data races and the inexorable costs of data movement. There is a dire need for race checkers based on formal principles, that handle irregular computations, and are usable. Reduced-precision floating-point is fundamental to reducing data movement. Rigorous and scalable error analysis methods are essential to licensing the use of reduced precision. Our research is contributing to the development of race checkers for OpenMP. They have caught multiple data races in large-scale projects. Our research is also contributing to rigorous floating-point error analysis that has pushed the boundary up by several orders of magnitude. We expect both our race checking methods and error estimation methods to be perpetuated.

*Ganesh Gopalakrishnan*

Currently, I see the following two challenges. 1) How do we learn to use formal methods and formal modelling in a practical context such that we can reap the benefits of formal methods (i.e. verification of properties and overall correctness; modelling for verification). This aspect hardly received attention, because so few researchers model substantial systems. 2) How do we change the (industrial) society such that they will incorporate formal methods within their production processes on a large scale. This is hardly scientific, as it requires tool builders, service providers, willing managers, probably changes of the law regarding software responsibility, etc.

*Jan Friso Groote*

A main challenge in formal methods is to keep advancing the theory, methodologies and tools for handling new problems arising in the real-world, as well as new application areas. My work is aimed at developing new approaches to new problems while exploiting and incorporating established "old" knowledge into the novel ones, as well as combining ideas from other disciplines. Another important challenge is education, to make sure that mainstream software engineers are aware of formal methods, understand them and are willing to apply them. I teach a yearly large undergraduate course on formal methods to software engineering students, which then carry the knowledge into industry. They apply formal methods and develop industrial verification tools. My research students join academic research or lead verification groups in industry.

*Orna Grumberg*

Formal methods are well established in a few specialized sectors in industry where risk is very high, financially or even in the form of endangering people. In these cases, formal methods are most often used by domain and method experts. It is a big challenge to evolve formal methods in such a way that they can be used successfully in other sectors as well. For several years, I used formal methods in different sectors: safety critical systems, hardware verification, software model checking and cryptocurrencies. Now, my main goal is teaching students to make them aware of the possibilities of using formal methods, to write formal specifications, and to use formal analysis tools on real problems. I deem it very important to give a realistic view of advantages but also limits of formal methods.

*Matthias Güdemann*

The main challenge for formal methods is very basic: how to discover loop invariants. We still can't get around that barrier. I do, however, see some reasons to believe that this problem will get more focus: the introduction of programming languages supported by proof systems. I do expect it to become more common with programming language implementations being born with program verifiers of various kinds. This can bring formal methods in the hands of software developers, and put pressure on the research community to address the right problems. It is a pleasure to see modern programming languages to an increasing degree look and feel like well known formal specification languages. There is a convergence it seems, of formal methods and programming language design and implementation.

*Klaus Havelund*

If formal methods should have a future, it is now time to focus more on their industrial adoption: More collaboration projects with industry are needed. In these projects, methods and tools should be industrialised and cost-benefit analyses should be carried out. It should be investigated how formal methods can be integrated into existing software life cycles and made simpler to use, e.g. by the provision of professional tools encapsulating the use of formal methods. First Movers (companies providing or applying formal methods) should help drive the marketing and spreading of formal methods to industry. In courses, we should not only teach theory and show toy examples, but also show how formal methods can be integrated into existing life cycles and used in industrial applications.

*Anne Haxthausen*

In the past, our research and most other formal methods research has focused on verifying, or detecting defects in, formal models of critical software systems. While such research was extremely valuable, ultimately, assurance is needed that the executable system code satisfies critical properties, such as safety and correctness. One relatively new application of formal methods, called run-time verification, aims at formally verifying the system code. This code will often rely on AI techniques, such as machine learning, to perform its functions. The aim of our recent research is to develop a comprehensive set of new formal methods and robust, usable tools that support run-time verification and provide assurance of machine-learning systems. This new technology will be used throughout the system development process to ensure that requirements are satisfied, with a focus on the safety and correctness of the important and steadily growing class of autonomous systems.

*Constance L. Heitmeyer*

I would like to put forward a reasonable conjecture that the next virus to threaten civilisation will be an infovirus. It may be accidental (due to software error), or due to rumour (misinformation), or to unsubstantiated but deeply held beliefs (e.g. about chloroquine or measles), or to an extreme political agenda (disinformation), or it may be due to malice (malware). To imagine the consequences, think of something like the present coronavirus attack, in which all connected computers, both public and private, have closed down. The closure could be permanent, because each component relies on another component to restore it to a stable state.

I would like to make an urgent plea that researchers from the entire computer science community should participate in a project to reduce the risks and consequences of a potential infovirus pandemic. Research into formal programming methods should aim to reduce the risk of accident, and increase the likelihood of recovery. We also need to define and verify security properties of basic hypervisors, supervisors, and other critical basic software. Finally all programmers should lobby their professional organisations, their employers, and their elected political representatives to establish and implement verifiable standards that keep the reproduction rate of the infovirus low.

*Tony Hoare*

I believe that there is considerable benefit in the further development of formal methods based tools that can perform analyses in real-time, interactively. In my own work, I have tried to develop two types of methods that can achieve this. The first concerns a swarm verification strategy that is aimed at large cloud computing environments, where we launch large numbers (up to millions) of very small, and very quickly executing search engines in the cloud, to jointly deliver a verification result with high confidence of full coverage of a complex problem domain. The second concerns the development of the Cobra tool (github/nimble-code) to perform interactive static analysis on large code archives, including, most recently, new analyses for cybersecurity vulnerabilities. More research in this area of interactive formal analysis is needed.

*Gerard Holzmann*

How to make software reliable is an important open challenge, and formal methods will play a crucial role in this. To address this challenge, it is essential to close the gap that currently exists between industrial practice and the academic state-of-the-art. This means that we have to make formal methods usable for engineers developing large-scale software. However, this is a challenge that will not be solved from one day to the other: it is a long-term process, and requires serious investment from both sides. As a scientific community, we can help this process by giving academic rewards for investing in (long-term) collaborations with industry, adapting tools to make them practically usable, and by providing good training material that can help engineers to apply suitable formal methods to their products.

*Marieke Huisman*

The future use of formal methods for the development of hardware/software systems depends on demonstrated cost reductions. Microprocessor vendors have been able to save tens of millions of dollars of development costs annually by proving the correctness of parts of their designs. A mathematical proof run on a single small machine can replace millions of hours of simulation run on thousands of machines. Formal specifications provide designers with clear targets, both for engineers implementing such specifications and for users of precisely-defined IP blocks and subroutines. My recommendation for the formal-methods community is to continue demonstrating and emphasizing the cost savings, quality improvements, and shortened time-to-market, that a formal-methods-based process can provide.

*Warren A. Hunt Jr.*

Formal methods have made tremendous progress in foundational underpinnings and tool development! But formal methods, as a School of Thought, may look alien to software developers who use Jira and Github, do Sprint plannings and Sprint reviews and, think of Epics, User Stories, and Tasks. Getting adoption and impact in that world means being able to support these activities and ways of thinking. "Formal methods" also needs to cultivate its branding and do some effective PR. It has a lot of — almost invisible — but lasting and far reaching impact, e.g. in modern programming languages. These are missed PR opportunities. At present, I work in a startup for privacy-preserving collaborative AI — designing efficient protocols. Future generations may appreciate that I have a nuanced and humble view of the place of formal methods.

*Michael Huth*

I am convinced that formal methods are the only means to tackle the ever growing complexity of systems, whether it is software or hardware or a combination thereof. I am not sure that we have already reached the point where the benefits really overcome the costs (except in specific domains, such as hardware), but I continue to try making formal methods percolate to industry. Formal methods will also probably be the only way to gain confidence in systems embedding machine-learning algorithms. This is actually a new and fantastic challenge, surely requiring new formal techniques. I sincerely hope that the current hype about AI, and the general movement of people (including students) and industry towards that direction will not end-up marginalizing even more the effort towards the industrialization of formal methods.

*Eric Jenn*

I believe that formal methods are best suited to express, analyze and organize models, and that interesting application domains for formal methods may be found not only in software engineering and computer science, but anywhere in science or society where correct models matter. I also believe that to increase the impact of formal methods, it may be useful to frame them as powerful extensions of simulation, which is a technique that everyone understands and accepts. In my work on the ABS modeling language, focus has always been on formal methods as executable high-level programs with clearly defined semantics and additional analysis support.

*Einar Broch Johnsen*

To me, the challenge is to get formal methods used all the way from formal specification through a verified design process: although it is possible to analyse code and detect classes of errors, it is more cost-effective to detect mistaken design decisions before further development is undertaken. For sequential programs, methods such as VDM and (Event-)B have shown that this can be done. Concurrency has proved to be more challenging, but ideas like the rely/guarantee method have shown that compositionality can also be achieved for concurrent programs. Tool support is essential if methods are to be used widely.

*Cliff B. Jones*

Formal methods developed enormously in the last 25 years. They led to various international standards and influenced more recent ones, such as ISO 26262, which prescribes formal methods for the automotive domain. Software tool capabilities made incredible progress, and major software and hardware companies invest considerably in making formal methods industrially applicable. To be successful in the future, I believe that we need much more emphasis on "lightweight" formal methods: techniques that can be applied on a daily basis by system engineers in the same way as they use compilers and debuggers. This requires orchestrated efforts by the research community and industry. Examples of such efforts at RWTH Aachen are the COMPASS toolset for AADL (developed together with FBK since 2008 with ESA funding) and an IC3-based software model checker (joint work with Siemens).

*Joost-Pieter Katoen*

Recall the parable of the Blind Men and an Elephant: e.g., an elephant is like a snake if we touch only its trunk. Thus, formal methods may seem to be automatic equivalence checking if we only compare related hardware models, but may seem to be interactive theorem proving (ITP) if we only verify deep properties. Let's embrace the entire elephant! Note that many ITP systems — including the one I know, ACL2, which is used daily at several companies — support the use of automatic tools in human+machine proof development. As hardware and software continue to grow in complexity, their successful verification will demand further education and research on a variety of mechanized formal methods.

*Matt Kaufmann*

It has been amply demonstrated that even deep formal methods like interactive proof can be used to good effect on real, deployed systems. The challenge is to make these methods scale to larger systems and at the same time make their application cheaper, to reach the point where deep formal methods are economically preferable to normal software development. Our projects such as the seL4 microkernel verification are within a cost factor of less than 4 compared to standard high-quality software development, and work on proof engineering, better tools, and better methods looks promising to overcome this factor in the future.

*Gerwin Klein*

Formal methods should become a more ecumenical community, as it is now understood that each approach has it own strengths and weaknesses, so that no formal method alone can be a silver bullet. One must thus combine several approaches by connecting modelling languages to the various verification engines available. I am addressing this long-term objective in the particular area of model checking. Specifically, I explore *adaptive* model checking, which tries to dynamically select the most appropriate algorithms and heuristics for a given model under verification and a given property to verify. Software competitions are helpful for this purpose, as they allow a fair comparison of tools and algorithms on a common set of benchmarks, and encourage the development of gateways between different software implementations, or even between different formal methods.

*Fabrice Kordon*

My vision, after more than 25 years in this field, is still that formal methods will become a "natural" part of industrial software and systems engineering like compiler and simulation technology already is today. This requires appropriately adapted methods, user-oriented tools, and solid integration into engineering curricula, such that these methods can be applied by engineers with only a basic understanding of the underlying concepts but without deep knowledge of theoretical results. I believe that making formal methods applicable is a research topic of its own which has been neglected in the past. It has to be different than the usual formal methods research, e.g., by involving user studies instead of theorems and proofs. But it is key for the future of our field.

*Stefan Kowalewski*

A major challenge confronting formal methods is usability. For too long we have ignored it entirely, and research shows this was ill-advised. Researchers still have very naive views of what "usability" even means: e.g., they assume one is speaking of slapping a GUI on a tool. Rather, human factors methods need to be applied to all parts of the pipeline, from the languages and notations we use for specifying problems, to the methods we use for presenting output, the modalities we offer for working, and so on.
I focus on two aspects. One, I am explicitly applying human factors to formal methods tools to tackle the issues listed above. Second, I am also applying education research methods to the way we teach formal methods, understanding what problems students actually confront and devising teaching methods and tools to address those.

*Shriram Krishnamurthi*

Formal methods are unavoidable when concurrency is at stake, but they often face the complexity wall of state-space explosion. I am confident that such limits can ultimately be overcome by using well-defined concurrent languages, together with compositional verification techniques based on divide-and-conquer approaches exploiting property-preserving reductions. To this aim, I contribute to the design and implementation of LNT, a next-generation language combining the best traits of imperative languages, functional languages, and process calculi. I am also advancing the effectiveness and user-friendliness of compositional verification, whose implementation in the CADP toolbox successfully tackled all the CTL and LTL parallel problems proposed at the RERS 2019 competition.

*Frédéric Lang*

Industrial impact requires an evolution of both their methods and our tools, potentially in several iterations of collaboration with academia. Formal methods tools must fit development methodology applied by industry, and it may be necessary to create domain specific formalisms for maximal impact. Sustained industrial use needs repeated committed collaboration. For increased impact it is important that more academic tools become available on commercial terms from spin-out companies. As a next important step I envisage that synthesis of correct-by-construction control components of critical systems will disrupt the way that such systems are currently constructed. The complexity of such synthesis may even benefit from the use of machine-learning techniques.

*Kim G. Larsen*

Personally, I think that the main challenge for the industrial application of formal methods at large is that 1) it generally requires highly skilled formal methods champions, 2) it takes too long to produce and analyse formal models and 3) the formal methods tools are much less user friendly than conventional programming language tools. In my own research, I also target the 2nd and 3rd of these, trying to combine such formal methods models based on discrete mathematics with models from different kinds of mathematics, for example representing physical elements in cyber-physical systems (for example in a digital twin context). I see many future possibilities for using formal approaches in this context in the future.

*Peter Gorm Larsen*

The main challenge is to ensure that systems, whether legacy, current or under design, meet their requirements by modelling the reasoning explaining why they are/were designed that way. This approach is universal, allows to capitalize knowledge and to improve (safety/security) when reusing/modifying/improving, by keeping track of the design decisions. This is particularly important for the (critical) infrastructures that are expected to survive decades — especially when their designers left or retired. Formal proofs of correct interoperability and correct design have been performed over the last 5 years on real railway systems under exploitation, with quite a number of safety-related findings. Other domains would benefit from this approach.

*Thierry Lecomte*

The key challenge is understanding the role of formal methods in engineering. It is not, as is often stated, to prove a system correct. Formal proofs are statements about relationships between models, not statements about some physical-world realization of a system. And, as George Box famously said, “all models are wrong, but some are useful”. What makes a model useful? To a scientist, a model is useful if it reasonably accurately describes some physical-world system. To an engineer, however, a model is useful when a useful physical-world system can be constructed that reasonably accurately behaves like the model. From the engineering perspective, all physical-world systems are wrong, but some are useful. Formal methods shine when they make statements about models that are accurately emulated by useful physical-world systems.

*Edward A. Lee*

Formal methods have reached a milestone in the last 15 years: the formal verification of functional correctness (not just safety) for the actual source code (not just models) of general-purpose, reusable systems software, such as the seL4 microkernel, the FSCQ and BilbyFS file systems, the miTLS secure communication library, and the CompCert and CakeML compilers. Much future work remains to verify infrastructure software. One challenge is to convince industry to pick up the effort, which goes beyond what academics can do. Another challenge is to better integrate specification and proof with programming, preferably at the level of programming languages and tools. Finally, it may be time to re-think priorities in formal methods research, with less emphasis on abstraction and automation and more emphasis on compositionality and reuse of verifications.

*Xavier Leroy*

Formal methods are the key to building dependable systems, but formal methods are mainly used in domains like cryptocurrencies, avionics or railways. Why are formal methods not used more often? In my opinion, their use is often not cost-effective. Potential software failures can typically be mitigated by simple updates without spending the extra effort for using formal methods. But when financial loss is huge or when faulty software threatens life, the situation is different. Does this mean formal methods cannot be applied in the majority of software system projects? In my opinion, no. But we have to make their application more cost-effective, i.e. cheaper. Formal methods research has mostly concentrated on developing new methods. Now that we have a variety of powerful methods and tools, research should rather focus on methodology, on applicability in the software engineering process, and on education of software engineers.

*Martin Leucker*

Formal models can help understanding critical systems and mastering their complexity. One challenge is helping humans understand the formal models better, e.g., by visualizations, interactions or automated extraction of knowledge. This is of particular importance for domain experts, who may not be familiar with the particular formal notations and concepts being used. If successful, formal models can play the role of interactive requirement or specification documents. Another challenge is to put formal models into the loop at runtime. This allows to use formal models embedded in a real-life system, either as a demonstrator or for test and certification purposes. This could also pave the way for more intelligent systems making predictions based on the formal models.

*Michael Leuschel*

When I was a graduate student I spotted a deadlock. The program was taken from a book ("books never lie!", I thought) and the deadlock was hard to reproduce. I spent days figuring out how all those threads could get stuck, up to the point I was able to convince my course mates and my teacher that the book was wrong. A few months later I met the SPIN model checker; I was able to find the deadlock in milliseconds! I was so excited that I decided to use SPIN for my Ph.D. and shortly after that I was having fun spotting bugs with AI algorithms (and earning a Ph.D. for it). Now that we talk about programming everything (including life!), we need new generations of students excited about novel ways to apply and extend formal methods. Life can't get stuck!

*Alberto Lluch Lafuente*

Formal methods, to me, is a very broad term: any technique that provides a logical and computational lens to the study of systems falls within its purview. Thus, it is difficult to summarize all the challenges in the field. I shall thus confine myself to one challenge: a better understanding of temporal behavior of continuous-state, continuous-time dynamical systems. Dynamical systems arise naturally when we study computational systems interacting with the physical world. My current research focuses on the application of formal methods principles, such as abstraction and composition, and tools, such as logics and automata, to the analysis of dynamical systems. I believe these principles and tools will be crucial to building high-confidence, autonomous, cyber-physical systems.

*Rupak Majumdar*

Beating the same path, trying to push complicated methods and manual approaches, is not likely to lead to fundamentally different results. The significant paradigm change that uses formal methods to power a transformational change is to start using models and DSLs instead of programs; properties and property analysers (even general purpose model checkers) instead of testing; and code generation (through certified or verified compilers) to generate correct by construction and optimised code that nobody needs to "maintain" anymore. If something changes, the action is on the models, the properties or the generation toolchain, and a new generation and deployment occur. See tools like CINCO, DIME and earlier jABC. This makes the effect of programming available to the masses, bypassing legions of hand-coders and eliminating costly code maintenance.

*Tiziana Margaria*

I believe that formal methods will become an essential piece in the design process of industrial systems. But, for this to happen, it is crucial to increase the scalability of analyses by designing better algorithms and tools, and to devise user-friendly formalisms, which are easier to learn and use. My current efforts towards these goals are focused on the extension of MCL, a temporal specification language for concurrent, value-passing, and probabilistic systems, together with the design of model-checking algorithms for MCL, which are made available as part of the CADP toolbox and used in many industrial applications. I also believe that greater efforts are needed to instill a "formal methods culture" to university students, especially in courses on software engineering, distributed systems, and hardware design.

*Radu Mateescu*

Formal methods have a long record of success in terms of algorithms, academic papers and tools, and relevant applications. This research community also contributed to other areas where concurrency and real-time are critical aspects. However, such level of maturity in research in the last 40 years did not impact as expected mainstream tools and practices to develop software. The use of rigorous methods for design and testing software is still rare. In my opinion, expanding the use of formal methods and tools to capitalize on the huge knowledge is not a role for the research community. Expanding software skills is for universities updating their bachelor degrees, and for companies creating the software development kits. Teaching formal methods only at master's and doctoral level for years gives the wrong impression that only few selected people will be ready to use them and they will never become general software engineering tools. Are we are still on time?

*Pedro Merino*

Handling formally the domain knowledge in design models is a challenge in system and software engineering. Domain knowledge is mainly related to the domain expert and the system under construction is in fact manipulating concepts that are valid according to knowledge. When developing justifications in the proof process, one can request knowledge that is known only by the expert. A formal and effective link should be defined between formal methods and knowledge-based techniques. Currently we are considering the definition of reusable mathematical theories for HMI or CPS. Moreover, we develop specific lectures for training master students using effective formal techniques together with case studies borrowed from our past and current scientific projects, while focusing on domain engineering.

*Dominique Méry*

In my opinion, research in formal methods focuses too much on methods and notations. I believe that the main benefit comes from clearly documenting algorithms and designs at a suitable level of abstraction, above the code level. The skills of mathematical thinking and rigorous reasoning are important: notation and support tools should help express and analyze / verify precise specifications in a way that corresponds to the problem rather than forcing users to shoehorn their thinking into a narrow formalism. An important challenge for researchers in formal methods is to provide useful and highly automated feedback for expressive formalisms.

*Stephan Merz*

Future developments in formal methods should broaden its impact. Rather than focusing mostly on verification, they should support system design, validation, evolution and maintenance from the earliest stages. For this, use of executable formal specifications for fast system modeling and analysis before implementation are crucial. In my own experience, formal executable languages like Maude have shown how this can be done for designing new web browsers, new cloud storage systems, correct-by-construction distributed real-time systems, or for fully specifying languages like Java or C. This supports what I call the "system specification" part, which is already scalable. The "property verification" part supports formal verification of properties specified in the logics of theorem provers and model checkers. Increasing verification's scalability is a key challenge ahead.

*José Meseguer*

Today formal methods consist largely of algorithms and tools for the automated analysis of system and software models. Many such tools are incorporated in software development environments for "behind the scenes" analysis and are routinely used with great success. Unfortunately, these applications are often given new names, probably to avoid the perception that formal methods are difficult to use. Formal verification tools have recently been used to verify important safety properties of models used to generate code for critical systems. While failure of these systems can expose a company to crippling liabilities, formal verification is still not seen as a valuable complement to testing. Since bottom-up technology transfer is not working, emphasis needs to be put on convincing senior management to make the use of formal verification a priority.

*Steve Miller*

As the co-founder of the startup TrustInSoft, I am developing formal-methods-based tools for the software industry. These tools perform advanced static source-code analysis, with comprehensive mathematical analyses that formally guarantee the absence of complete families of software flaws. Thus, our users know exactly in which conditions their software can be trusted. The main breakthrough of our approach is: "be modest with formal methods". We train our users to gain trust incrementally, on limited parts of their software first, but with a very precise plan to reach, depending on their time constraints, the largest possible perimeter of trust. This is the only path to success in commercializing formal methods: adapt the tools to non-specialists and fit within the time constraints of the industry.

*Benjamin Monate*

My research has focused on automated tools that do not require formal method expertise from end users, though designing and implementing these tools requires that expertise. I worked on the Astrée static analyzer: though it helps if the end user understands invariants, no advanced abstract interpretation expertise is needed. I am now working on improving the CompCert formally certified compiler with optimizations. Certified high-level synthesis tools are promising: the user specifies the design in a suitable high-level language, and target code is generated by formally proved compilation or optimization phases. This could be much safer and less human-intensive than the manual approaches still commonly used. One could for instance wish to synthesize concurrent communication structures automatically.

*David Monniaux*

Let's identify an important subset of formal methods... and stop calling it formal methods. Teach it early, between "programming introduction" (for all) and "harder-core" formal methods (for eventual specialists only). Start it with assertion-labelled flowcharts (Floyd style); call assertions "comments"; and name it "programming continuing". Teaching that would improve significantly the quality of the IT industry at its intermediate levels, where most programmers work and which affects our everyday lives so much. I teach such "(in)-formal methods" at roughly second-year level (6 times now); and the course has been picked up by other institutions. Lessons from where that has worked, and the effect it has had, and how it could be improved: they are concrete things that could be passed on.

*Carroll Morgan*

We have witnessed some very successful applications of formal methods in a number of different fields and industrial sectors; these include applications in medical and healthcare devices, and railway and automotive industry. We must recognise and celebrate these success stories, but also draw some conclusions from our failures. In my opinion, some of our failures were due to our obsession with developing sophisticated linguistic constructs, rather than focussing on usability and scalability of verification techniques for those aspects that the domain experts deem most relevant. To replicate and amplify the success stories of formal methods, we need industrial-strength tools and integration of various verification techniques with a focus on usability and scalability. We should be ready to embrace scalable and possibly non-exhaustive formal verification methods to deal with the heterogeneous complex systems of the future.

*Mohammad Reza Mousavi*

A growing number of companies is looking for formal methods to adopt in order to develop safe and secure software systems. Thankfully, many automated reasoning tools and analysis platforms are now available to all as open source projects. For program proof, the future is bright. Industrial tools for program proof should consider partial verification as the norm, allowing for different levels of assurance; strive to include executable specifications in the programming language as some form of contracts; distinguish specification-only and verification-only code, also known as ghost code; and consider manual proof as a programming activity in the auto-active style of manual proof. Rustan Leino once said: "Program verification is unusable. But perhaps not useless". Program verification will remain hard. But definitely useful.

*Yannick Moy*

It is disappointing that computer science and cyber security have taken over the language of biology — in particular the word "virus" — to indicate malign agents. In biology researchers are struggling to understand how building blocks operate and to deal with the complexities of scale. In computer science and (largely) cyber security we do understand how building blocks operate and are at most left with the complexities of scale. The use of terms from biology to discuss the vulnerabilities of IT systems risks putting expectations too low — to the extent that managers and policy makers remain unaware of what formal methods (full blown or "light weight") might achieve. If the language of biology cannot be avoided, formal methods should perhaps be explained as a "vaccine" deployed during software development.

*Flemming Nielson*

I see the next challenge for formal methods as the modelling and verification of autonomous systems. The challenges include dealing with the autonomy, learning and adaptation present in such systems, as well the fact that they do not operate in isolation and often in unknown environments with human interaction. My current work is focused on formal models for stochastic games. Such games combine nondeterminism, representing the adversarial, cooperative and competitive choices, stochasticity, modelling uncertainty, and concurrency, representing simultaneous execution of interacting agents. This research is still in its early stages, but a critical direction to carry forward is model partial observability.

*Gethin Norman*

We need to build formal methods tools that can scale and are accessible to non-experts. Among such tools are model checkers and theorem provers. Almost all model checkers available do not scale very well. Theorem provers have several tasks left for humans, such as lemma conjecture. It is hard for non-experts to construct proof scripts for theorem provers. We have been working on some techniques that may make model checking scale better and building a tool that supports the technique. We also have been working on a flexible way to construct proofs and building a tool that produces proof scripts from such proofs and scales well. I would like to pass on my experiences of case studies on formal methods and my knowledge accumulated through the experiences to younger generations.

*Kazuhiro Ogata*

No mature technology has ever dispensed with a formal basis in its evolution. The founding fathers of software engineering very early emphasized the need for software to be based on theoretical foundations playing a role similar to those of other established branches of engineering (Garmisch NATO conference, 1968). Fifty years are, at best, 'childhood' in the life of any technological field. Wait and see what comes with 'adulthood'. With software taking over all fields of (what used to be) human activity, the challenges are enormous. Insecurity, risk of malfunction/failure in increasingly complex systems will reach unprecedented levels, opening up a great opportunity for formal methods.

*José Oliveira*

Time is ripe for formal methods in mainstream software development, since: the "winner-takes-all" nature of the software industry justifies up-front investment into system quality; industry is realizing that standard validation techniques are insufficient; and industrial success stories on using formal methods are emerging (e.g., at Amazon). Achieving this goal requires modeling languages and analysis methods that scale to today's systems, and developers who appreciate formal methods. I try to contribute to this goal by: (i) developing simple and intuitive modeling languages for complex real-time and cyber-physical systems (using Maude and AADL); (ii) developing complexity-reducing formal patterns where verifying a CPS is reduced to simpler problems; and (iii) writing an introductory formal methods textbook and teaching a second-year formal methods course with 50 students.

*Peter Csaba Ölveczky*

Application of formal methods to learning-enabled systems, i.e. systems that use machine learning (neural networks) is a big challenge due to the approximate nature of machine-learning algorithms. It is difficult to write formal specifications for such systems. Perhaps probabilistic properties can be written but there is very little work in that direction. I am personally working on property inference and also probabilistic analysis of neural networks, that will hopefully address some of these challenges.

*Corina Pasareanu*

We need to focus on performing simple verification tasks: integrate (even hide!) backend light formal methods in graphical tools (in Bell Labs, we had such a success with analyzing message sequence charts), monitor executions to perform run-time verification, allow light-weight automatic verification tasks that can be run by programmers or engineers during system development; such tasks should be performed automatically in the background, or as a simple extension to hardware or software development tools. I am currently concentrating on run-time verification, developing algorithms and tools that are immediately applicable for system development. I also worked on making model checking more efficient (e.g. partial order reduction and LTL translation) and integrated it in automatic genetic synthesis of code from specification.

*Doron Peled*

Formal methods have enjoyed many areas with successes. But it is still difficult to use them to get a real system safe. Admittedly, getting real systems working, let alone safe, is exceedingly difficult and time-consuming, no matter what approach is used. Formal methods have an opportunity to help with this challenge, however, but only if they broaden their scope to cover full systems.
Resting on the logical foundations of cyber-physical systems, my research is pursuing this question in multiple complementary ways: 1) Forming logical links between models and reality with ModelPlex; 2) Verified compilation to executables that inherit safety theorems from verified models; 3) Formally supported development processes for incremental development that benefit from formal results about prior designs.

*André Platzer*

The challenge of FMs as a science is to contribute to a fundamental question: How can one use computers not only to solve a problem (say, autonomous driving) but also to build the solution to the problem. The problem may be complex, but building the solution is an even more complex problem. A science is a language, and the challenge lies in finding the language that allows us to translate aspects of practical issues into concrete research questions. Until now, the lingua franca in FMs has been logic; what is the language for the kind of systems we will have to deal with in the future? FMs is an attractive research area since it offers many variations of self-reflection, a theme dear to computer scientists from the very beginning. A typical example is the question whether the requirements on a system are correct. What are the requirements for system requirements, and how can we check them? What are the requirements on a system, say, for autonomous driving?

*Andreas Podelski*

Rather than coming up with more powerful formalisms and better tools, one of the biggest challenges — and opportunities — in formal methods is still to get very basic formal notions into the minds of our students, so that it becomes natural for them to tackle problems thinking in concepts such as finite state machines and grammars, or in terms of object invariants and types when coding. The bulk of all security problems is due to hand-written parsers of overly complex and poorly specified input formats and protocols, in long prose documents with odd, informal diagrams. This is downright embarrassing, given that formal languages and parser generation are some of the oldest formal methods around. This is something I hope to improve as part of the LangSec community.

*Erik Poll*

Computer systems have become pervasive in all walks of our lives. Formal methods have contributed to both foundational understanding and construction of tools for practical analysis and validation of computer systems. While we should continue to be critical of how we frame problems and introspect about the impact we are having, we should also be happy that formal methods researchers have won Turing awards, and every major SW and HW company has groups developing, building, and deploying tools based on formal methods. Looking to the future, systems are getting larger and more complex and diverse. We have systems driven by AI and ML in their core, cyber-physical systems, and autonomous systems, and will have quantum and biological computers. We should continue work in foundational understanding of such systems, develop tools and techniques that work at industrial scale, and constantly strive to close the gap between theory and practice.

*Sriram Rajamani*

The foremost challenge for formal methods remains to demonstrate their applicability in industry. The second challenge for formal methods is to educate students. Yet the standard computer science graduate leaves the university with either no knowledge of formal methods or, even worse, a hatred for them. The third challenge for formal methods is to find ways of combining them with ML and AI techniques. With companies, I work on applying formal methods to their challenges. With colleagues, I have written a book "Formal Methods for Software Engineering", I have organised the "1st International Workshop on Formal Methods — Fun for Everybody". I am collaborating with colleagues from the ML/AI community. I would very much hope all three of my contributions would be picked up ☺

*Markus Roggenbach*

Formal methods, despite the significant progress in recent years, are still not yet readily available to average engineers. I believe that the main challenges are: the creation of robust, open and usable infrastructures for research and industrial application; the design of a standard formalism to exchange benchmarks and models; and the creation of adequate and attractive teaching material. Finally, widen the application to emerging areas (e.g. trustworthy AI). My main contribution was taking an active part in the development of the NuSMV open source model checker. NuSMV and its derivatives have been integrated in several commercial and academic verification tools, and are used in other domains (AI planning) as a reasoning framework. Finally, it is widely used as a teaching tool in several formal methods courses at different levels.

*Marco Roveri*

The next frontier in formal methods is to make them *usable* and *practical*. Our main challenge is still the specification bottleneck: formal methods are highly dependent on specifications. We must know where we get specifications from, how we measure their quality, and how we best organize and maintain them. If we support non-experts to semi-automatically extract unambiguous, analyzable specifications, then formal methods are *usable*. Formal methods must also be *practical*: such as reasoning under constraints on time, memory, knowledge, and other resources. How do we create living, changing, hierarchical models and specifications that tie formal verification to the real system, at different levels of abstraction, throughout its lifecycle? How do we build formal verification into systems, and build formal verification tools so they become one with the systems they are meant to specify, validate, and verify?

*Kristin Yvonne Rozier*

Formal methods allow us to calculate properties of computational systems, just as computational fluid dynamics allow us to calculate the flow of air over a wing. The challenge is, and always has been, to automate this efficiently. With modern SMT solvers that can deal with quantified formulas, nonlinear arithmetic, and complex data types, we are almost there.
The next challenge is effective use of these capabilities and here I see two big opportunities. First is to embed them, invisibly, inside every tool for software, hardware, and system development with a view to improving their fault detection and, consequently, their productivity and the quality of the artifacts produced. Second is to find contributions to the predictability and safety of modern autonomous systems largely driven by machine learning and AI.

*John Rushby*

Formal methods are the only way to develop high-quality software and hardware. There are still difficult problems to be tackled: performance, scalability, usability, etc. The main challenge, in my opinion, is to make these methods become mainstream. To this aim, I have been working with companies (Naver, Nokia, and Orange) to show how such methods could be used to solve industrial problems. I have been developing tools supporting the development and verification of component-based software, business processes, and IoT applications, where formal methods are hidden within software development platforms. Such press-button approaches are a promising solution, which allows formal methods to be used by anyone without requiring a high level of expertise.

*Gwen Salaün*

The advancement of modern formal methods and their successful industrial applications have been a consequence of some key factors: automation, continuous education, the "hidden formal methods" approaches, and integration with other areas like natural language processing, (semi-formal) graphical notations and system testing. There are several challenges for an even more significant industrial insertion; I single out scalability as a major concern. My research focus has been on compositional analysis in the context of model checking, and industrial applications of formal strategies for test case generation from natural-language requirements, in a partnership with Motorola/Lenovo and Embraer. More recently, I have been exploring formal modelling, simulation and analysis of autonomous systems.

*Augusto Sampaio*

The challenge for formal methods is to be integrated in system design flows to enhance their rigorousness. The objective is to break with the promise of "absolute correctness" and focus on understanding and accountability. Design flows should be model-based, to allow semantic coherency achieved by translation into a single host language. Additionally, they should be component-based, meaning that they rely on a common and general component model and theory for building systems bottom-up from components. The third requirement is correctness by construction, achieved by property-preserving source-to-source transformations and extensive use of architecture patterns. I have played a leading role in the BIP project of Verimag. My efforts focus on the design of dynamic reconfigurable systems, autonomous systems in particular.

*Joseph Sifakis*

Some of the future challenges for formal methods include verification of algorithms and systems developed for quantum computers which is a long term challenge, verification of systems developed using artificial intelligence such as autonomous vehicles and verification of applications in security and privacy. I am currently working on — (i) automated methods for verifying privacy and accuracy properties of differential privacy mechanisms, (ii) verifying properties of autonomous vehicles controlled by AI techniques. For verifying differential privacy mechanisms the challenges include wider applicability, speed of verification as well as handling inputs of all sizes. For autonomous vehicles, we are exploring run-time verification of safety properties. Here the challenges include handling deep neural networks and modeling the environment.

*A. Prasad Sistla*

We have to move from thinking in terms of individual methods and tools to thinking in terms of adequacy for solution: which methods fit where and under which paradigm? CINCO, our meta-tooling suite, has morphed into a DSL-driven correctness-by-construction environment in this way, where language design has become a prime means for guaranteeing system correctness. The corresponding rich meta models require strong formal methods-based support for static semantics checking. Required analysis tools are in turn built automatically within our environment, establishing a bootstrapping-based continuous improvement cycle. Our experience suggests that this way of DSL-driven development may well become a popular new style of system development.

*Bernhard Steffen*

I'm quite optimistic about formal methods: type checking is now standard, model checking is heavily used in hardware verification, model-based testing is daily practice, and Simulink, UML and SysML are rooted in formal methods. To achieve ambitious goals, we should get really serious about software tools: there are far too many tools. Rather than everyone working on her own research prototype, we should, as a community, work on joint tools. On tools that have impact. On tools that come with excellent GUIs and visualization features. On tools that have decent user manuals, training sessions and even customer support. We should not wait for a start-up company to commercialize one of the research prototypes. No, if we really want formal methods to have impact, we must change the way we handle tool development. This requires communication, coordination and courage, but that is what science needs.

*Marielle Stoelinga*

I believe formal methods are most useful as a mindset, i.e., a systematic, rigorous way of approaching and solving problems. Being trained in formal methods, I often find it easier to see what a problem really is, how to generalize it and how to approach it. My students on the contrary can't seem to differentiate the problem/concept from the implementation. The challenge is thus to install the mindset of formal methods early in the mind of the students. Beyond working as a mindset, formal methods tools and techniques apply only if the return (in terms of improved safety and security) outweighs the cost of applying formal methods. It thus implies that all we have to do is to drive down the cost and demonstrate the return on real-world systems.

*Jun Sun*

Based on my decades-long experience, I think that one of the most serious impediments to the adoption of formal methods in industry is a generalized lack of education and training in formal specification and reasoning, especially in the US. Few companies employ computer scientists and engineers who have a working knowledge of logic beyond propositional logic. This usually makes it hard to even convey what a formal methods tool can offer, let alone how to use it. I am convinced that to go from its current technical successes to a wide adoption of formal methods, the formal methods community needs a concerted and sustained effort aimed at making the teaching of logic and formal specification an integral part of computer science and engineering curricula. Until then, we are condemned to developing wondrous tools that the majority of our intended user base will not be able to use.

*Cesare Tinelli*

During my last visit to Oded Maler, in Oct. 2017, he was even more philosophical than usual. Maybe he knew he would not live much longer (he passed away on Sept. 3, 2018). To create industrial impact, Oded argued, we need to come up with *simple ideas*. Most of the research in the formal methods community is just way too complex. He mentioned his work on verification and synthesis of timed/hybrid systems. Mathematically appealing, but with limited practical impact due to the decision algorithms' complexity. In contrast, Oded argued, his results on signal temporal logic are mathematically trivial, but the industrial impact is the highest from all his work. Of course, it is not always clear what simple means: while the functionality of SAT solvers is simple, sophisticated algorithms are used underneath. Still, there is much wisdom in Oded's words, and I use them to guide my research.

*Frits Vaandrager*

Systems should do the right things and do them right. Much of the work goes into finding out and understanding what are the right things. It requires understanding user needs and their consequences much better than how users express them. Computers obey instructions precisely, no matter how stupid the outcome is from the human point of view. To prevent stupid outcomes, software professionals must understand both the computers' and the end users' worlds, and build a bridge between them. Formal methods can be a good tool here. However, good analysis and reasoning skills in the informal side are a must. Unfortunately, teaching thinking is difficult. Software education seems to more and more focus on blind use of ready-made components and prescribed methods. I find it worrying.

*Antti Valmari*

Formal methods offer a promising path towards building reliable systems; there exist numerous examples that highlight the benefits. From my own personal experience at ESA (European Space Agency), the main obstacle to the widespread adoption of formal methods is that they tend to address limited aspects of the design space, using notations that require expert knowledge, which imposes significant upfront investment. Industrial users often perceive the proliferation of notations as a risk. The integration of formal and informal techniques, and their embedding in process standards, is crucial for successful industrial adoption; it would help if scientists spent time in an engineering setting to understand these challenges. To capitalise on the positive impact that formal methods have had in niche applications to date, industry requires long-term support for professional formal tools, with stability and performance guarantees.

*Marcel Verhoef*

In my opinion, formal methods will become essential to successfully handle the ever increasing complexity of software systems, their design, construction and maintenance, at a much larger scale than they are now. However, it is not only academia that will make the spread of application of formal methods happen. Also industry will foster the uptake of formal methods as soon as this will be economically beneficial and I expect that it will. Therefore, academia should focus on its own agenda, of course keeping in touch with developments elsewhere. Top priorities are the development of a theory of formal methods, encompassing and integrating the myriad of approaches, and the building of a community or a society of researchers and practitioners that goes beyond scattered conferences and journal publications.

*Erik de Vink*

I see two primary challenges for formal methods over the next years. 1) Be able to document requirements specifications formally so that all stakeholders can be involved and understand the specification, and also then use that to improve validation of requirements. 2) Use formal methods to produce effective and fully integrated model-based engineering tool chains, that help us build complex systems and also assure properties of those systems with a high enough degree of confidence. Currently assurance and development are not integrated well enough. They will have to be in order for us to produce safe, secure and dependable complex systems. Our group is working on modeling this integration.

*Alan Wassyng*

Among all the challenges for formal methods, the analysis for worst-case execution time (WCET) is a very difficult one, which must be performed on executable code, as instruction semantics, memory allocation, machine-register use, and compiler optimizations heavily influence the execution times. This analysis must search for the longest path in an enormous space of combined program paths and architectural paths. Thus, sound WCET analysis is only feasible through appropriate abstraction of the execution platform. We used Abstract Interpretation to obtain a reliable, precise, and scalable analysis method, which was implemented by the spin-off company AbsInt and instantiated for many architectures. The resulting tools are the only ones widely used in industry, and are validated by EASA (European Aviation Safety Agency) for time-critical tasks in several Airbus plane generations [21].

*Reinhard Wilhelm*

Scalability and usability of model checking and model-based testing remain major challenges in formal methods, and, in particular, in their adoption in practice. While breakthroughs in the past (e.g. symbolic methods) have paved the way to analyse systems of immense complexity, there is a huge gap between our academic languages and solutions, and the languages used in industry, offering fancy data types, language constructs, etc. Bridging this gap is among the most important challenges in formal methods research. Through case studies, I identify weaknesses and strengths in our academic solutions, often in the context of our mCRL2 tool set, and also expose situations when this gap is minimal. I believe research into languages, (fixpoint) logics and game theory are needed to narrow this gap.

*Tim Willemse*

Trustworthy AI: We are seeing an astounding growth in deployment of AI systems in critical domains such as autonomous vehicles, criminal justice, healthcare, and public safety, where decisions taken by AI agents directly impact human lives. Can these decisions be trusted to be correct, reliable, fair, and safe, especially under adversarial attacks? Just as for trustworthy computing, formal methods could be an effective approach for building trust in AI-based systems. However, we need to extend the set of trust properties to include fairness, robustness, probabilistic accuracy under uncertainty, and other properties yet to be identified. Further, we need new specification and verification techniques to handle new kinds of artifacts, e.g., data distributions and machine-learning models.

*Jeannette M. Wing*

Formal methods are a cornerstone of computer science. They form the scientific basis for the design, validation and verification of software systems. Formal methods have been, are and will continue to be the starting point for many successful companies. Artificial intelligence and novel computing paradigms such as quantum and molecular computing are the next challenges for formal methods. How can we make AI systems safe and secure? How can we build reliable mixed systems that combine quantum and molecular components with conventional hardware and software?
Currently, I am working on safe machine-learning algorithms and how to integrate them into a rigorous development process for collective adaptive systems. But there are many other pressing research questions for formal methods. May formal methods live and prosper!

*Martin Wirsing*

Today, the functionality as well as economical value of most industrial systems and products, such as cars, airplanes, and medical devices, is defined and realized by software as embedded system. The ability to deploy software updates dynamically is critical for security, new features, and customization of next-generation embedded systems. But such deployments are not possible today because we lack the techniques to guarantee that the updated system remains safe. In 2019, I received an ERC advanced grant for the CUSTOMER project (Customizable Embedded Real-Time Systems). The mission of CUSTOMER is to develop a new paradigm supported by powerful model-based tools for building embedded systems which can be updated on demand dynamically, safely, and securely over their operational life-time.

*Wang Yi*

I am working on a new model of the domain of networking, called "compositional network architecture," with networking expert Jennifer Rexford at Princeton University [24]. We believe it will have a major long-term impact on education, practice, and especially verification in the field of networking. For education, we are writing a textbook based on the terminology, patterns, and principles derived from our model. We are also engaged in a research project to embody the model in an implementation, so we can explore model-driven design, development, and verification of networks. The biggest benefit of our model for network practitioners is that it explains layering in a completely new way that is realistic, precise, and offers strong modularity to exploit for reuse and verification.

*Pamela Zave*

We live in a revolutionized digital era: smartphones, self-driving vehicles and online education turn the planet into a global village. Guaranteeing correctness of such products is a cornerstone for modern society. Formal methods is arguably the most convincing methodology to achieve both performance and dependability; interdisciplinary in nature, it integrates established disciplines like control theory and language processing and promising new directions like machine learning and quantum computing. I am contributing to developing tools for learning models and verifying probabilistic systems, and I am excited to witness the birth of new fundamental theories and advanced tools. Formal methods still have a long way to be standardized in industry, familiarized in universities and popularized in society: this will happen and needs the efforts of us all. Long live formal methods, congratulations to FMICS 25th anniversary and looking forward to the next 25 years!

*Lijun Zhang*

**Acknowledgements**

We heartily thank all our colleagues who participated in the survey. This includes all those whose position statement appears in Section 9. Thanks also are due to those experts who took the time to answer our questionnaire, but did not provide a position statement, namely Bernhard Aichernig, Roderick Bloem, Arne Borälv, Rocco De Nicola, Cindy Eisner, Dimitra Giannakopoulou, Georges Gonthier, Susanne Graf, Aarti Gupta, Thomas Henzinger, Holger Hermanns, Michael Hinchey, César Muñoz, Tobias Nipkow, Joël Ouaknine, Charles Pecheur, Alastair Reid, Ina Schieferdecker, and Jim Woodcock. Finally, we are grateful to Nicolas Amat, Pierre Bouvier, Alessio Ferrari, Arnd Hartmanns, Ajay Krishna, Rom Langerak, Lina Marsso, Franco Mazzanti, and Wendelin Serwe, who tested four successive beta-versions of our questionnaire and provided us with many wise observations; Pierre Bouvier, Alessio Ferrari, Dejan Ničković, and Wendelin Serwe also proof-checked the author version of the present report.

## References

1. AbsInt: Astrée software (2020), http://www.absint.com/astree
2. Bartocci, E., Beyer, D., Black, P.E., Fedyukovich, G., Garavel, H., Hartmanns, A., Huisman, M., Kordon, F., Nagele, J., Sighireanu, M., Steffen, B., Suda, M., Sutcliffe, G., Weber, T., Yamada, A.: TOOLympics 2019: An overview of competitions in formal methods. In: Beyer, D., Huisman, M., Kordon, F., Steffen, B. (eds.) Proceedings of the 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'19), Prague, Czech Republic. Lecture Notes in Computer Science, vol. 11429, pp. 3–24. Springer (Apr 2019). https://doi.org/10.1007/978-3-030-17502-3_1
3. Basile, D., ter Beek, M.H., Fantechi, A., Gnesi, S., Mazzanti, F., Piattino, A., Trentini, D., Ferrari, A.: On the industrial uptake of formal methods in the railway domain: A survey with stakeholders. In: Furia, C.A., Winter, K. (eds.) Proceedings of the 14th International Conference on Integrated Formal Methods (iFM'18), Maynooth, Ireland. Lecture Notes in Computer Science, vol. 11023, pp. 20–29. Springer (Sep 2018). https://doi.org/10.1007/978-3-319-98938-9_2
4. ter Beek, M.H., Borälv, A., Fantechi, A., Ferrari, A., Gnesi, S., Löfving, C., Mazzanti, F.: Adopting formal methods in an industrial setting: The railways case. In: ter Beek, M.H., McIver, A., Oliveira, J.N. (eds.) Proceedings of the 3rd World Congress on Formal Methods (FM'19), Porto, Portugal. Lecture Notes in Computer Science, vol. 11800, pp. 762–772. Springer (Oct 2019). https://doi.org/10.1007/978-3-030-30942-8_46
5. Bjørner, D., Havelund, K.: 40 years of formal methods – some obstacles and some possibilities? In: Jones, C., Pihlajasaari, P., Sun, J. (eds.) Proceedings of the 19th International Symposium on Formal Methods (FM'14), Singapore. Lecture Notes in Computer Science, vol. 8442, pp. 42–61. Springer (May 2014). https://doi.org/10.1007/978-3-319-06410-9_4
6. Bowen, J.P., Hinchey, M.G.: Ten commandments of formal methods. IEEE Comput. **28**(4), 56–63 (Apr 1995). https://doi.org/10.1109/2.375178
7. Cerone, A., Roggenbach, M., Davenport, J., Denner, C., Farrell, M., Haveraaen, M., Moller, F., Körner, P., Krings, S., Ölveczky, P., Schlingloff, B.H., Shilov, N., Zhumagambetov, R.: Rooting formal methods within higher education curricula for computer science and software engineering: A white paper. In: Cerone, A., Roggenbach, M. (eds.) FMFun 2019, Revised Selected Papers. Communications in Computer and Information Science, Springer (2020), to appear

8. Clarke, E.M., Wing, J.M.: Formal methods: State of the art and future directions. ACM Comput. Surv. **28**(4), 626–643 (1996). https://doi.org/10.1145/242223.242257
9. Davis, J.A., Clark, M.A., Cofer, D.D., Fifarek, A., Hinchman, J., Hoffman, J.A., Hulbert, B.W., Miller, S.P., Wagner, L.G.: Study on the barriers to the industrial adoption of formal methods. In: Pecheur, C., Dierkes, M. (eds.) Proceedings of the 18th International Workshop on Formal Methods for Industrial Critical Systems (FMICS'13), Madrid, Spain. Lecture Notes in Computer Science, vol. 8187, pp. 63–77. Springer (Sep 2013). https://doi.org/10.1007/978-3-642-41010-9_5
10. Ferrari, A., Mazzanti, F., Basile, D., ter Beek, M.H., Fantechi, A.: Comparing formal tools for system design: a judgment study. In: Proceedings of the 42nd International Conference on Software Engineering (ICSE'20). pp. 62–74. ACM (2020). https://doi.org/10.1145/3377811.3380373
11. Garavel, H., Graf, S.: Formal methods for safe and secure computer systems. BSI Study 875, Bundesamt für Sicherheit in der Informationstechnik, Bonn, Germany (Dec 2013), https://www.bsi.bund.de/DE/Publikationen/Studien/Formal_Methods_Study_875/study_875.html
12. Garavel, H., Mateescu, R.: Reflections on Bernhard Steffen's physics of software tools. In: Margaria, T., Graf, S., Larsen, K.G. (eds.) Models, Mindsets, Meta: The What, the How, and the Why Not? Lecture Notes in Computer Science, vol. 11200, pp. 186–207. Springer (Jun 2019). https://doi.org/10.1007/978-3-030-22348-9_12
13. Gates, B.: Trustworthy computing (Jan 2002), https://www.wired.com/2002/01/bill-gates-trustworthy-computing, e-mail memo to Microsoft employees
14. Gnesi, S., Margaria, T. (eds.): Formal methods for industrial critical systems: A survey of applications. Wiley (2013). https://doi.org/10.1002/9781118459898
15. Hall, A.: Seven myths of formal methods. IEEE Softw. **7**(5), 11–19 (Sep 1990). https://doi.org/10.1109/52.57887
16. Huisman, M., Gurov, D., Malkis, A.: Formal methods: From academia to industrial practice. A travel guide. CoRR **abs/2002.07279** (2020), https://arxiv.org/abs/2002.07279
17. Jones, C.B., Thomas, M.: The development and deployment of formal methods in the UK. CoRR **abs/2006.06327** (2020), https://arxiv.org/abs/2006.06327, submitted to Annals of the History of Computing
18. Miller, S.P.: Lessons from twenty years of industrial formal methods. In: Proceedings of the 20th High Confidence Software and Systems Conference (HCSS'12) (2012), http://cps-vo.org/node/3434
19. Rushby, J.: Formal methods and the certification of critical systems. Tech. Rep. SRI-CSL-93-7, Computer Science Laboratory, SRI International, Menlo Park, CA (Dec 1993), http://www.csl.sri.com/papers/csl-93-7/, also issued under the title "Formal Methods and Digital Systems Validation for Airborne Systems" as NASA Contractor Report 4551, Dec 1993
20. Steffen, B.: The physics of software tools: SWOT analysis and vision. International Journal on Software Tools for Technology Transfer **19**(1), 1–7 (2017). https://doi.org/10.1007/s10009-016-0446-x
21. Wilhelm, R.: Real time spent on real time. Commun. ACM (Oct 2020), to appear
22. Wing, J.M.: A specifier's introduction to formal methods. IEEE Computer **23**(9), 8–22 (Sep 1990). https://doi.org/10.1109/2.58215
23. Woodcock, J., Larsen, P.G., Bicarregui, J., Fitzgerald, J.: Formal methods: Practice and experience. ACM Computing Surveys **41**(4), 19:1–19:36 (2009). https://doi.org/10.1145/1592434.1592436
24. Zave, P., Rexford, J.: The compositional architecture of the Internet. Commun. ACM **62**(3), 78–87 (Feb 2019). https://doi.org/10.1145/3226588