New Opportunities for Integrated Formal Methods

Mario Gleirscher, Simon Foster, Jim Woodcock
University of York, United Kingdom
Email: firstname.lastname@york.ac.uk

***Abstract*—Formal methods have provided approaches for investigating software engineering fundamentals and also have high potential to improve current practices in dependability assurance. In this article, we summarise known strengths and weaknesses of formal methods. From the perspective of the assurance of robots and autonomous systems (RAS), we highlight new opportunities for integrated formal methods and identify threats to their adoption to be mitigated. Based on these opportunities and threats, we develop an agenda for fundamental and empirical research on integrated formal methods and for successful transfer of validated research to RAS assurance. Furthermore, we outline our expectations on useful outcomes of such an agenda.**

***Index Terms*—Formal methods, strengths, weaknesses, opportunities, threats, SWOT, challenges, integration, unification, research agenda.**

Nomenclature

(A)SIL	(Automotive) Safety Integrity Level
DA	Dependability Assurance
DSL	Domain-Specific Language
FI	Formal Inspection
FM	Formal Method
iFM	integrated Formal Method
MBD	Model-Based Development
MDE	Model-Driven Engineering
ML	Machine Learning
QA	Quality Assurance
RAS	Robots and Autonomous Systems
RE	Requirements Engineering
RCA	Root Cause Analysis
SACM	Structured Assurance Case Meta-Model
SMT	Satisfiability Modulo Theory
SWOT	Strengths, Weaknesses, Opportunities, Threats
UTP	Unifying Theories of Programming

I. Introduction

A plethora of difficulties in software practice and momentous software faults have been continuously delivering reasons to believe that a significantly more rigorous discipline of software engineering is needed. Researchers such as Neumann [1] have collected plenty of anecdotal evidence on software-related risks substantiating this belief.

In *dependable systems engineering*, researchers have turned this belief into one of their working hypotheses and contributed formalisms, techniques, and tools to increase the rigour in engineering workflows. Examples of activities where formalisms have been brought to bear include requirements engineering (e.g. [2]), architecture design, test-driven development, program synthesis, and testing, to name a few. The field of *formal methods* was born and has been an active research area for many decades. FMs have served as a powerful tool for *theoretical researchers* and as a paradigm to be adopted by *practitioners* and, hence, to be further investigated by *applied researchers*. Applications of these methods showed their strengths but also their weaknesses.

Recently, we can observe this plethora of difficulties with *robots and autonomous systems (RAS)*. Such systems are going to be more broadly deployed in society, therefore, increasing their level of safety criticality [3]. Hence, their regulatory acceptance requires assurance cases with comprehensible and indefeasible safety arguments. However, assurance cases associated with standards like IEC 61508 and DO-178C can be laborious to create, complicated to maintain and evolve, and must be rigorously checked by the evaluation process to ensure that all obligations are met and confidence in the arguments is achieved [4], [5].

In spite of the weaknesses of current FMs, and encouraged by their strengths, we believe that the integration of formal methods can reduce critical deficits observable in dependable systems engineering. Farrell et al. [6] state that "there is currently no general framework integrating formal methods for robotic systems." The authors highlight the use of what are called *integrated formal methods (iFM)*[1] in the construction of assurance cases and the production of evidence as a key opportunity to meet current RAS challenges. Particularly, machine-checked assurance cases can greatly increase confidence in the sufficiency of assurance cases, and also aid in their maintenance and evolution through modularisation of arguments and evidence. Integration of formal methods, in particular modern virtual prototyping and hybrid verification tools, can improve the automation of the evidence gathering process, and highlight potential problems when an assurance case changes.

A. Contribution

With this work, we investigate the *potentials for the wider adoption of iFMs in dependable systems engineering*, taking RAS as a recent opportunity for making progress with iFM research and for its successful transfer into practice.

We carry through an analysis of *strengths, weaknesses, opportunities, and threats* to the use of iFMs in the practical assurance of robots and autonomous systems. For this analysis, we surveyed literature on FM research transfer and application

[1] We reuse the term from the homonymous conference series [7].

The first author of this work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – GL 915/1-1/2.

Reference Format: Gleirscher, M., Foster, S., & Woodcock, J.. *New Opportunities for Integrated Formal Methods* (June 12, 2019). Unpublished working paper. Department of Computer Science, University of York, United Kingdom. arXiv:1812.10103 [cs.SE]

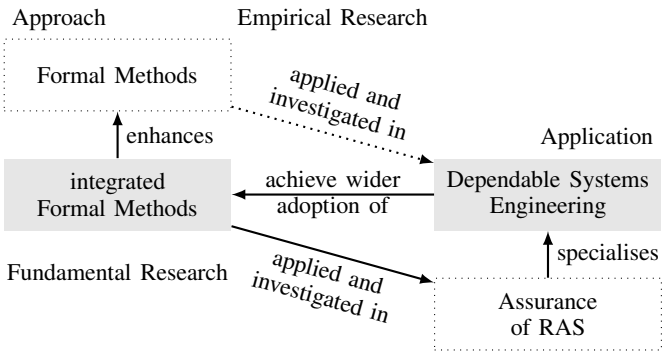

Fig. 1. RAS assurance as an opportunity for the wider adoption of iFMs in dependable systems practice

and on dependable systems engineering. Figure 1 depicts how we deduce a research agenda for iFMs with a focus on RAS assurance from a research agenda for FMs in dependable systems engineering.

From the *strengths*, we see in recent research, and from the *opportunities* in current RAS assurance, we argue why RAS assurance—an instance of assurance in dependable systems engineering (cf. Figure 1)—is a key opportunity for making substantial iFM research progress. Throughout this work, we indicate how iFMs can meet typical challenges in dependability assurance that also apply to RAS assurance.

From the *weaknesses*, we observe in recent research, and from the *threats* general FM research transfer is exposed to, we derive the directions of foundational and empirical research to be taken to transfer iFMs to RAS assurance and use them to their maximum benefit.

Our analysis 1) elaborates on the analysis and conclusions of Hoare et al. [8], 2) extends their suggestions with regard to FM experimentation and empirical evidence of effectiveness focusing on collaboration between FM research and practitioners, and 3) develops a specific research and research transfer roadmap within the application domain of robots and autonomous systems.

B. Overview

We provide some background including terminology (Section II-A) and related work (Section II-B) in the following. Then, we carry through an analysis of strengths, weaknesses, opportunities, and threats of iFMs for RAS assurance (Sections III and IV). Based on this analysis, we formulate our hypotheses (Section VI), pose research questions based on this hypotheses, derive a research agenda, and specify the outcomes we expect from this agenda (Section VII).

II. Background

A. Terminology

For sake of clarity among readers of different provenance, we restrict the meaning of some terms we use in the following and introduce convenient abbreviations.

We view *robots and autonomous systems* as both dependable systems and highly automated machines capable of achieving a variety of complex tasks in support of humans. We can consider such systems looking at four layers: the plant or process composed of the operational environment and the machine; the machine itself; the machine's controller, and the software embedded into this controller. We treat "embedded system" and "embedded software" as synonyms. Machine, controller, and software can be distributed.

By *dependable systems engineering*, we refer to error-avoidance and error-detection activities in control system and embedded software development (e.g. according to the V-model). Avizienis et al. [9] provide a comprehensive terminology and an overview of the assessment and handling of a variety of faults, errors, and failures. For critical systems, such activities are expected to be explicit (e.g. traceable, documented), to employ best practices (e.g. design patterns), and to be driven by reasonably qualified personnel (e.g. well-trained and experienced engineers or programmers).

In the applications we consider, the need for *dependability* arises from the embedding of software into a cyber-physical context (i.e., an electronic execution platform, a physical process to be controlled, other systems or human users to interact with). *Dependability assurance* (DA, or assurance for short) encompasses the usually cross-disciplinary task of providing evidence for an assurance case (e.g. safety, security, reliability) for a system in a specific operational context [10].

By *formal methods*, we refer to the use of formal (i.e., mathematically precise and unambiguous) modelling languages to describe system elements, such as software, hardware components, and the environment, and to subject these models to analysis, the results of which are targeted at DA [11], [12]. FMs always require the use of both *formal syntax* and *formal semantics* (i.e., the mapping of syntax into a mathematical structure). Semantics that allow the verification of refinement or conformance across different FMs are said to be *unifying* [13], [14]. *Integrated formal methods* (iFMs) allow the coordinated application of several potentially heterogeneous formal methods, supported by several interrelated layers of formal semantics [15], [16].

FMs stand in contrast to *informal methods*, which employ artefacts without a formal syntax or semantics, such as natural language descriptions and requirements. In the gap between informal methods and FMs there is also a variety of *semi-formal methods*, including languages like UML and SysML, whose syntax and semantics have frequently been subject of formalisation in research (e.g. [17], [18], [19], [20]).

FM-based tools represent software for the automation of modelling and reasoning along with the use of a FM. Through *model-based development* (MBD) and *model-driven engineering* (MDE), FM-based tools have been successfully applied in dependable systems projects [21], [22].[2]

We speak of *applied or practical FMs* to signify successful applications of FMs in a practical context, for example, to develop embedded control software deployed in a commercial product marketed by an industrial company. We consider the use of FMs in research projects still as *FM research*. *Empirical*

[2]The SCADE Design Verifier (http://www.esterel-technologies.com) and the seL4 microkernel (http://sel4.systems) represent good although less recent examples.

FM research investigates practical FMs, for example, using surveys, case studies, or controlled field experiments [23]. We speak of *FM transfer* if FM research is transferred into practice with the aim to effectively apply and practice FMs. FM transfer, as discussed below, is crucial for strong empirical FM research and progress of FM research in the long term.

B. Related Work

Many researchers have suggested that FMs will, in one way or another, play a key role in mastering the mentioned difficulties and in achieving the desired guarantees (e.g. dependability, security, performance) of future critical systems.

Expecting an increased use of FMs to solve practical challenges in the mid 1990s, Clarke and Wing [11] suggested FM integration, tool development, and continuous specialist training to foster successful FM transfer to practice.

In 2000, Van Lamsweerde [24] observes a growing number of FM success stories in requirements engineering. Evaluating several FM paradigms, he outlines weaknesses (e.g. isolation of languages, poor guidance) to be compensated and challenges to be met towards effective FM use, particularly, their integration into multi-paradigm specification languages.

In the mid 2000s, Hinchey et al. [25] spot a decline of internet software dependability in the context of an increased level of concurrency in such software systems. Their observation is backed by an earlier comparative software/hardware dependability discussion by Gray and Brewer [26]. Hinchey et al. highlight achievements in FM automation enabling an increased use of lightweight FMs in "software engineers' usual development environments." Furthermore, they stress the ability to use several FMs in a combined manner to verify distributed (embedded) systems, avoid errors and, hence, stop the decline of software dependability.

Hoare et al. [8] issue the manifesto of the "Verified Software Initiative." Based on a consensus of strengths, weaknesses, opportunities, and threats in the software engineering community, they propose a long-term international "research program towards the construction of error-free software systems."

Outlining an agenda for FM transfer, Jhala et al. [27] raise the need for improved benchmarks, metrics, and infrastructure for experimental evaluation, the need for revised teaching and training curricula, and the need for research communities interested in engaging with practitioners and working on ways to scale FMs up to large systems and to increase the usability of FMs. The authors specify several applications with great opportunities for FM transfer.

The applied researchers and practitioners interviewed by Schaffer and Voas [28] convey an optimistic picture of FM adoption in practice, highlighting the potentials to improve IT security, particularly, in cyber-physical systems. Chong et al. [29] share the view that FMs are the most promising approach towards acceptably dependable and secure systems. The challenges they list for the security domain are similar to the challenges we perceive in RAS assurance: FM integration, sound abstraction techniques, compositional guarantees, sufficient evidence for sustainable transfer.

With their survey of FMs for RAS verification, Luckcuck et al. [30] identify difficulties of applying FMs in the robotics domain and summarise research results and their limitations. They conclude (i) that formalisation remains the most critical and most difficult task, (i) that the surveyed approaches do not provide "sufficient evidence for public trust and certification," and (i) that iFMs would be highly desirable if the current lack of translations between the most relevant of the surveyed techniques (e.g. model checking) could be overcome. Our survey complements their observations by further analysis of the lack of unification of iFMs and the missing empirical evidence for the effectiveness of FMs and iFMs as well as by a corresponding research roadmap.

III. Strengths and Weaknesses of Formal Methods for Dependability Assurance

Following the guidelines for SWOT analyses in [31], we provide an overview of *Strengths* and *Weaknesses* of FMs.

A. Reputation, Education, and Training

The question "Are formal methods essential, or even useful, or are they just an intellectual exercise that gets in the way of building real-world systems?" in the guest editor's introduction of the "50 Years of Software Engineering" IEEE Software special theme issue [32] invites us to deliberate on this topic and summarise its highlights. Applied researchers have raised the issue of *limited effectiveness and productivity* of FMs, particularly, in large practical systems with changing requirements [33], [34]. FMs are known to be *difficult to apply in practice*, and *bad communication* between theorists and practitioners sustains the issue that FMs are taught but rarely applied [33]. In contrast, they are considered to have significant potential to cope with the toughest recent engineering problems: certifiable RAS assurance [6].

Since the beginning of software engineering there has been a controversial debate on the usefulness of FMs. FMs are well-suited to *substantially improve modelling precision, requirements clarity, and verification confidence*. FM applications in requirements engineering such as the "Software Cost Reduction" toolset [35] even carry the hypothesis of FM cost-effectiveness in its name. Already in the 1990s, FM researchers have started to examine FM usefulness with the aim to respond to critical observations of practitioners [36], [37], [38], [39], [40]. Some of these efforts culminated in empirical studies [41], [42] suggesting *high error detection effectiveness*, though with some controversy also caused by employed research designs [43], [44].

Jones and Bonsignour [45, Sec. 3.2, Tab. 3.2] suggest that the combination of *formal*[3] *inspections*, static analysis, and formal testing has been the best approach to defect prevention with *up to 99% of accumulated defect removal efficiency*. FMs can be seen as a particularly rigorous and systematic form of this approach, though even less often applied. In Appendix A, we make a brief excursion to the relationship between FMs and formal inspection and try to estimate the size of the population of FM users.

From two larger surveys, one in the early 1990s [46] and another one in the late 2000s [21], [22], we obtain a more

[3]Here, the word "formal" does not imply the use of formal semantics.

comprehensive picture of the typical advantages of FM use and barriers to FM adoption as seen by practitioners and practical FM researchers. In two recent surveys [47], [48], [49], we made two, not necessarily surprising but empirically supported, observations underpinning the main findings of the former studies: many practitioners *view FMs as promising* instruments with high potential, and would use these instruments to their maximum benefit, whether directly or through FM-based tools. However, the beneficial use of FMs is still hindered by severe obstacles (e.g. hard to learn, difficult to integrate in existing processes, too expensive, fallacy of invalid abstractions, difficult to maintain).

B. Transfer Efforts

FMs can be effective in two ways: First, they can reduce cost in incremental system design when being used as a prototyping technique, as formal test-driven development, or by crafting module assertions prior to programming. Once an initial formalisation (particularly, invariants) is available, it is argued for families of similar systems that, from the second or third application (e.g. iteration or increment) onwards, the benefit of having the formalisation outperforms the cumulative effort to maintain the formalisation up to an order of magnitude [50], [51]. Second, FMs can be relatively easily employed through knowledge extraction from existing artefacts and using automated tools such as, for example, formal or model-based post-facto testing tools or post-facto use of code assertion checkers [52], [53]. However, the second way of utilising FMs is known to be more compatible with everyday software practice.

Achievements collected in [54], [55], [56] show that many researchers have been actively working *towards successful FM transfer*. Moreover, researchers experienced in particular FMs draw positive conclusions from FM applications, especially, in scaling FMs through adequate tool support for continuous reasoning in agile software development [57], [58]. Other researchers report about progress in theorem proving of system software of industrial size (e.g. [59]) and about FM-based tools for practical use (e.g. [22], [60], [53]).

Furthermore, MBD and MDE have a history of wrapping FMs into software tools to make access to formalisms easier and to help automating tedious tasks via domain-specific languages (DSL) and visual notations.

Static (program) analysis is another branch where tool-supported FMs have been practically used successfully (e.g. [53]). However, by far not all static analysis tools are based on FMs and many of these tools are known to be exposed to the problem of reduced effectiveness because of *high false-positive rates*, particularly, if occasionally complex settings are not perfectly adjusted to the corresponding project [61].

Furthermore, the concolic testing technique [62], a post-facto FM, has seen multiple successes in industry [63], [64]. It exercises all possible execution paths of a program through systematic permutation of a sequence of branch conditions inferred by an instrumented concrete execution. It uses these symbolic execution paths and SMT solving to obtain a series of inputs that exercise the full range of program paths. It does not depend on a pre-defined model of the program, but effectively infers one based on the branch conditions. It can therefore readily be used on existing program developments, and has notably been used by Samsung for verification of their flash storage platform software [63]. Indeed, it is a belief of the authors of this latter work that post-facto methods provide greater opportunities for adoption of FMs into industry.

C. Evidence of Effectiveness

Whichever of these two directions is taken, *strong evidence* for the efficacy of FMs in practice is *still scarce* (e.g. [42]) and more anecdotal (e.g. [54], [56], [55], [28]), rarely drawn from comparative studies (e.g. [41], [42]), often primarily conducted in research labs (e.g. [65], [66]), or not recent enough to reflect latest achievements in verification tool research (e.g. [67]). We observe that a large fraction of empirical evidence for FM effectiveness can be classified as level 6 or 7 according to [23, Tab. 2], that is, too weak to draw effective conclusions.

Researchers from the software engineering measurement community [45, Sec. 4.4, p. 220] support this observation by stating that “there is very little empirical data on several topics that need to be well understood if proofs of correctness are to become useful tools for professional software development as opposed to academic experiments.”

Graydon [68] observes this lack of evidence of FM effectiveness in assurance argumentation. More generally, Rae et al. [69] notice insufficiently evaluated safety research. About 86% of works lack guidance to reproduce results, hence, forming a barrier to the advancement of safety practice. Although their study is limited to one conference series, it indicates deficiencies in the evaluation of DA research.

Overall, it is important to understand that the mentioned lack of evidence and successful transfers constitutes great opportunities for successful FM research and transfer and not necessarily risks of failure.

D. Expressivity

An often quoted weakness of MBD when applied to robotics is the “reality gap” [70], [71] that can exist between a naively constructed model and its corresponding real-world artefact. According to [70], over-reliance on simulation to test behaviour using naive and insufficiently validated robot models can lead to effort being applied to solving problems that do not exist in the real world. Worse, programs for robotic controllers developed in a model-based setting may fail when executed on real-world hardware, because “it is very hard to simulate the actual dynamics of the real-world” [70]. This problem is not only true of simulation, but any form of model-based analysis, including FMs.

The fundamental problem here is that it is impossible to model the behaviour of any physical entity precisely [72], unless we replicate the original. Moreover, as models become more detailed, their utility decreases and they can become just as difficult to comprehend and analyse as their real-world counterparts, an observation highlighted by the famous paradox of Bonini [73]. Nevertheless, as statistician George Box said “all models are wrong but some are useful” [74]: we

must evaluate a model not upon how "correct" it is, or how much detail it contains, but on how informative it is. According to [72], the antidote is not to abandon the use of models, but to recognise their inherent limitations and strengths, and apply them intelligently to reasoning about a specific problem. This means selecting appropriate modelling paradigms that enable specification of behaviour at a sufficiently detailed level of abstraction, and using the resulting models to guide the engineering process, with iteration where necessary.

E. Integration and Coordination

Modelling notations usually employ a particular paradigm to abstract the behaviour of the real-world. For example, the state-based paradigm, employed by FMs like Z, B, and refinement calculus, considers how the internal state of a system evolves, whilst the event-driven paradigm, employed in process calculi like CSP, CCS, and π-calculus, considers how behaviour may be influenced by external interactions. Consequently, individual formal methods are usually limited to considering only certain aspects or views of a system's behaviour [75], [76], which can limit their effectiveness when used in isolation. Many researchers have therefore sought to overcome this weakness by FM integration [75], [77], [76]. In their FM summary, Clarke and Wing [11] also stress the demand of FM integration.

The 1990s saw a large number of works in the direction of semantic unification and method integration [75], [77]. Theoretical foundations were provided by Hehner, in his seminal work on semantic unification using the "programs-as-predicates" approach [78], [79] and comparative semantics [80]. At the same time, refinement calculi were being developed [81], [82], [83], that would underlie the work on linking heterogeneous notations through abstraction. Meanwhile, Woodcock and Morgan [84] explored the integration of state- and event-based modelling using weakest preconditions, and several other works on this topic followed [85], [86], [87]. Hoare proposed a unified theory of programming [88] that links together the three semantic styles: denotational, operational, and algebraic. These developments culminated in Hoare and He's Unifying Theories of Programming [13] (UTP), a general framework for integration of semantically heterogeneous notations by application of Hehner's approach [89] to the formalisation of a catalogue of computational paradigms, with links between them formalised using Galois connections. This framework enabled a definitive solution to the integration of states and events, along with other computational paradigms, in the Circus language family [90], [91], [92].

Another result of these developments was a number of seminal works on FM integration [75], [77], [76]. Paige, taking input from earlier work on systematic method integration [93], defined a generic "meta-method" that aimed at integration of several formal and semiformal methods using notational translations with a common predicative semantic foundation, which builds on Hehner's work [79]. Meanwhile, Galloway and Stoddart [77], building on their previous work [86], likewise proposed the creation of hybrid FMs with a multi-paradigm approach. Moreover, Broy proposed that FMs should be integrated into the traditional V-method with common semantic foundations to link the various models and artefacts across development steps [76].

These diverse efforts eventually led to the founding of the Integrated Formal Methods (iFM) conference series in 1999 [7], with the aim of developing theoretical foundations for "combining behavioural and state-based formalisms". For the second iteration of the iFM conference [15], the scope broadened to consider all the different aspects of FM integration, including semantic integration, traceability, tool integration, and refinement. A few years later, a conference series was also established for UTP [94], with the aim of continuing to develop unifying semantics for diverse notations within the UTP framework.

However, there is as yet no agreed and general methodology for integrating FMs that could be applied to RAS [6]. Overall, integration is of particular pertinence to RASs, since they are multi-layered systems possessing a high degree of semantic heterogeneity. As Farrell et al. [6] state, they "can be variously categorised as embedded, cyber-physical, real-time, hybrid, adaptive and even autonomous systems, with a typical robotic system being likely to contain all of these aspects." When we consider RAS, we must consider advanced computational paradigms like real-time, hybrid computation with differential equations, probability, and rigid body dynamics. This implies the use of several different modelling languages and paradigms to describe the different aspects, and therefore a variety of analysis techniques to assure properties of the overall system. Assurance of autonomous systems will certainly therefore require iFMs [6]. Figure 1 summarises this relationship.

IV. Opportunities for Integrated Formal Methods

This section continues with the "environmental analysis" part of our SWOT analysis. Several key opportunities for the transfer of iFMs arise from ongoing challenges in DA, particularly, in RAS assurance and from looking at what other disciplines do to cope with similar challenges. In the following, we describe four such opportunities.

A. The Desire for Early Removal of Severe Errors

Summarising major challenges in automotive systems engineering in 2006, Broy [95, p. 39] indicated that practised modelling languages are not formalised and the *desired benefits cannot be achieved from semi-formal languages*. Moreover, *software engineering is not well integrated* with core control and mechanical engineering processes. Domain engineers would produce software/hardware sub-systems and mechanical sub-assemblies in undesirable isolation. Broy referred to a lack of iFMs for *overall architecture verification*.

Has the situation changed since then? Liebel et al. [96] report on significant drawbacks of model-centric development in embedded software practice (e.g. based on UML/SysML) if methods and tools are not well integrated or trained personnel is missing. Likely, Broy's criticism remains in contemporary automatic vehicle engineering and assurance practice. In fact, he has a recent, clearly negative, but not pessimistic answer to

this question [97]. Moreover, this view is shared by the Autonomy Assurance International Programme's discussion of assurance barriers,[4] that is, current challenges in the assurance of RAS applications. These barriers (e.g. validation, verification, risk acceptance, simulation, human-robot interaction) might be addressed by formal engineering models and calculations based on such models to be used as evidence in corresponding assurance cases.

A specific opportunity for the use of formal methods in (through-life) dependability assurance lies in model-based assurance [98], [99], which uses models of system elements to form the structure of an assurance case. The Structured Assurance Case Meta-Model (SACM)[5] represents a standardised DSL suitable for integrating system-level assurance evidence and, thus, from a specific branch of MDE.

Leading voices from applied software engineering research keep mentioning the role of FMs as a key technology to master upcoming challenges in assuring critical software systems [100]. A round table about the adoption of FMs in IT security [28] positively evaluates their overall suitability, the combination of FMs with testing, and the achievements in FM automation. The panellists notice some limitations of FMs in short-time-to-market projects and in detecting unknown vulnerabilities as well as shortcomings in FM training and adoption in practice.

However, even for mission-critical systems, high costs from late defect removal and long defect repair cycles [45] as well as dangerous and fatal[6] incidents indicate that assurance in some areas is still driven by practices failing to assist RAS engineers in overcoming their challenges.

Moreover, Neumann, an observer of a multitude of computing risks, states that "the needs for better safety, reliability, security, privacy, and system integrity that I highlighted 24 years ago in my book, Computer-Related Risks, are still with us in one form or another today" [101], [102], [103].

For example, artificial intelligence software, particularly, machine learning (ML) components have been developed at a high pace and used in many non-critical applications. Recently, ML components are increasingly deployed in critical domains. For verification and error removal, such software has to be transparent and explainable. Preferring verifiable algorithms to heuristics, Parnas [104] reminds of the corresponding engineering principle: "We cannot trust a device unless we know how it works." One way to follow this principle and establish transparency is to reverse engineer (i.e., to decide) the functionality of an ML component even if this is not possible in general [105]. FMs can help extract knowledge and reverse engineer abstractions of ML systems to improve explainability. Obviously, we might then ask to which extent the reverse engineered and verified functionality serves as a substitute for the original ML component.

These anecdotes make it reasonable to question current assurance practice. Seen through the eyes of assurance, these anecdotes suggest that we might again be facing a dependable software engineering *crisis* similar to the one from the late 1960s [97], [106].

> *Opportunity 1:* We as researchers and practitioners could really learn from this crisis and improve the way we correctly engineer and certify highly automated systems.

B. The Desire to Learn From Accidents and Their Root Causes

In the title of Section IV-A, the word "severe" refers to the negative consequences *potentially* caused by errors we want to remove using iFMs. The more severe the potential consequences of an error, the more critical is its early removal. The usefulness of iFMs, thus, positively correlates with their support in the removal of critical errors. However, the estimation of severity often also requires the careful study of past field incidents [107].

We speak of *field incidents* to refer to significant operational events in the *field* (i.e., in the environment where a technical system is operated), undesired because of their safety risks and their severe harmful consequences. Field incidents typically range from minor incidents to major accidents. It is important to separate the observed effect, the field incident, from its causes or, more precisely, from the *causal chains of events* leading to the observed effect. Hence, this analysis depends on the considered system perimeter (see, e.g. [9]). Depending on the possibilities of *observation* and the depth pursued in a *root cause analysis* (RCA), a conclusion on a possible cause can result in any combination of, for example, overall system failure, human error, adverse environmental condition, hardware fault, software fault, or specification error.[7]

There are many databases about field incidents, some are comprehensive including RCA, others are less detailed, and some are confidential, depending on the regulations in the corresponding application domain or industry sector. Based on such databases, accident research, insurance, and consumer institutions occasionally provide brief root cause statistics together with their accident statistics (e.g. [108]).[8]

Accident statistics allow certain predictions of the safety of systems and their operation, for example, whether risk has been and will be acceptably low. Such statistics are also used in estimations of the amount of field testing necessary[9] to sufficiently reduce risk [110].

However, without the analysis of accident causes, such statistics are *of little use in decisions on measures for accident prevention* [111], for example, on improvements of engineering processes, methods (e.g. iFMs), and technologies (e.g. iFM tools) used to build these systems. For this, we require *deep RCA* and statistics that *relate error removal by iFMs and incident root causes*. To this extent, deep RCA is crucial for the investigation of iFM effectiveness.

[4] See https://www.york.ac.uk/assuring-autonomy/body-of-knowledge/.

[5] See https://www.omg.org/spec/SACM/About-SACM/.

[6] For example, the fatal accident involving a Tesla advanced driving assistance system, https://www.theguardian.com/technology/2018/jun/07/tesla-fatal-crash-silicon-valley-autopilot-mode-report.

[7] Specification errors are also called development failures [9] and can be seen as flaws in the process of requirements validation.

[8] Apart from the databases mentioned in Section II, further examples of such databases are mentioned in [109, Sec. 1.1].

[9] For example, according to "As Low As Reasonably Practicable" or "So Far As Is Reasonably Practicable." See http://www.hse.gov.uk/risk/theory/alarpcheck.htm: "Something is reasonably practicable unless its costs are grossly disproportionate to the benefits."

To understand the current RCA situation, we studied [112] a sample of 377 reports from open field incident databases (in aviation, automotive, rail, energy, and others) finding the following:

1) RCAs in these reports were of poor quality, either because they were not deep enough, economically or technically infeasible, or inaccessible to us.
2) Particularly, root causes (e.g. software faults, specification error) were rarely documented in a way that useful information about the used technologies (e.g. software) or about consequences in the development process could be retrieved from the reports.
3) Reports in some sectors contain deeper RCAs (e.g. aerospace, rail, power plants, process industry) than others (e.g. automotive) because of different regulations.
4) Some sectors operate official databases (e.g. NHTSA[10] and NTSB[11] in the US transportation sector) and others do not (e.g. German road transportation sector).
5) Our findings suggest that, even in domains with regulated RCA, reports in open databases tend to be less informative than reports in closed databases.
6) The reports from the automotive industry exhibited a relatively small fraction of technology-related errors (e.g. software-related errors).

To validate our study and to better understand the context of our findings, we performed seven semi-structured interviews with safety practitioners [112], [47]. One takeaway was that, because of an unclear separation of technologies and a lack of explicit architectural knowledge, a desirable classification of root causes is sometimes infeasible. Hence, accident analysts can conclude their reports at a level of detail too low to draw helpful conclusions. Additionally, one expert stated that *the hidden number of software-related or software-caused field incidents in dependable systems practice is likely much larger than the known number*. This matches our intuition but we are missing clear evidence.

Ladkin demands regulations to mandate the use of systematic RCAs.[12] In support of his view, we believe that systematic deep RCA (based on iFMs) can be helpful to gain clarity about actual root causes. Again, beyond this undesirable form of late error removal, such data is essential for the *measurement of the effectiveness* of error removal techniques, particularly, iFMs.

The "Toyota unintended acceleration" incident [113] exemplifies the difficulty of drawing conclusions without using powerful RCA techniques: A first RCA concluded that floor mats and sticky throttle pedals caused a fatal car mishap. A second RCA carried out by NASA experts and based on *testing and automated static analysis* of the control system (i.e., software and hardware) was not conclusive. A third RCA[13] based on *code reviews*—we could not find out which level of formal inspection (Appendix A) was used—detected defects in the control software and safety architecture, demonstrated to be likely the causes of the accident [113].

[10]National Highway Traffic Safety Administration, https://www.nhtsa.gov.
[11]National Transportation Safety Board, https://www.ntsb.gov.
[12]From personal communication.
[13]See expert interview by embedded software journalist from 2013 on https://www.eetimes.com/document.asp?doc_id=1319903&page_number=1.

Opportunity 2: We could invest in the employment of integrated formal methods for certifiable RAS assurance to prevent field incidents, major product recalls, and overly lengthy root cause investigations.

C. The Desire of Assurance to Form a Mature Discipline

In his Turing Award acceptance speech about 40 years ago, Tony Hoare reviewed type safety precautions in programming languages and concluded: "In any respectable branch of engineering, failure to observe such elementary precautions would have long been against the law" [114].

Inspired by this comparison, it can be helpful to look at other engineering disciplines such as civil, mechanical, or electrical engineering to identify transfer opportunities for iFMs. There, engineers use FMs in many of their critical tasks. However, nowadays these methods are often hidden behind powerful software tools usable by qualified professional engineers. However, we do not see such a high level of FM adoption in dependable systems practice.

For example, in mechanical engineering, vocationally trained engineers use computer-aided engineering, design, and manufacturing software. Whether for designing machine parts for serial production (i.e., specification) or for calculations (e.g. dimensioning, force or material flow simulations), for these parts and their assembly (i.e., for prototype verification), these engineers use tools based on canonical mathematical models.

Nowadays, drawings from computer-aided mechanical design carry at least two types of semantics, one declarative based on calculus for dimensioning (1), and one procedural for the synthesis of Computer-Numerical-Control programs for production machines processing materials to realise the drawings (2). Note that the unifying base of these two semantics is geometry, a well-studied mathematical discipline. Although higher levels of complexity demand more sophisticated analytical expertise, typically from engineers with several years of work experience, many tasks can be accomplished by less trained engineers using the corresponding tools.

Whereas in computer-aided mechanical design both semantics seem to be used to a similar extent, in DA we observe that analogous semantics are rarely used even if tools are available, and less often we see (1) and (2) being consistently used. Low adoption might result from the semantics for dimensioning and production automation being usually less abstract than the semantics for verification (1) and synthesis (2) of computer programs. Accordingly, Parnas suggests a shift from correctness proof to property calculation to develop practical formal methods [34, p. 33].

Patterns have had a long history in many disciplines. In mechanical engineering, patterns are better known as *machine elements* and are particularly useful in high-reliability applications. Machine elements (and standardised forms thereof) have a stabilising impact on the outcome of an engineering project. The process of element selection and composition can take tremendous advantage not only from the *reuse* of proven design knowledge but also from the reuse of complex calculations (e.g. from gear transmissions, injection moulding

tools, skeleton framings). Moreover, modern tools typically foster the use of *element libraries* and *parametric design*. Importantly, because the properties of such elements are in many cases well known, calculations for assemblies (i.e., *compositional* verification) get relatively easy. However, the higher the required precision of these calculations the more expensive is their computation.

These observations are in line with what we know from collaborations in robotics, like mechatronics, a discipline where many engineering domains have to play together well: FMs are heavily used for the analysis of robot controllers and for various kinds of simulations and tests [115], [116].

Digital circuit engineering is a domain where FMs such as model checking have already been successfully applied decades ago. However, systematic hardware errors, such as Spectre and Meltdown, and the *unavailability of temporal specifications* of highly optimised operations (e.g. branch-prediction and speculative execution) discontinue the verifiability of recent computer architectures. This lack of verifiability of the assumptions (e.g. partitioning, information flow) about the execution platform complicates the verifiability of the software (e.g. an OS) running on such a platform.[14]

Opportunity 3: Dependability assurance has not yet successfully adopted iFMs as a vital part of their key methodologies. If FMs seem relatively well established in other disciplines, we might also be able to successfully transfer iFMs to RAS assurance and to dependability engineering practice in other domains.

D. The Desire for Adequate and Dependable Norms

A striking finding in one of our recent discussions of dependable systems standards (e.g. IEC 61508, ISO 26262, DO-178C) is that *normative parts for specification* (i.e., requirements engineering, RE), for specification validation (i.e., avoiding and handling requirements errors), *and for hazard and risk analysis* (particularly, in early process stages) seem to be *below the state of the art* [47], [103], despite several observations that significant portions (e.g. 44% [117]) of the *causes* of safety-critical software-related incidents fall into the category of *specification errors* [118], [119].

The literature provides plenty of evidence of undesired impacts of specification errors dating back as early as the investigations of Lutz [120] and Endres [121]. As reported by MacKenzie [108], the 92% of computer-related field incidents caused by human-computer interaction also illustrate the gap between specifications and capabilities of humans to interact with automation. Despite these older figures, we are talking of one of the most critical parts of standards. Practitioners could expect to receive strong guidance from these parts and requirements to show conformance to these parts should not be vacuous.

Many standards define *specific sets of requirements* (i.e., for error removal and fault-tolerance) depending on the level risk a system (or any part of it) might cause. The higher the risk level, the more demanding the allocated requirements, for example, ASIL C-D, systematic capability 3-4, SIL 3-4, Design Assurance Level A-B.[15] Even for the highest such levels the mentioned standards only "highly recommend" but not mandate the use of FMs.

Guidelines for embedded software development such as MISRA:1994 [122] *recommend* FMs for SIL 4, although MISRA:2004 does no more include such information and instead refers back[16] to MISRA:1994. As already mentioned, ISO 26262 as the overriding standard does not go beyond high recommendation of FMs for SIL 4. Koopman [113] reports in 2014 that, in the US, car manufacturers are not required to follow the MISRA guidelines and that there are no other software certification requirements. Note that this applies to autonomous road vehicles as well.

As an interesting anecdote, Ladkin, a researcher involved in the further development of IEC 61508, reports on his lack of success in introducing systematic hazard (and risk) analysis methodology into normative parts of this standard [123]. Moreover, he reports[17] on unsuccessful attempts to strengthen the role of FMs in IEC 61508 and on the "broken standardisation" in assurance practice. In reaction to that, he proposes the use of evidently independent peer reviews to "dampen committee-capture by big-company bully players."

Knight [119] complements: "There is an expectation by the community that standards will embody the best available technology and that their presentation will allow determination of conformance to be fairly straightforward. A criticism that is seldom heard is that some standards are, in fact, technically flawed and poorly presented." He exemplifies his critique by several issues with IEC 61508 and RTCA DO-178B and suggests to make the meaning of "conformance [or compliance] with a standard" more rigorous. Particularly, he encourages to replace *indirect* (i.e., process-related) evidence (e.g. documentation of specification activities) in assurance cases by *direct* (i.e., artefact-related) evidence (e.g. unsuccessful checks for presence of certain specification faults, successful checks for absence of implementation errors).[18] With the observation in software quality control that "there is little evidence that conformance to process standards guarantees good products," Kitchenham and Pfleeger [125] deliver a reasonable basis for Knight's suggestion.

Regarding the integration of dependability approaches and FMs, Bowen and Stavridou [126] state already in 1993 that they "do not know how to combine formal methods assurance with metrics collected from other techniques such as fault-tolerance." Is this still an issue? From a practical viewpoint, standards such as, for example, IEC 61508, ISO 26262, and DO-178C, provide recommendations about techniques for the reduction of both random hardware failures (e.g. by fault-

[14] See blog post on the seL4 microkernel, https://research.csiro.au/tsblog/crisis-security-vs-performance/.

[15] Automotive Safety Integrity Level, Safety Integrity Level.

[16] This is likely also the case for MISRA:2012 from March 2013. We are not aware of the opposite but also were not able to receive a copy of this version.

[17] See System Safety Mailing List message from 4/11/2018, http://www.systemsafetylist.org/4183.htm and [124].

[18] While formal verification serves the check of absence of property violations, conventional testing can only serve as a check of presence of such violations.

tolerance techniques) and systematic hardware and software failures (e.g. by FMs, static analysis, and testing). If iFMs can support the combined application of the recommended techniques and achieve an improvement in practice then we should really strive to demonstrate this.

We believe that critical fractions of strong direct evidence can be delivered through the use of FMs. In support of Feitelson's argument [103], we see a strong opportunity for an assessment of how the corresponding guidelines in these standards can be extended and aligned with recent results in FM research.

Opportunity 4: No more can we afford poorly regulated and poorly certified high-risk software in a time where dangerous autonomous machines are about to get widely deployed in our society.

V. Threats to the Adoption of Integrated Formal Methods in Assurance Practice

This section closes the "environmental analysis" part of our SWOT analysis by identifying potential threats to the success of FM transfer as well as challenges that arise from alternative or competing approaches taking the opportunities mentioned in Section IV. We also hint to some remedies to these threats.

The development of effective iFMs and their successful transfer into practice can be impeded by

- a lack of agreement on a sound semantic base for domain-specific and cross-domain iFMs (Sections V-A and V-B),
- a missing support of widely used and established tools (Section V-B),
- a lack of interest in practical problems on the side of FM researchers (Section V-C),
- a lack of incentives for FM researchers to engage with current practice and for software practitioners to engage with recent theoretical results (Section V-C),
- a bad reputation among practitioners and applied researchers (Section V-C),
- proofs that are faulty or do not scale (Section V-D),
- the quest for soundness overriding the quest for usefulness (Section V-E).

We discuss these threats and barriers in more detail in the following.

A. Difficulties and Misconceptions of Unification

According to [95], the successes and failures of semi-formal languages (e.g. UML or SysML) suggest that FMs, once wrapped in FM-based tools, get exposed to the *quest for a unified syntax*, one main objective of the UML movement in the 1990s. Rather than a unified syntax, it is more desirable to have a unified semantics and several well-defined mappings to domain-specific syntax wherever convenient (Section III-E). This approach is occasionally taken up by DSLs in MDE (Section III-B). It has been argued [127] that one cannot achieve proper integration of methods and notations without a unifying semantics. This argument carries over to the problem of tool integration as already discussed in Section III-E and revisited below. Particularly, the following challenges apply to FMs when used in MDE:

1) the maintenance of a single source of information serving in the (automated) derivation of downstream artefacts [128] (e.g. proof results, code via synthesis),
2) a clear mapping between the DSL presented to the engineer (using intuitive notation) and the DSL semantics serving as the basis of formal verification,
3) the embedding of a lean domain-specific formalism into a *common* [129] data model suitable for access and manipulation by engineers through their *various* [130] tools.

These challenges are corroborated by irreducible unidirectionalities in automated transformations (e.g. model-to-code) limiting the desirable round-trip engineering [131] (i.e., the change between views of the same data).

We discussed SACM as an assurance DSL in Section IV-A. Likewise, *architecture description languages* (e.g. the Architecture Analysis & Design Language [132], EAST-ADL [133]) are DSLs for overall embedded system design. DSLs can be seen as one shortcut to the still ongoing efforts of arriving at a reduced version or a variant of UML where a semantics can be defined for the whole language (e.g. [20]).

At a higher level of abstraction, so-called *architecture frameworks* (cf. ISO 42010, e.g. the Department of Defense Architecture Framework) and *artefact and traceability models* (e.g. [134], [135]) have been proposed, aiming at the standardisation of specific parts of the systems and software engineering life-cycle and of the documentation and data models used there. These frameworks and models are similar to the models used in product data/life-cycle management in fields like mechanical or civil engineering.

To our best knowledge, no cross-disciplinary semantic unification has been undertaken yet (see Section III-E), serving as a basis for dependable systems engineering. Although many of these approaches have not been developed with the aim of formalisation and the unification of semantics, we believe that this effort has to be made when developing powerful iFMs.

FM integration and refinement-based software engineering could be aligned with artefact models (see, e.g. [135], [136]), particularly, because formal semantics can help establishing traceability among the artefacts and handling traceability issues in the engineering process (see, e.g. [137], [138]).

B. Reluctant Integration Culture and Legacy Processes

Tool integration is about the integration of engineering IT, e.g. tools for requirements specification, computer-aided software engineering, computer-aided mechanical design. Among the wide variety of solutions to capture and track model data, the majority deals with linking or merging data models [130] in one or another shallow way (e.g. software repositories, data exchange formats, product/engineering/application data or life-cycle management systems).

Some tools with sustainable support are heavyweight, making it difficult to agree on lean model semantics, others are proprietary, accompanied with interest in hiding model semantics. The surveys of Liebel et al. [96, pp. 102,104],

Mohagheghi et al. [128, p. 104], and Akdur et al. [139] confirm that method and model integration are not yet solved in MBD, MDE, and dependable systems practice. Moreover, frequent proposals by researchers (e.g. [17], [19], [20]) to formalise the syntax and semantics of fragments or variants of UML and SysML have not yet received wide attention by practitioners and standardisation authorities.

However, modern DSL-based integrated development environments get close to what is suitable for FM-based tools relying on a lean representation of the formal semantics integrating the model data. For successful iFM transfer to assurance practice, tools need to be built on a lean and open central system model (see, e.g. [140], [141]).

An even greater barrier than loosely integrated tools are legacy *language and modelling paradigms*, an *established tool and method market* carried by *legacy stakeholders* and, possibly, a *neglected continuous improvement of FM education and training*. However, education through teaching and transfer through training, application, and feedback are decisive.

C. Reluctant Transfer Culture and Exaggerated Scepticism

Finally, the vision of introducing iFMs into assurance practice might be hindered by a lack of FM researchers able or willing to engage with industrial assurance practice, as diagnosed in [21]. It is certainly hard work to collect sufficient evidence for FM effectiveness in assurance practice because of intellectual property rights and other legal issues but also because of a lack of awareness [21] among FM researchers. However, for credible method comparison experiments, Jones and Bonsignour [45] recommend a sample of 20 similar projects split into two groups, 10 projects without treatment (i.e., not using FMs) and 10 projects with treatment (i.e., using FMs) to establish strong evidence (i.e., evidence of level 5 or above [23]).

Exaggerated scepticism on the side of practitioners and applied researchers that has piled up through the years might be one of the most important barriers to cross. Early failures to meet too high expectations on FMs and FM transfer might have led to what can be called an "FM Winter." However, we think crossing a few other barriers first might make it easier to cope with scepticism in the assurance community and initiate an "FM Spring" at least in assurance practice.

D. Too Many Errors in Proofs and Failure to Scale

From the perspective of measurement, Jones and Bonsignour [45, Sec. 4.1] state that "proofs of correctness sound useful, but [i] errors in the proofs themselves seem to be common failings not covered by the literature. Further, large applications may have thousands of provable algorithms, and [ii] the time required to prove them all might take many years." For [i], the authors oppose 7% of erroneous bug repairs to up to 100% of erroneous proofs, though stating that the latter is based on an anecdote and there is little data around. Jones and Bonsignour provide an example for [ii]: Assuming one provable algorithm per 5 function points[19] and on average 4 proofs per day, Microsoft Windows 7 (160,000 function points) would have about 32,000 provable algorithms, taking a qualified software engineer about 36 calendar years. They highlight that typically only around 5% of the personnel are trained to do this work, assuming that algorithms and requirements are stable during proof time.

[19]A function point is a measure of the conceptual complexity of an IT system relevant for the estimation of the amount of work required to engineer this system.

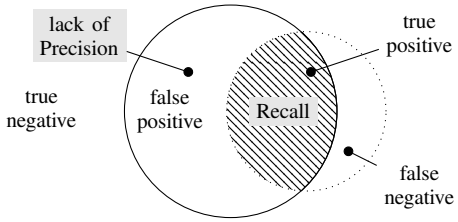

Fig. 2. Precision and recall vs. soundness and completeness

E. Failure to Derive Useful Tools

Being loosely related to erroneous proofs, the *information overload through false-positive findings of errors* is a well-known problem in static program analysis. Semi-formal pattern checkers, such as PMD and FindBugs, are heavily exposed to this threat [61]. Additionally, FM-based verification tools, such as Terminator and ESC/Java [142], can also be unable to correctly report all potential problems, because they are bounded and, therefore, unsound. While such tools can be very helpful, confronting developers with too many irrelevant findings can lead to a decreased use of FM-based tools.

Figure 2 helps to relate the two information retrieval metrics *precision* and *recall* with two adequacy criteria of proof calculi, *soundness* and *completeness*. Completeness, although unachievable for richer theories, would correspond to recall and soundness would correspond to a precision of 1. On the one hand, the usefulness of the calculi underlying FMs is directly proportional only to their completeness and (traditionally) expires with a precision of < 1. On the other hand, the usefulness of semi-formal pattern checkers leaves great freedom as it is directly proportional to both precision and recall of their findings. Practical tool usefulness might hence lie somewhere in the middle between these two extremes.

VI. A Vision of Integrated Formal Methods for Dependability Assurance

The following discussion applies to many domains of dependability assurance. However, the complexity of robots and autonomous systems forms a key opportunity for the progress of iFM research and for its successful transfer. Accordingly, Table I summarises the discussion in Sections III to V with an interpretation into RAS assurance practice. Based on the strengths and opportunities described in Sections III and IV, we formulate our vision in terms of working hypotheses:

1) From Section IV-A: Computer-assisted tools for the construction of arguments and production of evidence using *iFMs can meet the challenge* of assuring RAS safe. Machine-checked assurance cases will greatly increase confidence in their sufficiency, and also aid in their

TABLE I
OVERVIEW OF OUR SWOT ANALYSIS (ACCORDING TO [31]) OF "IFMS IN PRACTICAL RAS ASSURANCE"

Method Strengths: iFMs raise the potential of ... • improvement of RAS models, specification of RAS requirements, automation of RAS verification (Section III-A) • early detection of systematic errors in RAS designs (Section III-A) • integration and coordination of several FMs to consistently reason about interrelated RAS aspects (Section III-E) **Community Strengths: iFM research can rely on ...** • many transfer re-entry points from former RAS case studies in industrial and academic labs (Section III-B) • many assurance practitioners who perceive FM usefulness as positive (Section III-A)	**Method Weaknesses: iFMs have been suffering from ...** • being difficult to learn and apply, many assurance practitioners perceive ease of use of FMs as negative (Section III-A) • low effectiveness of formal models because of the reality gap (Section III-D) • fragile effectiveness and productivity in RAS engineering (Section III-C) **Community Weaknesses: iFM progress has been hampered by ...** • no agreed framework for integration of FMs (Section III-E) • lack of convincing evidence of FM effectiveness in RAS engineering (Section III-C) • research ineffectively communicated in iFM teaching/training (Section III-A)
Key opportunities for iFM research transfer and progress: • The desire for early removal of erroneous RAS behaviour and model-based assurance (Section IV-A) • The desire to learn from RAS accidents and their root causes (Section IV-B) • The desire of RAS assurance to be a mature discipline (Section IV-C) • The desire for adequate and dependable RAS norms (Section IV-D)	**Method Threats: iFM research could be threatened by ...** • misconceptions of semantic unification in RAS practice (Section V-A) • iFMs do not scale up to industry-size RAS applications (Section V-D) • faulty, tedious, or vacuous proofs (Sections V-D and V-E) • poor integration with RAS engineering tools and processes (Section V-B) **Transfer Threats: iFM transfer could be threatened by a ...** • lack of roboticists' education in iFMs (Section V-B) • lack of iFM researcher engagement in transfer to RAS practice (Section V-C) • lack of comprehensive access to quality data from RAS practice (Section V-C)

maintenance and evolution through modularisation of arguments and evidence.

2) From Sections IV-A and IV-C: iFMs, in particular modern verification tools, will enable *automation of the evidence gathering process*, and highlight potential problems when an assurance case changes.
3) From Sections IV-A and IV-C: Moreover, there is *no stable path to assured autonomy without the use of iFMs*. Acceptable safety will be much more likely achieved with iFMs than without their use.
4) From Section III-E: However, the success of iFMs depends on the ability to *integrate a variety of FMs* for different aspects of RAS (e.g. human-machine interaction, safety-security interaction, missing human fallback, environment/world modelling, uncertain prediction/behaviour), which is not currently possible.
5) From Sections III-D and III-E: Sophisticated techniques for *model integration and synchronisation are necessary* to support MDE with iFMs. This way, iFMs will make it easier to express consistent RAS models covering all relevant aspects, make their modelling assumptions explicit, and improve future assurance practices.
6) From Sections III-A to III-C and V-C: iFMs can be beneficial in the short term. However, an important engineering principle is to be conservative and, therefore, not to change procedures unless there is compelling evidence that iFMs are effective. Such evidence can be delivered through *empirical research* (e.g. [42], [41], [21], [143] on FMs in general) and collaboration between academia and industry. Moreover, such evidence *is required to re-evaluate research and foster research progress and transfer*.
7) From Section IV-B: The demonstration of cost effectiveness in addition to technical effectiveness of new iFM research is necessary to justify further research progress.
8) From Section IV-D: Norms are a lever of public interest in dependability [103]. Current norms seem to deviate from the state of the art and may fail to guarantee product certification procedures compliant with these interests.

Figure 3 assigns these hypotheses to the relationships between foundational and transfer-directed iFM research by example of the RAS application domain.

Overall, we believe that iFMs have great potential and are believed to improve assurance but practitioners do not use them accordingly.

Opportunity 5: We could take and enhance *credible measures to convince assurance practitioners* of our results and effectively transfer these results. For this to happen, we have to answer further *research questions*.

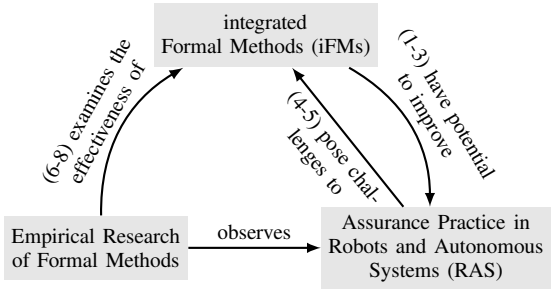

Fig. 3. Progress of research on integrated formal methods through transfer into and improvement of assurance practice of robots and autonomous systems

VII. Empirical, Applied, and Foundational Research in Integrated Formal Methods

Based on the aforementioned working hypotheses, we state several objectives for foundational and transfer-directed iFM research, formulate research questions, and show our expectations on desirable outcomes of such research.

A. Research Objectives and Tasks

To validate and transfer our research results, we need to

- re-evaluate how assurance case construction and management for RAS can be improved by iFMs.
- debunk or justify arguments against the use of FMs or FM-based tools in RAS assurance practice.
- foster successful FM research transfer to RAS industries performing assurance and certification.

Taking an iFM foundational point of view, we need

- foundational research on integration and unification of FMs to tackle the complexity of RAS [6].
- a unified semantic foundation for the plethora of notations in RAS assurance, to enable method and tool integration. There are currently a number of promising research directions being undertaken here [13], [144].

Taking an evidence-based point of view (as already highlighted in 1993 in [126]), we need to

- understand the difference between the state of assurance practice and state of assurance research.
- understand in which ways current RAS assurance practices fail and suggest effective approaches from assurance research. In this way, we can be sure that assurance practice is equipped with state-of-the-art assurance technology for holding up against potential liability claims, and assurance practitioners do not fail in fulfilling their obligations.
- understand how results from assurance research can be validated. In this way, we can be sure that assurance research follows promising directions with high potential of success in assurance practice.

Based on that, we need to

- set concrete directions for empirical FM research in RAS assurance.
- train FM researchers in applying empirical research designs in their research of rigorous assurance cases. Woodcock et al. [21] corroborate this objective by saying that "formal methods champions need to be aware of the need to measure costs."
- avoid biases as found in various branches of scientific research, such as e.g. nutrition and medical sciences.
- increase the level of evidence of FM research to level 2 according to the hierarchy in [23, Tab. 2].
- avoid knowledge gaps about whether (a) RAS practice is keeping up with state of the assurance art, and (b) whether recent academic or industrial research is going in the right directions. In this way, we can be sure that we are doing our best to inform and serve the society.

Using appropriate research designs, we need to

- invite the RAS industry to enhance their efforts in engaging with recent iFM research.
- foster goal-oriented interaction (a) between assurance practitioners and researchers and (b) between the FM researchers and assurance researchers. In this way, we can be sure to do everything to keep FM and assurance researchers up to date with respect to practical demands.
- join the FM research and applied assurance research communities (Figure 1), both vital for the progress and transfer of assurance research into RAS assurance practice. This way, we can be sure to foster necessary knowledge transfer between these two communities.
- further summarise achievements in practical application of iFMs for constructing assurance cases.
- suggest improvements of curricula for RAS assurance.
- guide assurance and certification companies in process improvement, training, and tool support.
- guide vendors of FM-based assurance tools to assess and improve their tools and services.

B. Some Research Questions addressing these Objectives

The research questions below are relevant for FMs in general. We consider these questions as crucial to be answered for RAS assurance to address the aforementioned objectives:

1) What is the true extent of computer-related accidents up to 2019 [108]? What would these figures mean for the RAS domain?
2) Does the use of formalism detect severe errors to a larger extent than without the use formalism [42], [41]?
3) Does the use of formalism detect severe errors earlier than without using formalism?
4) Why would such errors be a compelling argument for the use of FMs?
5) Apart from error avoidance and removal, which other benefits of iFMs in practice are evident and can be utilised for method trade-offs?
6) Beyond scalable FM-based tools, which other criteria play a central role in measuring iFM effectiveness?
7) How would Commercial-off-the-Shelf and System-Element-out-of-Context verification by iFMs pay off?
8) Which hurdles need to be overcome to use iFMs in practice to the maximum benefit?
9) How do we know when these hurdles are actually overcome?
10) How can FMs (from different disciplines) be used together (iFMs, unification)?
11) How can FMs be used to assure systems involving AI techniques like machine learning, deep neural networks, and computer vision?
12) How can FMs be integrated into assurance cases to support certification against international safety and security standards?
13) How do we combine formal and informal methods?
14) How do we present formal requirements, evidence, and artefacts in an assurance case?
15) How can empirical research help in successfully demonstrating the capabilities of iFMs for rigorous and certifiable autonomy assurance?

This list of research questions can easily be extended by further more detailed empirical questions from the settings discussed in [45, Sec. 4.4].

C. Envisaged Research Outcomes

Our vision of *rigorous RAS assurance* implies foundational iFM research to result in:

- novel semantic frameworks unifying best practice methods, models, and formalisms established in RAS,
- new concepts for iFM-based development environments,
- new computational theories to support formal modelling and verification of RAS
- evaluations of
  - assurance tools, languages, frameworks, or platforms used in practice regarding their support of iFMs
  - the integration of iFMs into modelling and programming techniques, assurance methods, and assurance processes
  - languages for linking informal requirements with evidence from iFMs
  - (automated) abstraction techniques used in assurance and certification.
- opinions, positions, and visions on FM integration and unification for rigorous practical assurance.

Our vision of *rigorous RAS assurance* implies applied and empirical iFM research to result in:

- comparisons of
  - projects applying iFMs in assurance practice with similar practical projects applying non-iFM approaches
  - iFM-based (embedded software) assurance with assurance approaches in traditional engineering disciplines
- checklists, metrics, and benchmarks (for and beyond tool performance) for
  - the evaluation and comparison of iFM-based assurance approaches (e.g. confidence level)
  - relating error removal and incident root cause data (e.g. efficiency and effectiveness in removal of severe errors or in avoidance of severe accidents, cf. [108])
  - usability and maturity assessment of iFMs (e.g. abstraction effort, proof complexity, assurance case complexity, productivity)
  - the evaluation of FM budget cases (cf. [145] in electronic hardware development).
- experiences in or surveys (e.g. systematic mappings and reviews of assurance case research, interview studies with assurance practitioners) of
  - iFM transfers and applications (e.g. case studies in assurance and certification projects)
  - challenges, limitations/barriers, and benefits of iFMs in assurance and certification projects,
- research designs (e.g. for controlled field experiments) for the practical validation of iFMs in assurance and certification projects
- opinions, positions, and visions on future research, education, and training in FM-based assurance.

VIII. Summary

Along the lines of Hoare et al. [8], we carried through an analysis of strengths, weaknesses, opportunities, and threats to determine the potential of integrated formal methods to improve the practice of dependability assurance. From the particular perspective of the robots and autonomous systems domain, we outline several working hypotheses to express our scientific vision. From these hypotheses, we derive a research and research transfer agenda with the objective of (i) enhancing the foundations of integrated formal methods, (ii) collecting evidence on the effectiveness of integrated formal methods in practice, (iii) successfully transferring integrated formal methods into the assurance practice of robots and autonomous systems, and (iv) fostering research progress, education, and training from the results of this transfer effort.

Acknowledgements: We would like to thank Michael Butler, Ana Cavalcanti, John Fitzgerald, Cliff Jones, and Peter Gorm Larsen for very helpful feedback and discussions on the direction of this work.

References

[1] P. G. Neumann, "Risks to the public," *ACM SIGSOFT Software Engineering Notes*, vol. 43, no. 2, pp. 8–11, may 2018.

[2] C. A. Gunter, E. L. Gunter, M. Jackson, and P. Zave, "A reference model for requirements and specifications," *IEEE Software*, vol. 17, no. 3, pp. 37–43, May/June 2000.

[3] J. Guiochet, M. Machin, and H. Waeselynck, "Safety-critical advanced robots: A survey," *Robotics and Autonomous Systems*, vol. 94, 2017.

[4] J. Rushby, "Logic and epistemology in safety cases," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 1–7.

[5] W. S. Greenwell, J. C. Knight, C. M. Holloway, and J. J. Pease, "A taxonomy of fallacies in system safety arguments," in *24th International System Safety Conference*, 2006.

[6] M. Farrell, M. Luckcuck, and M. Fisher, "Robotics and integrated formal methods: Necessity meets opportunity," in *Proc. 14th. Intl. Conf. on Integrated Formal Methods (iFM)*, vol. LNCS 11023. Springer, 2018, pp. 161–171.

[7] K. Araki, A. Galloway, and K. Taguchi, Eds., *Proc. 1st Intl. Conf. on Integrated Formal Methods*. Springer, 1999.

[8] C. A. R. Hoare, J. Misra, G. T. Leavens, and N. Shankar, "The verified software initiative," *ACM Computing Surveys*, vol. 41, no. 4, pp. 1–8, oct 2009.

[9] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, no. 1, pp. 11–33, 1 2004.

[10] T. P. Kelly, "Arguing safety – A systematic approach to safety case management," Ph.D. dissertation, University of York, Dept. of Computer Science, 1999.

[11] E. M. Clarke and J. M. Wing, "Formal methods: State of the art and future directions," *ACM Computing Surveys*, vol. 28, no. 4, pp. 626–643, dec 1996.

[12] K. J. *et al.*, *Formal Methods Supplement to DO-178C and DO-278A*. RTCA, Inc., 2011.

[13] C. A. R. Hoare and H. Jifeng, *Unifying Theories of Programming*. Pearson College Div, 1998.

[14] R. J. van Glabbeek, *Handbook of Process Algebra*. Elsevier, 2001, ch. 1. "The Linear Time - Branching Time Spectrum I: The Semantics of Concrete, Sequential Processes", pp. 3–99.

[15] W. Grieskamp, T. Santen, and B. Stoddart, Eds., *Proc. 2nd Intl. Conf. on Integrated Formal Methods*, ser. LNCS. Springer Berlin Heidelberg, 2000, vol. 1945.

[16] E. Börger, M. Butler, J. P. Bowen, and P. Boca, Eds., *Abstract State Machines, B and Z*, ser. LNCS. Springer Berlin Heidelberg, 2008, vol. 5238.

[17] M. Giese and R. Heldal, "From informal to formal specifications in UML," *The Unified Modelling Language*, pp. 197–211, 2004.

[18] R. Wieringa and E. Dubois, "Integrating semi-formal and formal software specification techniques," *Information Systems*, vol. 19, no. 4, pp. 33–54, 1994.

[19] R. Breu, U. Hinkel, C. Hofmann, C. Klein, B. Paech, B. Rumpe, and V. Thurner, "Towards a formalization of the unified modeling language," in *ECOOP'97 – Object-Oriented Programming*. Springer Berlin Heidelberg, 1997, pp. 344–366.

[20] E. Posse and J. Dingel, "An executable formal semantics for UML-RT," *Software & Systems Modeling*, vol. 15, no. 1, pp. 179–217, feb 2016.

[21] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald, "Formal methods: Practice and experience," *ACM Comput. Surv.*, vol. 41, no. 4, pp. 19:1–19:36, Oct. 2009.

[22] J. C. Bicarregui, J. S. Fitzgerald, P. G. Larsen, and J. C. P. Woodcock, "Industrial practice in formal methods: A review," in *FM 2009: Formal Methods*, A. Cavalcanti and D. R. Dams, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 810–813.

[23] C. L. Goues, C. Jaspan, I. Ozkaya, M. Shaw, and K. T. Stolee, "Bridging the gap: From research to practical advice," *IEEE Software*, vol. 35, no. 5, pp. 50–57, 9 2018.

[24] A. van Lamsweerde, "Formal specification: A roadmap," in *Proceedings of the Conference on The Future of Software Engineering*, ser. ICSE '00. New York, NY, USA: ACM, 2000, pp. 147–159.

[25] M. Hinchey, M. Jackson, P. Cousot, B. Cook, J. P. Bowen, and T. Margaria, "Software engineering and formal methods," *Commun. ACM*, vol. 51, no. 9, pp. 54–59, 2008.

[26] J. Gray and E. Brewer, "Dependability in the internet era," in *Proceedings of the High Dependability Computing Consortium Conference*, 2001.

[27] R. Jhala, R. Majumdar, R. Alur, A. Datta, D. Jackson, S. Krishnamurthi, J. Regehr, N. Shankar, and C. Tinelli, "Formal methods: Future directions & transition to practice," National Science Foundation, Workshop Report, 2012. [Online]. Available: http://goto.ucsd.edu/~rjhala/NSFWorkshop/

[28] K. Schaffer and J. Voas, "What happened to formal methods for security?" *Computer*, vol. 49, no. 8, pp. 70–79, aug 2016.

[29] S. Chong, J. Guttman, A. Datta, A. Myers, B. Pierce, P. Schaumont, T. Sherwood, and N. Zeldovich, "Report on the NSF workshop on formal methods for security," National Science Foundation, Tech. Rep., 2016.

[30] M. Luckcuck, M. Farrell, L. Dennis, C. Dixon, and M. Fisher, "Formal specification and verification of autonomous robotic systems: A survey," *ArXiv e-prints*, 2018.

[31] N. Piercy and W. Giles, "Making SWOT analysis work," *Marketing Intelligence & Planning*, vol. 7, no. 5/6, pp. 5–7, 1989.

[32] H. Erdogmus, N. Medvidovic, and F. Paulisch, "50 Years of software engineering," *IEEE Software*, vol. 35, no. 5, pp. 20–24, sep 2018.

[33] R. L. Glass, *Facts and Fallacies of Software Engineering*. Pearson Education (US), 2002.

[34] D. L. Parnas, "Really Rethinking 'Formal Methods'," *IEEE Computer*, vol. 43, no. 1, pp. 28–34, 2010.

[35] C. Heitmeyer, A. Bull, C. Gasarch, and B. Labaw, "SCR: a toolset for specifying and analyzing requirements," in *Proceedings of the Tenth Annual Conference on Computer Assurance, Systems Integrity, Software Safety, and Process Security - COMPASS'95*. IEEE, 1995.

[36] A. Hall, "Seven myths of formal methods," *IEEE Software*, vol. 7, no. 5, pp. 11–19, 1990.

[37] J. P. Bowen and M. G. Hinchey, "Seven more myths of formal methods," *IEEE Software*, vol. 12, no. 4, pp. 34–41, Jul 1995.

[38] J. C. Knight, C. L. DeJong, M. S. Gibble, and L. G. Nakano, "Why are formal methods not used more widely?" in *Fourth NASA Formal Methods Workshop*, 1997, pp. 1–12.

[39] L. M. Barroca and J. A. McDermid, "Formal methods: Use and relevance for the development of safety-critical systems," *Comp. J.*, vol. 35, no. 6, pp. 579–99, 1992.

[40] B. Littlewood, I. Bainbridge, and R. E. Bloomfield, "The use of computers in safety-critical applications," London, UK, 1998. [Online]. Available: http://openaccess.city.ac.uk/1955/

[41] A. Sobel and M. Clarkson, "Formal methods application: An empirical tale of software development," *IEEE Transactions on Software Engineering*, vol. 28, no. 3, pp. 308–320, mar 2002.

[42] S. L. Pfleeger and L. Hatton, "Investigating the influence of formal methods," *Computer*, vol. 30, no. 2, pp. 33–43, 1997.

[43] A. K. Sobel and M. Clarkson, "Response to comments on "Formal methods application: An empirical tale of software development"," *IEEE Transactions on Software Engineering*, vol. 29, no. 6, pp. 572–575, jun 2003.

[44] D. Berry and W. Tichy, "Comments on "Formal methods application: an empirical tale of software development"," *IEEE Transactions on Software Engineering*, vol. 29, no. 6, pp. 567–571, jun 2003.

[45] C. Jones and O. Bonsignour, *The Economics of Software Quality*. Addison-Wesley Professional, 2011.

[46] S. Austin and G. Parkin, "Formal methods: A survey," National Physical Laboratory, Teddington, Middlesex, UK, techreport, Mar. 1993.

[47] M. Gleirscher and A. Nyokabi, "Safety practice and its practitioners: Exploring a diverse profession," Department of Computer Science, University of York, UK, Unpublished working paper, 2018.

[48] M. Gleirscher and D. Marmsoler, "Formal Methods: Oversold? Underused? A survey," Department of Computer Science, University of York, Unpublished working paper, 2018.

[49] ——, "Electronic supplementary material for "Formal methods: Oversold? Underused? A survey"," Zenodo, 2018.

[50] S. P. Miller, "The industrial use of formal methods: was darwin right?" in *Proceedings. 2nd IEEE Workshop on Industrial Strength Formal Specification Techniques*. IEEE Comput. Soc, 1998.

[51] S. P. Miller, D. A. Greve, M. M. Wilding, and M. Srivas, "Formal verification of the aamp-fv microcode," NASA, Tech. Rep. NASA/CR-1999-208992, 1999. [Online]. Available: https://ntrs.nasa.gov/search.jsp?R=19990018441

[52] K. R. M. Leino, "Accessible software verification with dafny," *IEEE Software*, vol. 34, no. 6, pp. 94–97, nov 2017.

[53] D. Kästner, S. Wilhelm, S. Nenova, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, and X. Rival, "Astrée: Proving the absence of runtime errors," in *Proc. of Embedded Real Time Software and Systems (ERTS2)*, vol. 9, 2010.

[54] B. K. Aichernig and T. Maibaum, Eds., *Formal Methods at the Crossroads. From Panacea to Foundational Support*. Springer Berlin Heidelberg, 2003.

[55] S. Gnesi and T. Margaria, *Formal Methods for Industrial Critical Systems: A Survey of Applications*. Wiley-IEEE Press, 2013.

[56] J.-L. Boulanger, *Industrial Use of Formal Methods: Formal Verification*. Wiley-ISTE, 2012.

[57] S. P. Miller, M. W. Whalen, and D. D. Cofer, "Software model checking takes off," *Communications of the ACM*, vol. 53, no. 2, pp. 58–64, feb 2010.

[58] P. W. O'Hearn, "Continuous reasoning," in *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science - LICS'18*. ACM Press, 2018.

[59] G. Klein, J. Andronick, M. Fernandez, I. Kuz, T. Murray, and G. Heiser, "Formally verified software in the real world," *Communications of the ACM*, vol. 61, no. 10, pp. 68–77, sep 2018.

[60] J. Peleska and W. ling Huang, "Industrial-strength model-based testing of safety-critical systems," in *FM 2016: Formal Methods*. Springer International Publishing, 2016, pp. 3–22.

[61] M. Gleirscher, D. Golubitskiy, M. Irlbeck, and S. Wagner, "Introduction of static quality analysis in small and medium-sized software enterprises: Experiences from technology transfer," *Software Quality Journal*, vol. 22, no. 3, pp. 499–542, 9 2014.

[62] P. Godefroid, N. Klarlund, and K. Sen, "Dart: Directed automated random testing," in *Programming Language Design and Implementation (PLDI)*, ser. ACM SIGPLAN Notices, vol. 40, no. 6. ACM, 2005, pp. 213–223.

[63] M. Kim, Y. Kim, and Y. Choi, "Concolic testing of the multi-sector read operation for flash storage platform software," *Formal Aspects of Computing*, vol. 24, pp. 355–374, 2011.

[64] P. Godefroid, M. Y. Levin, and D. Molnar, "SAGE: Whitebox fuzzing for security testing," *Communications of the ACM*, vol. 55, no. 3, 2012.

[65] A. J. Galloway, T. J. Cockram, and J. A. McDermid, "Experiences with the application of discrete formal methods to the development of engine control software," *IFAC Proceedings Volumes*, vol. 31, no. 32, pp. 49–56, sep 1998.

[66] A. Chudnov, N. Collins, B. Cook, J. Dodds, B. Huffman, C. MacCárthaigh, S. Magill, E. Mertens, E. Mullen, S. Tasiran, A. Tomb, and E. Westbrook, "Continuous formal verification of Amazon s2n," in *Computer Aided Verification*. Springer International Publishing, 2018, pp. 430–446.

[67] N. Cataño and M. Huisman, "Formal specification and static checking of gemplus' electronic purse using ESC/java," in *FME 2002:Formal Methods—Getting IT Right*. Springer Berlin Heidelberg, 2002, pp. 272–289.

[68] P. Graydon, "Formal assurance arguments: A solution in search of a problem?" in *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*, 6 2015, pp. 517–528.

[69] A. Rae, M. Nicholson, and R. Alexander, "The state of practice in system safety research evaluation," in *5th IET International Conference on System Safety 2010*. IET, 2010.

[70] R. A. Brooks, "Artificial life and real robots," in *ECAL*, F. J. Varela and P. Bourgine, Eds. MIT Press, 1992, pp. 3–10.

[71] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *ECAL*, ser. LNCS, vol. 929. Springer, 1995.

[72] E. A. Lee and M. Sirjani, "What good are models?" in *FACS*, ser. LNCS, vol. 11222. Springer, 2018.

[73] C. P. Bonini, *Simulation of information and decision systems in the firm*. Prentice-Hall, 1962.

[74] G. E. P. Box and N. R. Draper, *Empirical model-building and response surfaces*. Wiley, 1986.

[75] R. F. Paige, "A meta-method for formal method integration," in *Proc. 4th. Intl. Symp. on Formal Methods Europe (FME)*, ser. LNCS, vol. 1313. Springer, 1997, pp. 473–494.

[76] M. Broy and O. Slotosch, "Enriching the software development process by formal methods," in *Applied Formal Methods – FM-Trends 98*, ser. LNCS, vol. 1641. Springer, 1998, pp. 44–61.

[77] A. J. Galloway and B. Stoddart, "Integrated formal methods," in *Proc. INFORSID*. INFORSID, 1997.

[78] E. C. R. Hehner, "Predicative programming," *Communications of the ACM*, vol. 27, no. 2, pp. 134–151, 1984.

[79] ——, "A practical theory of programming," *Science of Computer Programming*, vol. 14, pp. 133–158, 1990.

[80] E. C. R. Hehner and A. J. Malton, "Termination conventions and comparative semantics," *Acta Informatica*, vol. 25, 1988.

[81] J. M. Morris, "A theoretical basis for stepwise refinement and the programming calculus," *Science of Computer Programming*, vol. 9, no. 3, pp. 287–306, 1987.

[82] R. J. R. Back and J. von Wright, "Refinement calculus, part i: Sequential nondeterministic programs," in *Proc. REX Workshop*, ser. LNCS, vol. 430. Springer, 1989.

[83] C. Morgan, *Programming from Specifications*. Prentice-Hall, January 1996.

[84] J. C. P. Woodcock and C. Morgan, "Refinement of state-based concurrent systems," in *3rd Intl. Symp. of VDM Europe*, ser. LNCS, vol. 428. Springer, 1990, pp. 340–351.

[85] A. S. Evans, "Specifying and verifying concurrent systems using z," in *Proc. 2nd. Intl. Symp on Formal Methods Europe*, ser. LNCS, vol. 873. Springer, 1994.

[86] A. Galloway and B. Stoddart, "An operational semantics for zccs," in *Proc. 1st Intl. Conf. on Formal Engineering Methods*. IEEE Computer Society, November 1997.

[87] A. W. Roscoe, J. C. P. Woodcock, and L. Wulf, "Non-interference through determinism," in *ESORICS 94*, ser. LNCS, vol. 875. Springer, 1994.

[88] C. A. R. Hoare, "Unified theories of programming," Oxford Computing Laboratory, Tech. Rep., July 1994, also published by Springer in *Mathematical Methods in Program Development*, 1997.

[89] E. C. R. Hehner and C. A. R. Hoare, "A more complete model of communicating processes," *Theoretical Computer Science*, vol. 26, pp. 105–120, 1983.

[90] M. Oliveira, A. Cavalcanti, and J. Woodcock, "A UTP semantics for Circus," *Formal Aspects of Computing*, vol. 21, pp. 3–32, 2009.

[91] A. Butterfield and P. Gancarski, "The denotational semantics of slotted-Circus," in *Formal Methods 2009*, ser. LNCS, A. Cavalcanti and M. Damm, Eds., vol. 5850. Eindhoven, Netherlands: Springer, November 2009.

[92] K. Wei, J. Woodcock, and A. Cavalcanti, "Circus Time with Reactive Designs," in *Unifying Theories of Programming*, ser. LNCS, vol. 7681. Springer, 2013, pp. 68–87.

[93] K. Kronlöf, Ed., *Method Integration: Concepts and Case Studies*. Wiley, 1993.

[94] S. Dunne and B. Stoddart, Eds., *1st Intl. Symp. on Unifying Theories of Programming*, ser. LNCS. Springer, 2006, vol. 4010.

[95] M. Broy, "Challenges in automotive software engineering," in *Proceedings of the 28th International Conference on Software Engineering - ICSE'06*. ACM Press, 2006.

[96] G. Liebel, N. Marko, M. Tichy, A. Leitner, and J. Hansson, "Model-based engineering in the embedded systems domain: An industrial survey on the state-of-practice," *Software & Systems Modeling*, vol. 17, no. 1, pp. 91–113, mar 2016.

[97] M. Broy, "Yesterday, today, and tomorrow: 50 Years of software engineering," *IEEE Software*, vol. 35, no. 5, pp. 38–43, September/October 2018.

[98] I. Habli, I. Ibarra, R. Rivett, and T. Kelly, "Model-based assurance for justifying automotive functional safety," in *Proc. SAE World Congress*, April 2010.

[99] R. Hawkins, I. Habli, D. Kolovos, R. Paige, and T. Kelly, "Weaving an assurance case from design: A model-based approach," in *Proc. 16th IEEE Intl. Conf. on High Assurance Systems Engineering (HASE)*. IEEE, 2015.

[100] A. Moore, T. O'Reilly, P. D. Nielsen, and K. Fall, "Four thought leaders on where the industry is headed," *IEEE Software*, vol. 33, no. 1, pp. 36–39, jan 2016.

[101] P. G. Neumann, *Computer-related Risks*. NY, USA: Addison-Wesley, 1995.

[102] L. Hoffmann, "Promoting common sense, reality, dependable engineering," *Communications of the ACM*, vol. 61, no. 12, pp. 128–ff, nov 2018.

[103] D. G. Feitelson, "Tony's law," *Commun. ACM*, vol. 62, no. 2, pp. 28–31, Jan. 2019.

[104] D. L. Parnas, "The real risks of artificial intelligence," *Communications of the ACM*, vol. 60, no. 10, pp. 27–31, sep 2017.

[105] S. Ben-David, P. Hrubeš, S. Moran, A. Shpilka, and A. Yehudayoff, "Learnability can be undecidable," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 44–48, jan 2019.

[106] B. Randell, "Fifty years of software engineering - or - the view from garmisch," *CoRR*, vol. abs/1805.02742, 2018, iCSE keynote speech.

[107] C. M. Holloway and C. W. Johnson, "How past loss of control accidents may inform safety cases for advanced control systems on commercial aircraft," in *3rd IET International Conference on System Safety*, Oct 2008, pp. 1–6.

[108] D. MacKenzie, "Computer-related accidental death: an empirical exploration," *Science and Public Policy*, vol. 21, no. 4, pp. 233–248, aug 1994.

[109] M. Gleirscher, "Behavioral safety of technical systems," Dissertation, Technische Universität München, 2014. [Online]. Available: http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20141120-1221841-0-1

[110] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" RAND Corp., Research Report RR-1478-RC, 2016.

[111] A. Hopkins, "Quantitative risk assessment: A critique," Australian National University, Working Paper 25, 2004. [Online]. Available: http://regnet.anu.edu.au/sites/default/files/publications/attachments/2015-05/WorkingPa

[112] D. Yang, "Hazards from high system entropy: An explorative analysis of case reports," Technical University of Munich, Master's thesis, 2016.

[113] P. Koopman, "A case study of Toyota unintended acceleration and software safety," 2014. [Online]. Available: https://www.slideshare.net/PhilipKoopman/toyota-unintended-acceleration

[114] C. Hoare, "The Emperor's Old Clothes," *Communications of the ACM*, vol. 24, no. 2, pp. 75–83, 1981, the 1980 ACM Turing Award Lecture.

[115] T. Lozano-Perez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *CACM*, vol. 22, no. 10, pp. 560–570, 1979.

[116] M. Meng and A. C. Kak, "Mobile robot navigation using neural networks and nonmetrical environment models," *IEEE Control Systems*, vol. 13, no. 5, October 1993.

[117] Health and Safety Executive, *Out of Control*. HSE Books, 2003. [Online]. Available: http://www.hse.gov.uk/pubns/priced/hsg238.pdf

[118] J. C. Knight, "Safety critical systems: Challenges and directions," in *Proceedings of the 24th International Conference on Software Engineering*, ser. ICSE '02. New York, NY, USA: ACM, 2002, pp. 547–50.

[119] J. Knight, "Safety standards – A new approach," in *System Safety Symposium*, 2014. [Online]. Available: https://scsc.uk/r126/1:1

[120] R. R. Lutz, "Analyzing software requirements errors in safety-critical, embedded systems," in *IEEE Int. Symp. Req. Eng.* IEEE, 1993, pp. 126–33.

[121] A. Endres, "An analysis of errors and their causes in system programs," *IEEE Transactions on Software Engineering*, vol. SE-1, no. 2, pp. 140–149, 6 1975.

[122] Motor Industry Research Association, *Development Guidelines for Vehicle Based Software*. Motor Industry Research Association, 1994. [Online]. Available: https://www.misra.org.uk

[123] P. B. Ladkin, "Root Cause Analysis: Terms and definitions, AcciMaps, MES, SOL and WBA," University of Bielefeld, Tech. Rep., 2013. [Online]. Available: https://rvs-bi.de/publications/Papers/LadkinRCAoverview20130120.pdf

[124] ——, "Standards for standards improving the process," 2013. [Online]. Available: https://rvs-bi.de/publications/WhitePapers/RVSsfssPrinciples.pdf

[125] B. Kitchenham and S. L. Pfleeger, "Software quality: The elusive target," *IEEE Software*, vol. 13, no. 1, pp. 12–21, 1996.
[126] J. Bowen and V. Stavridou, "Safety-critical systems, formal methods and standards," *Software Engineering Journal*, vol. 8, no. 4, p. 189, 1993.
[127] D. Harel and B. Rumpe, "Meaningful modeling: What's the semantics of "Semantics"?" *IEEE Computer Society*, vol. 37, no. 10, October 2004.
[128] P. Mohagheghi, W. Gilani, A. Stefanescu, and M. A. Fernandez, "An empirical study of the state of the practice and acceptance of model-driven engineering in four industrial cases," *Empirical Software Engineering*, vol. 18, no. 1, pp. 89–116, 1 2012.
[129] M. H. B. Broy, M. Feilkas, M. Herrmannsdoerfer, S. Merenda, and D. Ratiu, "Seamless model-based development: From isolated tools to integrated model engineering environments," *Proceedings of the IEEE*, vol. 98, no. 4, pp. 1–21, 2010.
[130] M. Gleirscher, D. Ratiu, and B. Schätz, "Incremental integration of heterogeneous systems views," in *1st Int. Conf. Systems Engineering and Modeling, ICSEM 2007, Herzliyya-Haifa, Israel, March 20-23, 2007*, 2007, pp. 50–9.
[131] P. Stevens, "Is bidirectionality important?" in *ECMFA*, ser. LNCS, vol. 10890. Springer, 2018.
[132] P. H. Feiler, B. Lewis, S. Vestal, and E. Colbert, "An overview of the SAE architecture analysis & design language (AADL) standard: A basis for model-based architecture-driven embedded systems engineering," in *IFIP The International Federation for Information Processing*. Springer-Verlag, 2004, pp. 3–15.
[133] V. Debruyne, F. Simonot-Lion, and Y. Trinquet, "EAST-ADL — an architecture description language," in *IFIP The International Federation for Information Processing*. Springer-Verlag, 2004, pp. 181–195.
[134] B. Ramesh and M. Jarke, "Towards Reference Models for Requirements Traceability," *IEEE Trans. Soft. Eng.*, vol. 27, no. 1, pp. 58–93, January 2001.
[135] J. Whitehead, "Collaboration in software engineering: A roadmap," in *Future of Software Engineering (FOSE'07)*. IEEE, may 2007.
[136] Mendez-Fernandez, Penzenstadler, Kuhrmann, and Broy, "A meta model for artefact-orientation: Fundamentals and lessons learned in requirements engineering," in *MODELS'10*, 2010.
[137] M. Broy, "A logical approach to systems engineering artifacts: semantic relationships and dependencies beyond traceability—from requirements to functional and architectural views," *Software & Systems Modeling*, vol. 17, no. 2, pp. 365–393, sep 2017.
[138] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2009.
[139] D. Akdur, V. Garousi, and O. Demirrs, "A survey on modeling and model-driven engineering practices in the embedded software industry," *Journal of Systems Architecture*, vol. 91, pp. 62–82, nov 2018.
[140] V. Aravantinos, S. Voss, S. Teufl, F. Hölzl, and B. Schätz, "Autofocus 3: Tooling concepts for seamless, model-based development of embedded systems," in *ACES-MB&WUCOR@MoDELS*, ser. CEUR Workshop Proceedings, vol. 1508. CEUR-WS.org, 2015, pp. 19–26.
[141] F. Huber, B. Schätz, A. Schmidt, and K. Spies, "AutoFocus — a tool for distributed systems specification," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1996, pp. 467–470.
[142] C. Flanagan, K. R. M. Leino, M. Lillibridge, G. Nelson, J. B. Saxe, and R. Stata, "Extended static checking for java," in *Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation - PLDI'02*. ACM Press, 2002.
[143] R. Jeffery, M. Staples, J. Andronick, G. Klein, and T. Murray, "An empirical research agenda for understanding formal methods productivity," *Information and Software Technology*, vol. 60, pp. 102–112, apr 2015.
[144] G. Rosu and T. F. Serbanuta, "An overview of the K semantic framework," *JLAP*, vol. 79, no. 6, 2010.
[145] A. Darbari. (2018) The budget case for formal verification. [Online]. Available: http://www.techdesignforums.com/practice/technique/the-budget-case-for-formal-verification/

APPENDIX

A. Formal Inspection versus Formal Methods in Practice

Formal Inspection (FI) encompasses a variety of techniques (e.g. peer reviews, walk-throughs) where critical process artefacts (e.g. program code) are checked (e.g. manually or using software tools) against a variety of criteria (e.g. checklists), usually by a group of independent qualified engineers. For sake of simplicity of the following discussion, we assume that FMs can be seen as a particularly rigorous variant of FI where formal specifications serve as a particular way of formulating checklists.

Now, we compare the use of FMs with the use of FI. According to [45, Sec. 4.4], formal inspections are used in more than 35% of commercial defence, systems, and embedded software projects, and FMs are estimated to be applied in less than 1% of overall commercial software engineering projects. To get an idea of these coverage data, we perform an analysis of the global embedded software market based on other global software market indicators in Table II. We found estimates of systems and software professionals world-wide and estimates of annual US business values.[20]

A uniform distribution would entail roughly 37000 USD/year per person in the general software domain and 10000 USD/year per person in the embedded software domain. Clearly, geographically strongly differing salaries and part-time engagement rule out a uniform distribution, yet providing figures helpful for our purposes.

Next, we apply the following proportions: From a world-wide population of around 18.5 million software developers in 2014, about 19% live in the US, 10% in China, 9.8% in India, 36% Asia/Pacific region, 39 % live in Europe, the Middle East, and Africa; and 30% in the Americas.[21] The *design to quality assurance* (i.e., verification and test) *cost ratio* is observed to be approximately $30 : 70$.[22] About 20% of embedded software personnel are quality assurance engineers (i.e., test, verification, or validation engineers).[23]

The estimates in Table II suggest that around 2% of the overall pure software market are allocated to the embedded pure software market. 35% coverage of formal inspection in about 13.5% of the overall software market $(161/(689 + 515) = 0.134)$ would result in roughly 4.7% coverage of all software projects by formal inspection versus at most 1% coverage by FMs. However, from these data we can hardly know whether rates of FM use get close to or beyond 10% in high-criticality systems projects.

Assuming that in about 35% of embedded software projects the quality assurance personnel would use formal inspection and that in every fifth $(1 : 4.7)$ of such projects formal methods would be used, the current population of regular practical FM users would globally amount to about 5040 $(= 72,000 * 0.34 * 0.20)$ persons. Note that these numbers are rough estimates. However, we believe their order of magnitude is realistic. Moreover, given that these persons would on average earn about 100,000 USD/year each, we would speak of round USD 504 million of annual business value.

Importantly, from these data we can determine minimum sample sizes for surveys. For example, assume we want to have 95% confidence in our test results and are fine with a

[20] See https://en.wikipedia.org/wiki/Software_industry and https://softwareengineering.stackexchange.com/questions/18959/what-proportion-of-program
[21] See https://adtmag.com/Blogs/WatersWorks/2014/01/Worldwide-Developer-Count.aspx.
[22] See https://www.slideshare.net/pboulet/socdesign.
[23] See https://de.slideshare.net/vdcresearch/searching-for-the-total-size-of-the-embedded-so

TABLE II
DATA FOR THE ESTIMATION OF THE SIZE OF THE FORMAL INSPECTION AND FORMAL METHOD MARKET

Global market/project indicators [Unit]	Professional engineers / developers [million]	Ann. business value [billion USD/year]	Quality assurance personnel [million]	QA business value [billion USD/year]	FI [%]	FM [%]	Devices [billion/year]
General IT hardware and devices (incl. personal computers)		2018: 689					2010: 10
Embedded systems (hardware, software, connected embedded devices) in all domains	2014: 1.2	2009: 88 2018: (161)[a]			(35)		2010: 9.8
Industrial embedded systems							2016: 2
Defence, systems, and embedded commercial software engineering					2011: 35		
Embedded software	2014: 0.36	2009: 3.4 2018: (10)	**2010: (0.072)**	**2010: (2.38)**			
General software (overall commercial software engineering)	2014: 11 US: 2.1 (19%)	2013: 407 2018: (515)			(4.6)	2011: 1	

[a] The numbers in parentheses include estimates for 2018 based on the other numbers and corresponding average growth rates.

TABLE III
OVERVIEW OF OUR GENERAL SWOT ANALYSIS (ACCORDING TO [31]) OF "FMS FOR PRACTICAL DEPENDABILITY ASSURANCE"

Method Strengths:
- Improvement of modelling precision, requirements clarity, verification confidence
- High error detection effectiveness, early error removal

Community Strengths:
- Many transfer re-entry points from former case studies with industry
- Many dependable systems practitioners perceive FM usefulness as positive

Method Weaknesses:
- Difficult to learn and apply, many dependable systems practitioners perceive ease of use of FMs as negative
- Fragile effectiveness and productivity

Community Weaknesses:
- Lack of compelling evidence of FM effectiveness
- Ineffectively communicated in teaching and training

Key Opportunities for iFM Transfer and Research Progress:
- Desire for early removal of severe errors (Section IV-A)
- Desire to learn from accidents and their root causes (Section IV-B)
- Desire to be a mature discipline (Section IV-C)
- Desire for dependable norms (Section IV-D)

Method Threats:
- Lack of method scalability
- Faulty, tedious, or vacuous proofs
- Lack of user education
- Poor tool integration, legacy tools and processes

Transfer Threats:
- Lack of researcher engagement in FM transfer
- Lack of access to comprehensive high-quality data

confidence interval of $\pm 7\%$. Then, for *regular practical FM users*, a population of the size of 5040 persons would require us to sample 189 independent data points (e.g. questionnaire responses). The population of *regular practical FI users*, 25200 ($= 72,000 * .35$) would imply a minimum sample size of 194. For any sample of such size, any survey has to argue why the sample *represents* the population. This step depends on the possibilities given during the sampling stage. Obviously, reaching out to 189 out of 5040 persons whose locations might be largely unknown is an extremely difficult task that might only be tackled in terms of a global group effort among researchers. Overall, these figures suggest that it is realistic to run surveys for the collection of confident evidence.

B. Formal Methods for Dependable Systems Practice

As depicted in Figure 1, Table III provides a more general SWOT analysis than the one shown in Table I.