On the Heroism of really Pursuing Formal Methods

(Title inspired by Dijkstra’s “On the Cruelty of really Teaching Computing Science” [1])

Dino Mandrioli
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano
Milano, Italy
Dino.Mandrioli@polimi.it

Abstract—Formal methods have been “preached” as a means to achieve better reliability and other qualities in software and systems for half a century. Despite many success stories confirmed their effectiveness, there is still much reluctance (not only) in the industrial world to apply them on a large scale. In this paper I add my modest personal view to the rich and long-standing debate on the reasons of this apparent failure; I examine the main challenges to face and the most promising symptoms to hope into an improved state of the affairs.

Index Terms—Half-empty-half-full glass, incrementality, model-checking, education.

I. INTRODUCTION

It is customary to associate the birth of Formal Methods (in Computer Science) with the pioneering work by John McCarthy [3]. At that time McCarthy’s and others’ hope/prophecy was that in a decade or so most software should/would have been delivered together with a formal certification of its correctness. We all know that the hope soon turned into a hype and after more than half a century a small community of Formal Methods (FMs) heroes is still struggling to promote wider adoption of FMs within the development of industrial systems. In such a time the community made an impressive effort to develop, promote, and apply FMs in practice but nevertheless the community is still small, compared to other fields of computer science and is still “preaching in the desert.”

Needless to say much effort has also been devoted to understand the reasons for such difficulties, to find appropriate remedies, to provide evidence of the benefits offered by FMs (see, e.g., [2,6,7]); thus there is little I can add to what is already widely available and has been already tried in previous efforts; so I will only try to confront with you my personal view and interpretation of the state of the art and of the perspectives of FMs with no claim of systematic analysis (only personal, anecdotic experience) and no “miracle recipe” to revolutionize our world. I will illustrate my feelings at the light of the “half glass” analogy: the pessimist sees a half-full glass as almost empty, whereas the optimist sees the same glass as almost full.

II. THE HALF-EMPTY GLASS

There is a general consensus that perhaps the most serious obstacle to achieve a wide application of FMs is of cultural and educational nature: the mathematical background needed to master typical computing engineering FMs is much less taught in normal schools and universities than continuous mathematics which is exploited in traditional engineering from centuries: whether it is due to historical traditions and habits or to some intrinsic conceptual difficulty, it is a fact that many students learn more easily and more willingly to compute derivatives and integrals (I deliberately emphasize “computing” as opposed to “understanding”) than to manage (possibly nested) quantifiers in logic formulas. This situation certainly does not improve if we refer to less young people who already work from years in any field of engineering.

I believe, however, that if we “zoom out” from this particular technical case and we examine the attitude of the average students towards mathematics in general, we find a general “social” attitude against rigorous reasoning, and, with a further “zoom out”, against spending effort to obtain a good result as opposed to saving time and effort and obtaining just any result. “Time to market” is the overwhelming goal and quality is secondary. The market itself favors such an attitude in a vicious circle since, “in the small”, we see the long midnight lines of people eager to buy the last smartphone, but, “in the large” huge contracts are signed selling technology that, at best, is not yet “mature”, with the unavoidable consequences of penalties and, often, bankruptcies.

This general “sociological laziness” towards rigorous reasoning and mathematics, which is its natural “support”, exhibits some distinguishing peculiarities in the context of SW engineering. Thanks to its natural malleability and incrementality—which are certainly useful qualities—SW tends to give a false impression of ease and immediateness; this impression is often encouraged by teachers to attract their students who, naturally, wish to achieve “tangible results” as soon as possible: they are so happy to run the typical “hello world” program at their first class in Computer Science; they write loops but quite rarely they realize–and are taught–that behind any loop there is an invariant which explains its rationale; they feel happy enough, in doing their homeworks, as soon as they come up with “anything running”.

1
Software has the nice but dangerous property that, up to some point, it can be learned almost exclusively “by doing” rather than “by studying”. It is certainly good that anybody learns easily to use spreadsheets, word processors, browsers, search engines, etc.; this apparent ease, however, becomes a risk when in the realm of professional engineering: many non-computer-engineers are convinced that they can themselves build the software that implements the control of their robots, plants, trains, etc. … with results which can be easily measured … after a while.

What do we – the computing and software engineering teachers – do in this context? We (I mean: the majority of us, certainly not me!) advocate “lego teaching” so that students can have the satisfactory feeling of “having built something that works in the first day of school”; for instance, nowadays many teachers even offer their students – in the first class in Computer Science – the chance to build their first video-game or app; a similar attitude is often advocated in many papers recommending to “revolutionize” the teaching of basic Computer Science: again, “learning by doing” overwhelming “learning by studying” (I too, when a kid, liked very much playing with lego and similar “boxes”, but I never thought I was learning engineering).

We avoid going to teach “real computer science” in the “Bronx” of industrial engineering schools (and complain when our colleagues in those schools claim that they are able to teach what their students need about programming better than we do); we claim that teaching “soft skills” is more important than teaching the “hard ones”; some colleagues even oppose creative reasoning against rigorous reasoning, just as the two things were contradictory; when unavoidably teaching some mathematical machinery we try to avoid or to hide excessive technical difficulties, we propose “copy and paste exercises” at the exam, because if students are unhappy with those difficulties, then their evaluation of our teaching may be too low … and this may have consequences on our career; etc. In summary, there is a long term vicious circle of mutual feedback between social behavior and educational system which hampers rigorous reasoning: no wonder managers are at best reluctant at investing in applying FMs in their industries.

Side remark: FMs are just the “vanguard” of a more general attitude towards SW engineering best practices: still now how many industries just adopt, say, UML, systematic testing, etc.? Curiously enough many SW engineers have towards FMs the same prejudices as other engineers have towards SW engineering recommendations.

Particularly discouraging is the fact that I do not see symptoms of a change of tendency. On the contrary, consider the history of programming languages (PLs), their definition and implementation. No doubt, technically, this field tells one of the major success stories of FMs, starting from the advent of formal language and automata theory, through deterministic parsing up to formal semantics and, occasionally, fully automatic compilation. Even if not all the conceptual milestones of PLs history have enjoyed wide success and industrial usage, phenomena such as Pascal and Ada itself are in my opinion good examples of the quality that can be obtained by rigorous application of sound principles and suitable formalisms; Java too has been fully formalized [8] though I cannot assess the impact its formalization had on its development. The scenario of modern PLs instead is a considerable step backwards and we witness a plethora of widely used languages (Python, Perl, JavaScript, just to mention a few of them) which just “put together” a collection of unordered constructs with no rational design … and regularly, after a while, we see plenty of highly critical reviews on their features (see [14] for a deeper analysis of the case of PLs.)

As a totally different recent experience let me mention that among 157 project proposals by young researchers to be funded by the Italian government, only 4 were self-classified as “theoretical” and all of them where narrowly focused on cryptography and security; none of 157 proposals had a minimum reference to formal system analysis and verification. Are FMs heroes extinguishing? Certainly, even within academia, the environment is not quite friendly: think, e.g., to the crazy “number game” of bibliometric parameters [9].

Certainly our community has its own faults for this unpleasant half-empty glass. Major mistakes were:

- Overselling FMs promises (hopes and hypes): guaranteed SW cannot be obtained magically whichever method is adopted for its development.
- Overlooking the difficulties of technology transfer.
- Too little attention (at least initially and at least from many FM professionals) to the peculiar needs of the various application fields and to the managerial and political aspects that are always dominant in strategic industrial decisions.
- Too much love for the mathematical elegance of formalisms which unavoidably led to subjective preferences and useless debates about reciprocal qualities and defects.
- The lack of well-engineered tools supporting the application of methods to large non-toy systems.

These and various other mistakes were more or less promptly recognized and many of us tried hard to recover therefrom: see, e.g., [7, 2] for the general consciousness of the community and [10] for a personal, dated, but perhaps still valid, experience. “Many” success stories of practical application of FMs to real-life systems have been written but such “many” ones are still quantitatively negligible w.r.t. the present state of SW development.

In summary the glass of FMs is still far from full.

III. THE HALF-FULL GLASS

Nevertheless, the glass can also be seen as partially full. First the success stories mentioned above, though numerically few if compared with most industrial SW, showed unequivocally that the promises offered by FMs can be kept: in general, systems developed by exploiting FMs are more
reliable, exhibit fewer errors during verification, and most often require shorter times and less economic resources. Only few exceptions of failure of FM-based projects are reported [4].

Also, a major agency such as NASA is now fully committed to exploit FMs and officially reports satisfactory usage thereof [12]. More generally, there is an increasing agreement that, with the increasing pervasiveness of SW, the increasing demand for higher and higher reliability standards can be satisfied only by resorting to the precision and rigor of mathematical modeling and analysis. This mostly applies to safety and security critical systems such as avionic, automotive, e-health, energy production and distribution, banking … ones, but is gaining more and more impact also on our every-day life.

It is certainly good news that our heroic community survived, despite the difficulties mentioned above, … and that it is welcome by the larger SW engineering one at least in this workshop. It is also part of the half-full glass the fact that several Turing prizes have been awarded to major contributors to the field of FMs.

A major point in favor of FMs within the larger computing community has been scored by the advent of model checking. No doubt model checking produced a major breakthrough in the field of automatic, guaranteed verification and ignited much intriguing research to widen at most the frontiers (and the limits: see the challenges launched by Vardi [11] and Leyton-Brown et al. [13]) of its applicability. Certainly model checking produces a strong attractiveness toward (lazy) practitioners thanks to its “push button promise”. Going back for a short while to the half-full glass, however, it would be wise to avoid falling again from too much hope back to hype: the intrinsic complexity challenges of most model checking algorithms have still to be properly addressed and an industrial practitioner could hardly accept to push the button of a model checker without knowing a priori whether it will run for a few seconds, minutes or weeks, … or it will run out of time and/or memory, no matter how powerful the running HW; I will get back to this issue in Section V.

IV. SO WHAT?

It would be unfair to close such a paper without any indication to help moving towards a full glass; but it is also difficult, after several decades of hopes, hypes, attempts, novelties, failures, self-criticism, recommendations and commandments … to find any new recipe or simply suggestion to help the FM community to help the society to build more reliable, higher quality systems. Most of such recommendations are valid and effective, at least in the long term; thus, let me simply recall the recommendations that I believe most important on the one side, and, on the other side, suggest a few ones that are more controversial within the general computing community and perhaps even within our own one.

- Apply incrementality at any level: industrial environments, unlike what they claim in their commercial ads, are very conservative and quite reluctant to apply radical changes –and, in most cases, this is a wise attitude. Thus, do not pretend to obtain the full potentiality of a method in an environment which is totally unaware thereof; rather design a step by step strategy: often a rigorous use of a semiformal notation such as UML (syntax) or so-called lightweight FMs [5] are already an important improvement w.r.t. existing practices; in a further step a good formal specification paired with informal verification can be better “digested” than pretending formal verification paired with specification, etc.

- Offer much user friendliness. Do not pretend to create familiarity with cumbersome mathematical notations, rather provide natural user interfaces; often graphical “state-based” notations are better understood and welcome than formulas with nested levels of quantifiers. Strictly connected with this requirement is the usefulness of reliable, effective, and user-friendly tools supporting whichever method in whichever non-toy application. The recent tendency to avoid developing new tools from scratch in favor of augmenting few widely known and robust ones (e.g. core theorem-provers and/or model-checkers) with suitable external interfaces deserves being further pursued.

- When (trying to) introduce a FM within an industrial environment be mainly a teacher and a promoter rather than a pure researcher: try to present already well-established practices rather than following intriguing but not mature ideas; on the other hand, personally, I often went back to be a researcher as a … “student”, i.e., by learning problems, needs, and practices of the environment I was interacting with, and using them as a source of inspiration to improve, tailor to specific needs existing methods, and possibly even invent new ones. There is a kind of beneficial mutual feedback between transferring research results in industrial practices and taking inspiration therefrom to formalize their problems and invent suitable solutions. Mixing up these two activities, however, should be done with much care or avoided at all: industrial processes in most cases cannot afford the risks that are typical of research; they are confronted with strict deadlines and must be highly predictable.

This is certainly no to say that research should be carried over exclusively within academia and then transferred to industry in a “one-way” style: just the opposite, integrated teams are often very productive and offer much fun to their members … provided that the managers are well aware of the difference between looking for original, “new generation”, solutions and guaranteeing well-planned and timely results.

As a side remark, based exclusively on personal experience, it may also happen that industrial people, once involved in some research activity, get even “too enthusiastic” and keep looking for, or proposing, ever new ideas, techniques, even notations, thus hampering real advancements: in such cases a curious change of roles may occur.
All these and various other recommendations recalled, I maintain my position that the major hurdles against a wider and more convinced adoption of FMs are of social and educational nature, and this leads us back to the empty part of the glass since these aspects are only partially (and this is an euphemism!) under control of our community.

From a political point of view, let aside ethical aspects of human social behavior, which however do have an impact on the attitude towards rigorous reasoning and therefore towards accepting mathematical formalisms to support it, standards are a typical means that social organizations adopt to (try to) enforce acceptable levels of safety, security, interoperability. Standardization however, are unavoidably influenced by often contrasting political and/or commercial interests, so that in most cases, rather than producing precise and rigorous choices derived from sound and well-thought principles they deliver compromises which at least partially satisfy the interests of the various stakeholders. In principle, FMs are a natural means to produce precise and safe standards in most application fields, but is this what the powerful stakeholders involved really want? How much power has our community to influence such “top-level strategic decisions?”

Our power should increase in the education system, of which we are a natural part. Here too, however, I see a few major difficulties. As I mentioned above, I see a general tendency towards “soften” the teaching of engineering principles; this requires a certain amount of “heroism” to convince students that not everything can be obtained without effort. I see however, another more subtle enemy of rigorous reasoning in engineering disciplines which I found in most communities, including our own. Despite almost everybody advocates interdisciplinarity in the teaching of engineering disciplines, the way through which this buzzword is interpreted is often paired with a sort of NIMBY attitude: “my discipline is the core business and must be taught in depth; other ones are useful soft skills which can be learned with little effort”. As a consequence we see mechanical engineers who pretend to be able to teach by themselves just the “little of computing that is necessary for their students”; conversely, many SW engineers claim that their students do not have to waste their time with the principles of mechanics and electromagnetism.

Are we sure that the same attitude does not manifest itself even within our own teaching of FMs? Isn’t it that, when teaching FMs we actually present only or mainly our own favorite one focusing more on its peculiarities than on the general principles that it shares with others? In my strong opinion rigorous reasoning can be better developed if applied to many fields from philosophy to social sciences, from mechanical to SW engineering, down to Petri nets or temporal logics.

Software now pervades and impacts on practically every aspect and instant of our life; thus, SW engineers should be ready, more than others, to understand the problems, the methods and the solutions of fields of science other than their own. For the same reason, FMs, as a major tool to achieve high reliability and trustworthiness in any critical system should be first of all a mental attitude of any engineer and FMs specialists should in turn be able to interact in depth –and breadth– with any other engineer.

Am I advocating a sort of encyclopedic teaching? Of course not, but specialization should deserve lower priority than breadth of cultural coverage … in SW engineering and FMs too.

V. A FEW WORDS ABOUT RESEARCH

In this paper I deliberately have focused on the “teaching” aspects –in a generalized sense– of FMs for two main reasons: I believe that their future depends much more on convincing more and more people of their usefulness than on producing more technical novelties; this of course does not negate that research remains always the fundamental locomotive which pulls the train of technological progress. The other reason is that, simply, I am not so presumptuous to pretend to propose directions towards which addressing further research effort. Rather, I feel myself as an interested listener –in this workshop– to better understand what is going on in our field.

I easily avoid talking of my present research goals, which my coworkers and I can document elsewhere; rather, I want to mention my surprise in realizing how recently computational complexity problems are dealt with in the context of FMs (and not only). Up to a few years ago I was used to think of NP-complete problems as those beyond the barrier of computational tractability. It is a fact, however, that many tools operating in practice, such as SAT- or SMT-based solvers, “often” deliver their results in “acceptable time”; the terms “often” and “acceptable time”, however, are deliberately in quotes. As I said, this reminds me the intriguing editorial by Moshe Vardi [11] which wished more foundational research to fill up the gap between the typical worst case analysis of many algorithms and the fact that “in practice they often run in acceptable time”. For instance, it is juts folklore that the simplex algorithm has been –and still is– used satisfactorily for decades without even knowing that the linear programming problem it solves is NP-complete ... but a fairly atypical one. As an external observer, instead, I see that many recent papers focus their attention in positioning a given formalism –typically, the satisfiability of its formulae– in the class NP, rather than PSPACE or EXPTIME, and more. I certainly agree that such a classification has a high theoretical value, and possibly even a practical impact; given that anyway problems in these classes have intolerable worst-case performances; however, I wonder whether this type of investigation should be the main theoretical interest for many of us. Certainly, on the opposite side many “theorems” are paired with serious empirical evaluation, benchmarks, heuristics, etc.; on the other hand, I often witnessed the “expert” of a given tool explaining the novice that he had to abort the run because he wrote the input formula with an inappropriate style, e.g., with explicit quantifiers; or that the tool he chose is not adequate for the problem he wants to solve despite that, in principle, the problem is in the family of solvable problems.

I agree with Vardi that, if we really wish that present tools are well accepted and used in industrial environments as real “push-button” ones, we need more efforts to fill up the gap
between the traditional worst-case, or even statistical, analysis and the empirical folklore on the –relative– performances of present state-of-the-art tools. In other words, it would be nice if some “scattered wisdom” available in some blogs or Wikipedia pages, or through informal consultancy with experts and gurus, were digested into a systematic and well-structured theory.

VI. CONCLUSION

In a highly competitive world, where even academia is importing attitudes and rewarding methods typical of business and commercial endeavor, seriously pursuing FMs requires a good amount of heroism. Unlike “superheroes” of the movies, unfortunately, real heroes accept to fight for a right and commendable objective even in an inferiority situation and even if they know that most likely their reward will not compensate the effort. But just for this reason a possible even partial victory will be more valuable for them.

FMs heroes must combat on several fields: not only they should produce novel and valid research; they should also address their teaching (and technology transfer) in a way that is not shared by the majority of colleagues –including, unfortunately, many SW engineers–; they should look for allies among various policy makers, including politicians, funding agencies, etc. A few important allies, however, already exist as in fact our glass is also half-full and the hope that, thanks to further heroism, the glass will be filled up is still alive.

ACKNOWLEDGMENTS

A grateful thought to the pioneers of FMs and to the more recent heroes who keep our flag high, including those who maintain a workshop such as Formalise in the context of ICSE and, in general, the great FME group. Thanks to Matteo Pradella for hints and advises about several points touched in this paper.

REFERENCES