Applying Concurrency Research in Industry Report on a Strategic Workshop

Luca Aceto* Jos Baeten† Wan Fokkink‡
Anna Ingolfsdottir* Uwe Nestmann§

Abstract

This article summarizes the main discussion points that were raised during the presentations at the Workshop on Applying Concurrency Research in Industry, co-located with CONCUR 2007 in Lisbon, and the ensuing panel discussion. It also recalls some of the questions that the audience asked the panel members at the workshop and their answers, and presents the views provided by other experienced members of the concurrency theory community.

1 Prologue

As some of our readers may know, we organized the Workshop on Applying Concurrency Research in Industry, co-located with CONCUR 2007 in Lisbon, on behalf of IFIP WG1.8 "Concurrency Theory". The event was held on the afternoon of Friday, 7 September 2007, and ran concurrently with the last two sessions of the main conference.

The aim of the workshop was to highlight the challenges that arise in applying concurrency-theory research in an industrial setting, broadly construed. The event was meant to be a forum for the discussion of the state of the art in the transfer of results from concurrency theory to industry, and for distilling the lessons to be learned from the successes and failures so far. Moreover, we intended to discuss, e.g., how to increase the impact that concurrency research can have in industry, the role of software tools in this technology transfer effort, and what are possible novel industrial application areas of concurrency theory

*School of Computer Science, Reykjavík University, Kringlan 1, 103 Reykjavík, Iceland. Email: luca@ru.is, annai@ru.is.

†Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands. Email: josb@win.tue.nl.

‡Vrije Universiteit Amsterdam, Department of Computer Science, Section Theoretical Computer Science, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands. Email: wanf@cs.vu.nl.

§TU Berlin (TUB), Fakultät IV (EECS), Franklinstrasse 28/29, D-10587 Berlin, Germany. Email: Uwe.Nestmann@TU-Berlin.DE.

research. The ultimate goal of the meeting, and subsequent discussions, will be to establish a road map for the concurrency-theory community, or parts thereof, in applying its research in industrial settings.

In order to fulfill our aims at least in part, for the benefit of the concurrency-theory community as a whole, in this article we will try to summarize the main discussion points that were raised during the presentations at the workshop and the ensuing panel discussion. We will also recall some of the questions that the audience asked the panel members and their answers. We also present the answers to those questions that we received by other experienced members of the concurrency-theory community. We hope that some of our readers will feel enticed to contribute their opinions on the discussion items and to give their own answers to those questions. A live record of the answers that we will receive will be maintained on the blog of WG1.8.

2 Report on the Invited Presentations

The workshop featured four invited presentations delivered by Hubert Garavel, Vincent Danos, Kim G. Larsen and Jan Friso Groote. A cursory look at the titles of those presentations reveals the importance that the invited speakers attach to tool development in applying concurrency theory in industrial settings. Indeed, Hubert Garavel, Jan Friso Groote and Kim G. Larsen have been amongst the prime movers behind the long-term development of tool sets, which have now matured to a point that they can be applied to tackle non-trivial problems. We find it instructive to report on the main messages of their talks, and on their views regarding the role of, and the challenges in, tool development.

Hubert Garavel gave an overview of his twenty-year work on the CADP tool set, which he considers to be the oldest concurrency-theoretic tool still in activity. A thread underlying his work in what he called "applied concurrency theory" is that one *must* transfer the main results of our research area to industry and that this is only possible with the development of strong tool support for our methods. He also stated that one of his design principles has been to restrict the tool's functionality for efficiency reasons, and that elegant semantics and efficient execution are two separate, and at times conflicting, issues.

Developing a software tool over twenty years or so involves a large software engineering effort. The software must be maintained and one must strive to ensure that the results of tool development are not lost over the years. One of the issues that was raised during Hubert Garavel's presentation is that the concurrency-theory community has lost over the years a lot of code doing, e.g., bisimulation minimization. Tool developers have simply not been consistent enough in preserving some of our earlier tooling efforts for the present generations of developers. Jan Friso Groote said that current bisimulation minimization algorithms do not scale up to the size of the systems that are currently being analyzed, and asked whether it would be appropriate to rekindle research and implementation efforts on efficient and scalable bisimulation algorithms.

Kim G. Larsen's talk was based on his experience with the ten-year develop-

ment of the Uppaal tool, and reported on the knowledge-transfer activity that is part of his involvement in CISS, the Centre for Embedded Software Systems in Aalborg. Apart from surveying the development of Uppaal and its recent off-springs, Kim G. Larsen's talk sent out the following general messages to the audience.

- Tools are a necessary condition for the successful transfer of concurrency-theoretic ideas in industry. Tool development is labour intensive, and one needs the sustained effort of many people over many years to produce good software tools.

- Academic courses offered to students and to industry practitioners play a key role in the transfer of research in concurrency theory into industrial practice.

- Concurrency researchers should try and target different communities of potential users. It is very hard to foresee from where successful applications of our research results are going to stem.

- A good beginning is useful! Being able to start with a success story may lead to further successes and long-term collaborations. However, one should not assume that a good beginning is a guarantee of a good ending. To wit, Kim G. Larsen recounted the story of the Aalborg collaboration with Bang and Olufsen, which disappeared from sight after Klaus Havelund, Arne Skou and he found and fixed a major error in one of their protocols using Uppaal. See the paper *Formal modelling and analysis of an audio/video protocol: An industrial case study using UPPAAL*.

- The success of CISS shows that several companies are very interested in applying concurrency-theoretic ideas and tools because this reduces time to market and increases the quality of their products.

- The impact of one's work on the application of concurrency-theoretic research in industry is not always directly proportional to the amount of effort one puts into the research work itself. Kim G. Larsen gave the example of the synthesis of software controlling the temperature and humidity in an actual pig stable in Denmark. This software was synthesized using Uppaal Cora in a short time and is actually running to the satisfaction of its users.

- Finally, Kim G. Larsen called for an expansion of the scope for use of concurrency theory. He urged the community to link its work to testing, optimization etc. and to embed it into standard software engineering methodologies, which are familiar to practitioners.

Jan Friso Groote presented an overview of the mCRL2 tool set, and gave a demo of several features of the system. He stated his belief that proving the correctness of systems by hand gives one understanding that mere button-press model checking does not yield. However, one needs tools to transfer knowledge

to industry and to validate one's manual verifications. He also shared Hubert Garavel's belief that the initial emphasis on abstract data types in CADP and μCRL was a mistake, but that formalisms supporting data are necessary in applications. One of Jan Friso Groote's messages was that high-performance in tools is underestimated, and that achieving it requires a lot of subtle work. He also stressed the importance of teaching courses to our students that involve modelling and analysis of real-life industrial case studies. In that way, when students go and work in industry, they will know that there are effective modelling languages and tools that they can use efficiently to check their designs before implementing them. Jan Friso Groote also mentioned the importance of visualization tools.

On the day of the workshop, Vincent Danos had delivered a truly inspiring tutorial on his work on rule-based analysis of biological signalling. Listening to his tutorial, we were left with the impression that concurrency theory may be on the brink of really having an impact in the life sciences, and that experimental biologists might very well use his tools based on concurrency-theoretic ideas. At the workshop, Vincent Danos presented another way in which results from concurrency theory can help experimental biologists in their work. Experimental biology has a huge knowledge representation problem. (Vincent Danos mentioned that there are two papers published in that area each minute!) In his opinion, experimental biologists can/should

- use concurrency-inspired languages to express biological understanding and
- display this information in a wiki-type system.

This will allow them to construct models systematically and to simulate them (exploiting the executable nature of concurrency-theoretic models) well beyond what is possible today by means of in-vitro experiments.

3 The Panel Discussion

The workshop ended with a panel discussion, which was driven by questions to the invited speakers from the audience. Below, we list the questions, together with some of the responses from the speakers. We cannot claim complete accuracy in our rendition of the speakers' opinions, but we hope that we are not misrepresenting their viewpoints and that you will find their answers stimulating.

1. *What is the role/importance of real time in modelling? Does industry want dense-time or discrete-time models?*
    - Kim G. Larsen (KGL): The vast majority of computing systems in operation today are embedded. Proper modelling and analysis of the behaviour of embedded systems requires some representation of the passage of time. However, industry does not really seem to care

whether the model of time used in the models is discrete or continuous. In the analysis of embedded systems, quantitative analysis techniques are needed and I expect that stochastics will play an increasing role in the future.

- Jan Friso Groote (JFG): Basically, industry do not know what they want and there is little point in chasing their ever-changing needs. Modelling and analysis of computing systems should be based on a uniform calculus, which is rich enough to model the problem scenarios at hand. As far as real time is concerned, it should be expressible in the uniform calculus.

2. *How does one involve small- and medium-size companies in collaborations with concurrency theoreticians/practitioners? Does "company size" matter?*

   Both JFG and KGL reported on several examples of interaction with industry where there seemed no relation between "size" and the success of the interaction. KGL described a successful collaboration on testing of GUI applications with a one-person company having basically no technological expertise. This was compared with the collaboration with Bang and Olufsen, which was a failure despite the resounding success of their first cooperation.

   JFG pointed out the successful cooperation with industry within the Laquso laboratory in Eindhoven.

   Hubert Garavel (HG) stated there are three necessary conditions for a successful collaboration with industry. The company should have

   (a) a strong interest in quality,

   (b) a lot of money and

   (c) a formal modelling group in house.

   These criteria are difficult to find in a small- or medium-size company. In particular, the third criterion above is useful because it opens up the possibility of embedding a company engineer within the academic research group.

3. *Is there any need for stochastic and probabilistic modelling in applications? More pointedly, have you met an example that you could not model because your tool does not support stochastic or probabilistic phenomena?*

   There seemed to be a general opinion amongst the panel members that probabilistic modelling is nice, but not necessary. More precisely, none of the speakers had yet met an example that they could not model adequately because their models and tools do not permit stochastic or probabilistic modelling.

   JFG stated that he wants to work on research topics that can have applicability in real-life scenarios. He wants to have interaction with industry

mainly as a way to learn what are the good and the bad aspects of his modelling language and his tools. He feels that one should push for the use of the *same* model for verification and performance evaluation.

4. *How can we, as a community, foster the building of industrial-strength tools based on sound theories?*

   The general feeling amongst the panelists was that our community does not have the infrastructure to support tooling efforts. HG pointed out how the situation is better in France, where the development of tools and languages is supported and recognized by institutions like INRIA. (In hindsight, this is reflected by the success of long-term efforts like those that led to Esterel, Coq and CAML, amongst others.)

   The panelists suggested that conferences and journals should be keener to accept tools and case studies as refereed contributions on a par with papers. JFG pointed out that Science of Computer Programming has now a track devoted to "expositions on implementations of and experiments with novel programming languages, systems and methods". The journal's web page also states that

   > "It must be emphasized that papers describing new software tools of relevance to SCP are welcome under the strict condition that the source code of the tools is open."

   KGL also stated that the TACAS conference series was initiated precisely to give more visibility to tooling efforts in our community.

   There was general agreement that one should carefully consider the "economics of tool development". Tools are difficult to build; attracting users is even more difficult; and maintaining tools requires a lot of time and resources. Our community should create mechanisms for rewarding the work that goes into the building of good tools.

5. *What has concurrency theory offered industry so far? What are the next steps that the concurrency community should take in order to increase the impact of its research in an industrial setting? And what are future promising application areas for concurrency research?*

   - JFG: Theory does not have much to offer to industry. We should probably view concurrency theory as a nice mathematical theory that need not have any real-world application.

     As for what is it that we can do as a CONCUR community to assist in the work on tools, JFG's answer is to make sure that as many students as possible are being taught their use and effective application. One of the biggest problems that we are facing is that far too few people in industry understand what formal methods and their tools can effectively bring to industry. To be on the safe side, they do not see where the techniques are effective and what they offer. They,

however, understand that there are other pressing needs to invest time in.

If we teach formal methods we should teach the most advanced ones that the students can swallow. If they understand the advanced methods, they can certainly apply the more straightforward techniques. Of course the reverse does not hold. Don't teach your students UML and expect them to understand mCRL2. But if you teach them mCRL2, they will not have any conceptual difficulty in applying UML.

- KGL: We should make our techniques fit into the whole system development process. We should also make sure that existing model-based tools that are already in use by industry have much stronger semantic foundations.
- HG: Our community is in trouble! The model of concurrency that is prevalent is industry is Java-like (threads and shared variables). Our foundational beliefs are exotic for industry and message-passing, as used in our process calculi, is not the order of the day. Our major challenge is in pushing the clean model of concurrency we like into industrial usage. Every computer science department in every university should play its part in achieving this aim. Education is one of our best weapons to make industry accept our models and the techniques based upon them.

4 Some Responses to the Questions Raised at the Workshop

After the workshop, we asked several members of the concurrency-theory community with a strong interest in the application of concurrency in an industrial setting to provide their answers to the questions that the audience asked the invited speakers. We are most pleased to provide the answers that were provided by Rance Cleaveland, Joost-Pieter Katoen, Frits Vaandrager and Moshe Vardi.

Rance Cleaveland

1. *What is the role/importance of real time in modelling? Does industry want dense-time or discrete-time models?*

   In my experience with Reactis, my company's testing and verification tool, Reactis customers absolutely need real-time support. This is due to their applications, which are in automotive and aerospace, and their focus on developing embedded control software. The most widely used commercial modelling languages (Simulink, Stateflow, SCADE) also include real time as an intrinsic part of their semantics.

   Ironically, given the sound and fury in the academic community, the industrial people I have interacted with for the most part do not care whether

time is discrete or continuous. Sometimes, I have encountered customers who want to do hybrid-systems style modelling, and for these people continuity is important.

2. *How does one involve small- and medium-size companies in collaborations with concurrency theoreticians/practitioners? Does "company size" matter?*

   Regarding SMEs (small- and medium-size enterprises, a common acronym among US policymakers), I think that the best way to involve them is via projects funded by third parties (governments or a large partner). SMEs generally don't have the overheads to support "blue-sky" research, and their investment-return horizons are necessarily of shorter duration. At both Reactive Systems and Fraunhofer, our concurrency-oriented SME collaborations have either involved collaborations on government research grants or project work on behalf of a larger customer. In the latter cases, it was important that we work in commercial notations (e.g. Simulink) rather than research-oriented ones.

   Large companies do have resources to put into more basic research, but there is another phenomenon to be aware of: researchers in these companies often view outside researchers as competitors for their internal research funds. So collaborations with these organizations are highly dependent on the personal connections between company and non-company researchers. So-called "business unit" personnel are often the easiest to deal with, but in this case there needs to be a clear, typically short-term, pay-off to them for the collaboration.

3. *Is there any need for stochastic and probabilistic modelling in applications? More pointedly, have you met an example that you could not model because your tool does not support stochastic or probabilistic phenomena?*

   We support simple probabilistic modelling in Reactis in the form of probability distributions over system inputs that we sample when creating tests. This feature, however, is almost never used by our customers. The reasons for this mostly boil down to a lack of training these engineers receive in stochastic modelling and control, which in turn is tied into the lack of good (or maybe standard) theory for stochastic differential equations.

   More precisely, the engineers in automotive and aerospace industry that I've dealt with are usually mechanical or electrical engineers with background in control theory. The feedback control they use relies on plant models (i.e., "environments") being given as differential equations, which are deterministic. The plant models they devise for testing their control-system designs often have parameters that they tweak in order to test how their ideas work under different conditions.

   These engineers talk in the abstract about how useful it would be to develop analytical frameworks for probabilistic plants, but tractable theories

of probability spaces of differential equations are unknown, as far as I can tell.

4. *How can we, as a community, foster the building of industrial-strength tools based on sound theories?*

   To have an industrial following, tools have to work with languages that industry uses. For most research tools this is a problem, because the input languages are typically invented by the tool developers.

   I see two possibilities. One is to work on commercial languages such as Simulink. These languages are often a catastrophe from a mathematical perspective, but they also usually contain subsets that can be nicely formalized for the purposes of giving tool support. If tools have a nice "intermediate notation" into which these cores can be translated, then this offers a pathway for potential industrial customers to experiment with the tools.

   The second approach is to become involved in standardization efforts for modelling languages. UML 2.0 has benefited to some extent from concurrency theory, but there are many aspects of that language that remain informal and imprecise.

5. *What has concurrency theory offered industry so far? What are the next steps that the concurrency community should take in order to increase the impact of its research in an industrial setting? And what are future promising application areas for concurrency research?*

   I think the best way to answer the first question is to "trace backward" from commercial tools / modelling languages that have some basis in concurrency. Such tools would include those based on Statecharts (Stateflow, STATEMATE, BetterState); others based on Message Sequence Charts (Rational Rose, other UML tools); the French synchronous-language tools (e.g. SCADE and Esterel); tools that include model checkers (the EDA, "electronic design automation" industry); tools that use model-checking-based ideas for other analyses (Reactis, DesignVerifier).

   Unfortunately the process-algebra community has had relatively little impact on commercial tool development. This is not due to shortcomings in the theory, in my opinion, but in the inattention that compositionality continues to receive in the (non-research) modelling community. In my experience, event-based modelling is also relatively uncommon, at least in the automotive and aerospace industry: sampling of "state variables" is the preferred modelling paradigm.

   I personally would like to see work on semantically robust combinations of specification formalisms (e.g. Message Sequence Charts plus state machines, or temporal logic plus process algebra) and related tools; theoretically well-founded approaches to verifying systems that use floating-point numbers; and compositional, graphical modelling languages (lots of work done already, but still no commercial interest).

Joost-Pieter Katoen

I'd like to answer the question on the need for stochastic and probabilistic modelling (and analysis). Some concrete examples of case studies provided by industry for which probabilistic aspects are very important are listed below. The importance of explicitly modelling random effects explicitly stands or falls with the kind of property to be established, of course, so I am definitely not claiming that these examples cannot (and should not) be modelled by using techniques that do not support random phenomena.

1. Leader election in IEEE 1394: in case of a conflict (two nodes pretend to be a leader), the contending nodes send a message (be my parent) and randomly wait either short or long. What is the optimal policy to resolve the contention the fastest? (This turns out to be a slightly unbiased coin).

2. In the Ametist EU-project, the German industrial partner Axxom generated schedules for a lacquer production plant. While doing so, they abstracted from many details that the lacquer producer supplies such as: the average fraction of time a resource is not operational, the fraction of (operational) time the resource can be used because necessary human support is present, and so forth. In their abstraction they scheduled tasks conservatively and they were interested in whether they could improve upon their schedules while reducing the probability to miss the deadline. Clearly, a stochastic modelling is needed, and indeed such a modelling has been carried out using a stochastic process algebra.

3. Hubert [Garavel] and Holger [Hermanns] should be able to say much more about a recent project they are pursuing with a French company on the validation of multiprocessor multi-threaded architectures. I do not know exactly what they are investigating, but they use stochastic process algebras to model such architectures! See

   `http://www.inrialpes.fr/vasy/multival/`

   for more information on this project.

Finally, let me say that (as Moshe [Vardi] is also indicating) the interest in probabilistic modelling is growing steadily. To give an example, the European Space Agency (ESA) is currently considering to use probabilistic modelling and analysis in the context of AADL, an architecture specification language where an important ingredient is the failure rates of components.

All in all, it is fair to say that there is a quest for probabilistic techniques!

Frits Vaandrager

1. *What is the role/importance of real time in modelling? Does industry want dense-time or discrete-time models?*

When modelling embedded systems and protocols, real-time aspects have to be dealt with very frequently. If one prefers not to use a real-time model checker, one can often encode real-time constraints in finite-state model checkers. For instance, in an STTT paper from 2002, Brinksma, Mader and Fehnker analyzed a PLC control schedule using SPIN, and obtained results that were (at that time) competitive with Uppaal. Their trick was to advance the discrete clock variable not one by one, but directly to the point of the next event.

In his PhD thesis, Rajeev Alur has made a good case for dense-time models. In the case of open systems with components that have different clocks, dense time is conceptually the right thing to do. But in practice one often gets away with discrete-time models.

Industry has no preference for dense-time or discrete-time: any approach that solves their problems is fine.

2. *How does one involve small- and medium-size companies in collaborations with concurrency theoreticians/practitioners? Does "company size" matter?*

   The maturity of model checkers has increased enormously over the last years and as a result it becomes much easier to apply them, also for non-experts. I give two examples.

   - Allen B. Downey has written *The Little Book of Semaphores*, a nice book in which he presents solutions to dozens of concurrency problems, ranging from classical ones (like the barbershop) to tricky and obscure problems (like the Sushi bar). This year I asked a class of first-year computer science students with just a few hours of model-checking experience to pick a problem from Downey's book (a different problem for each pair of students) and to model and analyze Downey's solution for it using Uppaal. As a result, my students spotted several mistakes in Downey's book, mistakes which have been confirmed by the author.
   - Matthijs Mekking, a student interested in implementing protocols and with no particular expertise in formal methods just completed an MSc thesis project at NLnet Labs on the SHIM6 protocol, a proposed IETF standard for multi-homing. At some point he started to model the protocol using Uppaal and became enthusiastic. He managed to find several nontrivial mistakes in the standard and greatly impressed the protocol designers. His results directly influenced the new version of the standard. (See `http://www.ita.cs.ru.nl/publications/papers/fvaan/SHIM6/`.)

   SMEs usually don't have verification specialists in house, so they need some advice on modelling and analysis. But with today's technology it is possible to get results rather quickly, and they can do part of the work

themselves. The situation will further improve when MSc and PhD graduates with a background in model checking get jobs at these companies. My group is collaborating with several SMEs and the verification problems they provide us with are often interesting from an academic perspective.

Unlike SMEs large companies are able to invest in long-term research.

3. *Is there any need for stochastic and probabilistic modelling in applications? More pointedly, have you met an example that you could not model because your tool does not support stochastic or probabilistic phenomena?*

   Yes!! There is a great need for stochastic and probabilistic modelling and analysis techniques, and I would for instance welcome a tool that combines the functionality of Uppaal and PRISM.

   The Zeroconf protocol, for instance, is full of probabilistic features that we could not model using Uppaal. If we really want to make impact in the area of wireless sensor networks, mobile ad-hoc networks, and P2P networks we need to extend model checkers with probabilities since the design of these networks can only be properly understood if we take probabilities into account.

4. *How can we, as a community, foster the building of industrial-strength tools based on sound theories?*

   Theory for the sake of theory is good and important but in my humble opinion the concurrency community has too much of it. In order to really push model-checking technology into industry the performance and functionality of these tools must be further improved by several orders of magnitude. This can only be achieved by a combined and focused effort of a large team of researchers and is also a major academic challenge.

   In order to test new ideas one needs prototype tools to play with. However, I believe it is not healthy that almost every research group on model checking has its own tool. Only a few groups manage to keep their model checking tool in the air for more than a decade. Developing an industrial-strength model checking tool requires a huge effort. I think academic groups have to join forces if they want to build (and maintain!) industrial-strength tools. Uppaal has been highly successful in part due to the continued efforts from teams in Aalborg, Uppsala, Nijmegen, Twente and elsewhere. So why don't the CADP and muCRL teams join forces? Why isn't it possible to establish stronger ties between Uppaal and IF? Why are there so many probabilistic model checkers?

   By combining efforts we can be much more effective.

5. *What has concurrency theory offered industry so far? What are the next steps that the concurrency community should take in order to increase the impact of its research in an industrial setting? And what are future promising application areas for concurrency research?*

Contributions are very powerful modelling languages and concepts, and of course model checking.

A major challenge ahead of us is to combine features. In model checking people proposed symmetry reduction, partial order reduction, CEGAR, etc. What about the combination of all these features? How can we prove it is sound. Can we combine probabilistic choice, hybrid aspects, real time, and hierarchy as in state-charts? I believe it will be possible to define such combinations but I expect we will need theorem provers and proof assistants to help us to manage the resulting complexity.

I am aware of applications in other areas, but I believe computer system engineering will remain the most important application area for concurrency theory the coming decade.

Moshe Vardi

1. *What is the role/importance of real time in modelling? Does industry want dense-time or discrete-time models?*

   Modern complex hardware designs are locally synchronous, but globally asynchronous. Using multiple clocks to specify temporal properties is very important, but I personally saw, so far, no application that required dense-time reasoning. (Perhaps timing analysis of circuits requires dense time?)

2. *How does one involve small- and medium-size companies in collaborations with concurrency theoreticians/practitioners? Does "company size" matter?*

   Large companies can afford to think longer term and invest in internal or external research. This is quite harder to do with small companies, which typically look for immediate solutions to their problems.

3. *Is there any need for stochastic and probabilistic modelling in applications? More pointedly, have you met an example that you could not model because your tool does not support stochastic or probabilistic phenomena?*

   I have not seen it so far, but I do hear people starting to talk about probabilistic behavior of computer circuits. Perhaps we'll see a growing need for stochastic modelling in a few years.

4. *How can we, as a community, foster the building of industrial-strength tools based on sound theories?*

   Academia can rarely build industrial-strength tools. Academic tools are often the work of a single graduate student. Industry can put several PhD-level people on a single project.

5. *What has concurrency theory offered industry so far? What are the next steps that the concurrency community should take in order to increase*

the impact of its research in an industrial setting? And what are future promising application areas for concurrency research?

In my humble opinion, the biggest successes of the concurrency-theory community, broadly conceived, is model checking and the development of synchronous languages. At the same time, many research direction in concurrency theory, such as process calculi and bisimulation theory have had fairly minimal impact. The theory community is often attracted to research based on its theoretical appeal, which does not always correlate with its industrial applicability. This does not mean that industrial applicability should be the guiding principle of research. Nevertheless, it would be worthwhile to pause once in a few years and examine the potential applicability of theoretical research.

5 Epilogue

Since the beginning, the theory of concurrency has shown promise for model-based development of computing systems. Its potential field of application has increased enormously with the advent of embedded computing, and with the correctness and other quality guarantees that embedded software must satisfy. Moreover, since the beginning of the field, concurrency theorists have placed emphasis on the development of prototype tools embodying their theories, taking advantage of the executable nature of their underlying formalisms. We feel that our field can boast some good success stories in applications, and the responses from our colleagues to the questions that were asked at the workshop indicate that concurrency theory may be on the verge of achieving recognition in industry and even greater success in applications.

It is not up to us to suggest a road map that the community may follow in order to increase its impact in industry. We hope, however, that the workshop and our report on it will help to foster a lively discussion amongst concurrency theorists.

One of the messages that we feel emerged clearly from the discussions at the workshop and from the opinions reported in Section 4 is that education is one of our main weapons to make our ideas and tools reach industry. The practitioners of tomorrow are the students of today. Let's embed our research in concurrency theory (model checking, model-based testing, process calculi, verification and validation and so on) into all aspects of our curricula. Let's write textbooks which are accessible to undergraduate students. Let's convince our students that the theory that they learn in our courses is not only "beautiful", but is also useful and needed. If we succeed in this enterprise, we will have built a strong bridge between our research and industry.

The workshop discussion may have given the impression that industrial impact can solely be achieved by means of tools and joint case studies. However, we believe that the development of theoretically sound and clean specification languages that are actually used by industry is another area in which in our

community can (and we dare say "should") have an impact. In a personal communication to us, Moshe Vardi wrote that

> In fact, I believe that much of my industrial impact has been achieved through the development of clean and useful theory.

Applicability does not have to necessarily come at the expense of elegance!

We close this, admittedly rather non-standard, article by listing below some further questions that have crossed our mind in putting together this summary, in no particular order. We hope that readers of this piece will contribute to the on-going discussion in our community by sharing their answers to these questions (and to the ones listed above) with us.

- Do you think that the concurrency-theory community must transfer the main results of its research to industry in order to be successful?
- Are tools necessary for technology transfer? Or can our community have industrial impact in other ways?
- What is the role of elegant theory in industrial applications?
- Should we make our techniques fit into the whole system-development process?
- Should concurrency theorists work on commercial languages like Simulink? If so, what should their priorities be?
- What is the role of bisimilarity in the application of concurrency theory in industrial settings?
- What are the uses of bisimulation minimization algorithms you are aware of in industrial applications? Do you think that the concurrency-theory community should rekindle research and implementation efforts on efficient and scalable bisimulation algorithms?
- Should academic tool developers combine their tooling efforts and make all of the code for their tools openly available? Should the code for earlier incarnations of tools be kept accessible for anybody who wishes to use it?
- Should the concurrency-theory community actively pursue the development of tools that combine real-time and stochastic modelling? If so, what theory should such tools be based on?
- Do you think that formalisms and tools that allow for a first-class treatment of data are necessary in (industrial) applications?
- How can the community improve the education of future software systems engineers and computer scientists? Should the concurrency theory community create some repository containing classroom-tested teaching material (including exercises and projects) that can be used off the shelf to create courses where theory and its applications are covered?