Kalpa Publications in Computing
Volume 3, 2017, Pages 89–95
RV-CuBES 2017. An International Workshop on
Competitions, Usability, Benchmarks, Evaluation,
and Standardisation for Runtime Verification Tools

A Few Things We Heard About RV Tools (Position Paper)

Sylvain Hallé, Raphaël Khoury, and Sébastien Gaboury

Laboratoire d'informatique formelle
Université du Québec à Chicoutimi, Canada

Abstract

Over the past five years, we have met with representatives of more than a dozen companies in various domains, trying to showcase the features of our runtime verification tool. These meetings gave us some very instructive feedback about the way RV is perceived by people from the industry, and about the shortcomings of current RV solutions. In this paper, we discuss a few of the comments we heard, and use them to suggest new ways of developing and evaluating RV tools in the future.

1 Foreword

The Runtime Verification community has a long history of research results that comprise both theoretical and practical contributions. Of particular interest is the vast inventory of tools and case studies that have been presented over the years, and which reveal a genuine interest in applying theoretical advances to real-world problems. Perhaps lesser known is what happens to the projects which, for various reasons, failed along the way and did not end up as case study papers in a conference. Obviously, technical problems can be a cause for such failures; a tool may not be expressive enough, or fast enough for the intended task. However, it may also be the case that some interesting industrial collaborations never materialized, for the simple reason that they never started in the first place.

This paper attempts to shed some light on this latter issue. It is based on formal and informal meetings we had in the past five years with multiple representatives from the industry, among which video game companies (Frozen Studios, Fungi Entertainment, GreenHaze), web development and design studios (Solstice, TLC), one of the largest insurance companies in Canada (JB), as well as about a dozen of other companies from the health care (PluriMedic), IT and service sectors.[1]

These meetings were intended to showcase the various capabilities of our own Runtime Verification tool, called BeepBeep 1 [2], and to explore the potential of using it for their monitoring and log analysis tasks.[2] BeepBeep 1 was an open source monitor written in Java, which used a first-order extension of Linear Temporal Logic, called LTL-FO$^+$, as its specification language. Events in BeepBeep 1 were represented as data-rich XML documents comprising

[1] Due to non-disclosure restrictions, all the names of these companies have been replaced by fictitious names in the text.

[2] The current version of this monitor, BeepBeep 3, largely supersedes the capabilities of the original.

$$\begin{aligned}
&\mathbf{G}\,(\exists retAddrVal \in \text{return-address} : ((\text{instruction} = \text{call}) \wedge (\neg((\mathbf{F}\,(((\text{instruction} = mov)\\
&\qquad \wedge (\text{output/type} = \text{general-register})) \rightarrow (\exists regA \in \text{output/name} :\\
&(\mathbf{F}\,((((\text{instruction} = mov) \wedge (\text{output/type} = \text{general-register})) \wedge (\text{input/type} = literal)) \rightarrow\\
&\qquad (\exists regB \in \text{output/name} : (\exists constAddr \in \text{input/value} : (\mathbf{F}\,(((\text{instruction} = \text{cmp}) \wedge\\
&\qquad\quad (\text{output/type} = regA)) \rightarrow (\exists loc \in \text{location} : (\mathbf{F}\,((((\text{instruction} = mov) \wedge\\
&\qquad\qquad (\text{output/type} = \text{general-register})) \wedge (\text{output/name} = regA)) \wedge\\
&\qquad\qquad\quad ((\text{input/name} = regB) \wedge (\text{input/type} = ptr))))))))))))))\\
&\qquad\qquad\qquad \mathbf{U}\,((\text{instruction} = return) \wedge (\text{function-returned} = retAddrVal))))))
\end{aligned}$$

Figure 1: A wall of text. This is an actual LTL-FO$^+$ formula we used to analyze traces of assembly-level calls in a program. Can you figure out what it does? (We could some time ago.)

multiple nested elements; using system pipes or other types of data streams, BeepBeep could be connected to a variety of event sources. It could work either offline or online.

In the following, we shall tell —in a lighthearted tone— a few anecdotes that actually happened to us, and try to extract a few reasons that made us come back from these meetings empty-handed, more often than not.

2 "It's the attack of the wall of text"

Our first anecdote starts at Fungi Entertainment during one of our first official presentations to members of the industry. Our talk started with a pre-recorded video demo of our monitor, catching bugs in realtime during the execution of an open source video game, and logging these bugs automatically in a bug tracker. After about twenty minutes of show and tell, things were going rather smoothly, with our audience listening attentively. This was before we reached the slide that showed the temporal logic formula that was used to catch one of the bugs, which filled the screen, had six first-order quantifiers, temporal operators and XPath expressions all mixed in a happy soup of math symbols. It looked really nice (much like the one in Figure 1).

The silence was broken by one of the representatives from Fungi Entertainment, who exclaimed that the syntax for this language was horrendous, and called it the "attack of the wall of text". He went on to explain, accompanied with energetic nodding from his colleagues (the top brass from the QA department), that such a specification language was too low-level and verbose to have a chance of being used in the real world. He summarized it rather bluntly as "the kind of thing that will be maintained for a week and then be abandoned."

As our experiences have shown, the specification language of current RV tools is the first stumbling block preventing their adoption by practitioners. These languages look (relatively) simple to us, but are seen as disturbingly alien rocket science to most people in the business. Jack, the QA manager at Frozen Studios, expressed his fear that with such languages, QA testers would have to be replaced by QA *analysts* —people with a higher education required to understand the properties, and hence require higher wages. RV as a financial burden: we had never seen it that way.

Indeed, a few works have been devoted to improve the usability and intuitiveness of the process of writing properties, but we are still far from the day where some random IT worker

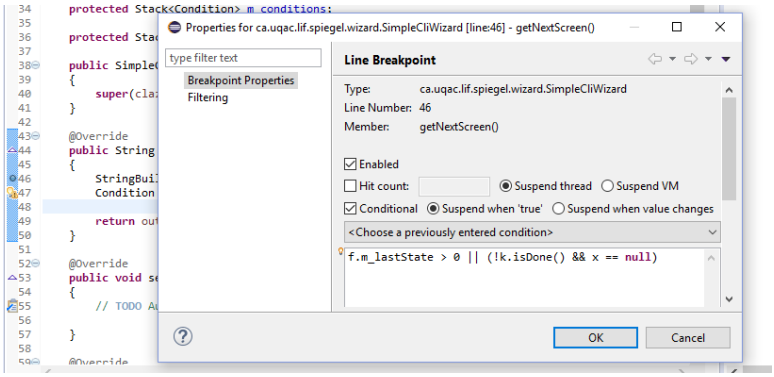

Figure 2: A conditional breakpoint. According to some, research in RV has simply been trying to replicate this dialog box for a decade.

will write properties by himself on a regular basis. One might even question if this is really what we should strive for; since programmers are so at ease with writing code, perhaps writing monitoring specifications should look more like an extension of writing code, instead of being a separate (and rather formal) activity.

Research Question 1. *Should we revise our expectations regarding the usability of our specification languages?*

3 "We can already do this"

One of the most recurring (mis)conceptions we encountered during our meetings with the industry, is that runtime verification provides no added value with respect to current practices. Many of the people we met were sincerely convinced that the features we were presenting could easily be achieved by tweaking or slightly extending the set of existing software and programming practices they were already using —and thus did not warrant the adoption of an untrusted, academic-proof-of-concept software and its equally suspicious input language.

At Frozen Studios, one of the IT people patiently listened for the end of our talk, and then told us that the runtime bug finding tasks we had showcased in our demo could all be done by simply inserting *conditional breakpoints* inside the code (Figure 2). We went on to explain that conditional breakpoints cannot correlate multiple states of the program (unless you start adding variables into it to keep track of that state), whereas our RV system could take care of that by itself; we even added that mixing test code with production code was not a particularly clean way of working. This did not seem to be a particularly moving argument, and soon after we were politely escorted to the door.

At Quidnovi, the same argument was involved for a different reason. This time, a simple relational database was the proposed solution. "Why don't you just put everything in MongoDB and query it with SQL?" We replied that we had already tried [3, 4], and that in some cases, computing the equivalent of temporal operators in SQL requires many self-joins of a large table, making the query extremely inefficient, verbose and illegible.[3] Again, this argument seemed to contradict their intuition, and was quickly dismissed.

[3]We had learned from our previous "wall of text" objection, and this time tried to use it to our advantage.

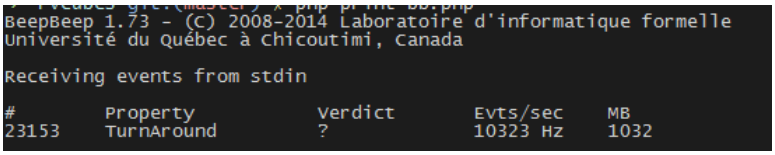

Figure 3: BeepBeep 1 running from the command line. When the monitored property is violated, the sole feedback a user gets is the change of a single character in the screen. Can you guess which one?

The list could go on and on; every new meeting seems to reveal yet another way of being told we are reinventing the wheel. Fungi Entertainment mentioned commercial solutions like Scribe and DeltaDNA, plus some other secret internal tools they could not talk about —but that was already doing everything we did. The overall impression, in many of our meetings, was that Runtime Verification, practiced in the way we put forward, could always be approximated in some quirky and inappropriate way by using what was at hand.

Research Question 2. *Do we have measurable (and quantitative) evidence of the benefits brought by the adoption of RV?*

4 "What are we supposed to do with a Boolean?"

In our demos, we always expected the "aha!" moment to be when the monitor catches a bug during the execution of the program, and manifests itself in some way (lighting a red light, printing a message, or something of that kind; see Figure 3). Unfortunately, in most of our presentations, the only enthusiastic person in the room at that point was the presenter himself. For all its complexity, the fact that a runtime monitor only returns a fail/pass verdict leaves the audience largely unimpressed. "All this for a Boolean" was roughly the comment we got from Fungi Entertainment.

And indeed, judging from past literature, recent RV tools enjoy more expressive specification languages, allow the computation of aggregate values, the use of timers, and so on. Yet, not much work has been done to provide a richer verdict than $\top$ or $\bot$. A monitor that merely stops along a trace when a property is false provides scarce information to a potential user, apart from saying that "the problem is in there". Some temporal logics do have a few more truth values, but they mostly provide subtle ways of saying "I don't know". Something tells us that this would not be sufficient to appease the people we met.

We came out from these meetings with the conviction that, if one is to see monitors as testing and diagnostics tools, they ought to shed their Boolean skin and evolve to provide a verdict that contains more information. What this information should be, and how to compute it, is an open question. This could prove a very fertile ground for both theoretical and practical research. In this respect, the fact that the Competition on Runtime Verification plans to give credit for tools based on the richness of their verdict represents, in our view, an encouraging first step.

Research Question 3. *What kind of useful feedback can a monitor give, beyond true or false?*

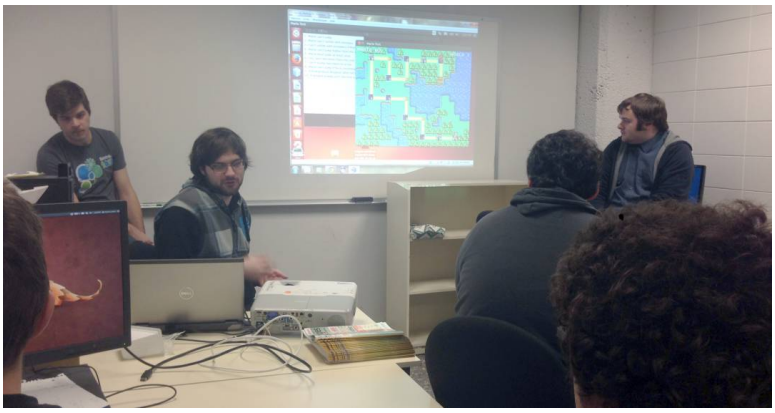

Figure 4: Some of our students doing a demo of our monitor on a video game. See the sad look on their faces: they have been writing properties all day.

5 "But you have to write the properties"

In our presentations with Quidnovi and Olevia, the representatives we met asked us a puzzling question: "do you have to write the properties?" Our positive answer was met with visible deception (as in Figure 4). This time, the problem was not that the properties must be written in a rebuffing language —it was that they must be written *at all*, which seemed rather baffling to us. Our meeting with Olevia gave us more information on this point. We were suggested to have a look at Data Mining, as it can "find patterns in data without the need to tell what you are looking for". A representative from NuageCloud told us something similar.

At first sight, this might look like a naive manifestation of the "Data Mining Is Magic" phenomenon that seems to be common these days. However, Data Mining aside, this is a valid objection, which applies to any specification-based technique. The need to exhaustively codify the expected behavior of a system, before any testing can ever take place, is an important limitation of current RV tools, which representatives from the industry have sensed very well.

It is not clear at all how one could be relieved from inputting a specification. More intuitive languages (see above) could help, but automated or semi-automated techniques for obtaining (learning?) a specification could also prove useful.

Research Question 4. *Can we remove or reduce the need for an* a priori *specification?*

6 "How long does it take?"

Contrary to the previous points, this is actually a question we have seldom been asked. As a matter of fact, in some of our meetings, more or less the opposite happened.

A prime example of this is our meeting with executives from PluriMedic for a potential collaboration. We had no idea about how much data there was to process, and since our monitor was still under development, no idea either as to how fast it would run. Therefore, we tentatively asked: if you started a job in the afternoon and got your result by the next morning, would that be OK? Surprisingly, the answer was: "sure, we don't mind waiting".

This came as a surprise, given the emphasis on overhead and throughput we find in RV literature. Yet, in all of the meetings we had before and since, performance has never been the

Figure 5: The Bloodhound SSC is one of the fastest cars in the world —provided you don't want to go anywhere but a closed circuit track. (Source: Wikipedia)

first element to be discussed, and has never been the issue that makes or breaks the deal. What seems to count first and foremost is the *value* one can extract from the results of the tool: can I discover new patterns I currently can't find? What new insight can I get by using this piece of software? If a result is considered to provide added value, the interval of time that is deemed reasonable to obtain this result may be larger than we tend to think. And conversely, there is no point in being fast at answering a question no one is asking; after all, the fastest car in the world is only fast at going on a straight line, which very few people would find useful for their everyday errands (Figure 5).

Research Question 5. *Do we know what are reasonable performance requirements for our use cases? Are we over-emphasizing speed over other matters?*

7 Afterword

As we have seen, our experience at presenting RV tools to members of the industry has raised a lot of important issues. As disparaging as some of these remarks may seem at first sight, each of them contains some very instructive tips about the way RV tools are perceived.

At the RV-CuBES workshop where this paper was presented, a wise person commented on the fact that some of the issues we faced might be related to the way we approached representatives of the industry. Rather than barge in, give a one-hour sales pitch and cross our fingers (a close-enough description of what we did), a more fruitful technique would involve listening to them first, identify potential application points —and only then, suggest a tailored solution that involves RV. We shall also mention that, despite all the misfortunes and anecdotes we have told in these few pages, we did succeed in working with companies on a handful of interesting projects. For the most part, we collaborated with people that were open and motivated, and which provided us with lots of insight, suggestions for future work, and useful data. A great deal of the success or failure of a potential collaboration, it turns out, depends on the person you end up talking to.

Some of these remarks have already oriented the focus of our own work on RV; for example, the criticism of our first monitor's Boolean verdict has led to the creation of a new, and much more general version of BeepBeep [1]. Similarly, the "wall-of-text" propensity of RV languages and their reliance on pre-written properties are currently being worked on, and new features of BeepBeep addressing these issues shall be unveiled in future research papers. Overall, it is hoped that the comments discussed here will provide food for thought for future research on Runtime Verification, not just for us, but for the community as a whole.

References

[1] Hallé, S.: When RV meets CEP. In: Falcone, Y., Sánchez, C. (eds.) Runtime Verification - 16th International Conference, RV 2016, Madrid, Spain, September 23-30, 2016, Proceedings. Lecture Notes in Computer Science, vol. 10012, pp. 68–91. Springer (2016), https://doi.org/10.1007/978-3-319-46982-9_6

[2] Hallé, S., Villemaire, R.: Runtime enforcement of web service message contracts with data. IEEE Trans. Services Computing 5(2), 192–206 (2012), https://doi.org/10.1109/TSC.2011.10

[3] Mrad, A., Ahmed, S., Hallé, S., Beaudet, É.: Babeltrace: A collection of transducers for trace validation. In: Qadeer, S., Tasiran, S. (eds.) Runtime Verification, Third International Conference, RV 2012, Istanbul, Turkey, September 25-28, 2012, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7687, pp. 126–130. Springer (2012), http://dx.doi.org/10.1007/978-3-642-35632-2_14

[4] Vallet, J., Mrad, A., Hallé, S., Beaudet, É.: The relational database engine: An efficient validator of temporal properties on event traces. In: Bagheri, E., Gasevic, D., Hallé, S., Hatala, M., Nezhad, H.R.M., Reichert, M. (eds.) 17th IEEE International Enterprise Distributed Object Computing Conference Workshops, EDOC Workshops, Vancouver, BC, Canada, September 9-13, 2013. pp. 275–284. IEEE Computer Society (2013), http://dx.doi.org/10.1109/EDOCW.2013.37