Kalpa Publications in Computing
Volume 3, 2017, Pages 1–9
RV-CuBES 2017. An International Workshop on
Competitions, Usability, Benchmarks, Evaluation,
and Standardisation for Runtime Verification Tools

A Report of RV-CuBES 2017

Giles Reger

University of Manchester, Manchester, UK

Abstract

RV-CuBES was an international workshop that took place alongside the 17th International Conference on Runtime Verification in Seattle during 13-16th September, 2017. The focus of the competition was to consider tools for Runtime Verification (RV). The acronym CuBES stands for Competitions, usability, Benchmarks, Evaluation, and Standardisation. The workshop consisted of poster presentations of tool overview papers and a discussion session of position papers. This report gives an overview of submissions and discussions.

1 Introduction

The RV-CuBES workshop was integrated into the 17th International Conference on Runtime Verification [22] which took place in Seattle during 13-16th September, 2017. The purpose of the workshop was to act as a focus for issues related to the Competition on Runtime Verification (CRV)[5, 16, 26] that ran from 2014 to 2016 but took a hiatus in 2017. The acronym CuBES stands for Competitions, usability, Benchmarks, Evaluation, and Standardisation. The u was purposefully given a lower status than the other topics.

The workshop encouraged two kinds of submission. The first was a tool overview paper, with the aim of performing a lightweight survey of actively developed runtime verification tools (e.g. tools that may take part in CRV). The second was a position paper on one of the cognate topics (Competitions, usability, Benchmarks, Evaluation, or Standardisation) to be discussed during a discussion session at the workshop. Workshop activities (poster and discussion sessions) were integrated into the main conference to maximise involvement of the wider runtime verification community.

There were 18 submissions with 13 tool overview papers and 5 position papers. All papers received at least 3 reviews from the PC. All 18 papers were accepted for presentation at the workshop; the aim of the workshop was to encourage discussions and the starting point was to allow everybody to participate. Initially, 7 tool overview papers and 2 position papers were accepted unconditionally for the post-proceedings with the remainder being conditionally accepted. After further review and shepherding, all papers were accepted for publication in the post-proceedings. However, one paper was withdrawn by its authors due to a conflict with a concurrent submission.

This report discusses the tool submissions in Section 2 and the discussed position papers in Section 3 before summarising in Section 4

Table 1: An overview of tools described in post proceedings (and some other relevant tools).

Tool	Target	Specification	Key features
Aerial [9]	Outline (OCaml)	MTL/MDL	Event-rate independent
ARTiMon	Matlab/Simulink	ARTiMon	Also for hybrid systems [24]
BeepBeep 3 [18]	XML documents	LTL-FO$^+$ [20]	Stream processing, modular
DANA [23]	Network systems	UML state machines	Eclipse plugin
detectEr [1]	Erlang	μHML [17]	Asynchrony. Adaption [10]
E-ACSL [14]	C	E-ACSL [14]	Integrated with Frama-C [21]
Larva [12]	Java	DATEs [11]	Case studies & extensions
Lola [13]	Logs/streams	Lola	Stream processing
MonPoly [8]	Log files	MFOTL [7]	Highly scalable [6]
TemPsy-Check	CSV logs	TemPsy	Based on spec patterns
R2U2 [28]	UAS	MTL, Bayes Net etc	Deployed on a drone!
Valour	Outline	Rules	
MarQ [25]	Outline (Java)	QEA [3]	Parametric trace slicing
LogFire [19]	Outline (Scala)	Rule system	Internal DSL
TraceContract [4]	Outline (Scala)	TL, FSM, rules etc	Internal DSL

2 Tool Submissions

The workshop had 13 tool overview papers covering 14 separate tools. One of the features of the workshop was not to require attendance for a tool overview submission. The aim was to introduce as few barriers as possible to authors describing their tools in this lightweight survey. Six of the tools were presented at a poster session during the runtime verification conference. During the conference there was a prize for *Most Interesting Poster*, which was awarded to Lola[1]. This prize was judged by attendees of the RV conference and no further guidance was given beyond the name of the prize.

Table 1 gives an overview of 12 of the 14 tools presented in the overview papers and 3 further tools that were not described as they are developed by the workshop chairs. This presents a relatively high-level view of these actively developed tools and I encourage the reader to look at the relevant papers to learn more (I have not cited the papers included in this proceedings but these should act as a first resource). Indeed, some parts of the table represent an oversimplification. For example, I state that Larva uses DATEs for its specification language but in reality they more often define a domain-specific language to compile into DATEs, similarly BeepBeep 3 provides a query language that hides the LTL-FO$^+$ logic. The key features were selected by this author, not the tool authors, and suggest some points of interest only.

The table separates the *Target* of the tool (e.g. XML files, Java programs), the form of *Specifications* that it considers (e.g. state machines, temporal logic), and also any other key features of the tool (which remains relatively abstract). In the *Target* column I write Outline when the tool provides an API and could, therefore, be used with any target but I also include the native language (where using such an API would be most straightforward). I omit links to the actual tools. In general, these can be found either via the papers in these proceedings.

[1]The authors of Lola chose not to include their submission in the post-proceedings due to conflict with another submission.

It is perhaps interesting to note that 6[2] of the 15 described tools have entered the competition before (BeepBeep 3, E-ACSL, Larva, MonPoly, MarQ, and LogFire), with E-ACSL and MarQ being the only tools that entered all three years. Of the remaining tools, Aerial, TemPsy-Check, Valour, and TraceContract could straightforwardly fit into the current competition format, but the format may need to be altered somewhat to allow the other tools (ARTiMon, DANA, detectEr, R2U2) to meaningfully participate.

3 Discussion Session

The workshop consisted of two 45 minute discussion sessions. The first session considered the general topic of what we can do as a community to encourage usage of runtime verification (tools). The second session considered more specifically what can be done to improve the competition. Each session began with 5 minute presentations from position paper authors followed by open discussion. *It should be noted that the submitted position papers significantly expand and complement the discussion points given here.*

3.1 Making RV Useful

There were two positions in this session that can be summarised as follows:

- *COEMS - open traces from the industry.* The authors present their experience in implementing a repository for storing traces originating from their COEMS industrial project. There is a focus on the infrastructure required to support a range of data and to provide open access to data. One recommendation is the use of the *Common Trace Format* (CTF) as a format that is used by industry. Another recommendation is for the resulting repository to be used by the community (and competition) for sharing data.
- *A Few Things We Heard About RV Tools.* The authors reflect on their previous five years experience attempting to engage industry with runtime verification. They present their findings in the form of 5 research questions. Those notably specific to RV include whether we can provide more informative feedback than a boolean verdict, whether the specification burden can be reduced, and whether monitoring speed is over-emphasised.

Together, the authors of these position statements suggested three questions for the discussion session to discuss:

1. How can we improve the applicability of RV in the industry?
2. Does the RV community need a joint repository and a joint format?
3. Should RV embrace the Open Data movement in Computer Science, and how?

The following reports on the outcome of the discussions where different strands of the discussion have been grouped under relevant headings.

How to engage industry. It was pointed out that our community is not alone in finding it difficult to engage with industry. An early point was that to engage industry it is necessary to start with *their problem*, rather than *our solution*. For example, one should engage with an industrial partner to understand the specifics of their problem, especially the terminology

[2] Although an earlier version of TemPsy-Check called OCLR-check was entered into the competition.

and applications-specific requirements, rather than extolling the virtues of a certain approach (e.g. runtime verification) and looking for areas where it may be applicable. It was noted that cases where communities close to ours have successfully engaged with industry (examples given included PSL [15] and Code Contracts [2]) speak the language of the application domain.

What do industry want from RV? The next topic considered was what industry is looking for i.e. what kind of research should we be doing to attract industry interest. The following areas were proposed based on the participants interactions with industry:

- *Explanation.* One participant relayed an experience where an industrial partner was underwhelmed by the ability to detect a 'boolean result' (e.g. whether the specification held at runtime or not). It was felt that, in general, industry want *more* from their traces i.e. data that they can *use* in their development process. Principally, this appeared to mean *explanations* of verdicts (violations or success). A simple example of this would be to provide a minimal violating subtrace relating an error back to a point in the code that generated it. Another direction was the generation of *statistics* from a trace i.e. the view that a monitor is a *transducer* from the domain of runtime events to statistical events.
- *Metrics.* It was noted that industry like 'metrics' e.g. statistics on what has been achieved by the development process (in this case runtime verification). For example, they may be interested in the *coverage* of a monitor, both with respect to the monitored code and the specification. The second point was expanded in discussions to explain that it may be interesting to see that a certain behaviour allowed by the monitor was never exercised at runtime. Parallels were drawn to similar metrics produced in the field of testing.
- *Debugging.* It was felt that industry wanted to be able to integrate runtime verification techniques into their general debugging workflow. This might mean altering the interface between monitoring technique and user to talk in the language of the debugging process. Importantly, it would need to relate the monitoring activity directly to the source of the system being monitored. Another angle suggested here was the need for *specification debugging* to identify potential bugs in specifications.

What common formats do we need? There were two areas discussed where joint/common formats may be desirable: in the storage of traces and in the specification of properties. One of the position statements explicitly discussed the former, and the latter is a key issue in CRV and will be discussed further below.

With relation to trace formats, it was noted that industry are already frequently recording traces and analysing these and there exist a number of formats (mostly ad-hoc) for recording this information[27], although these formats are not designed to record the information needed for runtime verification. It was agreed that more should be done to take advantage of the abundance of existing traces being produced. One aspect that was discussed was the *cost of entry* to using a format as some formats (such as Common Trace Format) are difficult for a human to read and understand.

With relation to common specification languages, a few points were made specifically related to the interests of industry. There was some disagreement around which kinds of languages are appropriate to ask software engineers to engage with. An early point was that temporal logic is too hard/unintuitive mainly due to its declarative nature. However, two counterpoints were the success of PSL and the fact that regular expressions are declarative and enjoy particular success in software engineering in general. In particular, it was commented that the *Until* rule can be confusing (especially when nested) and this can sometimes be avoided

3.2 The Future of CRV

There were three positions in this session that can be summarised as follows:

- *On the Evaluation and Comparison of Runtime Verification Tools for Hardware and Cyber-Physical Systems.* The author gives a comprehensive account of how current approaches to evaluation (particularly in CRV) must be modified to account for the differences when monitoring hardware and cyber-physical systems. This is an element that has been missing from CRV and this contribution may lay the foundations for considering an extension in the future. The author also makes a range of general suggestions for the competition, in particular a list of inspirations that can be taken from other communities.

- *On the Risk of Tool Over-tuning in Run-time Verification Competitions.* The authors note that the previous format of the competition was open to *over-tuning* and make suggestions on how to combat this. The primary suggestion is to split benchmarks into public parts (for testing) and private parts (for evaluation).

- *Online Runtime Verification Competitions: How To Possibly Deal With Their Issues.* The author reviews the previous three years of the competition and reflects on various aspects, making suggestions of how things could be improved. Two major points are that the current structure makes too strong assumptions about what an RV tool is and requires too much time from organisers and participants.

Together, the authors of these position statements suggested three questions for the discussion session to discuss:

1. What does a RV hardware track benchmark look like?

2. How should RV hardware tools be evaluated?

3. What are the current issues and their possible solutions of online runtime verification competitions?

4. Can we decrease the risk of tool over-tuning by only providing the specification (syntax and semantics) of the property languages as part of the benchmark?

The following reports on the outcome of the discussions where different strands of the discussion have been grouped under relevant headings.

Why do we want a competition? It was discussed that before we can decide what a RV competition should involve, we should first decide what it is trying to achieve. At this stage the original aims of the competition were revisited (albeit briefly and from memory) - in [5] these are stated as

- To stimulate the development of new efficient and practical runtime verification tools and the maintenance of the already developed ones.

- To produce benchmark suites for runtime verification tools, by sharing case studies and programs that researchers and developers can use in the future to test and to validate their prototypes.

- To discuss the measures employed for comparing the tools.

- To compare different aspects of the tools running with different benchmarks and evaluating them using different criteria.

- To enhance the visibility of presented tools among different communities (verification, software engineering, distributed computing and cyber security) involved in software monitoring.

It was agreed that these were all reasonable goals in general, although it was not clear that the existing form of the competition is best suited to these goals or, indeed, that it has made progress in all of them. One participant proposed that the main aim of a competition should be *to allow practitioners decide which tool(s) they can, or should, use*. This formulation then suggests that one must first understand what practitioners want and it was generally felt (as indicated above) that in many cases they may prefer *usability* over highly optimised efficiency.

What should the competition measure? So far the competition has attempted to measure *efficiency* of the monitoring algorithm and *expressiveness* of the specification language via a combined scoring system. It was argued that the second part effectively conflates the effort required to write the specification with the expressiveness of the language, and therefore does not measure expressiveness as well as intended.

In the discussion an early comment was that *correctness* was essential. It was clarified that this should mean *soundness* of monitoring result. However, one participant pointed out that we might want to consider settings where certain levels of false positives are acceptable. The example of intrusion detection as an application of RV was given.

It was agreed that topics such as expressiveness and usability can be more important than efficiency but are typically harder to measure or even quantify. There was also the question of whether one should separate the usability of a tool from the specification language it uses. With respect to efficiency, there was the point that different resources (time, memory, network utilisation) are more or less important in different contexts and a competition should avoid combining them into a single score.

Where should benchmarks come from? Currently, the competition has had a benchmark submission phase as part of the competition and there has been minimal reuse from year to year. A proposal that has been repeated often is that the community build and maintain a common set of benchmarks from which the competition can draw benchmarks (as is done in other communities). It was generally agreed that the current format of making benchmark submission part of the competition made the time commitment too high.

On this topic, it was briefly discussed what a benchmark should consist of. No conclusive answer was given but it was suggested that it should consist of multiple workloads (in the case of programs) or traces (for offline benchmarks) to allow a separation of testing and evaluation.

What should we standardise and how? It was generally agreed that the competition would greatly benefit from a standard specification language but what this should look like was unclear. One previous proposal was to fix a language and force the community to adopt it in their own tools. However, it was felt that there were already enough barriers to entry. Another suggestion was to define a single reference language such that each participant only need translate to and from this language and their own.

What was not discussed. Due to time constraints there were some topics that did not receive any discussion time. In particular, there were no concrete discussion of how runtime verification techniques related to hardware (either monitoring hardware or using specific hardware to accelerate monitoring) could be incorporated into a competition.

4 Summary

The goals of the RV-CuBES workshop were to perform a lightweight survey of actively developed RV tools and to engage the community in a discussion of how the development of RV tools should be supported by the community. Both goals were achieved and the workshop will hopefully have a positive impact on the RV community in the future.

References

[1] Duncan Paul Attard and Adrian Francalanza. *A Monitoring Tool for a Branching-Time Logic*, pages 473–481. Springer International Publishing, Cham, 2016.

[2] Mike Barnett. Code contracts for .net: Runtime verification and so much more. In *Runtime Verification - First International Conference, RV 2010, St. Julians, Malta, November 1-4, 2010. Proceedings*, pages 16–17, 2010.

[3] Howard Barringer, Yliès Falcone, Klaus Havelund, Giles Reger, and David Rydeheard. Fm 2012: Formal methods: 18th international symposium, paris, france, august 27-31, 2012. proceedings. pages 68–84. Springer Berlin Heidelberg, 2012.

[4] Howard Barringer and Klaus Havelund. *TraceContract: A Scala DSL for Trace Analysis*, pages 57–72. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[5] Ezio Bartocci, Yliès Falcone, Borzoo Bonakdarpour, Christian Colombo, Normann Decker, Klaus Havelund, Yogi Joshi, Felix Klaedtke, Reed Milewicz, Giles Reger, Grigore Rosu, Julien Signoles, Daniel Thoma, Eugen Zalinescu, and Yi Zhang. First international competition on runtime verification: rules, benchmarks, tools, and final results of crv 2014. *International Journal on Software Tools for Technology Transfer*, Apr 2017.

[6] David Basin, Germano Caronni, Sarah Ereth, Matúš Harvan, Felix Klaedtke, and Heiko Mantel. Scalable offline monitoring of temporal specifications. *Form. Methods Syst. Des.*, 49(1-2):75–108, October 2016.

[7] David Basin, Felix Klaedtke, Srdjan Marinovic, and Eugen Zălinescu. Monitoring of temporal first-order properties with aggregations. In Axel Legay and Saddek Bensalem, editors, *Runtime Verification: 4th International Conference, RV 2013, Rennes, France, September 24-27, 2013. Proceedings*, pages 40–58, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[8] David Basin, Felix Klaedtke, Samuel Müller, and Eugen Zălinescu. Monitoring metric first-order temporal properties. *J. ACM*, 62(2):15:1–15:45, May 2015.

[9] David Basin, Srdjan Krstić, and Dmitriy Traytel. Almost event-rate independent monitoring of metric dynamic logic. In Shuvendu Lahiri and Giles Reger, editors, *Runtime Verification: 17th International Conference, RV 2017, Seattle, WA, USA, September 13-16, 2017, Proceedings*, pages 85–102, Cham, 2017. Springer International Publishing.

[10] Ian Cassar and Adrian Francalanza. *Runtime Adaptation for Actor Systems*, pages 38–54. Springer International Publishing, Cham, 2015.

[11] Christian Colombo, Gordon J. Pace, and Gerardo Schneider. Dynamic event-based runtime monitoring of real-time and contextual properties. In Darren Cofer and Alessandro Fantechi, editors, *Formal Methods for Industrial Critical Systems: 13th International Workshop, FMICS 2008, L'Aquila, Italy, September 15-16, 2008, Revised Selected Papers*, pages 135–149, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[12] Christian Colombo, Gordon J. Pace, and Gerardo Schneider. Larva — safer monitoring of real-time java programs (tool paper). In *Proceedings of the 2009 Seventh IEEE International Conference on Software Engineering and Formal Methods*, SEFM '09, pages 33–37, Washington, DC, USA, 2009. IEEE Computer Society.

[13] Ben D'Angelo, Sriram Sankaranarayanan, Cesar Sanchez, Will Robinson, Bernd Finkbeiner, Henny B. Sipma, Sandeep Mehrotra, and Zohar Manna. Lola: Runtime monitoring of synchronous systems. In *Proceedings of the 12th International Symposium on Temporal Representation and Reasoning*, TIME '05, pages 166–174, Washington, DC, USA, 2005. IEEE Computer Society.

[14] Mickaël Delahaye, Nikolai Kosmatov, and Julien Signoles. Common specification language for static and dynamic analysis of c programs. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 1230–1235, New York, NY, USA, 2013. ACM.

[15] Cindy Eisner and Dana Fisman. *A Practical Introduction to PSL (Series on Integrated Circuits and Systems)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[16] Yliès Falcone, Dejan Nickovic, Giles Reger, and Daniel Thoma. Second international competition on runtime verification CRV 2015. In *Runtime Verification - 6th International Conference, RV 2015 Vienna, Austria, September 22-25, 2015. Proceedings*, pages 405–422, 2015.

[17] Adrian Francalanza, Luca Aceto, and Anna Ingolfsdottir. *On Verifying Hennessy-Milner Logic with Recursion at Runtime*, pages 71–86. Springer International Publishing, Cham, 2015.

[18] Sylvain Hallé. When rv meets cep. In Yliès Falcone and César Sánchez, editors, *Runtime Verification: 16th International Conference, RV 2016, Madrid, Spain, September 23–30, 2016, Proceedings*, pages 68–91, Cham, 2016. Springer International Publishing.

[19] Klaus Havelund. Rule-based runtime verification revisited. *International Journal on Software Tools for Technology Transfer*, 17(2):143–170, Apr 2015.

[20] Raphaël Khoury, Sylvain Hallé, and Omar Waldmann. Execution trace analysis using ltl-fo$^+$. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications: 7th International Symposium, ISoLA 2016, Imperial, Corfu, Greece, October 10-14, 2016, Proceedings, Part II*, pages 356–362, Cham, 2016. Springer International Publishing.

[21] Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. Frama-c: A software analysis perspective. *Formal Aspects of Computing*, 27(3):573–609, May 2015.

[22] Shuvendu K. Lahiri and Giles Reger, editors. *Runtime Verification - 17th International Conference, RV 2017, Seattle, WA, USA, September 13-16, 2017, Proceedings*, volume 10548 of *Lecture Notes in Computer Science*. Springer, 2017.

[23] Thomas Pramsohler, Mahmut Kafkas, Annette Paulic, Marc Zeller, and Uwe Baumgarten. Control flow analysis of automotive software components using model-based specifications of dynamic behavior. *SAE Int. J. Passeng. Cars Electron. Electr. Syst.*, 6:425–436, 04 2013.

[24] Nicolas Rapin. Reactive property monitoring of hybrid systems with aggregation. In Yliès Falcone and César Sánchez, editors, *Runtime Verification: 16th International Conference, RV 2016, Madrid, Spain, September 23–30, 2016, Proceedings*, pages 447–453, Cham, 2016. Springer International Publishing.

[25] Giles Reger, Helena Cuenca Cruz, and David Rydeheard. Marq: Monitoring at runtime with qea. In Christel Baier and Cesare Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems: 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings*, pages 596–610. Springer Berlin Heidelberg, 2015.

[26] Giles Reger, Sylvain Hallé, and Yliès Falcone. Third international competition on runtime verification - CRV 2016. In *Runtime Verification - 16th International Conference, RV 2016, Madrid, Spain, September 23-30, 2016, Proceedings*, pages 21–37, 2016.

[27] Giles Reger and Klaus Havelund. What is a trace? a runtime verification perspective. In Tiziana

Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications: 7th International Symposium, ISoLA 2016, Imperial, Corfu, Greece, October 10-14, 2016, Proceedings, Part II*, pages 339–355. Springer International Publishing, 2016.

[28] Johann Schumann, Patrick Moosbrugger, and Kristin Y. Rozier. *Runtime Analysis with R2U2: A Tool Exhibition Report*, pages 504–509. Springer International Publishing, Cham, 2016.