Kalpa Publications in Computing
Volume 3, 2017, Pages 96–105
RV-CuBES 2017. An International Workshop on
Competitions, Usability, Benchmarks, Evaluation,
and Standardisation for Runtime Verification Tools

COEMS — open traces from the industry *

Svetlana Jakšić[1,4], Martin Leucker[2], Dan Li[1,3], and Volker Stolz[1]

[1] Western Norway University of Applied Sciences, Bergen, Norway
{svetlana.jaksic,dan.li,vsto}@hvl.no
[2] Universität zu Lübeck, Lübeck, Germany
leucker@isp.uni-luebeck.de
[3] Guizhou Academy of Sciences, Guiyang, China
[4] University of Novi Sad, Novi Sad, Serbia

Abstract

The runtime verification community is still fragmented with non-interoperable specifications and tools. Within the EU H2020 project COEMS "Continuous Observation of Embedded Multicore Systems", we are contributing to the European Open Research Data Pilot that makes scientific data available to other researchers. We describe our first contributions and experience with the required data management and discuss technical issues such as metadata management, format and storage on practical examples. Based on our experience, we make suggestions on tools and formats for future RV Competitions.

1 Introduction

In the EU Horizon 2020 RIA project "COEMS — Continuous Observation of Embedded Multicore Systems" we have committed to the European Open Research Data Pilot. This requires us to contribute scientific data to a repository which makes it accessible to other researchers. The repository enables us and the others to refer to a particular data set, with recommendations on archival range from five to ten years. This timeframe is also the requirement to be able to obtain a Digital Object Identifier (DOI) for a data set.

In the domain of Runtime Verification, where the COEMS project aims to advance the state of the art in hardware-based monitoring, trace data will be directly useful to researchers that are working on verification and stream processing techniques, such as Complex Event Processing (CEP) [10]. It allows them to validate the precision and performance of their techniques, and to give them access to real-world scenarios that correspond to traces which have been observed in the wild by our industry partners. Obviously the trace data from our project can have significant scientific value that is worth to store and share with other researchers. We find other stakeholders have similar interests: the International Competition on Runtime Verification (RV Competition, since 2014) [2, 12] is in need of a stable repository for historic and future data

*Supported by the European Union's Horizon 2020 research and innovation programme under grant agreement no. 732016 and the EU COST Action IC1402 "Runtime Verification Beyond Monitoring (ARVI)".

to be used in the competition. In the *offline*-track of this competition, verification/monitoring tools read stored traces of events and analyse them.

In the following, we discuss our experience and design decisions in the creation of a data repository which is not only for depositing our trace data, but can also be used by other researchers, e.g. for the RV Competition. The remainder of the article is structured as follows: in the next section, we firstly list the requirements on the repository for the COEMS project. We then discuss the different alternatives to satisfy the requirements, our choice, and describe our evaluation of a particular setup. For this, we configure a test system, and play through the workflows of contributing data into the system, and accessing it. In the conclusion, we will revisit the shared requirements between the project and the runtime verification community, and argue that the chosen setup would equally well serve as the platform for the competition.

2 Requirements

In this section we elaborate the requirements for storing, accessing and managing the data generated within the COEMS project. The requirements are created according to the principles of making data findable, accessible, interoperable and reusable (FAIR), prescribed by the European Commission for projects participating in the Open Research Data Pilot. We also briefly review data published by RV Competition and discuss alternative choices to satisfy the requirements.

2.1 Trace format

In the domain of runtime verification, we expect to have a high volume of events with little data in each event. For example, in the CRV-14 competition, the set "OFFLINE/Team4" contains events of 48 bytes in size.

We expect the same of general tracing activities, e.g. in the area of concurrent programming: in such scenarios, event data mostly consists of the type of event, and its associated data, which very often is identified primarily by memory addresses (e.g. locking or unlocking of a mutex), and a thread identifier. Conceptually, the memory addresses (32 or 64 bit, 4 or 8 byte, respectively) are enough, and we do not need to communicate the actual data contained in the referenced data structure. Tools that work with this data must be able to quickly write or read from the event stream.

To the best of our knowledge, there is no commonly agreed on, particular, data format to capture trace of events on the processor level. For example, Intel Processor Traces uses its own binary format and provides a library to access captures [8]. Application-specific payload data contained in an event may in fact be beyond interpretation by anyone except the owner of the system that the data was obtained from. However, in order to share the data with the researchers around the world, the data repository should use a format that has a common understanding and acceptance.

The (Linux-) operating system community uses the *Common Trace Format* (CTF) [6] in some of its tools. CTF is a binary trace format designed to be very fast to write and already supported by tools such as Babeltrace[1], LTTng[2] and Trace Compass (Eclipse) for visualization. As events in a trace are not intended for consumption directly by humans, we will not focus on text-based formats such as XML, CSV or JSON (see [9, 12, 13] for a discussion of formats). But some information derived from a slice of relevant events may be presented to users, e.g.

[1] http://www.efficios.com/babeltrace
[2] http://lttng.org

as a counter example to a particular property that has been given in a property specification language of the respective monitoring tool.

Nonetheless, other stakeholders, such as the RV Competition, stored their trace data in the formats of XML, CSV and JSON in previous years. Thus, a certain degree of support for these formats is also necessary for our repository.

2.2 Storage

The trace data that we plan to contribute will vary drastically in size: on the smaller end, traces for basic validation of functionality may be between 100 and 1000 events, while much longer traces will be used to validate the performance of our hardware-based continuous observation technology. For example, an Intel processor can easily produce 500MB binary trace data per second, and ten times the size after decoding. These larger traces will be essentially limited by reasonable assumptions that we make about storage space and bulk transfer rates to consumers. For the RV Competition, some of the traces consist of just a few hundred events whilst others contain tens of millions of events. The traces varied in size from few KB up to tens of MB.

In general, we have the following options for storage: data could be hosted on dedicated (possibly virtual) infrastructure, or by a third party. Dedicated infrastructure may be hosted at a participating (academic) institution, or at a purchased service in a commercial data center. Hosting at a participating institution has the advantage of being entirely under control of the owner, but also entails the required commitment to software installation, maintenance and backups.

For practical reasons, we would like to avoid purchasing third-party services (either in the form of hardware or storage space), as the accounting for those expenses in an academic environment is complicated, especially if the resource is used by other parties than the one funding it. Fortunately, many academic institutions already have operating budgets for IT infrastructure, which will cover the necessary hardware and maintenance by local staff. Relying on an established institution also covers the eventuality of a service provider going out of business, and making any data inaccessible or at least find the need to re-publish it somewhere.

We considered online providers of data storage: version controlled storage such as GitHub, and the data-centric platforms such as Zenodo [16], a research data repository for the preservation and making available of research, educational and informational contents. It is hosted by CERN, and a by-product of CERN's need for extensive storage for data arising from particle physics experiments. Content may be uploaded free of charge by those without ready access to an organized data centre. Total file size limit per record is 50GB and the service is guaranteed for the next 20 years. Figshare [7], hosted by Amazon Web Services, offers similar services as Zenodo, but not free of charge. BIBSYS [4] in Norway offers hosting data, but requires agreements on the level of (national) academic institutions, thus placing a burden on entering into the system for the first time.

However, it sounds very attractive to host our data in one of these online repositories. As they are originally designed for general purposes, to archive documents and data in a wide variety of data formats, e.g. electronic copies of publications, statistics results of experiments, and images from fieldwork, all information has to be stored "as-is", and it is the users' responsibility to ensure data quality and provide some kind of data descriptions (metadata, recommended but not mandatory). These online repositories usually will not provide special support to particular data formats. In the COEMS project, we focus on handling trace data. We want the data not only to be stored and shared with the others, but also to have a mechanism to ensure data quality, and to provide more services for evaluating, converting, analyzing and reusing it.

Obviously, these systems may have difficulties in providing full support for the trace data.

We ruled out GitHub for similar reasons like the RV Competition, that is, the current restrictions on repository- and file sizes (1GB and 100MB), and we do not expect to version the trace data (stable/write once-traces, i.e. little benefit from using Git).

2.3 DOI and License

The DOI is an important mechanism for the data resources to be found and referred to. While it may not make sense to request a DOI for every data set published, DOIs should be registered for those data sets that are used in publications, and for those where demand indicates that other researchers intend to use them.

Traditionally, DOIs have been assigned to publications, for which publishers took the responsibility of ensuring availability for a decade or beyond. They are now increasingly used to reference data adopted in research as well. In this case, the role of the publisher is less clear, and we have found two general options: Digitial library services such as the Technische Informationsbibliothek TIB [15] and BIBSYS [4] offer assignment of DOIs to URLs, i.e. they do not host the data themselves. Some open data platforms already offer allocation of DOIs to uploaded data. This is for example the case for Zenodo and Figshare.

Unlike hosting of the data, providing a DOI is usually free of charge, but digital libraries may only offer services to particular groups. For example, TIB's service is free of charge to academic institutions in Germany, BIBSYS does the same for Norway. In the case that such a library service is used, it usually requires guaruantees on the availability of data. To qualify for a DOI at TIB, according to the "rules of good scientific practice" of the German Research Foundation (DFG), digital preservation and the accessibility of objects must be guaranteed for a minimum of ten years.

Contributed data to a repository can be covered under a number of different licenses. While many licenses are communicated just by attaching the license to the data, in some cases there may be the need to restrict or embargo access to certain data. In our setting, we do not see the need to cover the latter eventuality: whereas in different areas the actual data may be of interest, in our field of runtime verification it is rather the *methods* used to analyse data, and a particular data set is only a means to showcase a new analysis technique, and e.g. report the overall result or execution time of such an analysis. Therefore, we should restrict ourselves to in general liberal licenses that do not require access control on the repository for consumers of data, as we do not intend to collect sensitive data, which we define as containing personal data, or data that would disclose information that can be used to obtain remote access to partners' equipments.

In the publicly accessible repository of the RV Competition 2014, there is no licence on the data. Some of the data for 2015 is available under the MIT Licence and General GNU. For 2016 some data is under the Creative Commons Attribution Share-Alike and the MIT Licence.

For a system that should cover the needs of different communities, we need a system in which licences may be specified for each uploaded file. For some communities in the natural sciences, the main research result is data, detailing e.g. observations and their circumstances. As such, they have a need to embargo such data to a subset of visitors until e.g. reviewing of an article that presents this data is complete, or even until the article is published. Although the RV community intends to reference data sets in scientific works that will advance the state of the art, we do not directly see the need to technically support embargo periods. We expect advances will be within the *software* that processes the published data, and not the data itself. Likewise, we see no reason to artificially limit the life time of the data.

In addition, the European Commission recommends to have the open data repository registered to get the data discovery and directory services for the repository contents. It can be done at the Registry of Open Access Repositories (ROAR[3]) or the Directory of Open Access Repositories (OpenDOAR[4]).

2.4 Workflow

A data repository does not exist in itself. It lives through the contribution of data and through accessing of that data. Accesses are motivated through two main ways of discovery: firstly, through reference from an article which makes use of a data set, and secondly, through discovery via indexing. Furthermore, to ensure that the repository is not abused, we need a way of curating the data, and making sure that each data set is submitted with a complete and accurate set of metadata.

For most contributed data sets, a mechanism to upload the data through the internet (a web interface or any other file transport protocol) would be sufficient. For large traces, whose transfer would be severely affected by the transmission rate between the client and the repository, one could envision using storage media such as USB sticks or other external devices which are somehow sent to the repository maintainer. Clearly, the latter alternative could place a burden on the maintainer if frequently exercised. Fortunately, as discussed above in the section on Storage, we do not expect this to be often the case, due to planning on staying at the lower end of trace sizes.

After the data and its associated metadata have been uploaded to the repository, there should be a possibility for the curators to review the submission. For enhanced flexibility, and with respect to smaller communities, there should be an option to make the data available even before review. However, as we have seen on the section on DOIs, a certain degree of completeness must be enforced if contributors expect to reference a data set through a DOI in their publications. This means that any data set for which a DOI is requested is required to undergo review.

For example, the benchmarks at RV Competition allow open submissions of data that must conform to the given rules. In that case, there are 3 or 4 organisers who review the benchmarks uploaded to the servers by the participants. For the COEMS project, as contributions come from within the project, we just need a quality assurance process for completeness as well, and checks to catch mistakes.

After reviewing/quality assurance, the data should be publicly available and DOIs can be allocated. If need be, the system should offer a way to publish revised data. Updates to an existing data set are most likely in the case where we subsequently provide post-processed traces that serve to demonstrate a particular effect, and where we have been able to trim down the number of events (and hence the file size) of the trace without eliminating the effect.

3 Handling Traces of COEMS

In this section, we present the solution we have prototyped for the COEMS project. We explain the trace format we settled on, the metadata we record with each trace, motivate the storage solution, and look at a particular software, DKAN, that supports us in the workflow.

[3] http://roar.eprints.org/
[4] http://www.opendoar.org/

3.1 Trace format and metadata

Within the project, we will offer two kinds of trace data: i) synthetic data that we use in the lab to validate our tools and techniques, and ii) (sanitized) relevant trace data from our industry partners that showcase the real-world scenarios that we tackle in the scope. The COEMS project mostly deals with the lowest level of tracing possible: processor/instruction-level tracing. Through the *trace interface* available on modern processors, the COEMS hardware will receive the processor trace for specifically set watchpoints in a binary.

The trace format we finally decided to adopt for the COEMS project is CTF, as we expect to both generate and consume trace events at rates up to 1 GHz, and hence require support for fast writing and reading. Furthermore, in the project we already have an industry partner with experience in using CTF, which makes this a suitable technology for our purposes. CTF can also serve as an intermediate exchange format between other tools, as a C-library and e.g. Python-bindings for scripting are readily available and can convert between one format and the others.

Each CTF trace contains as metadata a textual description of the binary layouts of all the other streams in the Trace Stream Description Language which makes the data entirely self-described. In the accompanying textual description, we will refer to commonly used concepts from operation systems concepts, such as memory addresses, thread- and task ids, and operations thereon.

Most of COEMS data will be in the CTF format. There may still be some data stored as XML, JSON and CSV, or directly in the processor's original binary trace format, and its decoded formats, so we need a publishing platform that supports multiple types.

We plan to use the TIB services to obtain stable DOIs for some of our data, since the ISP at the University of Lübeck as hosting institution qualifies for its free services. DOI creation through TIB requires a mandatory set of fields in the metadata (Identifier, Creator, Title, Publisher, Publication Year, Resource Type). Additionally, we will use the recommended fields Contributor, Date and Description, as well as the optional fields Size, Format, Version and Rights.

Data contributed by the COEMS project will be made available under the permissive Creative Commons "Attribution Non-Commercial No Derivatives" (CC BY-NC-ND) license [5] that places minimal burdens on interested parties (see [1] for a comparison of different licenses). This allows other developers, e.g. to obtain and consume our data in their tools, but restricts e.g. redistribution of COEMS data directly or in modified form.

3.2 Implementation of the data repository

We have chosen to use the infrastructure in operation at the University of Lübeck (UZL). The infrastructure at UZL already provisions storage, both for flat files and version controlled data, as well as virtual servers. The operation of this service (day-to-day maintenance and backups) is thus already budgeted for. Furthermore, we consider both the infrastructure and the institution to be stable for the foreseeable future (in the 5–10 year range), which ensures that data will be maintained beyond the duration of the COEMS project. This longevity of data is also needed in order to obtain DOIs. UZL, as a German academic institution, will be responsible for obtaining DOIs from TIB.

Based on the previous discussions, we decided to implement our open data repository using DKAN[5], an open-source data management platform built on the top of Drupal, one of the

[5] http://getdkan.com/

popular content management systems. DKAN features an easy-to-use interface for cataloging, maintaining and visualizing data, that allows organizations to easily share their data with the public. It is also possible for DKAN to refer to data resources of other data repositories as remote files, or collect these data into DKAN database by data harvesting if that is allowed by their licences. The support for different open data licenses is already embedded in DKAN. It is also possible to define complex workflows in DKAN that allow data published by its creators to be reviewed and censored by the editors before publication.

In addition, DKAN provides a set of REST APIs to explore, search, add, describe, tag, group its data so it can automatically interact with open data systems or management systems of other organizations. Additional functionalities can be easily added to DKAN by installing new modules, either developed by the end users, or simply selected from thousands of contributed modules from the Drupal community. All those made DKAN an ideal tool for building our open data portal.

The DKAN portal is installed on an Linux server (running CentOS) provided by UZL to take the advantage of UZL's infrastructure, such as stability and regular backup. Considering the possibility to migrate the system to cloud platform in the future, we deployed the portal with two Docker containers, one is for the DKAN (version 7) itself and the other for its MySQL database system. The portal is already available on `http://dkan.isp.uni-luebeck.de`. Figures 1 shows the main page of the data portal.

With the portal, a registered user can, basically, create a dataset for the trace data she wants to publish, give it a description and choose an open data license. Then the actual trace data files can be successively uploaded as resources of the dataset. After reviewed and approved by the administer, the dataset can then be searched and accessed by the public.

However, the main trace format of COEMS project is CTF which is not natively supported by DKAN. Based on the Babeltrace Python library, we developed a CTF micro-service that provides a set of REST APIs to convert the trace data to/from CTF and other data formats, such as CSV, JSON and XML. Through the standard HTTP protocol, the micro-service can receive the events sequentially generated by users' programs running in a remote machine, and finally save the events as CTF resources in the server. Furthermore a client program can download CTF resources as event streams through fast socket-io connections, for analysis or simulation purposes. The CTF micro-service is also deployed in a Docker container, and bind to the DKAN to provide with the CTF support. Figures 2 shows an example dataset which contains the trace data in CTF format (as a .zip file) and the C program file that generates the trace data. These two files are directly uploaded to the portal by the user. Then the CTF APIs are used to convert the trace data into different formats, and retrieve the metadata of the CTF data.

We also plan to further extend DKAN with the functionalities to automatically apply/-generate the DOIs, as well as the SHA256 for data resources. On the project web-site `https://www.coems.eu`, we will provide an index page of all data made available. Each item will link to an information page about the respective data set, showing the meta data and providing the download URL for the actual data.

4 Conclusion

We have presented our view on the requirements on an open data repository for traces from our EU RIA COEMS project and the Runtime Verification competition. We discussed existing technologies and services that are already available for researchers to make their data findable, accessible, interoperable and re-useable (FAIR). Based on these requirements, we have chosen

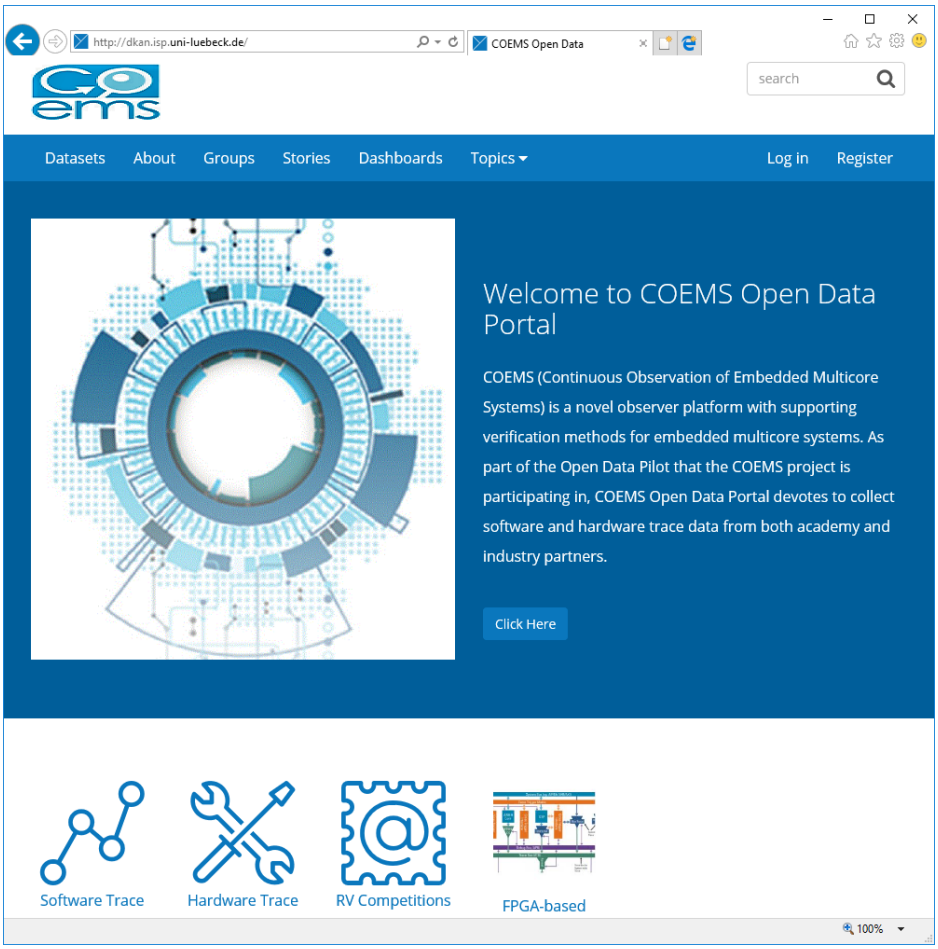

Figure 1: Front page of COEMS open data portal

and evaluated a particular setup in the form of the DKAN open data catalog content management system, running on a self-hosted virtual machine. We prefer the increased flexibility of such a set-up over choosing a (free) third-party hosting services such as Zenodo: in our own setup, we are for example able to offer micro-services that we have developed ourselves.

Other projects which operate with data similar to those that we have in COEMS and the RV Competition also chose to host their data on the servers of academic institutions. For example, the data collected within Malware Capture Facility Project [11], which captures, analyses and publishes real and long-lived malware traffic, is stored in a repository hosted by the institution in charge of the project. The backup of the data is stored with an external provider. The Termination Competition [14] runs on StarExec[6] platform which serves them as a benchmark (problem) library. The data base of problems is again stored in a repository hosted by one of the organising academic institutions. Apart from storage, the Termination Competition shares further challenge with the Runtime Verification competition that we have

[6] https://www.starexec.org/

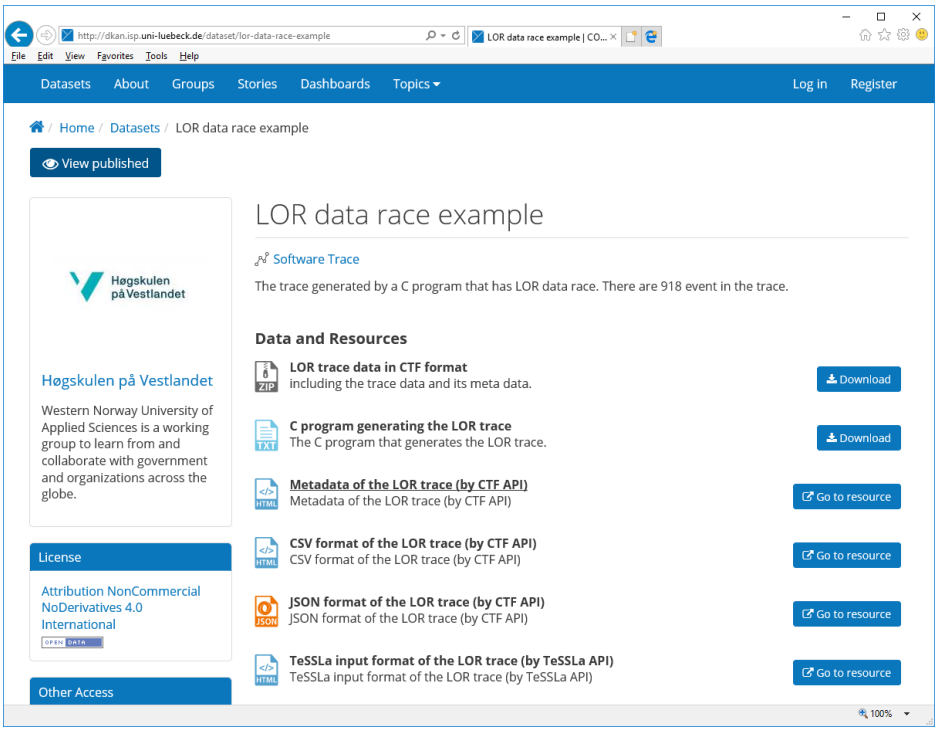

Figure 2: A dataset example.

not discussed here as it is out of the scope of the open data-focus of our article: since a goal of the competitions is also to compare execution time of the tools used to analyse the data, a uniform execution environment in terms of available software, to be able to subsequently execute runtime experiments on identically configured systems. The interested reader may refer to [3] for a discussion on technical challenges how to measure such experiments.

The particular nature of the COEMS data is one of the reasons which moved us to search for the solutions beyond already available services, such as Zenodo. We found that the self-hosting at UZL with the use of DKAN as management platform fulfils all the requirements. As shown in Section 3.2 we can register users of the portal which then may upload the data in the format of their choice, curate the data if needed before the publication, add tools which are essential for the COEMS specific data, liberally choose metadata we want to present, and choose different licences for individual traces. We plan to further develop the present implementation.

The data that we produce within COEMS will hopefully be useful to researchers working on runtime verification. Vice versa, we may find it useful to test COEMS tools on RV Competition trace data. We argue that requirements for the repositories of COEMS and RV Competition coincide in many aspects. In particular, they both need a stable solution which enables them to use accumulated trace data and upload new datasets, the size of the storage needed is similar, the data may be in several trace formats and published under different licences, and workflow should enable inspection and revision of the data.

We find that hosting data at UZL could be the right choice for both, COEMS and RV Competition, and that the implementation which we have presented here is suitable for the

repository of the competition as well.

References

[1] A. Ball. How to License Research Data, DCC How-to Guides. http://www.dcc.ac.uk/resources/how-guides, 2014.

[2] E. Bartocci, Y. Falcone, B. Bonakdarpour, C. Colombo, N. Decker, K. Havelund, Y. Joshi, F. Klaedtke, R. Milewicz, G. Reger, G. Rosu, J. Signoles, D. Thoma, E. Zalinescu, and Y. Zhang. First international Competition on Runtime Verification: rules, benchmarks, tools, and final results of CRV 2014. *Intl. J. on Software Tools for Technology Transfer*, pages 1–40, 2017.

[3] D. Beyer, S. Löwe, and P. Wendler. Benchmarking and resource measurement. In B. Fischer and J. Geldenhuys, editors, *22nd Intl. Symp. on Model Checking Software (SPIN 2015)*, volume 9232 of *LNCS*, pages 160–178. Springer, 2015.

[4] BIBSYS. http://www.bibsys.no/en/, 2017. Last accessed June 2017.

[5] Attribution Non-Commercial No Derivatives (CC BY-NC-ND). https://creativecommons.org/licenses/by-nc-nd/4.0/, 2017. Last accessed June 2017.

[6] M. Desnoyers. Common Trace Format (CTF) Specification (v1.8.2). http://diamon.org/ctf/#specification, 2017. Last accessed June 2017.

[7] Figshare. http://figshare.com/, 2017. Last accessed June 2017.

[8] Intel. Intel PT. https://software.intel.com/en-us/blogs/2013/09/18/processor-tracing, 2017. Last accessed June 2017.

[9] S. Jaksic and V. Stolz. COEMS Deliverable D7.2: Initial Data Management Plan, April 2017. To appear.

[10] D. C. Luckham. *The power of events – An introduction to complex event processing in distributed enterprise systems*. ACM, 2005.

[11] Malware Capture Facility Project. http://mcfp.weebly.com/, 2017. Last accessed June 2017.

[12] G. Reger, S. Hallé, and Y. Falcone. Third international competition on runtime verification. In Y. Falcone and C. Sánchez, editors, *Runtime Verification: 16th International Conference (RV 2016)*, volume 10012 of *LNCS*, pages 21–37. Springer, 2016.

[13] G. Reger and K. Havelund. What is a trace? a runtime verification perspective. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications: 7th Intl. Symp. (ISoLA 2016)*, volume 9953 of *LNCS*, pages 339–355. Springer, 2016.

[14] Termination Competition. http://termination-portal.org/wiki/Termination_Competition/, 2017. Last accessed June 2017.

[15] Technische Informationsbibliothek (TIB). https://www.tib.eu/en/publishing-archiving/doi-service/doi-registration/, 2017. Last accessed June 2017.

[16] Zenodo. https://zenodo.org/, 2017. Last accessed June 2017.