# Improving Trace Generation and Analysis for Medical Devices

Yoann Blein, Arnaud Clere, Fabrice Bertrand, Yves Ledru, Roland Groz, Lydie Du Bousquet

# Improving Trace Generation and Analysis for Medical Devices

***Yoann Blein[1], Arnaud Clère[2], Fabrice Bertrand[3], Yves Ledru[1], Roland Groz[1] and Lydie du Bousquet[1]***
***1 Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, F-38000, Grenoble, France***
***Email: firstname.lastname@imag.fr***
***2 MinMaxMedical, Grenoble, France***
***Email: arnaud.clere@minmaxmedical.com***
***3 Blue Ortho, Grenoble, France***
***Email: fabrice.bertrand@blue-ortho.com***

## Abstract

Y. Blein, A. Clère, F. Bertrand, Y. Ledru, R. Groz, and L. du Bousquet. *Improving TraceGeneration and Analysis for Medical Devices* In IEEE International Conference on Software Quality, Reliability and Security Companion, (QRS-C 2017), Pages 599-600, Prague, IEEE, Jul 2017

# Improving Trace Generation and Analysis for Medical Devices

Yoann Blein*, Arnaud Clère†, Fabrice Bertrand‡, Yves Ledru*, Roland Groz* and Lydie du Bousquet*
* *Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, F-38000, Grenoble, France*
*Email: firstname.lastname@imag.fr*
† *MinMaxMedical, Grenoble, France*
*Email: arnaud.clere@minmaxmedical.com*
‡ *Blue Ortho, Grenoble, France*
*Email: fabrice.bertrand@blue-ortho.com*

## 1. Introduction

A new generation of medical devices emerges to support increasingly more complex medical decisions and procedures. These Medical Devices (MD) combine data from novel sensors and existing modalities like scanners with elaborate software processing to assist caregivers in the same way Flight Management Systems help a pilot flying planes. For instance, Blue Ortho's MD allows performing Total Knee Arthroplasty (TKA, i.e. the placement of a femoral and tibial knee prosthesis) more precisely, potentially dividing the number of revisions by two [1]. Unfortunately, innovation and safety of such MD is hindered by the current software verification practices of the MD industry. Some formal methods have demonstrated their effectiveness for decades in the most safety-critical industries (defense, air, space, railways, nuclear power plants), but they failed to gain broader adoption.

MD provide an interesting opportunity for improving verification processes, not only during the manufacturers verification activities, but also in their clinical context for their whole lifetime. Indeed, these powerful systems can easily be equipped with elaborate tracing facilities, and exploiting their execution traces and sensor data can provide us with an unbiased and precise understanding of their behaviour in the field. Consequently, model-based verification of execution traces, also known as Runtime Verification, seems particularly fit to the task at hand. This strategy has, to our knowledge, never been applied to the MD industry, and a success in this highly visible domain could open the way to a broad range of formal methods and tools in numerous domains where only a small part of requirements are considered critical. For the MD industry, we believe runtime verification can advance the evaluation of MD safety and performance in the field. Although the generation of MD execution traces is a standard practice for a posteriori understanding when problems have been experienced, their systematic exploitation is new to the medical devices industry. A model-based, semi-automated analysis of these data provides a mean to verify that requirements are satisfied in any conditions, to validate the hypotheses made on the environment, and to understand the usage of the MD in a perspective of improvement. It can also bring disruptive progress in the postmarket surveillance of such systems which is usually limited to ineffective and informal followups of the device users. For instance, it will allow validating safety-related assertions on users behaviour [2] or hardware longevity [3]. Furthermore, some regulatory administrations like the FDA are interested in evidences of safety, i.e. factual data on the actual device.

The MODMED project gathers Blue Ortho, a MD manufacturer, MinMaxMedical, a software provider for MD and the LIG laboratory to propose a solution to model-based verification of execution traces of MD in two folds: a structured trace library, designed with low adoption costs in mind, and an intuitive property specification language along with its implementation of verification tools.

## 2. Trace Library

Software execution is almost always traced to facilitate its development, however such traces are difficult to use for formal verification purposes because they are often incomplete and they lack abstraction.

Existing trace libraries such as ETW[1], journald[2], or log4j[3] are focused on performance or flexibility at runtime but very limited in structuring the data of logged messages. When not limited to ambiguous text or primitive data types, those libraries require more work than most developers can spend to format structured data and describe it in external data schemas. The most common trace attributes (severity, timestamps, etc.) are usually standardized as with syslog[4], but there should also be an easy way to standardize the most prominent domain-specific data types and attributes (patient id, RMS, cut plane, medical terms, etc.) wherever they can be in the event data.

Few research is dedicated to improving trace generation tools and methodologies. Aspect-oriented programming tools like AspectJ [4] are fitted to automatically insert tracing code, but a survey of MinMaxMedical customers showed

1. https://msdn.microsoft.com/library/windows/desktop/aa363668.aspx
2. https://www.freedesktop.org/software/systemd/man/sd_journal_print
3. https://logging.apache.org/log4j
4. https://tools.ietf.org/html/rfc5424

that medical software project managers are reluctant to use tools modifying the generated executable. They would prefer approaches like LogEnhancer [5] that focus on improving traces quality before quantity. From their standpoint, the ideal solution would be to improve traces without even having to change trace points.

MODMED embraces this goal and proposes a C++ structured trace library that can greatly improve traces structure with existing trace points, and then, easily trace complex event data types without complicating trace points. This library[5] extends Qt logging[6] to:

- automatically collect and log event metadata such as category, function and abstract message format without requiring extra developer work or log space.
- structure event data using the source language trace expression and type system.
- unambiguously format event data in a TSV+JSON format fitted to human exploration and automated analysis (without having to maintain an external data schema).
- allow on-purpose classification of events based on their severity, category and abstract format message.

## 3. Expression and Evaluation of Properties

Trace verification can be automated when requirements are expressed as formal properties. Our main objective is to design a property specification language in such a way that it should be readable by quality engineers and easy to manipulate for software engineers when writing specifications. For this purpose, we analysed a large set of requirements for the aforementioned TKA to determine events of interests and how requirements rule the ordering of events. The result of our analysis is a language designed to concisely and intuitively capture the requirements for interventions and the desired behaviors of MD[7].

The language is an adaptation of the specifications patterns proposed by Dwyer et. al. [6]. Temporal properties can be expressed through a intuitive pairing of scope constraints and patterns. Scopes delimit the intervals of a trace where a property should hold, and patterns capture a requirement. Scopes may be freely nested and offer much more flexibility than the original specification patterns. Additionally, properties can be time-constrained through timed scopes, which not only delimits trace intervals according to events, but also according to elapsing time. Moreover, the language was built upon the idea that event parameters should have a central role. As a consequence, each time an event is referred to, its parameters may be extracted and constrained according to events seen previously.

Let us consider an example of requirement for TKA and its formalisation in the proposed language:

> *Before starting an acquisition phase, all the necessary sensor types must have been detected.*
>
> *If several lists of required sensor types are present, only the last one before starting must be considered.*

The beginning of the acquisition phase, a new sensor detection and the list of required sensors are reflected through the events `StartAcquisitions`, `SensorDetected` and `SearchSensors`, respectively. The type of a sensor is given by the `type` parameter of the second event and the list of required trackers by the `types` parameter of the last one. The requirement can be formalized as follows:

```
before each StartAcquisitions,
given last SearchSensors ss,
forall type in ss.types,
occurrence_of SensorDetected sd
  where sd.type == type
```

Although we believe that such properties are expressed in user-friendly manner, we are actively working on tools that will help express and understand them.

## 4. Conclusion

Prototype versions of the trace library and the language interpreter have been released under a free license[5]. The bridge between the two components is not automated yet, but they can be used as standalone tools. The verification of a few properties over a corpus of traces recorded during real surgeries already produced interesting results. For instance, it appears that one out of two surgeries includes at least one accidentally repeated click on the device's touchscreen, which impairs its usability. The frequentness of this defect, attributed to a hardware flaw in the device's stand, would not have been discovered without systematic and automatic verification of the execution traces.

## Acknowledgments

This work is funded by the ANR project MODMED (ANR-15-CE25-0010).

## References

[1] C. Schnurr, I. Güdden, P. Eysel, and D. P. König, "Influence of computer navigation on TKA revision rates," *International orthopaedics*, vol. 36, no. 11, pp. 2255–2260, 2012.

[2] A. Blandford, G. Buchanan, P. Curzon, D. Furniss, and H. Thimbleby, "Who's looking? invisible problems with interactive medical devices," in *Proceedings of the First International Workshop on Interactive Systems in Healthcare*. ACM Special Interest Group on Computer-Human Interaction, 2010, pp. 9–12.

[3] R. Sipos, D. Fradkin, F. Mörchen, and Z. Wang, "Log-based predictive maintenance," in *KDD'14*. ACM, 2014, pp. 1867–1876.

[4] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold, "An overview of AspectJ," in *ECOOP*, ser. Lecture Notes in Computer Science, vol. 2072. Springer, 2001, pp. 327–353.

[5] D. Yuan, J. Zheng, S. Park, Y. Zhou, and S. Savage, "Improving software diagnosability via log enhancement," *ACM Trans. Comput. Syst.*, vol. 30, no. 1, pp. 4:1–4:28, 2012.

[6] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *ICSE*. ACM, 1999, pp. 411–420.

5. https://forge.imag.fr/projects/modmed/
6. https://doc.qt.io/qt-5/qmessagelogger.html
7. https://sites.google.com/a/minmaxmedical.com/modmed/deliverables