WP6/D2
Method and Tools Study

Technical Report

Contributors:

Yves Ledru	LIG	Author
Lydie du Bousquet	LIG	Author
Arnaud Clère	MinMaxMedical	Author
Fabrice Bertrand	Blue Ortho	Author
Mohamad Hamamah	LIG	Author

Partners:

Partly funded by:

Contents

1. Introduction 3
2. Evaluation of modmedLog C++ trace library by MinMaxMedical 4
3. Evaluation of ParTraP and tools 5
3.1. Qualitative evaluation by Master's 2 students 5
3.2. Quantitative Evaluation of ParTraP 12
4. Overall evaluation of the approach and tools by Blue Ortho 16
5. Conclusion 17
Appendices 18
1. Multiple-choice questionnaire for the students (in French) 18

1. Introduction

This document features elements of evaluation about the approach and the tools developed by the MODMED project. These elements are provided by the partners of the project.

The first section is an evaluation of the Modmedlog C++ trace library, performed by MInMaxMedical.

The next two sections feature the results of two evaluations of ParTraP performed by master's students. The first one corresponds to a qualitative analysis performed after a tutorial on ParTraP given to Master's 2 students following the Advanced Validation Techniques course. The second one is a quantitative evaluation performed by a Master's 1 student during an internship at LIG.

The last section gives an overall evaluation of the MODMED approach and tools by Blue Ortho.

2. Evaluation of modmedLog C++ trace library by MinMaxMedical

Understanding the structure of traces “a posteriori” is hard, and MinMaxMedical experiments with TKA traces confirm this. Some simple properties can be verified with Python standard libraries if one invests enough time to parse unstructured traces up to the minimum required for the analysis at hand. However, it remains hard to draw reliable conclusions with this ad-hoc approach because obtained results require manual verification of extracted data (which is easy only for a reduced dataset), and sometimes, it is necessary to verify that no relevant data was ignored (which is usually much harder).

In effect, the modmedLog approach seems necessary to produce reliable analyses. As a reminder, this approach consists in extracting a maximum of data from source code to generate at runtime more explicit traces (eliminating numerous sources of ambiguity), optionally with more data, and more importantly with more event metadata, allowing to reliably distinguish all traced events. As a matter of fact, MinMaxMedical convinced most of its customers to use the commercial trace library (WP2/D2). Even if the MODMED project’s schedule and Blue Ortho’s technical constraints did not allow us to evaluate the modmedLog trace library in the course of the development of a new Blue Ortho SCPM, MinMaxMedical obtained feedback from other customers that are also SCPM manufacturers.

Regarding modmedLog’s ability to generate structured traces with few efforts from the developers: Surgivisio, instrumented their SCPM with WP2/D2 with 3782 tracepoints in a matter of 9 months. Generated traces are typically 10 times larger than those from TKA (300MB vs 30MB). Their JSON format can be directly analyzed by numerous tools. The approach being new to the developers, we still find data concatenated in text that need to be processed manually, but this is limited to only 1,5% of all the tracepoints which denotes that most event parameters can be conveniently handled by WP2/D2. In order to handle these exceptions, the initial approach was complimented with a tool to extract and catalog tracepoints in source code which allowed to highlight these deficient tracepoints to 1/ determine the associated data structure in existing traces and, 2/ guide a gradual improvement of these tracepoints. This tool also allows to identify abstract events (i.e. similar events for a given analysis) which occur at various places in executable code, which is in general impossible to do at runtime.

Regarding modmedLog’s performance: Orthotaxy, a MinMaxMedical customer which develops a surgical robot was experiencing insufficient performance from WP2/D2 to trace hardware Input/Outputs. The last WP2/D1 version increases performance by an order of magnitude, that is again improved by another order of magnitude in an experimental modmedLog prototype named « QBind » which will be integrated to the next commercial version WP2/D2 of MinMaxMedical InCAS Core library (version 2).

Regarding the dissemination of our approach to other industries: From our discussions with the Qt community (a de-facto standard framework for developing embedded devices in C++), we see two main barriers to adoption:

1. Outside of regulated industries, the usefulness of automating runtime traces verification is not obvious.
2. The relevance of processing the same data uniformly to generate different trace formats looks suspicious considering a/ the plethora of different formats and b/ the technical limits of existing solutions to the serialization problem.

Nonetheless, the « QBind » prototype shows that it is indeed possible to generate structured traces in formats as different as XML, JSON, CBOR and QDataStream (Qt proprietary format), and that these formats do not differ in their ability to express the same trace data, but only in the compromise they offer between performance, interoperability, usability and expressive power. For instance:

- System libraries based on « statically-defined » tracepoints offer the best performance at the expense of usability (necessity to maintain a data schema outside of traces and to use a dedicated code generator)
- QDataStream format, complimented with native handling of UTF-8 text, would attain the performance of the aforementioned libraries without their restrictions, at the expense of interoperability (a consumer must perfectly know the schema of produced data and errors cannot always be detected)
- CBOR format offer, in our opinion, a very good compromise between performance and expressiveness for structured traces at the expense of requiring users to use a dedicated tool to visualize traces.

To summarize, the modmedLog library is an original and relevant contribution for structured traces in particular, and for data serialisation/deserialisation in general. Its broader adoption remains subject to the integration in a broader ecosystem, like that of the Qt community, to address needs that remain secondary outside of regulated industries. MinMaxMedical thinks that the need for structured traces will grow in the future to 1/ better control software systems that become more and more heterogeneous, and 2/ better understand the actual usage of software in the field to guide their continuous improvement.

3. Evaluation of ParTraP and tools

3.1. Qualitative evaluation by Master’s 2 students

We took the opportunity of the Master’s 2 course on “Advanced Validation Techniques” to give a lecture on logging and ParTraP, followed by a practical session where students had the opportunity to use the ParTraP tools and were given several tasks to perform.

The students follow the Master’s 2 degree in “Génie Informatique”, which is aimed at educating software engineering professionals.

The ParTraP tutorial

The lecture lasted 1h30 and was supported by 39 slides. 17 slides were dedicated to logging, illustrated by the java.util.logging framework (standard logging framework provided in the Java distribution). 22 slides were dedicated to ParTraP and introduced:

- The JSON format supported by ParTraP,
- Unary properties of ParTraP (absence_of, occurrence_of),
- Unary scopes (before and after),
- Nesting of scopes,
- Combination of properties using Boolean operators,
- A classification of properties into required/assumed/usage properties,
- A classification of stages in the life-cycle where properties should be checked.

The lecture also gave a survey of additional features of ParTraP, such as binary operators and scopes, scopes involving physical time, or Python expressions in where clauses.

The lecture was followed by a practical session (1h30) where several tasks were performed by the students:

- Trace generation using the java.util.logging framework,
- Evaluation of a ParTraP property on several traces,
- Expression of three properties in ParTraP,
- Understanding the meaning of a given property.

The ParTraP properties only involved unary operators and scopes, in order to avoid overwhelming students with redundant constructs. This choice does not reduce the expressiveness of the language since binary operators and scopes are defined in terms of combinations of the unary ones. It focusses students on the concepts of the language.
The students were provided with command-line versions of the ParTraP tools. They also were given the URL of the eclipse update site, for those students who wanted to experiment with a more elaborated environment.

Evaluation by the students

After this tutorial and practical session, the students were provided with a simple questionnaire, and were asked to fill it in immediately after the tutorial. The questionnaire was anonymous.
The questionnaire included 4 multiple-choice questions:

1. What was the last question you answered?
2. Do you intend to use a logging system in your future practice of software development?
3. How easy/difficult is it to understand a ParTraP property?
4. How easy/difficult is it to write a ParTraP property?

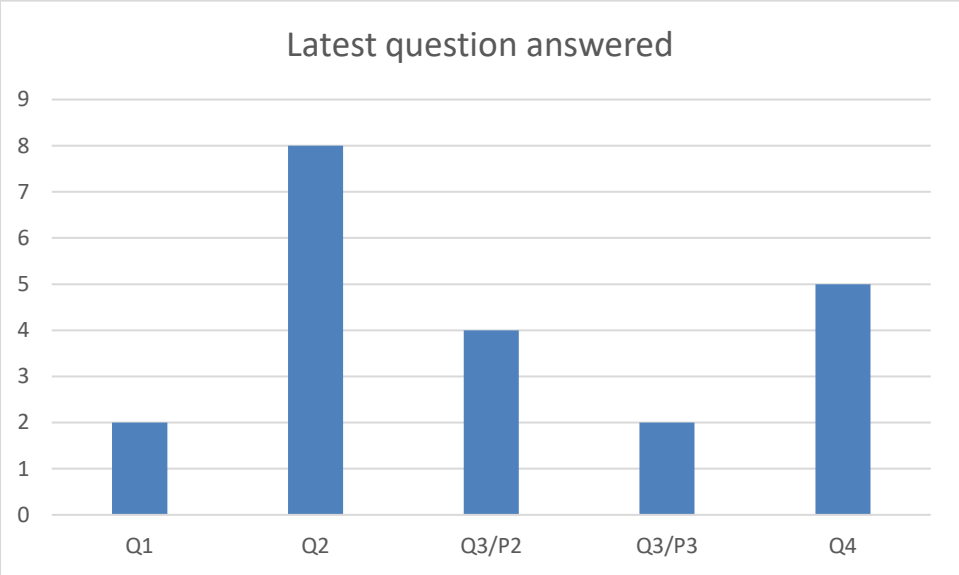
For questions 2-4, the students had four choices, ranging from a very positive to a very negative answer. The detailed questionnaire (in French) is given in appendix 1.

21 students answered the questionnaire. 31 students are registered for this course, but average attendance was around 24. So, we can conjecture that most of the students who attended the practical session did answer the questionnaire.

The first question states how far the students went in the practical session. The answers are summarized in the following table. Two students stopped after the first question. Five students answered all the questions. Actually, only half of the students tried to express properties with ParTraP.

Question 1 : nb of questions answered	
Q1	2
Q2	8
Q3/P2	4
Q3/P3	2
Q4	5

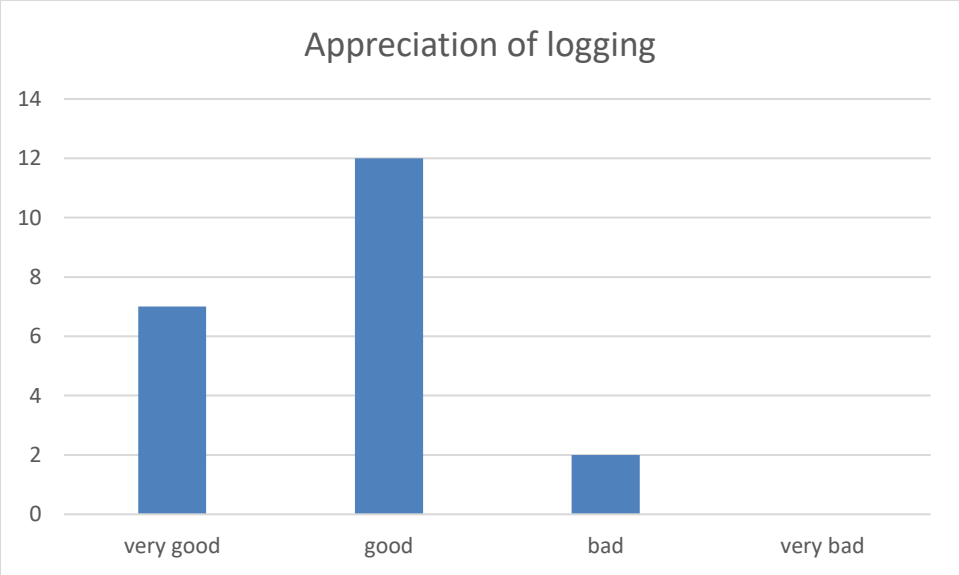
The following histogram corresponds to this table:



The second question was concerned with the appreciation of logging tools, without connection to ParTraP. 19 students (out of 21) had a positive appreciation of logging techniques and declared they are ready to use them. 7 of them are even proactive and intend to use such techniques at their own initiative.
These results are not surprising since these students choose to follow the course on advanced validation techniques, and are thus open to such practices.

Question 2 : Appreciation of Logging	
very good	7
good	12
bad	2
very bad	0

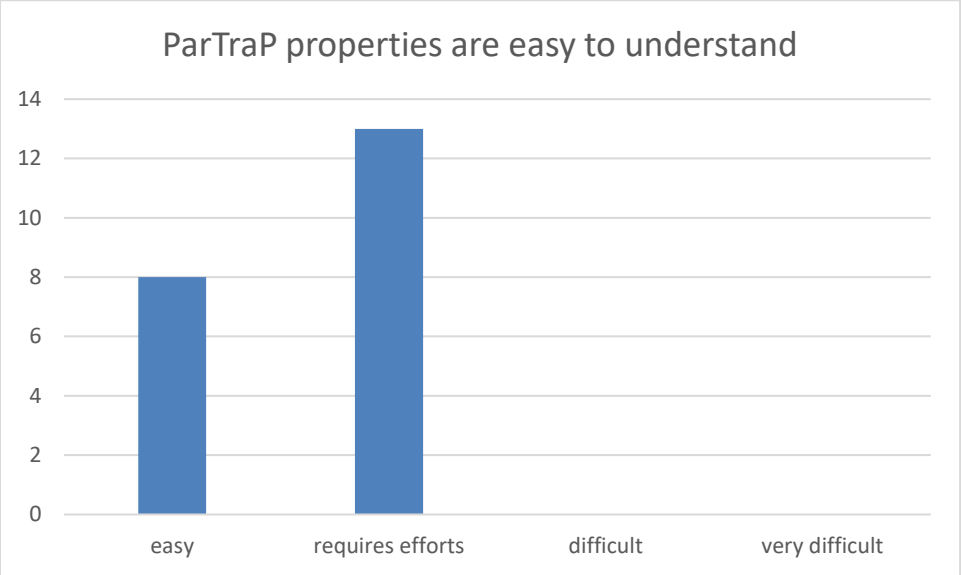
This gives the following histogram.



The next question was about easiness to understand ParTraP properties. Here all the students had a positive appreciation. This tends to show that ParTraP properties are readable with some effort, or at least that they don't frighten their readers. We will see that this impression is confirmed by the course final exam.

Question 3 : ParTraP properties are easy to understand	
easy	8
requires efforts	13
difficult	0
very difficult	0

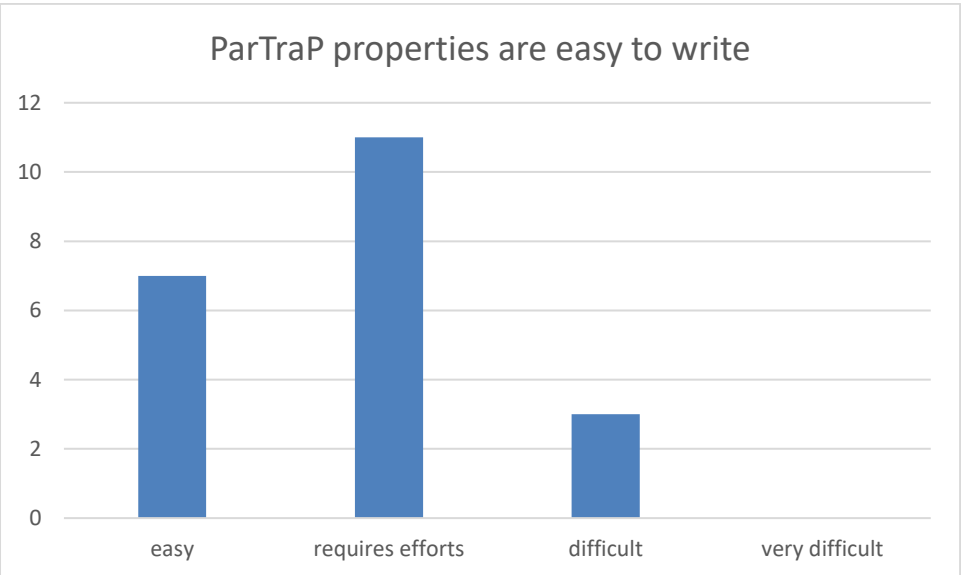
This gives the following histogram.



Finally, the last question was about easiness to write ParTraP properties. It must be noted that all students answered the question although half of them had not answered Q3/P2 which was the first question related to writing ParTraP properties. Only 3 students considered that writing ParTraP properties was difficult.

Question 4 : ParTraP properties are easy to write	
easy	7
requires efforts	11
difficult	3
very difficult	0

This gives the following histogram.



First Conclusion

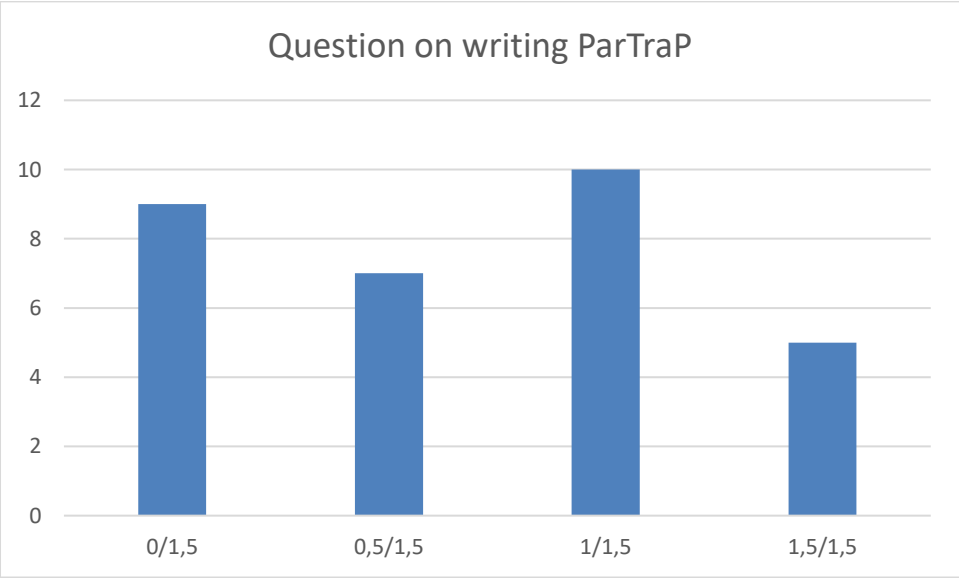
This tutorial about ParTraP which involved a course and a practical session went fine. The appreciations of the students were positive both about the easiness to read and write ParTraP.

Evaluation of the students at their exam

Following this tutorial, two questions were included in the final semester exam of the "advanced validation techniques" course. Since the students had mainly the opportunity to practice ParTraP during the tutorial, we decided to copy/paste two questions of the practical session into the exam. The first question was intended to evaluate how students write ParTraP properties. It was the exact copy of Q2/P2, which states the occurrence of two kinds of events. The second question was the copy of Q4, i.e. a question on understanding ParTraP properties. The property was an absence property involving nested scopes. The first question was evaluated on 1,5pts. The second question was evaluated on 1pt. The total amount of points for the exam was 20pts. So the ParTraP questions were worth 1/8th of the points. 31 students participated to the exam. This means that 10 of them did not answer the questionnaire, and presumably 7 to 10 of them did not attend the tutorial.

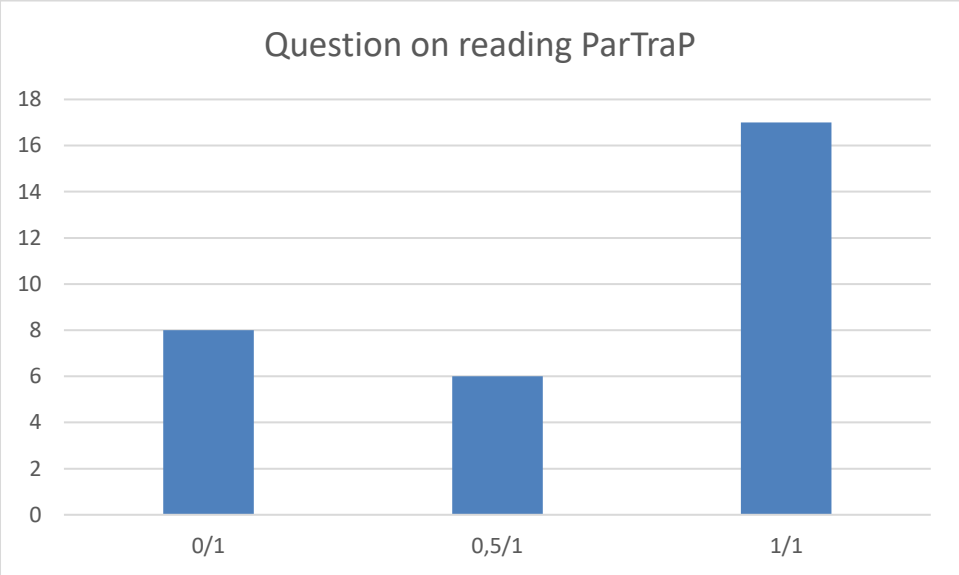
The question on writing ParTraP was expected to be the most difficult one. Nevertheless, 15 students succeeded this question, i.e. got 1/1,5pts or 1,5/1,5 points.

Question on writing ParTraP	
0/1,5	9
0,5/1,5	7
1/1,5	10
1,5/1,5	5



The second question, related to reading skills, had even better result. 17 students got the highest score, and 6 students got 0,5/1. Only 8 students failed on this question. This is a promising score since it corresponds to Q4 and only 5 students had reported that they had done Q4 during the practical session.

Question on reading ParTraP	
0/1	8
0,5/1	6
1/1	17



Conclusion

The results of this exam show that students are able to read ParTraP formulae after a short tutorial. Moreover, half of the students were also able to express satisfactorily ParTraP properties.

The questionnaire shows that the language did not frighten the students and that a large majority of them have a positive appreciation of the language. Moreover, a majority of students have a favourable appreciation of logging techniques, which are a first step towards the approach promoted by the MODMED project.

3.2. Quantitative Evaluation of ParTraP

During spring 2019, we carried out a group of experiments to study the factors that could affect the performance of ParTrap. Since properties are evaluated on traces, we considered both (traces and properties) as factors to design the experiments.

To generate the traces, we used a generator based on grammars. Given the grammar of the trace files, it was then possible to generate traces with random values (Fig. 1). Properties were built manually, to explore the different ParTrap operators. They were designed with the purpose to be evaluated either to true or to false (Fig. 2).

We generated 8 traces ranging from 300 to 10000 events. For each of the 8 generated traces, we ran the test 50 times. The execution times were collected and displayed with box plots. We ran the experiments on a Linux virtual machine with 3 processors and 4 GB RAM on top of a Mac-book Pro machine. The results of the execution time are given in milliseconds. The execution time of the properties on this virtual machine doesn't represent necessarily the performance of the ParTrap language but it is enough to study the impact of the factors.

Influence of the size of the traces
The size of a trace is the number of events in the file. When the size of the traces increases, there are more data, and thus more time is needed to analyse a property. This could need more time to analyse the traces. For a given property, it is expected that the time required to evaluate it on a trace is not worse than linearly dependent on the size of the trace.
We drew two sets of tests with properties that are evaluated to true (resp. to false). The size of the trace is progressively increased. Results are displayed Fig. 3 and 4. It can be noticed that, “on average, the time required to evaluate a property on a trace is not worse than linearly dependent on the size of the trace”.

Influence of the number of parameters
As mentioned previously, each event of a trace can be associated with a set of parameters. that the time required to evaluate a property on a trace is not worse than linearly dependent on the number of parameters associated to the events (on average). Fig. 5 and 6 show the time required on average to evaluate a property on a trace of 1000 events, each of them having 4 to 24 parameters (mandatory parameters are included). Again, it can be noticed that, “on average, the time required to evaluate a property on a trace is not worse than linearly dependent on the number of parameters associated to the events”.

Other experimentations were conducted to evaluate the influence of the type of data implied in the parameters and the influence of the complexity of the properties. However, the design of the experiments did not permit to achieve a relevant conclusion. Those experimentations have thus to be extended, before being able to conclude.

```
[
{"id":"EventMatch","Time":600000,"string1":"sed","string2":"id"},
{"id":"lacus","Time":600001,"string1":"augue","string2":"bibendum"}
,
{"id":"felis","Time":600002,"string1":"a","string2":"tortor"},
{"id":"sit","Time":600003,"string1":"et","string2":"pede"},
{"id":"dapibus","Time":600004,"string1":"turpis","string2":"dui"},
{"id":"justo","Time":600005,"string1":"massa","string2":"phasellus"
},
{"id":"sit","Time":600006,"string1":"eu","string2":"turpis"},
{"id":"ridiculus","Time":600007,"string1":"justo","string2":"augue"
},
{"id":"sit","Time":600008,"string1":"et","string2":"ut"},
{"id":"curae","Time":600009,"string1":"aliquam","string2":"potenti"
},
{"id":"sed","Time":600010,"string1":"posuere","string2":"primis"},
...
```


(a) An extract of a trace with 2 parameters for events in addition to "id" and "Time"


```
[{"id":"EventMatch","Time":600000,"string1":"elementum","stri
ng2":"suscipit","string3":"in","string4":"lorem","string5":"i
n","string6":"et","string7":"elementum","string8":"phasellus"
,"string9":"dolor","string10":"sed","string11":"turpis","stri
ng12":"ligula","string13":"eu","string14":"ut","string15":"ma
gnis","string16":"penatibus","string17":"ac","string18":"just
o","string19":"ante","string20":"amet","string21":"non","stri
ng22":"primis"},
{"id":"feugiat", ...
```


(b) An extract of a trace with 22 parameters for events in addition to "id" and "Time"

Fig. 1: Examples of traces

```
PropOR: after first EventMatch, EventMatch e followed_by
EventMatchh c where e.string1 == c.string1 || e.string2 ==
c.string2 || e.string3 == c.string3 || e.string4 == c.string4 ||
e.string5 == c.string5 || e.string6 == c.string6 || e.string7 ==
c.string7 || e.string8 == c.string8 || e.string9 == c.string9;

PropAND: after first EventMatch, EventMatch e followed_by
EventMatchh c where e.string1 != c.string1 && e.string2 !=
c.string2 && e.string3 != c.string3 && e.string4 != c.string4 &&
e.string5 != c.string5 && e.string6 != c.string6 && e.string7 !=
c.string7 && e.string8 != c.string8 && e.string9 != c.string9;
```

Fig. 2: Examples of properties

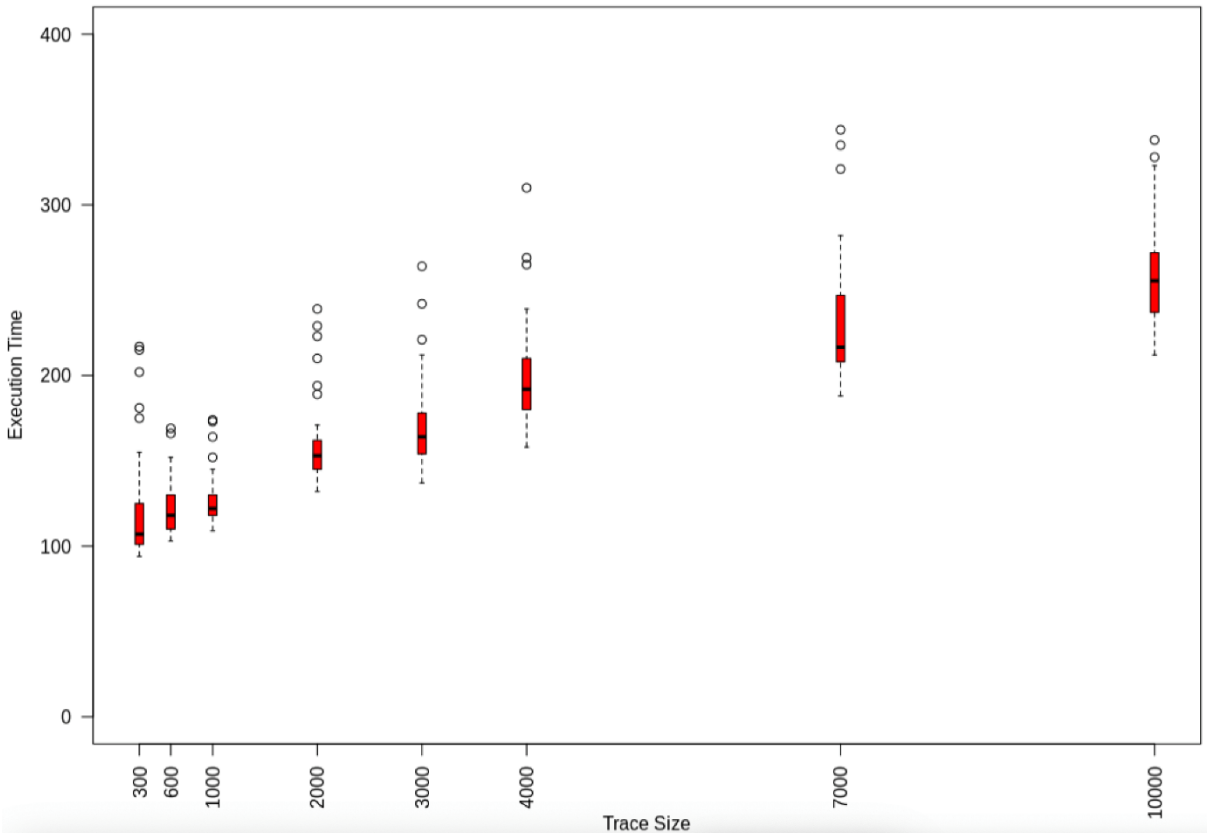


Fig. 3: Time (in milliseconds) required to evaluate a property with a true verdict w.r.t. the size of the trace

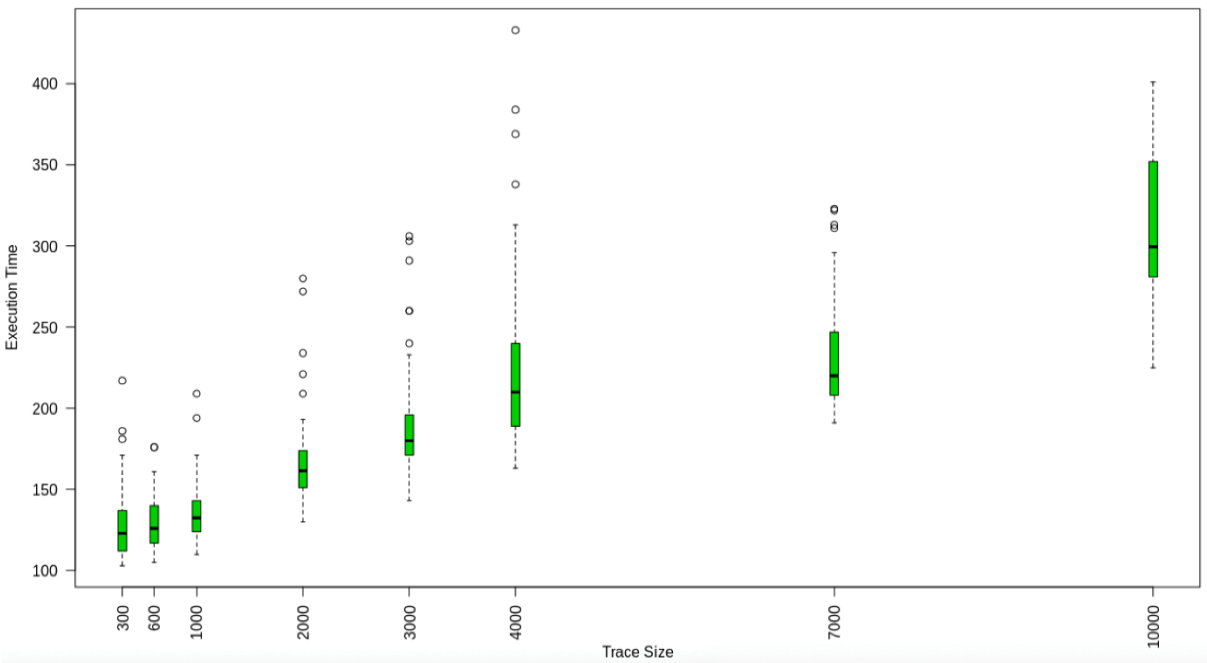


Fig. 4: Time (in milliseconds) required to evaluate a property with a false verdict w.r.t. the size of the trace

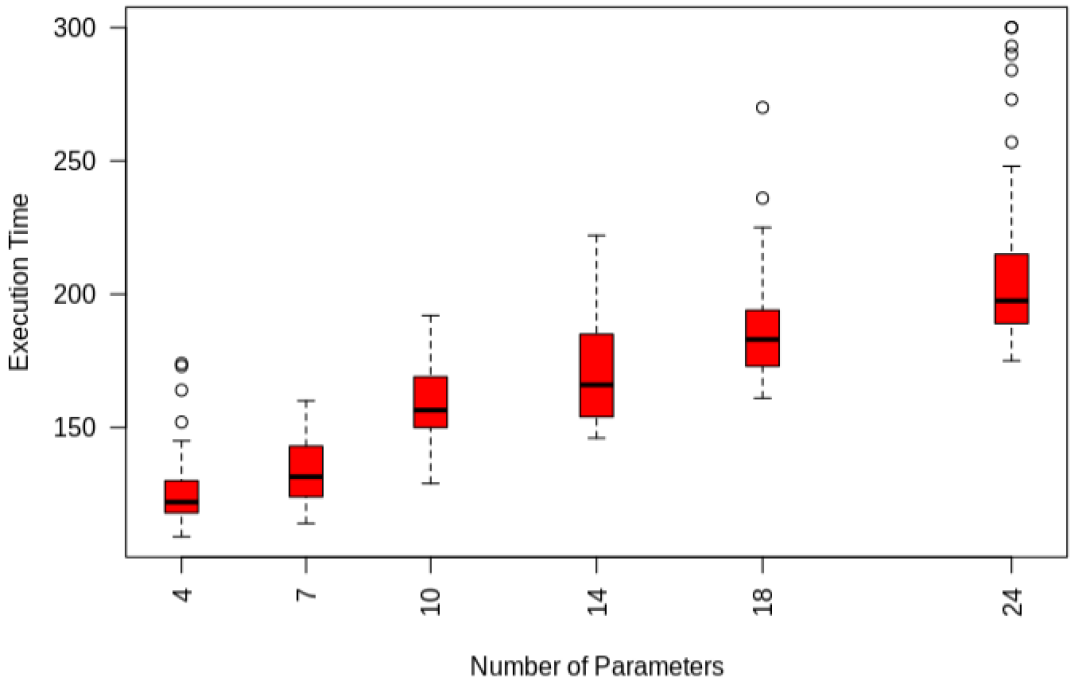


Fig. 5: Time (in milliseconds) required to evaluate a property with a true verdict w.r.t. the number of parameters

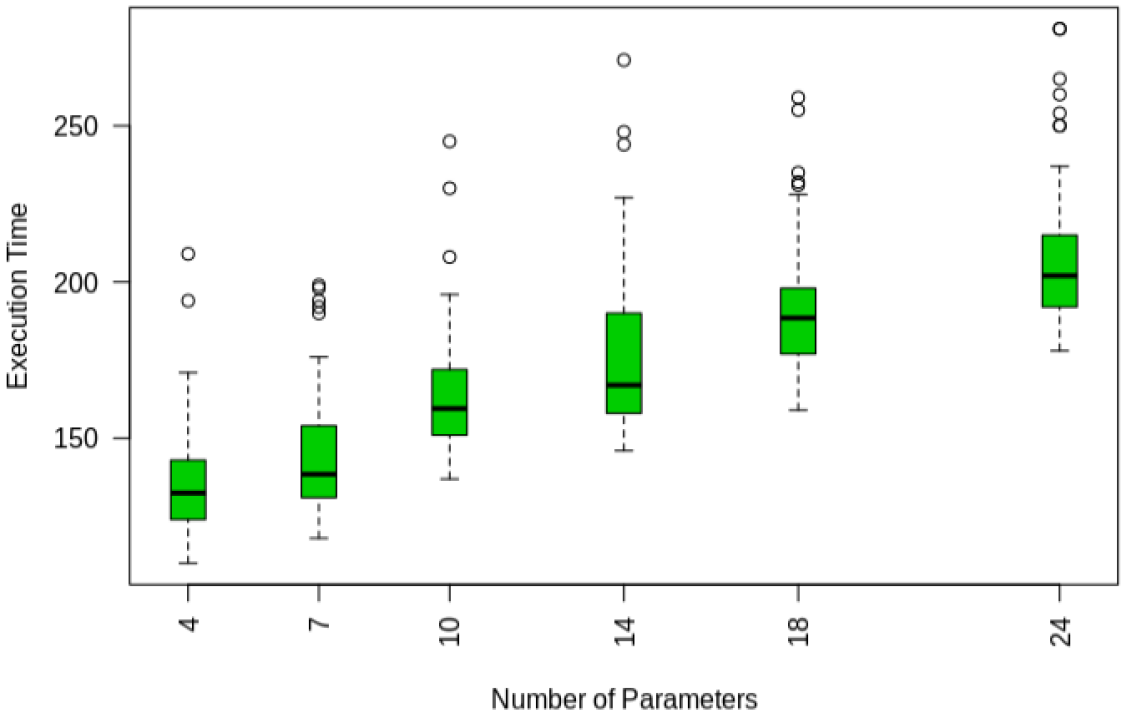


Fig. 6: Time (in milliseconds) required to evaluate a property with a false verdict w.r.t. the number of parameters

4. Overall evaluation of the approach and tools by Blue Ortho

Medical device industry has to deal with many constraints (regulatory, quality, documentation, validation, verification, audit, …) before releasing a product on the market.

Blue Ortho decided to not integrate the modmedLog trace library in a real released product in order to not put too many industrial and regulatory constraints to this research project.

Nevertheless, the main objective of verification of trace properties have been evaluated during a working session with Blue Ortho engineers. A subset of real traces has been transformed by a homemade software converter in order to make them compliant with ParTrap software (Json format).

Once transformed, the power of the ParTrap language has been evaluated using the different tools (IDE, libs, …) developed during the project. Driven exercises with graduated difficulties have been proposed to software development engineers. We explored 4 exercises during the session.

The ParTrap language looks like a query language and it has been well accepted by engineers. As a first contact, they feel comfortable with the syntax and the grammar. The small number of keywords in the grammar makes the DSL language easy to learn. Nevertheless, a risk of designing a property not responding to the real objective of the query has been identified. Engineers have to be comfortable with the global concept behind in order to write well-structured property using the right keyword. The learing curve should not be too long. More or less, similar design issue may appear on language manipulating dataset and operator (ie: SQL).

The IDE was quite pleasant to use (syntax coloration, feedback on syntax error, back trace or errors,…). In order to succeed in the industry, some additional work is necessary to transform this project in product. Easy installation, more robust, … are quality criteria expected by industrials.

Definitively, the DSL language responds to industrial needs and can be a good technology to support the monitoring process mandatory in the medical device industry.

5. Conclusion

This report has collected evaluations of the modmedLog library, on the one hand, and the ParTraP language with its associated tools on the other hand.

modmedLog facilitates the generation of more explicit traces than those produced manually. MinMaxMedical convinced several of its customers to use the commercial trace library. Their feedback is very positive regarding the ability of modmedLog to generate structured traces with few efforts from the developers. Regarding performances of the library, the initial version of WP2/D2 was recently improved to reach the expected standards of these customers. Nevertheless, the broader adoption of this technology, outside the regulated industries, requires its integration into a broader ecosystem like that of the Qt community.

ParTraP was evaluated by master students and by the Blue Ortho engineers. These evaluations produced positive results. Regarding the acceptance of the language, it was appreciated by both students and Blue Ortho engineers. The easiness to read and write ParTraP was also assessed by the answers of students to an evaluation questionnaire, and their good answers to the exam's questions. The Blue Ortho engineers also pointed out the risk to express a property not responding to their real objective. This risk has actually motivated the design of the ParTraP-EG (Example Generator) tool, which was not evaluated yet.

A quantitative evaluation was also performed by a master student. It shows that the complexity of properties evaluation is linear in terms of the size of the trace and the number of parameters of events. This tends to show that the generated monitors can scale up satisfactorily.

The Blue Ortho engineers also appreciated the Integrated Environment (which was not evaluated by the master students). Nevertheless, they stated that it remains a prototype and that its adoption requires to turn it into an industrial product.

In summary, modmedLog and ParTraP respond to industrial needs and address important issues for regulated industries such as the medical device industry.

Appendices

1. Multiple-choice questionnaire for the students (in French)

1 * À quelles questions avez-vous répondu lors du TD ?

- ☐ Question 1
- ☐ Question 2
- ☐ Question 3/Prop02
- ☐ Question 3/Prop03
- ☐ Question 3/Prop04
- ☐ Question 4

2 * Adopterez-vous la production de traces (pas nécessairement avec ParTraP) dans votre pratique du développement logiciel ?

- ○ Certainement, à mon initiative
- ○ Oui, si on me le demande
- ○ Avec des pieds de plomb, si on me l'impose
- ○ Jamais

3 * Les propriétés exprimées en ParTraP sont

- ○ Faciles à comprendre
- ○ Compréhensible avec des efforts
- ○ Difficiles à comprendre
- ○ Incompréhensibles, même en faisant de gros efforts

4 * Les propriétés exprimées en ParTraP sont

- ○ Faciles à écrire
- ○ Peuvent être écrites avec des efforts
- ○ Difficiles à écrire
- ○ Impossibles à écrire, même en faisant de gros efforts