Towards Complex Event Processing for Clinical Decision Support using FHIR

Gerhard Kober[1,*], Livio Robaldo[2] and Adrian Paschke[3,†]

[1]Visuapps GmbH, Vienna, AT
[2]Legal Innovation Lab Wales, Swansea University, UK
[3]Fraunhofer FOKUS and Freie Universitaet Berlin, Berlin, DE

Abstract
Clinical Decision Support (CDS) is designed to provide medical guidance and decision support based on patient information. More and more patient data is being collected from medical devices and fitness trackers, and some decisions need to be made quickly to allow for timely medical treatment. In our work, we combine a Semantic Web rule-based approach for representing medical guidelines with medical data represented in the "Fast Healthcare Interoperability Resources" (FHIR) standard. We implement a complex event processing strategy to deliver near-real-time decisions for a physician. This is relevant in intensive care and emergency medicine in particular, as medical treatment is time-critical and can decide on life or death of the patients. Different approaches exist for real-time clinical decision support. However, the integration of FHIR and Semantic Web technologies is missing. By integrating FHIR, medical ontologies, and Prova [1] as a semantic rule engine, we can define the medical guideline and include hooks for data extraction and decision points. We describe the entire process, starting with the medical device submitting the FHIR observations, through the data handling and decision points, to the notification of the attending physician. We extend the functionality of the existing rule engine for data handling and decision-making. In this work, we outline a solution that is capable of clinical decision support, including FHIR, and makes use of a semantic rule engine that allows medical guidelines to be expressed in RuleML. With these features, we also include the option of real-time complex event processing.

Keywords
CEP, FHIR, Healthcare, Rules, Medical Guidelines, Clinical Decision support

1. Introduction

Medical doctors must decide efficiently to provide the best possible treatment for their patients. The basis for the decision includes state-of-the-art knowledge about medical treatment as well as a patient-centric perspective and actual data. For example, in intensive care or in an emergency medical setting, physicians need to select fast which medical treatment to apply based on the patient's physical condition and the guideline that ensures the best care for the specific disease. Patients' real-time data and the medical guideline's actuality are fundamental. From a technical perspective, there are two main concerns: the high amount of patient-generated data (e.g.,

RuleML+RR'23: 17th International Rule Challenge and 7th Doctoral Consortium, September 18–20, 2023, Oslo, Norway
*Corresponding author.

gerhard.kober@visuapps.com (G. Kober); livio.robaldo@swansea.ac.uk (L. Robaldo); adrian.paschke@fokus.fraunhofer.de (A. Paschke)

from sensors and laboratory results) and the description of a medical guideline as a set of rules combined with the requirement to take near real-time decisions.

Medical guidelines are a crucial component of the clinical medical care process, since they enable cutting-edge medical treatment. The medical procedures are improving over time, and find their way to the latest recommendations in the guidelines.

Medical guidelines are evidence-based (i.e., they follow a standardized method of best practice and medical studies), ensure optimal patient treatment, and provide reassurance for the treating physician. Medical guidelines are typically described in natural language as workflows and diagnosis options. The guideline text also refers to literature, performed clinical trials, and respective outcomes and explains the purpose of the guideline design and introduction.

Fast Healthcare Interoperability Resources (FHIR) is an HL7 (Health Level 7)-Standard that defines the exchange and the structure of medical (data-) resources for building up interoperability among medical facilities[2].

In FHIR architecture, messages are exchanged using JSON, XML, and RDF formats, all transmitted using RESTful mechanisms. From a content standpoint, the format enables the description of so-called FHIR resources, which provide standardized building blocks for the instance-level description of healthcare entities. Resources exist, e.g., for a "Patient", "Observation", and basic Resource-types, e.g., as a "Bundle" or a "Code System". FHIR's resources and RESTful API access methods can cover various healthcare scenarios. A medical monitoring device can efficiently provide newly generated observations about a patient to a FHIR store for persistence. In a later step, data from the FHIR store can be selected to fulfill physicians' diverse needs. For example, regular measurements, such as heart rate, blood pressure, or oxygen saturation, can be recorded if a patient is undergoing clinical treatment. However, one can also imagine any other laboratory result or clinically relevant measurement that should be persisted in a FHIR store. The entire feed/query process for FHIR resources is maintained using FHIR's RESTful API.

Complex event processing is a strategy aimed at identifying meaningful events in real-time situations and reacting as quickly as possible [3][4]. Complex event processing is about event detection and reaction to complex events, i.e., compound event processed in near real-time. In the case studies considered in our research, processes involve, on the one side, a medical monitor (the event producer) and, on the other side, the physician's workstation (the event consumer) [5][6].

The medical domain faces many different sources of information. The problem we will address is integrating the workflow by combining the best practice medical guideline with the patient's real-time medical information encoded in FHIR and presenting the outcome to the physician for optimal medical treatment. Handling the problem can increase the quality of care for the patient, the safety of care, and the positive success rate in care for the clinician. By extending the capabilities of Prova [1], a Rule-based Complex Event Processing engine, for handling FHIR requests and decision-taking based on medical guideline rules, we plan to inform connected clients (physicians) of every decision to adopt during medical treatment of the particular patient.

The paper is structured as follows: section 2 describes the related work, and in section 3, we express our solution, containing a system architecture and depicting the flow of data. In section 4, we describe and evaluate our results. Finally, we discuss and conclude in section 5.

2. Related work

In previous work [7], we submitted a FHIR-Resource to Prova, extracted the coded values, and let Prova decide. However, the work in [7] was rather limited from several perspectives. This paper addresses these limitations thus presents a new more robust and flexible version of our framework.

First and foremost, we had the problem that we only referenced the guideline's content described in RuleML but included the details that make up the decision process in our implementation. Thus, the execution of the rule set was very much tied to the specific implementation and could not be used generically.

Second, the true or false decisions were not made solely by the ruleset but by configuration settings in the software that communicated with Prova.

Furthermore, we used multiple hooks to send messages between different services, which are expensive, and too slow for an external device that would send events within seconds. Finally, since we might require historical data during the processing, we need to query a FHIR store rather than hard-coding the data statically in configuration files. A related issue is that many relevant codes (e.g., LOINC, SNOMED-CT) can be in place in medical guidelines, but adding configurations for that and coding the values is mainly up to the developer, not to the guideline designer.

Another implementation that can be found in MuleSoft has an API in place that allows the detection of the different FHIR-Resources for event processing [8]. The event-detection tends to be meant for delivering received messages to other actors like payers and vaccination programs or child-health programs. This entails that MuleSoft can detect and forward FHIR resources, but the medical guideline decision path would eventually be missing.

Cotter et al. [9] enriched the FHIR data with Semantic Web technology and then asked queries. In their framework, incoming Health-Data is transferred to FHIR-RDF while persisting the generated graph in a triple-store. For subsequent queries, they were using SPARQL. Esper [10], a complex event processing engine, was used to discover conditions. Several challenges were identified in their approach: the query engine got overloaded by high-frequency queries; Furthermore, with each new incoming data set, the result of query-results accumulated which also led to an overload. Additionally, Esper is a language, compiler, and runtime for Java and .NET. It compiles Event Processing Language (EPL) to bytecode into a product. This compilation reduces the flexibility of a guideline specialist since a new guideline would result in a new product build.

De Laurentis [11], focuses on ECG (Electrocardiograms) and Images, where the ECG-events are detected matching defined criteria in the knowledge base. The analysis of historical data was necessary for detecting long-term anomalies. This was done by using and extending ISEQL [12] (Interval-based Surveillance Event Query Language). The work uses manual and automatic data acquisition. However, it seems the automated data is of a proprietary type and lacks the FHIR standard, thus limiting the interoperability and reusability of the approach.

This work, and our research in general, is instead focused on the FHIR standard, which we reason with by using the rule-based reasoner Prova. Different rule-based CEP engines, among which Prova (but also Prolog, Drools, ETALIS, etc.), are compatible with rule interchange standards such as Reaction RuleML, where we can implement medical guidelines with decision points. Still, the engines cannot directly deal with health-care-specific standards like FHIR. Furthermore, although Complex Event Processing (CEP) has been researched since decades, including research concerning the medical domain, integrating the FHIR standard and using a rule engine based on the RuleML standard was not done yet. This paper will contribute to these research directions in particular.

3. Methods

This paper, and the implementation, are part of a larger project, where we introduced a so-called Distributed Medical Rule engine[13]. Over time, the limitations were identified and the limited capabilities were re-engineered to a "Distributed Medical Service Engine" (DiMSE). This Service Engine can receive RESTful calls, perform different actions, and even encode medical information via Semantic Web technologies.

For the implementation part of the project, we are using Prova. Prova is a (semantic) Web rule language and a highly expressive distributed (Semantic) Web rule engine at the same time. It supports complex reaction rule-based workflows and rule-based complex event processing. Furthermore, distributed inference services, rule interchange, and rule-based decision logic are possible. The option of dynamic access to external data sources is highly relevant during the usage of our work[14]. Prova is built on top of ISO Prolog syntax from a syntactic perspective, and it offers serial Horn logic's expressiveness with a linear resolution for extended logic programs from a semantic perspective. Prova's capabilities allow us real-time event processing, as well as integrating external data sources, which are, in our case, FHIR stores.

3.1. Addressing real-time event processing

To address the problem of Complex Event Processing with FHIR, we are extending the capabilities of the DiMSE by using already implemented FHIR-RESTFul-APIs and injecting the FHIR standard into Prova. In more detail, the entire workflow (from a broader perspective) is described as follows and visualized in Figure 1:

1. Data-acquisition of FHIR-Observation.
2. Submission from FHIR-Observation to RESTFul API of the DiMSE.
3. Forward FHIR-Observation to Prova-Codebase (with the targeted medical guideline).
4. Parsing and extraction of the relevant values via Prova; these values are then provided to a subsequent internal function (compare-function).
5. The extracted value is validated against the target value.
6. The compare-function of Prova evaluated in Nr. 5 decides if the target value matches the FHIR value or not, and returns a truth value to the rule engine.
7. Decision/result is returned to Prova-initiating calls and initiating a WebSocket call to notify attached clients.

The data acquisition and health-status sensoring is a task of a medical device (like an ECG-Monitor). Such devices should be able to create patient-related FHIR-Observations to submit these to a designated endpoint. For example, a heart-rate measurement is permanently done,

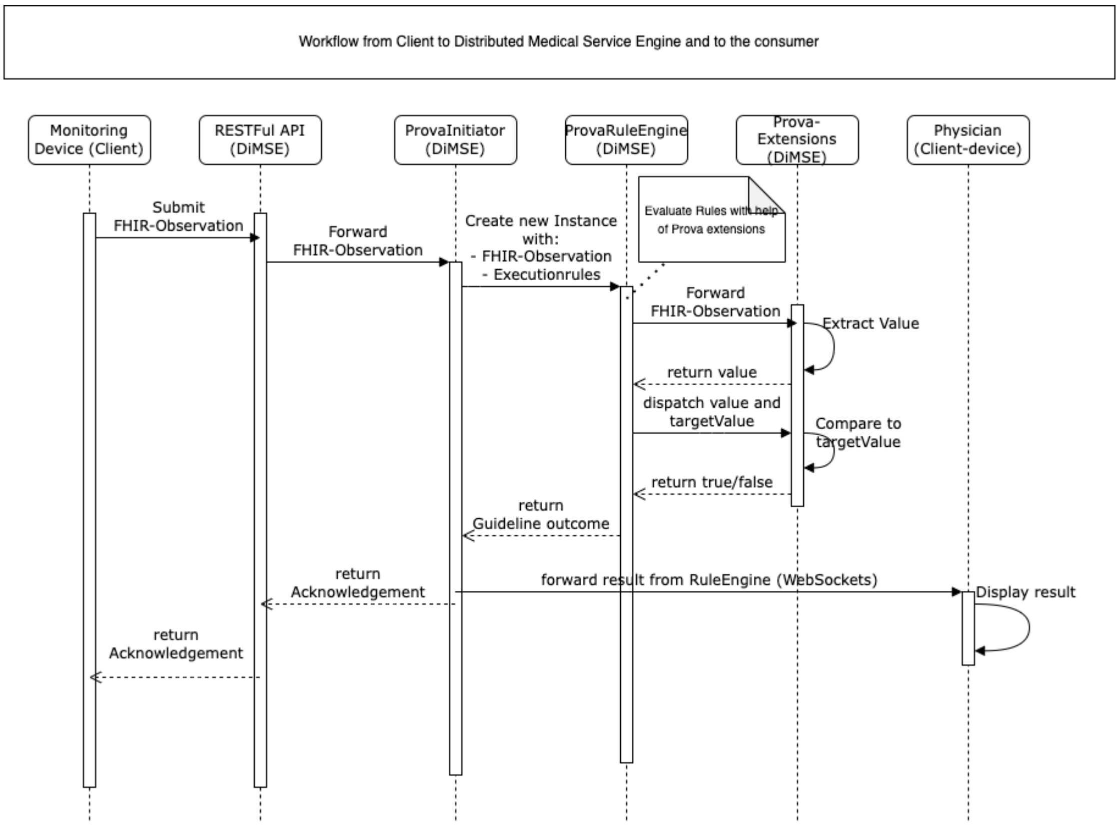
Workflow from Client to Distributed Medical Service Engine and to the consumer
Monitoring Device (Client)
RESTFul API (DiMSE)
ProvaInitiator (DiMSE)
ProvaRuleEngine (DiMSE)
Prova-Extensions (DiMSE)
Physician (Client-device)
Evaluate Rules with help of Prova extensions
Submit FHIR-Observation
Forward FHIR-Observation
Create new Instance with: - FHIR-Observation - Executionrules
Forward FHIR-Observation
Extract Value
return value
dispatch value and targetValue
Compare to targetValue
return true/false
return Guideline outcome
return Acknowledgement
forward result from RuleEngine (WebSockets)
Display result
return Acknowledgement

Figure 1: Workflow from a top-level perspective

and therefore a submission (for persistence purposes) should be performed at regular intervals. This applies to many different devices that serve different duties in the healthcare domain.

The FHIR observation submission is made using the FHIR standard methods, considering that FHIR is defined as a "RESTful" specification [15]. In our case, we defined and implemented a corresponding RESTFul-Service in the Distributed Medical Service Engine (DiMSE) capable of handling these requests. The FHIR-Observation-Resource that is implemented in the DiMSE is defined by the standard, allowing the DiMSE to handle all different sorts of Observations (such as heart rate, blood pressure, and breathing rate). Acting as a FHIR server allows us to operate as a proxy and intercept requests, where we can, on the one hand, use the FHIR resource content for the Prova-Rule-Engine, and, secondly, eventually forward it to another FHIR-Store.

Once we have received the FHIR-Observation, which is technically a JSON-Request, in the DiMSE, we parse it and call the implemented Prova-Service, which allows messaging to the rule base as well as messaging from the rule base back to our implementation. During the initialization phase of the Prova-Service, we load the rule base, which comprises the medical guideline. Furthermore, we send the FHIR observation as a payload to the Prova service to be used in the future execution process.

In the following step, the Prova-Rule engine with its capabilities comes into play. We are using Prova's "rcvMult''-function, which enables us to receive multiple incoming messages [16]. This function is used to accept the FHIR-Observation and pass it to subsequent functions in the rule base. We extended Prova to handle specific functions:

- fhirExtractValueFromObservation
- fhir_results
- fhirCompareBigDecimalValues

The fhirExtractValueFromObservation function: The signature of fhirExtractValueFromObservation contains the FHIR observation object, a QueryID, meant for temporarily holding the result available for the following function, and a code.

```
fhirExtractValueFromObservation(FHIR-Object,QueryID,``Code'')
```

The code expresses which Observation-Code-meaning should be extracted. It is a Code out of the LOINC-Codesystem, defining the value of the observation. For example, if a FHIR-Observation with a LOINC code "LP415755-0"[1] is submitted, this means that we are going to detect a heart rate and extract the observation's value. This value gets then assigned to a "QueryID".

The fhir_results function: The allocation is also implemented in a Prova function. This is sort of a decision support function. This was done to separate the tasks - to have the extraction and the following comparison disjunct and more globally available. Usually, the Prova functions return boolean values. In this case, we are not in the comparison phase, only in the "preparation"-phase.

```
fhir_results(QueryID, ObservationValue)
```

We need this assignment to have the observation value available in a later step in the process. Since we now have the extracted value as a single- entity (the type of this value is now BigDecimal) available, we can hand over this value now to a compare function.

The fhirCompareBigDecimalValues function: This function takes as input the value, the target value, and the comparison parameters. The implementation reads these values and evaluates the term as true or false.

```
fhirCompareBigDecimalValues(ObservationValue, 3.02, ``<='').
```

Allowed comparison parameters are: <,>,==,<=,>= and !=. Internally, the FHIR observation value, that was extracted beforehand, is compared with the target value. In the case, the comparison does not hold, we are returning "false" to the rule engine. In theory, this function is generic enough, to not only compare FHIR observation values, but any arbitrary BigDecimal values.

Since Prova rules are Horn rules, we have to evaluate our function to be true to ensure the entire rule is true. The following example shows the embedded fhirCompareBigDecimalValues-function within the Horn Rule in Prova:

[1]https://loinc.org/LP415755-0

```
Example:
normal(ObservationValue) :-
    %println(["normal - observationValue: ", ObservationValue]),
    fhirCompareBigDecimalValues(ObservationValue, 3.02,"<=").
```

This rule expresses that we want to evaluate for a "normal" result. If fhirCompareBigDecimal-Values evaluates to "false", the "normal" expression resolves to false, and the rule breaks the processing.

As a final step in the Prova rule base, we are sending back the result to the initiating Java code using the "sendMsg" function. This function has a corresponding section in the implementation, introducing the option to push the results directly to attached clients using WebSockets. In this implementation, we are registering web socket clients with their sessions. If a result from the rule engine is captured, we forward this result as a JSON formatted message to the client. We have a simple website for our simulation case that prints out the result.

3.2. Using ex-post-queries

The second option of integrating the FHIR-Data, where the interception is performed in a different way. The workflow, in general, is similar to the one described above. However, there are slight changes: Firstly, the monitoring device stores the FHIR-Observations as they are in a FHIR store. Secondly, an arbitrary trigger to the DiMSE can issue the execution of the medical guideline. These independent, asynchronous events, the FHIR submission, and the guideline event trigger, have the problem that they can overtake each other from a timing perspective, so the medical guideline execution engine is not aware of the latest FHIR observation. Even if triggers are correctly sequenced, FHIR queries to a FHIR store are required. To enable FHIR queries, we implemented two functions in the Prova rule engine:

- fhir_connect
- fhir_native_query

The fhir_connect function: This function handles the connection to which specific FHIR store the rule engine should bind, and the following query should be executed. As an example, we have

```
fhir_connect(Fhirserver,``https://hapi.fhir.org/baseR4/'')
```

where we define a particular naming for later use in the Prova rule engine (here, this is called "Fhirserver"). This function establishes and verifies if a connection to the destination endpoint is available. In our case, we connect to a public FHIR store, but any other FHIR store can be approached with the same method since the FHIR standard describes how to approach FHIR stores and the respective data.

The fhir_native_query function: It takes as input parameters the "Fhirserver" from the function above and then the needed URL-Parameters. For our example, we have the following parameters in place:

```
Observation?subject=Patient/1398961
&code=http://loinc.org|706-2&_sort=-date&_count=1
```

This means, that we connect to the FHIR store and search for Observation-Resources belonging to a particular patient. The patient's reference is encoded as "subject'' in the FHIR-Observation. Apart from the patient, we search for all Observations having a "code" from the code-system LOINC and a specific code - here 706-2. To ensure that we have the latest information from the FHIR store in place, we added sorting criteria to the date that is ordered in a descending way. This sorts all received observations from the newest to the oldest ones.

Our implemented Prova function should only care about the most recent observation. However, in such a FHIR query result, we receive many different observations. Furthermore, depending on the FHIR store's implementation, some sort of paging is in place that returns the first ten entries and provides links to the following pages. Since we need only one entry (the latest FHIR-Observation), we can "limit" the result by adding the parameter_count=1. The fhir_native_query-function also includes the comparison to the target value. This was done since there is no need to hold the particular FHIR-Observation for subsequent tasks.

The implementation details can be found on the GitHub repository associated with this paper[2], but to use the project's full capabilities, it must be packaged and deployed on a web server. This is necessary to support RESTFul requests and use the WebSocket API.

3.3. A medical example

The medical guideline on intensive care of cardiac surgery patients[17] describes the basic patient monitoring, an extended so-called "hemodynamic monitoring", the target parameters of cardiovascular therapy, and the therapy options themselves. The therapeutic goals of the cardiovascular system, through volume substitution or drug therapy, are sufficient body-tissue perfusion and, thus, an adequate oxygen supply to the organs.

"Hemodynamic monitoring" can be used to assess the pumping function of the heart. A "target value parameter" list is also available, which defines a wide range of values. For example, there is a target value of 65 mmHg as the target value for mean arterial blood pressure or a target value of >70% for central venous oxygen saturation. Depending on the values obtained from monitoring, a recommendation for medication (e.g., beta-blockers or catecholamines) is made. Depending on whether the target values are reached by therapy or not, a re-evaluation or therapy optimization takes place. Patients in intensive care units are usually connected to monitoring systems to enable continuous monitoring and adjustment of therapy.

The medical guideline describes a variety of parameters that are transmitted during monitoring and are taken into consideration for a therapy decision. Depending on the measured values transmitted for the treatment diagnosis, the outcome of the submitted values to the target values of the guideline determines which therapy to select. Four therapeutic options exist for so-called left heart failure, each using a distinct therapy. After treatment, the patient's status is re-evaluated towards the described measurement targets.

[2]https://github.com/gkober/CEP_FHIR_PROVA

4. Results and evaluation

The initial implementation considered solution number two as the favorite since it seemed that a simple FHIR client handling the calls might fulfill the requirement to fetch information for the clinical decision support. As a result of testing the solution with simulation data, it was determined that the many requests stored in the FHIR store and additionally triggered rule engine executions caused delays. Multiple threads were running, waiting to be completed. The bottleneck was mainly concerned with the connection to the FHIR store itself. In addition, the query's outcome is greatly influenced by how the particular FHIR store is implemented. Therefore, it cannot be guaranteed that the result is the most recent one to make the decision. Furthermore, if a medical device stores its data elsewhere, multiple queries are necessary, and therefore more network delays could occur.

When we encountered these issues, we implemented the described first approach: intercepting the request in the DiMSE and taking the decision there. We also executed our simulation, and the performance increased in comparison to FHIR-native queries to a FHIR store.

Since we want to assess performance and the operation of the implemented Prova workflow, the simulation is based on two random FHIR observations: The two observations distinguish between a "normal" and a "non-normal" heart rate. The RESTful call is omitted because the FHIR-Observation is directly submitted to the Prova-initiating function. However, in terms of performance, the dispatching approach is economical. The execution time for 10100 submissions was 84 seconds. This indicates that we can process 100 entries for decision-making on average every second. We discovered that the memory consumption was consistent for a streamed technique in which we continuously get FHIR observations, and no memory exception occurred. In figure 2, we depict the number of FHIR requests in conjunction with the duration in seconds and the memory consumption. The lower blue line outlines the duration: as expected, with an increasing number of requests, the overall execution time grows. The upper orange line describes the memory behavior of our test environment for the process. Initially, we have a low-level memory consumption; over time, the memory behavior stabilizes. Even a single run, with 120000 FHIR requests, did not exceed the memory limit.

5. Discussion and conclusion

We found that the two approaches can provide decision results for the medical guideline. These are described and executed as rules. With the integration of FHIR, we enabled the inclusion of medical devices capable of submitting FHIR observations. With this, we can overcome the limitations of [11]. With the usage of Prova, and its compatibility with RuleML, we define the medical guideline as a set of Rules as introduced in [18]. The work of Cotter et al. [9] and Kober et al. [18] transformed the FHIR JSON representation into FHIR RDF and also enriched these FHIR observations with ontologies for reasoning support. This enrichment is necessary for particular needs and can even fulfill more sophisticated queries using SPARQL. However, for the purpose of Complex Event Processing, the FHIR-to-RDF-transformation and the selective queries might cause too much overhead. The RDF approach is more suitable for ex-post-queries that are in use for "historical'' data (e.g., the heart rate value 5 minutes ago). So, with our

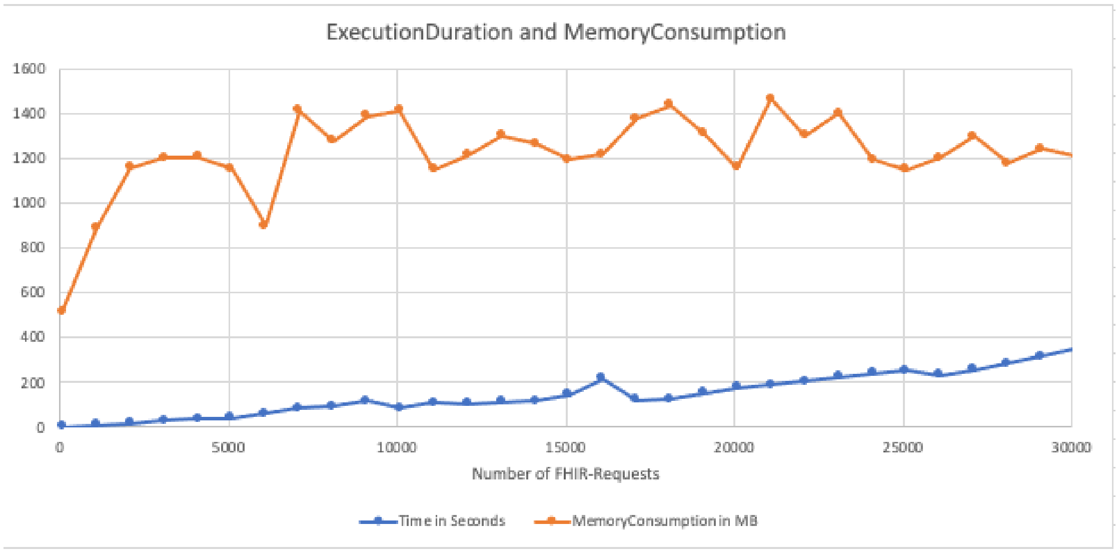

Figure 2: Duration and Memory consumption per FHIR-Requests

implementation, we can define in the rules which code for the value-extraction is needed to be evaluated, and Prova itself can describe the rules very precisely.

In the future, we need to investigate how to allow other FHIR resources as well, as currently, we are limited to the observation-resource. Furthermore, the comparison method takes care of decimal values, while textual representations might occur.

For processing FHIR-Observations, Prova has two different types of implementations ready. While the second solution does ex-post queries to a FHIR store, the first method intercepts the incoming request. We strive for request interception within a medical guideline when real-time execution is crucial for medical treatment. These queries to various FHIR stores are quite useful regarding historical data and medical standards that permit longer treatment times.

References

[1] A. Kozlenkov, A. Paschke, M. Schroeder, Prova language for rule based scripting of java and agents, and knowledge and information integration, 2023. URL: https://prova.ws/, accessed on June 14, 2023.

[2] Fhir v5.0.0, 2023. URL: https://www.hl7.org/fhir/, accessed on June 14, 2023.

[3] M. Eckert, F. Bry, Aktuelles schlagwort" complex event processing (cep)", Informatik-Spektrum (2009) 163–167.

[4] A. Buchmann, B. Koldehofe, Complex event processing, 2009.

[5] B. Linnert, Vorlesung alpv netzprogrammierung 2015, 2015. URL: http://www.mi.fu-berlin.de/w/SE/VorlesungALPVNetzprogrammierung2015, accessed on May 31, 2023.

[6] e. a. Francois Bry, Debs2009 event processing languages tutorial, 2009. URL: https://www.slideshare.net/opher.etzion/debs2009-event-processing-languages-tutorial, accessed on May 31, 2023.

[7] G. Kober, A. Paschke, Using prova-rule engine as dispatching-service for fhir-observation-resources., in: RuleML+ RR (Supplement), 2020, pp. 1–9.

[8] T. Huegle, Event-driven apis in the healthcare industry, 2022. URL: https://blogs.mulesoft.com/api-integration/event-driven-apis-healthcare/, accessed on May 31, 2023.

[9] D. Cotter, V. Bumgardner, Semantic enrichment of streaming healthcare data, arXiv preprint arXiv:1912.00423 (2019).

[10] Esper - espertech, 2022. URL: https://www.espertech.com/esper/, accessed on May 31, 2023.

[11] L. De Lauretis, F. Persia, S. Costantini, Intelligent Agents and Complex Event Processing to enhance Patient Monitoring, CEUR Workshop Proceedings 3203 (2022) 212–218.

[12] S. Helmer, F. Persia, Iseql, an interval-based surveillance event query language, International Journal of Multimedia Data Engineering and Management 7 (2016) 1–21. doi:10.4018/IJMDEM.2016100101.

[13] G. Kober, Distributed medical rule engine (dmre)-project., in: RuleML+ RR (Supplement), 2020, pp. 87–94.

[14] A. Paschke, Rules and logic programming for the web, Reasoning Web. Semantic Technologies for the Web of Data: 7th International Summer School 2011, Galway, Ireland, August 23-27, 2011, Tutorial Lectures 7 (2011) 326–381.

[15] Http - FHIR v5.0.0, 2023. URL: https://hl7.org/fhir/http.html, accessed 2023-06-16.

[16] A. Kozlenkov, A. Paschke, Prova rule language version 3.0 user's guide, Internet: http://prova. ws/index. html (2010).

[17] e. a. Sander, Prof. Michael, S3-Leitlinie zur intensivmedizinischen Versorgung herzchirurgischer Patienten. Hämodynamisches Monitoring und Herz-Kreislauf, Technical Report AWMF Register 001/016, Deutsche Gesellschaft für Anästhesiologie und Intensivmedizin e.V. (DGAI), 2017. Accessed 2023-06-18.

[18] G. Kober, L. Robaldo, A. Paschke, Modeling medical guidelines by prova and shacl accessing fhir/rdf. use case: The medical abcde approach., 2022.