Using PROVA-Rule Engine as Dispatching-Service for FHIR-Observation-Resources

Gerhard Kober[1] and Adrian Paschke[2]

[1] Tiani "Spirit" GmbH, Vienna, Austria
gerhard[DT]kober[AT]tiani-spirit.com
[2] Fraunhofer FOKUS and Freie Universitaet Berlin, Berlin, Germany
adrian[DT]paschke[AT]fokus.fraunhofer.de

Abstract. In a clinical medical environment, vital signs of patients are crucial information. In order to have the right information at the right time available, they need to be evaluated during the storage-process and then be submitted to special store-containers or to alerting-servers. To achieve the goal of observation-information-based dispatching, a rule engine is used to be highly flexible in the evaluation of health parameters and limits. The assumption is that such a declarative rule-based service is easier to maintain, and can also work in a distributed way. The distribution of the rule-engine is in the setting of a clinic-group (where many hospitals work together) fundamental to allow each hospital to work independently of the others.

Keywords: FHIR · PROVA · Rules

1 Introduction

In a clinical-medical environment, vital signs from patients are important for diagnoses and the upcoming therapy [5]. Usually, medical doctors are interested in two main things by evaluating the vital signs in clinical routine: first, if the vital signs are consistent over a timeline, and secondly, if there are outliers in medical values [13].

So the issue is to detect 'abnormal' values and decide dynamically where to store patient's data, to have it available for the physician (besides the as 'normal' defined vital-sings), who then needs to set actions for the patient.

This paper analyzes the use of a rule-engine to fulfill the needs of a future 'distributed medical rule-engine'. Furthermore, the values of the medical data, which exist in the form of FHIR-resources (Fast Healthcare Interoperability Resources) needs to be extracted, and the decisions need to be taken during the storage-process.

The goal of this work is to set up a system architecture and a workflow that allows us to receive FHIR-Observation-resources, which are meant for storage, extract the needed values of a particular type of information and decide on the retrieved information which storage-endpoint to use.

A use-case for this can be found for people who have diabetes, where the blood glucose level is highly relevant. So if the level falls under a certain level, a doctor (or another medical person) should get notified to help this person. Another feature is to have these occurrences for later medical analysis of these non-normal medicinal values.

A solution for the problem is needed to improve patient care, because of clear, relevant information and gathering information for answering medical-clinical research questions, based on vital signs.

FHIR (Fast Healthcare Interoperability Resources) is a standard created by HL7 (Health Level 7) [9]. FHIR defines resources, and they are defined by six layers to strap down the big healthcare domain [8]. These layers are:

1. Foundation Resources
2. Base Resources
3. Clinical Resources
4. Financial Resources
5. Specialized Resources
6. Resource Contextualization.

In this work, the clinical resources from layer three are used. To be more precise, the observation-resource is the target-resource. For the support of diagnosis or patient monitoring, the observations-resource is a central element in healthcare [6]. For the FHIR-Observation, it is mandatory to have the following items in the object, which will then be sent to the FHIR-Store: the status and the code. The 'Observations-status' describes the status of result value and is used to track the individual results. The observation-status is coded and can just take defined values. The code describes the observed entity to understand the meaning of the observation. [7]. Also, the value of the actual observation is useful (but not mandatory by definition). Other items in the observation can be included, but do not need to be. For the solution of the problem, the code and the observation-value are important during processing to decide if the value is relevant for making decisions, and if so, in which way to use it.

As a rule engine for this dispatching-case PROVA [12] will be used. It supports the usage of event-driven reaction rules. Since there is a need in the overall process for high flexibility and the ability of the solution to express event-patterns and reactions, the complex event processing-lifecycle supported by Prova is helpful. Related to [4][2] the CEP (complex event processing)-lifecycle contains:

– Event Production
– Event Definition
– Event Selection

– Event Aggregation
– Event Handling
– Event Consumption.

These events will be processed during the actual execution. Prova provides the ability to use a declarative programming paradigm in conjunction with an object-oriented programming paradigm. There is a separation on the program-logic (the declarative part), the data-access, and optional external procedural computations [1].

Other possible approaches besides the rule-based to be thought of are more restrictive and less flexible. For a very special use-case, a solution could be implemented 'hardcoded' by doing all the needed calculations and comparisons in the code. Still, if someone likes to change comparison-values or change the use-case, it takes a lot of changes in the program logic. Furthermore, if a second or a third factor for a decision needs to be taken into account, the 'hardcoded' solution will become complicated and hard to maintain. Therefore this is not the preferred solution. When taking into account that not only FHIR-resources or data structures defined in the medical domain are relevant for making decisions, a more generic solution is needed. It has to be based on standards and to be generic and flexible to build different options for needed decision points. For example, there is HL7-Arden-Syntax, which is used for representing procedural clinical knowledge in order to share health knowledge bases [3]. These knowledge bases are encoded in the Arden Syntax, which is specialized for the needs of the language of so-called MLMs (Medical Logic Module). So the Arden-Syntax has a strong focus on medical information sharing and is useful for clinical processes. Going beyond the clinics and using FHIR, the Arden Syntax has no capabilities for obtaining clinical-medical decision support.

2 System design

The following section describes the planned system architecture, as well as the workflow.

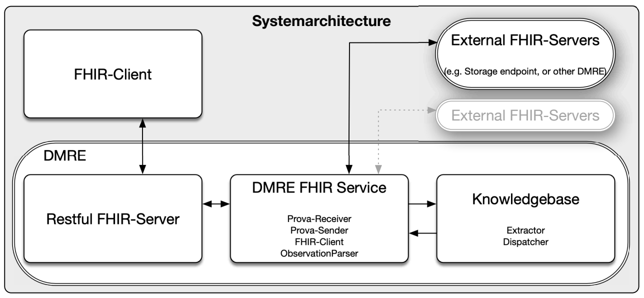

Fig. 1. Systemarchitecture

The following components are within the system architecture:

- FHIR-Client
- Restful FHIR-Server
- DMRE FHIR-Service
  - Prova-Receiver
  - Prova-Sender
  - FHIR-Client
  - ObservationParser
- Knowledgebase
  - Extractor
  - Dispatcher
- external FHIR-Server

The following section describes the relations between the components having in mind the workflow as well.

The system architecture: From an architectural point of view, the solution is based on FHIR as a state-of-the-art standard for sending and receiving medical data. Therefore, any (external) *FHIR-Client* is needed to send the FHIR-Observations to a so-called *'Distributed Medical Rule Engine'* (DMRE) which contains a *FHIR-Server*. The DMRE is a container for different services, which need to be accessed during transaction time.

The FHIR-Client is responsible for the creation of conformant JSON-objects, containing the appropriate values for the designated resource. The FHIR-Server provides a RESTful-service to receive FHIR-resources (e.g., Patient-resources, Observation-resources) by using HTTP. The server has to be able to accept HTTP-calls for inserting, updating, and deleting resources.

The *DMRE-FHIR-Service* is a service that provides the interfaces for the PROVA-rule-engine, as well as an FHIR-Client and an Observation parser. The rule-engine interacts with the *Knowledge base*. The knowledgebase contains the rules. There are rules which define which value needs to be extracted for further processing, and also rules for the dispatching of the FHIR-resources. These rules are providing the 'logic' in the workflow during execution-time.

The DMRE-FHIR-Service can send incoming requests from the FHIR-Server to the extractor-rule. Secondly, the DMRE-FHIR-Service can receive incoming requests from the Rules for upcoming processing. Here is meant to either get the information of extraction (which value to extract from the FHIR-Observation), or the dispatch information (where to process the FHIR-Observation to). For the extraction, the Observation-parser is needed, to grab the relevant information (requested code and value), if it exists in the observation. This DMRE-FHIR-Service exists as an extra service because of the option to have another incoming path than the FHIR-Server (which intercepts the request). The *external FHIR-Server* is the chosen endpoint by the rule-engine, executed by the FHIR-Client in the DMRE-FHIR-Service. This endpoint could be the goal (target) server, which is responsible for the persistence of the observation resource. It could be another DMRE-RESTful-FHIR-Server-instance, which proceeds other rules and takes a different decision (based on a different set of rules). That means it is possible to have the integrated rule-engine used in a distributed way, and no central component is needed. Still, it could be on the edge of a wide distributed inter-operable health system.

The system architecture is designed to intercept the incoming FHIR-Observation-request for the information extraction and make decisions based on the rules about the FHIR endpoint.

Workflow: For the current use-case, the workflow during the transaction process in this architecture is based on the fact that an observation-type (e.g. a heart-rate or glucose-level) and a decision for 'normal' or 'not normal' is needed.

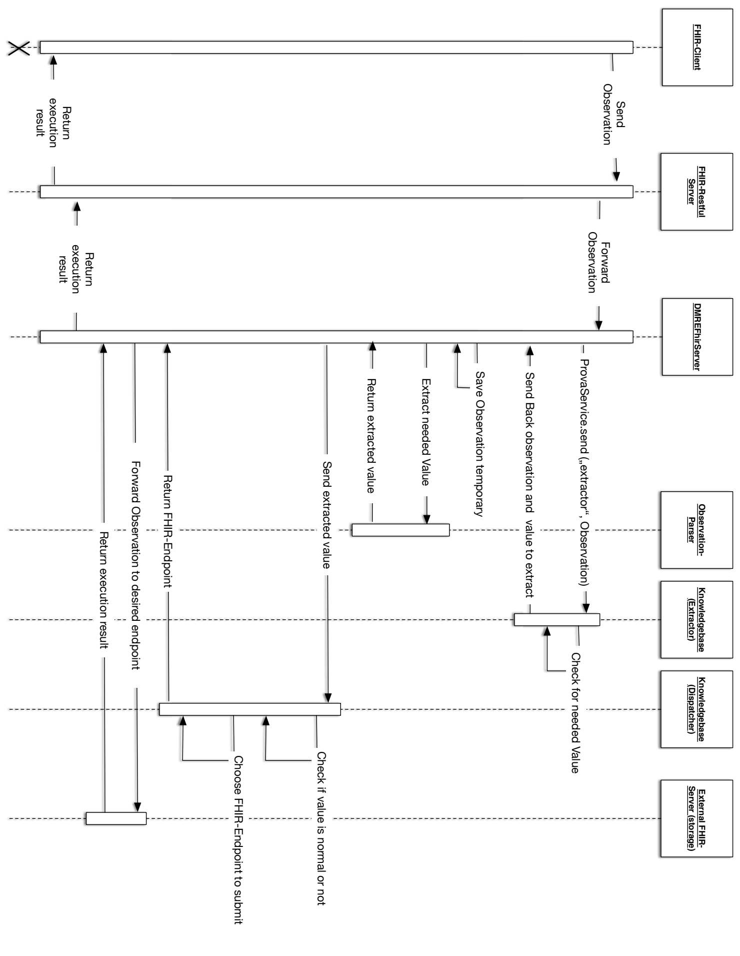
FHIR-Client
FHIR-Restful Server
DMREFhirServer
Observation-Parser
Knowledgebase (Extractor)
Knowledgebase (Dispatcher)
External FHIR-Server (storage)
Send Observation
Return execution result
Forward Observation
Return execution result
ProvaService.send („extractor", Observation)
Send Back observation and value to extract
Save Observation temporary
Extract needed Value
Return extracted value
Send extracted value
Return FHIR-Endpoint
Forward Observation to desired endpoint
Return execution result
Check for needed Value
Check if value is normal or not
Choose FHIR-Endpoint to submit

Fig. 2. Sequence FHIR-Client, Interception and final submission

The workflow starts by having an FHIR-Client submitting an FHIR-Observation to the DMRE-FHIR-Restful-Server. The DMRE-FHIR-Restful-Server intercepts the request by calling the DMRE-FHIR-Service and forwarding the observation there. This DMRE-FHIR-Service is the core component in the workflow since it coordinates requests and takes care of responding to the initial requestor. The DMRE-FHIR-Service forwards the observation to the 'extractor'-Rulebase. This extractor is aware of which value needs to be extracted to have the right information for the upcoming dispatching. The extractor-rule returns than the 'to-be-extracted' value and observation. This task of filling the needed value and an FHIR-Observation can also be done by any external component. The DMRE-FHIR-Service keeps the current observation to proceed with the following parsing and extraction. The observation-parser simply returns the extracted value, which will then be submitted to the 'dispatch'-rule. This rule decides (e.g., if a value is a normal value or a not-normal value), based on the configuration of the rules and the extracted value, which FHIR-Server-Endpoint to use for the (final) submission. The 'dispatch'-rule finally returns the endpoint to the DMRE-FHIR-Service, which then acts as an FHIR-Client and submits the initial observation to the decided endpoint. The endpoint returns an execution-result, which will be processed back through the entire processing-chain, to have the initial FHIR-Client informed about successful storage.

The general workflow during the execution process is for every observation-type and the attached values the same, but the specific details on 'what' goal need to be achieved can be changed due to the needs of the use case.

3 Implementation

The implementation of the Distributed Medical Rule Engine (DMRE) is done in Java by including Prova-libraries for the rule-engine-parts and the HAPI-FHIR-Library [10] for essential FHIR services.

For testing a sample use case, a reduced example from the HL7-FHIR-Resource page of samples is used. This one is then sent via HTTP-Post to the FHIR-Endpoint in the DMRE.

The HAPI-FHIR-Library provides a servlet, which allows us to set up an FHIR-Server in general, but without having specific resources implemented. This allows us to build up our own needs in the implementation, by creating for example a specific 'RestfulObservationResourceProvider', which then can call the DMRE-FHIR-Service-code. The DMRE-FHIR-Service acts as a client for sending the received JSON-Object to the rule-engine. The call is submitted to the extractor. A sample listening for the extractor is in Listening 1.1.

Listing 1.1. Sample Extractor.prova

```
:- eval(extractor()).
extractor() :-
        rcvMult(XID, Protocol, From, inform,
            {observation->Observation}),
          routeTo2(Observation).
routeTo2(Observation) :-
          sendMsg("XIDExtractor",osgi,"FHIR",inform,
               {glucose->Observation}).
```

The knowledgebase (extractor-rule) has the information WHAT to extract. It receives the observation as a string, and 'routeTo2' (which means send it back to the Java-process - 'Step 2') also has the information which observation-type to take care of. In the sample, it takes care of the glucose-coding.

The java-code contains a Prova-Callback, to receive calls from the rule-engine. Prova has a defined ordering of parameters [11] that is used for the decision if it is an extraction-task or a dispatching-task. The parameters are the following ones [11]:

– XID - conversion-ID of the message
– Protocol - Name of the message passing protocol
– Destination
– Performative - the message type broadly characterizing the meaning of the message
– Payload

For this task, the 'Destination' is used - either 'FHIR' for the data-extraction and further ongoing submission to the second rule-base or 'DISPATCHER' for executing the submission to the next FHIR-Endpoint.

Once the Prova-callback is hit by the Destination 'FHIR', the Java-code tries to parse the FHIR-Observation, extract the value from there, and return it, to then put this information in a further call to the Dispatcher-rule, to take the decision where to send the FHIR-Observation-Resource finally.

The dispatching-rule has information about the decision to submit to a different endpoint, and it also holds the information on the endpoint.

In Listening 1.2, a reaction-routing functionality is used to find the correct endpoint if the value of the observation is normal or another if it is abnormal. Once found out about the status of the value, the "routeToFHIR" either takes the event processing agent for FHIR-Endpoint A (http://hapi.fhir.org/baseR4) or B (http://hapi2.fhir.org/baseR4). This information is then sent back to the Prova-Callback with the Prova-Destination 'DISPATCH', to submit the FHIR-Observation to the meant endpoint, by using an implementation of an FHIR-Client.

Listing 1.2. Sample Dispatcher.prova

```
:- eval(dispatcher()).
dispatcher() :-
          rcvMult(XID,Protocol,From,inform,
               {value->Observation}),
          routeToFHIR(Agent,Observation),
          sendMsg("XID3",osgi,"DISPATCHER",inform,
               {endpoint->Agent}).
```

```
routeToFHIR("http://hapi.fhir.org/baseR4",Observation) :-
        abnormal(Observation).
routeToFHIR("http://hapi2.fhir.org/baseR4",Observation) :-
        normal(Observation).
abnormal(Observation) :-
        Observation>7.
normal(Observation) :-
        Observation<=7.
```

After processing the information to the next FHIR-Server, the response from the server is then returned to the client (for successful and also for not successful submissions). This is achieved by the HAPI-FHIR-Method 'MethodOutcome'. Important to say, that the initial JSON-Object is not changed in terms of content. The transformations are done for the necessary extraction to allow the decision based on values.

The code is available on Github https://github.com/gkober/DMRE. As a side note: in the code are two options implemented: first, using "hardcoded" observation-types (by configuration), and secondly, the opportunity for the rule-engine is available. The usage of the different options can be done by configuration.

4 Results and Discussion

The presented work shows the ability to achieve rule-based dispatching to different FHIR-Servers for obtaining patient's observations, which are critical for patient care. The solution is highly flexible, because of the rules which take control of the decision-points.

During the evaluation of the implementation, simple observations containing e.g., glucose-values or heart-rate values, the solution works appropriately by the extraction and the dispatching. There is a known issue in the implementation, which takes care of 'combined observations'. This means if an observation has two or more codes and values (e.g., blood pressure). The implementation is susceptible to the correctness of the FHIR-Observation. For example, an observation without a value returns in error. Such an observation is, by definition, correct, but for a production system, it is probably not relevant to take care of this topic.

Finally, the architecture and the decision to use the rule-engine Prova turned out to be a reasonable solution providing the required flexibility and adaptability as well as extendability of the architecture. This helps to improve the overall solution of the 'distributed medical rule engine'.

5 Future Work

Based on this work, further topics need to be addressed. One is to extend the solution for other FHIR-Resources, which are valuable for different types of information dispatching and extraction. In the rules, the user mentions the "type" of information which needs to be extracted - therefore, information retrieval from a terminology-service needs to found. Also, the dispatching of patient-basis and combinations is a field that needs to be addressed.

References

1. A.Paschke: Semantic Web Rules, https://pdfs.semanticscholar.org/d8f9/e961df8c4103a2d698f962d3460305d43564.pdf, accessed: 2020-05-06
2. A.Paschke: The Reaction RuleML Classification of the Event/Action/State Processing and Reasoning Space (12 2006), https://arxiv.org/ftp/cs/papers/0611/0611047.pdf
3. Arden Syntax v2.10 (Health Level Seven Arden Syntax for Medical Logic Systems, Version 2.10), https://www.hl7.org/implement/standards/product_brief.cfm?product_id=372, accessed: 2020-05-06
4. Semantic CEP with Reaction RuleML, https://de.slideshare.net/swadpasc/paschke-rule-ml2014keynote, accessed: 2020-05-06
5. Chalari E., Intas G., S.P.: The Importance of Vital Signs in the Triage of Injured Patients. Critical Care Nursing Quarterly **35**, 292–298 (2012). https://doi.org/doi:10.1097/CNQ.0b013e318255d6b3
6. Observation - FHIR v4.0.1, https://www.hl7.org/fhir/observation.html, accessed: 2020-05-06
7. Observation - FHIR v4.0.1 - Definitions, https://www.hl7.org/fhir/observation-definitions.html#Observation.code, Accessed: 2020-05-06
8. HL7 FHIR: Architect's Introduction, https://www.hl7.org/fhir/overview-arch.html, accessed: 2020-05-06
9. HL7 FHIR: Summary, https://www.hl7.org/fhir/summary.html, accessed: 2020-05-06
10. HAPI FHIR - The Open Source FHIR API for Java, https://hapifhir.io, accessed: 2020-05-06
11. Kozlenkov, A.: Prova Rule Language Version 3.0, User's guide (May 2010)
12. Prova rule language, https://github.com/prova/prova, accessed: 2020-05-06
13. Versloot, M.: Evidence based decisions in nursing and their effect on quality of care. Ph.D. thesis, Faculty of Medicine (AMC-UvA) (2012), https://hdl.handle.net/11245/1.385728