Building Health Policy Enforcement Solution Based on HL7 FHIR (Extended Abstract)

ALEXEY KOPTSEVICH, WOLF RUZICKA, VICTOR SHILO,
MIKKEL P. SCHULTZ, DMITRII VELIKII
Eastbanc Technologies, Washington, DC, USA
(*e-mail:* akoptsevich@eastbanctech.com)

submitted 14 September 2021; revised ; accepted

KEYWORDS: FHIR, s(CASP), ErgoAI

1 Business Problem

Health data management is in the process of standardization. Health Level Seven International (HL7) is an ANSI-accredited organization with a mission to provide frameworks and standards for the exchange, integration, sharing, and retrieval of electronic health information. The vision of HL7[1] is a world in which everyone can securely access and use the right health data when and where they need it. In March 2020[2], the Office of the National Coordinator for Health Information Technology (ONC) and the Centers for Medicare and Medicaid Services published interoperability regulation focused on key provisions of the 21st Century Cures Act[3] that governs use and exchange of electronic health information. ONC's standardization effort is based on HL7's Fast Healthcare Interoperability Resources (FHIR) standard, Release 4[4].

Avoidance of data leakage and cyber threats are the key aspects of health data management. The focus of our work is to design a Health Policy Enforcement Solution (HPES) that can be used as part of an FHIR-based interoperability platform and seamlessly inserted between an FHIR database and any application that provides an FHIR-based API. The purpose of the HPES is to ensure that legal requirements regarding sensitive personal information are accounted for every time a patient's data is returned upon request of the patient or their personal representative. The rules regulating access to health data can be based on various criteria: legislative and policy documents of various jurisdictions, patient and disease data, patient's acts of consent, attribution data (relation of the patient to the entity requesting the data), etc. Some rules may supersede other rules in different settings.

With a variety of health industry players involved, including insurance companies, medical service providers, clinicians, researchers, labs, etc., the market value of such an interoperability

[1] https://www.hl7.org/about/index.cfm?ref=common
[2] https://www.himss.org/news/final-cms-interoperability-regulation-what-you-need-know
[3] https://www.congress.gov/114/plaws/publ255/PLAW-114publ255.pdf
[4] https://www.hl7.org/fhir/overview.html

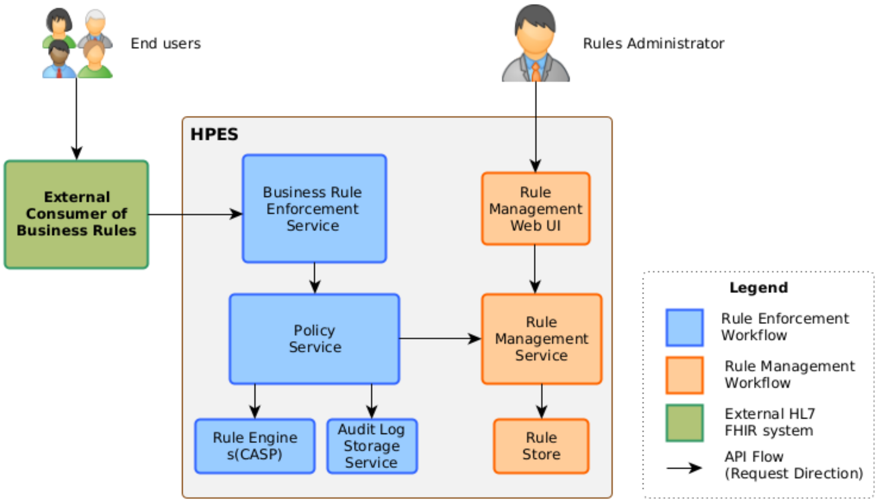

Fig. 1. HPES Architecture

solution is difficult to overestimate. Moreover, we expect that similar demand may emerge in other industries and business domains where HPES could be applied as rule-based access management Intelligent Agent[5].

The key features and functionalities of HPES are:

- Access policy is formulated as a set of rules. These rules implement logic embedded into laws of various jurisdictions and operate on input data comprised of patient health information, patient consent to release this information to various entities, and attribution data.
- There should be a way to allow one rule to supersede another.
- Rule creation and modification is tracked using version control.
- Rules may have time-related limitations, i.e., they may be effective only during specific time range or only apply to data produced during specific time range.
- Rule management shall be accessible to SMEs (Subject Matter Experts) that do not possess programming skills.
- Rule creation, modification, and deletion may need to be approved by a compliance officer before deployment to production.

2 Solution Architecture

HPES uses microservice-based architecture and consists of several components that work together to implement two primary workflows: rule management and rule enforcement. We designed HPES as a containerized solution to run on Kublr[6], which is an enterprise-grade frontend to Kubernetes that fully automates cluster deployments across various cloud environments.

Figure 1 outlines the high-level architecture that may have application not only in healthcare, but also in other business domains.

[5] https://en.wikipedia.org/wiki/Intelligent_agent
[6] https://kublr.com

Business Rule Enforcement Service is an HPES component that encapsulates the rule enforcement workflow: It either filters health data that is passed through it or communicates policy decisions based on it. This service provides APIs and implements authentication and authorization routines to communicate with External Consumer of Business Rules, which could be a wrapper application for an HL7 FHIR database or any other consumer in a general case.

Rule Management Service is a component that encapsulates storage and management of rule sets. A distributed version control system, such as git, would serve well as a backend for this service. Git offers workflows for all aspects of rule management: storage of rules and their metadata, reliable versioning of rule sets (via commits), approval (via branching and merging), backup and replication (via remotes), and electronic signing (via signed commits).

Policy Service is the key component of HPES. It encapsulates the logic of making policy decisions based on rules provided by Rule Management Service. At its core, it is built around a versatile logical engine that can perform logical inference based on input facts and a complex set of rules that are not required to be arranged into a rule hierarchy explicitly maintained by a developer or SME. Policy Service provides a wrapper around the rule engine, implementing an internal communication API consumed by Business Rule Enforcement Service.

A comparative market analysis of Business Rules Management Systems (BRMS) and Intelligent Agents with a focus on open-source solutions revealed ErgoAI and s(CASP) as the winners and Drools and OPA as runners-up. For details, please refer to Appendix A.

Our analysis shows that the winners, ErgoAI and s(CASP), score the highest overall, especially on the two criteria that we consider the most important: long-term maintainability of rule base and support of rule base growth in size and complexity over time. This is no surprise, as they are both based on logical programming language Prolog, that is designed with a goal of modeling of logical reasoning of humans. As a restrictive DSL (Domain-Specific Language), logical language provides a medium that is free of both the complexity of natural languages, from which rules originate, and the complexity of low-level general-purpose programming languages that are used to implement rules in production rule systems. Also, logical language gives an additional level of abstraction that is focused not on implementation aspects of rules, but on their logical content. Growth of the rule base in size and complexity over time makes availability of this level of abstraction more valuable, as it simplifies rule maintenance and debugging.

For comparison, defeasibility is not supported by production rule engines, such as OPA and Drools. This is risky in the long term. Our comparison to date is based on simple examples of input data and rule bases, with the goal of assessing efforts required to build HPES. Our experiments showed that only ErgoAI and s(CASP) are well-suited for the business problem. Limited amount of available input data prevented us from making the final selection, but s(CASP) appears to be the leader because all its functionality is provided as an open-source solution.

3 Conclusion

To proceed, and to make the final selection of the rule engine, we should assemble a realistic dataset that contains health care data, consent data, law-based rules, and example requests on a sample of impersonalized patients.

Our next milestone is to assess and compare performance of s(CAPS) and ErgoAI and develop an approach to scaling of HPES at high load. This will be addressed by creating uniform interfaces to compare the results of s(CASP) and ErgoAI, trying vertical and horizontal scaling approaches and monitoring availability for both engines to find their respective limits and the optimal approach.

Appendix A Rule Engine Selection

The most important criteria for rule engine comparison:

- **Support for complex law-based rules.** Laws are written in natural language that is not always easy to map to code. Rule sets can contain a large number of rules. It may be difficult for an SME, or even a software developer, to understand interconnections of rules in a large rule base.
- **Number and complexity of rules increases over time.** As a rule set grows, it can become difficult to maintain. New rules can contradict old ones directly or in convoluted ways, leading to errors that are difficult to catch. Code maintenance expenses may start growing super-linearly over time.
- **Snowball effect of commercial software development realities.** In a typical long-term enterprise project, the software product is a result of the efforts of many development teams. The initial architecture and design may be well thought-through, but if their ideas are not reflected in code constructs that speak for themselves, they will eventually be lost in the noise of new features and code refactoring. If this happens to rules that encapsulate complex logic of laws, the cost of error may be very high; hence maintenance expenses may start growing super-linearly. If a highly restrictive DSL is used to encode rules, it may help restrain developers in the confines of the initial design of the system and thus reduce maintenance costs.
- **Open-source solutions are strongly preferred to avoid vendor lock.**

Top candidates (in alphabetical order) and their characteristics (see Figure A 1 for comparison of the candidates):

- **Drools**[7] is a popular open-source production rules system that provides DSL focused on evaluation of conditions and on execution of actions based on them. It supports various integrations within Java ecosystem. Technical support is available from RedHat.
- **ErgoAI**[8] is an inexpensive commercial Intelligent Agent that provides Logic AI capabilities. It abstracts knowledge representation and reasoning using a DSL based on logical programming language Prolog. It provides support for several types of formal logic and advanced features, such as rule defeasibility, audit trail, and integration with templates written in natural languages. It is successfully applied in various domains, such as finance and healthcare (Grosof and Bloomfield 2018). An open-source version — Flora-2[9] — with reduced functionality is provided by developers of ErgoAI.
- **OPA**[10] is an open-source solution for policy enforcement for cloud-native environments. It focuses on integrations with cloud stacks and performance of issuing policy decisions. Its DSL provides dedicated constructs for low-level performance optimizations at the expense of maintainability.

[7] https://www.drools.org/

[8] http://coherentknowledge.com/product-overview-ergoai-platform/

[9] http://flora.sourceforge.net/

[10] https://www.openpolicyagent.org/

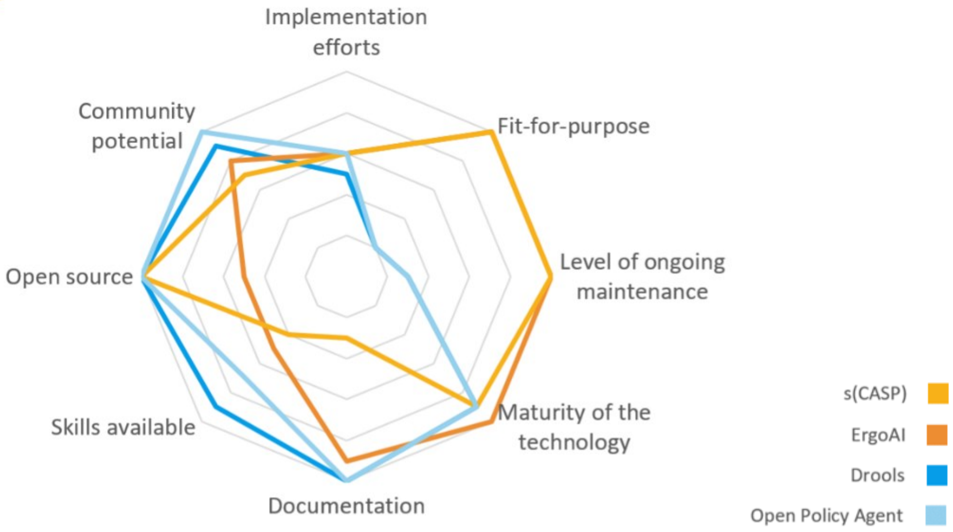

Fig. A 1. Scoring of Rule Engine Candidates

- **s(CASP)**[11] is an open-source Intelligent Agent based on Prolog which provides a solver for logical programming problems with constraints. It also provides support for advanced features like defeasibility and integration with templates written in natural languages. It was successfully applied to complex problems in the healthcare industry, such as building an experimental Physician Advisory System for Chronic Heart Failure Management Based on Knowledge Patterns (Chen 2018), where it was shown to outperform cardiologists in accuracy of decision-making.

References

Z. Chen et al., 2018. *An AI-Based Heart Failure Treatment Adviser System.* in IEEE Journal of Translational Engineering in Health and Medicine, vol. 6, pp. 1-10, 2018, Art no. 2800810, doi: 10.1109/JTEHM.2018.2883069

B. Grosof and J. Bloomfield, 2016. *Advanced Decision Analytics via Deep Reasoning on Diverse Data: For Health Care and More.* DecisionCAMP 2016

[11] https://gitlab.software.imdea.org/ciao-lang/sCASP