Developing a Reusable Clinical Decision Support System Module for Immunization Recommendations: A Case Study with OpenEMR and OpenMRS

Akash Shanmugan Boobalan*, MS[1], Samuil Orlioglu*, MS[1], Richard D. Boyce, PhD[2], Hua Min, PhD[3], Yang Gong, PhD[4], Dean F. Sittig, PhD[4], Paul Biondich, MD[5], Adam Wright, PhD[6], Christian Nøhr, PhD[7], Timothy Law, DO[8], David Robinson, MD[9], Arild Faxvaag, PhD[10], Nina Hubig, PhD[1], Ronald Gimbel, PhD[1], Lior Rennert, PhD[1], Xia Jing, PhD [1]

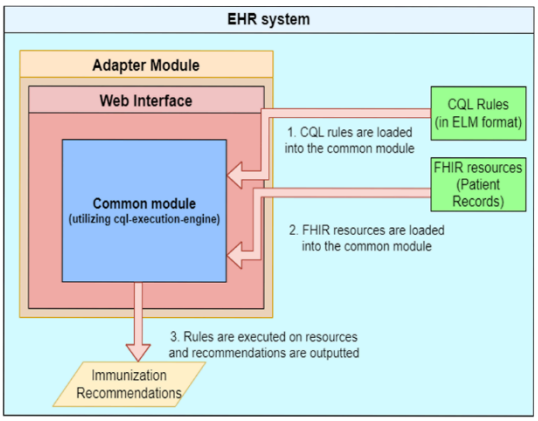
[1]Clemson University, Clemson, SC, USA; [2]University of Pittsburgh, Pittsburgh, PA, USA; [3]George Mason University, Fairfax, VA, USA; [4]University of Texas Health Sciences Center at Houston, Houston, TX, USA; [5]Indiana University, Indianapolis, IN, USA; [6]Vanderbilt University, Nashville, TN, USA; [7]Aalborg University, Aalborg, Denmark; [8]Ohio University, Athens, OH, USA; [9]Independent Consultant, UK; [10]Norwegian University of Science and Technology, Norway; *co authors contributed equally

Introduction

Clinical Decision Support Systems (CDSS) have been broadly used to improve care quality and patient safety, reduce medical errors, and enhance health service efficiency. A common use of CDSS is for immunization recommendations. Immunization is a vital strategy for preventing infectious diseases and a critical factor in prolonging life expectancy. Therefore, it is essential for care providers to provide consistent and up-to-date immunization administration for every patient every time. This type of CDSS is usually rule-based. A rule is a set of combined boolean conditions that map a patient's information (e.g., age, gender, diagnosis, previous vaccine record) to an action. Unfortunately, a manifestation of the interoperability challenge is that CDSS rules are often not reusable across institutions. Moreover, electronic health records (EHR) systems use diverse technological implementations. For instance, the OpenMRS[1] system is Java-based, while OpenEMR[2] is built with PHP. The lack of interoperability is exacerbated by the systems being tailored to differing specific concepts, thus lacking a cohesive ontology. Furthermore, the ever-evolving landscape of scientific and medical knowledge introduces new terminology, rendering reusable modules increasingly complex. These are just some examples of the challenges in developing a reusable CDSS module, which we are doing by leveraging reusable CDSS rules that can be particularly helpful in resource-limited settings. OpenEMR and OpenMRS are used as two examples of EHR systems, which are open source and mainly adopted and used in developing countries, and immunization recommendations are common CDSS in these settings.

Methods

Some commonalities between OpenMRS and OpenEMR can be used to create a common CDSS module. Both systems have support for Fast Healthcare Interoperability Resources (FHIR)[3] and have a web-based user interface. These characteristics provide a critical foundation for a common solution. However, different EHR systems may have different implementations of FHIR, therefore we use two systems to demonstrate the solution.



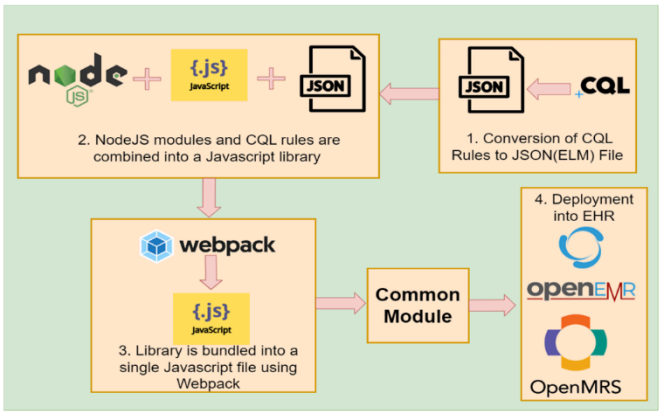


Figure 1: Internal processes during common module Operation (left)
Figure 2: Development and deployment process of the common module (right)

Figure 1 demonstrates the internal processes that the common module uses to execute rules to output a recommendation. Our strategy involves formulating CDSS rules in the Clinical Query Language (CQL)[4] format and developing a JavaScript-based CDSS module that retrieves patient information in FHIR format and uses the CQL rules to generate a recommendation for that patient. Because this module operates on the client-side, the data is safe since the client is already trusted if it has access to the EHR system. Given that both systems are web-based, they can readily accommodate JavaScript integration.

The development includes creating a common JavaScript-based CDSS module, adaptor module for the two EHR systems, and integrating the common module with OpenEMR and OpenMRS (Figure 2). The development of the common module entails the conversion of CQL rules into an expression logical model (ELM) file (a JSON file storing rules) and building the JavaScript module. The CQL rules are based on CDC-recommended immunization schedules. For the JavaScript module, we used cql-execution[5], which provides a Typescript/JavaScript library for executing CQL artifacts. We also used the cql-exec-fhir[6] library, an FHIR-based data source library, with the cql-execution library. It is used to parse FHIR data sources and execute CQL rules on them. Lastly, we used the cql-exec-vsac[7] library, which allows the CQL Execution Engine to execute CQL containing references to Value Sets that are published in the National Library of Medicine's Value Set Authority Center. Our JavaScript files that make up the common module are bundled into a single distributable JavaScript file using WebPack[8]. A single JavaScript file allows for integration with OpenEMR, OpenMRS and other web-based EHRs. We also create adapter modules for the two EHRs that install the common module and configure it to the EHR. Adapter modules integrate the common module into EHRs, thus allowing flexibility without comprising the system-independent nature of the common module.

Results

Currently, we have converted 19 vaccines and 465 rules from CDC Immunization Recommendations into a tabular format which enables us to convert to CQL format. We have successfully developed and are improving a preliminary version of a common module. This version was able to output consistent immunization recommendations on both OpenMRS and OpenEMR, given the same patient and a sample CQL ruleset. The initial success of this preliminary version confirms the validity and feasibility of our approach.

Next steps

Our next steps include completing conversion of all rules to CQL, building and testing the following components: functional CDSS module for all vaccines, CDSS rule management component for authorized users to modify them, and CDSS rule reporting component tracking rule utilization and performance. These will be implemented with REST APIS for ease of usage and future maintenance. We are also exploring ways to test and verify that rules output accurate recommendations based on sample patient inputs. A SMART-on-FHIR app for immunization can be a future endeavor.

Discussion

The adoption of disparate EHR systems provides ample opportunity for innovation that will enable higher-quality healthcare delivery. CDSS can assist physicians in providing safer clinical care. Our solution requires the EHR system to have certain specifications. First, it must support FHIR resources; second, the system must have a web-based component. A reusable CDSS module that can integrate with varied EHR systems could benefit patients; it will help standardize the industry and simplify implementing recommendations from authorities such as the CDC. The reusable CDSS rules and common CDSS module can be made available to the public to be shared, integrated into other systems, and reused. This will be especially helpful in resource-limited settings. Thus, a successful project will bring us one step closer to achieving CDS system interoperability.

Acknowledgments

The work is supported by the National Institute of General Medical Sciences (R01GM138589), Office of Data Science Strategy (3-R01-GM138589-03S1) of National Institutes of Health (P20GM121342 and T15LM013977).

References

1. Product – OpenMRS.org [Internet]. OpenMRS.org; 2023. Cited 2023 Sep 7. https://openmrs.org/product/.
2. OpenEMR [Internet]. OpenEMR Foundation, Inc; 2023. Cited 2023 Sep 7. https://www.open-emr.org/.
3. Index - FHIR v4.0.1 [Internet]. hl7.org; 2023. Cited 2023 Sep 7. https://www.hl7.org/fhir/R4/.
4. Clinical Quality Language (CQL) [Internet]. hl7.org; 2023. Cited 2023 Sep 7. https://cql.hl7.org/.
5. CQL Execution Framework [Internet]. Clinical Quality Framework; 2023. Cited 2023 Sep 7. https://github.com/cqframework/cql-execution.
6. CQL Execution FHIR Data Source [Internet]. Clinical Quality Framework; 2023. Cited 2023 Sep 7. https://github.com/cqframework/cql-exec-fhir.
7. CQL Execution VSAC Code Service [Internet]. Clinical Quality Framework; 2023. Cited 2023 Sep 7. https://github.com/cqframework/cql-exec-vsac.
8. webpack [Internet]. webpack; 2023. Cited 2023 Sep 7. https://webpack.js.org/.