MEDINFO 2019: Health and Wellbeing e-Networks for All
L. Ohno-Machado and B. Séroussi (Eds.)

doi:10.3233/SHTI190318

Implementation of Clinical Decision Support Services to Detect Potential Drug-Drug Interaction Using Clinical Quality Language

Binh-Phi Nguyen[a], Thomas Reese[c], Stefen Decker[a,b], Daniel Malone[d], Richard D. Boyce[e], Oya Beyan[a,b]

[a] Informatik 5, RWTH Aachen University, Aachen, Germany,
[b] Fraunhofer FIT, Sankt Augustin, Germany
[c] Department of Biomedical Informatics, University of Utah, Salt Lake City, Utah, USA
[d] College of Pharmacy, University of Arizona, Tucson, Arizona, USA
[e] Department of Biomedical Informatics, University of Pittsburgh, Pittsburgh, Pennsylvania, USA

Abstract

Potential drug-drug interactions (PDDI) rules are currently represented without any common standard making them difficult to update, maintain, and exchange. The PDDI minimum information model developed by the Semantic Web in the Healthcare and Life Sciences Community Group describes PDDI knowledge in an actionable format. In this paper, we report implementation and evaluation of CDS Services which represent PDDI knowledge with Clinical Quality Language (CQL). The suggested solution is based on emerging standards including CDS Hooks, FHIR, and CQL. Two use cases are selected, implemented with CQL rules and tested at the Connectathon held at the 32nd Annual Plenary & Working Group Meeting of HL7.

Keywords:

Potential Drug-Drug Interaction, Clinical decision support systems, Electronic health records.

Introduction

Drug-drug interactions (DDI) are biological processes that result in a clinically meaningful change to the response of at least one co-administrated drug [1]. Identifying potential DDIs during the care process is important to ensure patient safety and health care quality. DDIs can often be predicted and mitigated at the point of care. However, a large number of drugs, lack of knowledge of DDIs [2] hinders the ability of physicians to preemptively identify all potential DDIs [3]. Therefore it is important to guide care providers with well design computerized alerting systems during the medication prescribe process [4].

In the last two decades, numerous clinical decision support (CDS) systems have been developed to support physician by presenting alerts in real time to the clinician about the current medication's potential impact to patients [5]. Ideally, CDS should provide clinicians with the relevant reference information such as knowledge or suggestions, intelligently filtered and presented at appropriate times [6]. Currently, there are many CDS systems that provide drug interaction alerts [4]. Most of the systems curate their own knowledge base since there is no complete source of potential DDI (PDDI) knowledge [7]. Also, there are currently no broadly accepted standards to guide implementers in the organization and presentation of PDDI information [8]. Most EHR vendors either develop their own internal rule engines based on their custom standards, or contract with a third party. Since these rules are written in a way just for those systems, it is difficult to exchange, review and rewrite the rules. Lack of agreement on standards hinders the reuse of the rules by third parties, creating additional burden to rewrite rules at each side.

The minimum information models help to describe the PDDI information in a detailed, actionable and contextualizable format. Standardized PDDI information can be exchanged via broadly applicable formats. In this research, we will demonstrate the use of the PDDI minimum information model developed by Semantic Web in Healthcare and Life Sciences Community Group [9], by implementing them with CQL language and extend CQF Ruler[1] to provide decision support.

The aim of this work is to present an example implementation of representing PDDI logic in CQL and execution of them by using the FHIR Clinical Reasoning module. We selected Digoxin-Cyclosporine and Warfarin-NSAIDs as use cases because they are non-trivial PDDIs for which alerts can be contextualized to specific patient cases. The developed prototype detects a PDDI and provides alerts using Clinical Decision Support Hooks to Electronic Health Record systems that subscribe to the CDS services.

The following sections provide an overview of our methods, prototype architecture, implementation of PDDI with CQL, and evaluation.

Methods

PDDI: The PDDI Minimum Information Model [9] [10] is a standard proposed by the W3C Semantic Web in Healthcare and Life Sciences Community Group. It is aimed to help the clinicians to keep up with the PDDI evidence base, document and share PDDI information by summarizing PDDI evidence from primary sources using the information elements from the PDDI minimal information model.

The information model contains 10 core information items that should be used to describe every PDDI CDS knowledge artifact as follows: (i) Clinical consequences; (ii) Contextual information/modifying factors; (iii) Drugs involved, (iv) Evidence; (v) Frequency of exposure to the interacting drug pair; (vi) Frequency of harm for persons who have been exposed to the interacting drug pair; (vii) Mechanism of the interaction, (viii) Recommended actions; (ix) Seriousness rating; and (x) Operational classification of the interaction.

CQL: Clinical Quality Language (CQL)[2] is a Health Level Seven International (HL7) authoring language standard that can

[1] https://github.com/DBCG/cqf-ruler

[2] http://cql.hl7.org

be used with both CDS and electronic Clinical Quality Measures to represent the PDDI information. The CQL syntax provides a clinical focused, author-friendly and human-readable language to clinical domain experts, since it allows for rich, modular and flexible expression of the logic, and supports different data models including FHIR. CQL is a query language built up by combining rules, namely statements, to describe the available data in terms of a data model. For decision support, the rules will be evaluated in the context of a specific patient to produce a response at some specific point in a workflow.

CDS Services: CDS Service [11] is a role which provides real-time clinical decision support as a remote service. It enables a consumer to ask for clinical decision support based on his current context. The consumer gives relevant contextual information as part of the request and receives clinically relevant suggestion describing potential actions to be taken. The service implementation must comply with CDS Hooks specification.

CDS-Hooks[3] is an emerging standard gained considerable interest from EHR vendors. It is a "hook" based pattern designed to provide a simple way to initiate requests for CDS, from any point in a clinical workflow. It specified the basic actions of registering for CDS services, calling those services, and then receiving the CDS service response in form of simple cards providing appropriate information within the context of the EHR.

Concept

Architecture: As shown in the Figure 1, the PDDI CDS Implementation was designed as a self-contained service, with three main components including HAPI FHIR server[4], CDS Services, and CQL engine[5]. The first component is the HAPI FHIR server which is an option to store the definition of services, called PlanDefinition, supported by the system and knowledge base of CQL rules, called Library. The second component is the CDS services, which serve as a core logic to map the incoming request to the corresponding service defined by PlanDefinition, execute the CQL logic of the service defined in Library following the instruction from the PlanDefinition, and generate response cards. By this way, the new service can be introduced into the system without changing the core CDS services and the configuration. The third component is the CQL engine which provides CQL rule execution for the core CDS services.

As described in detail in the current draft of the PDDI Implementation Guideline (IG) [12], the CDS Discovery service is hosted at a stable endpoint, i.e. {baseUrl}/cds-services, and allows EHRs to discover the list of available supported CDS Service, e.g. {baseUrl}/cds-services/warfarin-nsaids-cds. The list of CDS Services contains information such as a description of the CDS Service, when it should be invoked, and any data that is requested to be prefetched.

PDDI CDS Implementation Workflow: The PDDI IG envisions two hooks which are medication-prescribe at order authorization, and medication-select at the time of selecting a medication and prior to the order authorization. For this paper, we focus only on medication-prescribe in the Level 1 Implementation.

Firstly, the EHR invokes the "medication-prescribe" hook and PDDI CDS service is called by sending an HTTP POST request, namely CDS Hooks Request, containing JSON to the service endpoint (e.g. {baseUrl}/cds-services/warfarin-nsaids-cds).

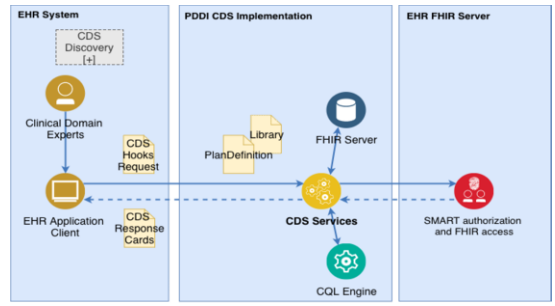

Figure 1 – The architecture and process of PDDI CDS Implementation

The CDS Hooks request contains specific information for the hook that was triggered including hook name FHIR server url, context data that the service will need, and prefetch data that is pre-queried data based on the CDS discovery. If the PDDI CDS service does not receive prefetch data in the request, it will query the EHR FHIR Server via network call with the authentication given by the EHR Application Client.

When CDS Service processes the request, the corresponding PlanDefinition and Library resources are loaded from FHIR server. Once the resources are loaded, the CQL logic in Library resource is decoded and evaluated by CQL engine with the data received either in the prefetch or from EHR FHIR server. The CQL engine evaluates the CQL logic following the guidance specified in the PlanDefinition resource.

After the evaluation is done, CDS Response cards are generated and returned to the client. Each Card has specified attributes including summary, detail, indicator, and list of suggestions providing actionable information. The specified attributes map to the core elements of the minimum information model (e.g. summary = Drugs Involved, detail = Clinical consequences, Seriousness, Mechanism of Interaction, and Evidence). The Card indicator element dictates how the EHR presents the alert (e.g. indicator = "hard-stop" could be a modal alert).

CDS Services Workflow: When the PlanDefinition is loaded by PDDI CDS services for the corresponding request, the services will load the CQL library defined in the PlanDefinition and decode it based on base64format.

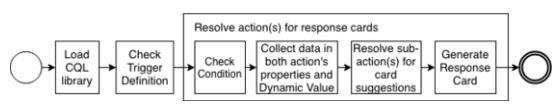

Figure 2 – The flow of processing Plan Definition

After the CQL library is loaded, the services check the hook event specified in the incoming request with the trigger definition defined in the PlanDefinition. If the hook event meets the trigger definition, the services begin to resolve the actions by evaluating the condition defined in CQL library to see whether or not the definition specified in the Action is to be applied. Then, the services collect the data in the action's properties and evaluate the dynamic value specified in the CQL library for the customizable properties. For the suggestions, the services resolve the sub-actions as the same approach as resolving actions.

[3] https://cds-hooks.org
[4] http://hapifhir.io
[5] https://github.com/DBCG/cql_engine

When the data is ready, the corresponding response card is generated and the services continue to resolve the next action until no actions left. Once finished, the services return the response cards as a result to the client.

Results

The implementation focuses on two PDDI CDS use cases which are Digoxin + Cyclosporine and Warfarin + NSAIDs. This section only discusses the Digoxin + Cyclosporine use case. Please refer to our repository (http://doi.org/10.5281/zenodo.1481220) for the full implementation and the PDDI IG [12] for the detail including PDDI model of the two use cases.

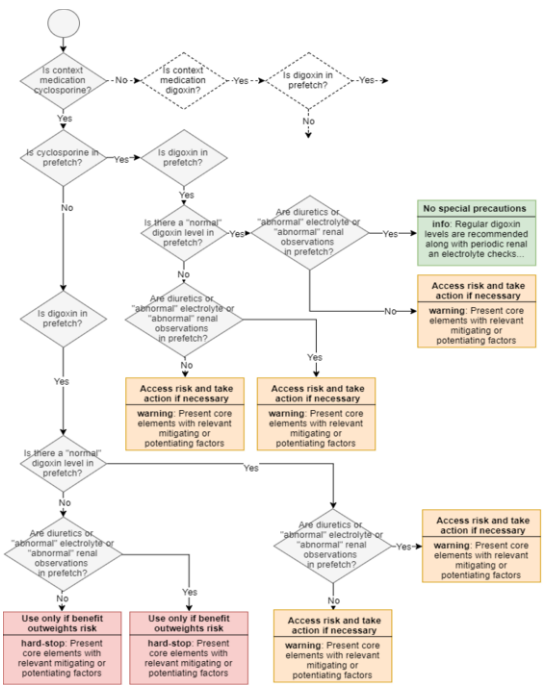

Figure 3 – The partial of decision tree for Digoxin Cyclosporine use case [12]

Use Case

The Digoxin + Cyclosporine exemplar artifact has two main decision blocks. Whether the patient is taking digoxin and/or cyclosporine at the time of the current order for digoxin or cyclosporine, and whether the patient has risk factors that may potentiate the risk of digitalis toxicity.

As shown in the Figure 2, each decision block has a certain suggestion in rectangle shape which presents core elements of PDDI model, necessary actions and level of indicator (i.e. "info" – green, "hard-stop" – red, and "warning" – orange).

Implementation of Plan Definition: PlanDefinition is a definition of the service served as a guide for the incoming request. PlanDefinition contains a set of definition including Library, Trigger Definition, Condition, Action, and Dynamic Value (Figure 3).

The Library element defines the reference to the logic used by the PlanDefinition. An example of digoxin-cyclosporine-cds PlanDefinition is the reference to Library/digoxin-cyclosporine-cds Library. This Library contains the CQL logic library encoded in base64 format as a string used by the PlanDefinition.

The Trigger Definition uses the Name Event, which allows triggering of an event opposed to a scheduled or fixed event. As an example, by specifying medication-prescribe as an event, the service only serves for the incoming request sent from medication-prescribe hook.

The condition element is used to determine whether or not the CDS logic specified in the Library is to be applied. By specifying "Inclusion Criteria" as a CQL statement for the condition, an action(s) is initiated once the condition is satisfied (i.e., true or false).

The Action element defines a list of response card needed to be generated. The subaction element indicates a list of suggestion which recommends a set of changes in the context of the current activity.

The Dynamic Value enables customization of the response card's properties by collecting the dynamic data defined in CQL statements. For the action, we define three dynamic values, such as "Get Summary" statement for the card title, "Get Detail" for the card description and "Get Indicators" for the card level. Since each decision block for PDDIs has one or more individualized information components, integrating patient-specific and product-specific data into Card elements is facilitated by the Dynamic Value element.

Implementation of CQL: All artifact logics of CQL for the Digoxin Cyclosporine use case are wrapped in the Digoxin_Cyclosporine_CDS library. A set of declarations, including data model, included libraries, valuesets, parameters, and context, provide information about the library.

The PDDI CDS uses FHIR model, version 3.0.0 as the primary data model to support for the FHIR resources.

A common library, namely PDDICDSCommon, contains all supported statements. The PDDI IG provides a number of valueset used in this library. Each valueset describes RxNorm codes drawn from one or more code systems for certain drugs. For example, the "Digoxin" valueset has this number of codes which is 197604 for Digoxin 0.125 MG Oral Tablet, 245273 for Digoxin 0.0625 MG Oral Tablet and other RxNorm codes. CQL evaluate the rules using all these RxNorm codes.

A ContextPrescription parameter refers to the list of Medication Request prescribed by the clinician specified in the incoming request. The Patient context restricts the information within a scope of single patient.

As specified in the condition element of the PlanDefinition, the "Inclusion Criteria" statement is used to check whether the patient of the incoming request is taking digoxin and/or cyclosporine at the time of the current order for digoxin or cyclosporine.

```
define "Inclusion Criteria":
  ( "Is Context medication cyclosporine"
      and "Is digoxin in prefetch"
  ) or (
    "Is Context medication digoxin"
      and "Is cyclosporine in prefetch"
  )
```

To express "Is Context medication cyclosporine" criterion, we need to check the existence of the list of Medication Request containing Cyclosporine specified in the ContextPrescription. The "Cyclosporine Prescription" statement requires that all codes from medication of the ContextPrescription parameter belong to the valueset identified by "Cyclosporine".

```
define "Is Context medication cyclosporine":
  exists ("Cyclosporine Prescription")
```

```
define "Cyclosporine Prescription":
  ContextPrescriptions P
    where Common.ToCode(P.medication.coding[0]) in
"Cyclosporine"
```

Next, the second criterion is "Is digoxin in prefetch". This criteria requires that the code of MedicationRequest is the code in the valueset identified by "Digoxin". Since we are in the Patient context, this query retrieves all Medication Request for only current Patient in the context. To filter the results which authored within the 100-day look-back period, we had to construct an interval from the 100-day look-back to infinitive.

```
define "Is digoxin in prefetch":
  exists ("Digoxin Rx")

define "Digoxin Rx":
  [MedicationRequest: "Digoxin"] MR
    where MR.authoredOn.value in Interval[Today()-100 days,null]
```

We use the same approach for the remaining criteria.

As defined in the dynamicValue element of the PlanDefinition for the “summary” property of the response card, the “Get Base Summary” statement provides the short description of the drugs involved in this interaction.

```
define "Get Base Summary":
  'Potential Drug-Drug Interaction between digoxin ('
    + (if "Is Context medication digoxin"
        then Common.GetDrugNames("Digoxin Prescription")
      else Common.GetDrugNames("Digoxin Rx")
    )
    + ') and cyclosporine ('
    + (if "Is Context medication cyclosporine"
        then Common.GetDrugNames("Cyclosporine Prescription")
      else Common.GetDrugNames("Cyclosporine Rx")
    )
    + ')'
```

For the “detail” property of the response card, the “Get Base Detail” statement provides detail of PDDI minimum information.

```
define "Get Base Detail":
  'Increased risk of digoxin toxicity...'
```

For the “indicator” property of the response card, the “Get Base Indicator” statement specifies the importance of what this card conveys.

```
define "Get Base Indicator":
  if "Is Context medication cyclosporine" then
    if "Is cyclosporine in prefetch" then
      if "Is there a normal digoxin level in prefetch" then
        if "Are diuretics or abnormal electrolyte or abnormal
renal observations in prefetch"
          then 'info'
        else 'warning'
      else 'warning'
    else
      if "Is there a normal digoxin level in prefetch" then
        then 'warning'
      else 'hard-stop'
  else
    if "Is digoxin in prefetch" then
      if "Is there a normal digoxin level in prefetch" then
        if "Are diuretics or abnormal electrolyte or abnormal
renal observations in prefetch"
          then 'info'
        else 'warning'
      else 'warning'
    else 'warning'
```

The first criterion of the “Get Base Indicator” statement is "Is there a normal digoxin level in prefetch". This criterion requires that all codes of Observation are in the valueset identified by "Digoxin LOINC". The normal digoxin observation is identified by measure the number of quantity less than 0.9ng/mL. To filter the results which take effects within the 30-day look-back period, we had to construct an interval from the 30-day look-back and infinitive. Because the most recent of Observation is important for the check, we select the latest result after ordered by the effective date.

```
define "Is there a normal digoxin level in prefetch":
  exists ("Normal Digoxin Observation")

define "Normal Digoxin Observation":
  Last (
    [Observation: "Digoxin LOINC"] O
      where O.effective.value in Interval[Today()-30 days, null]
        and Common.ToQuantity(O.value) < 0.9 'ng/mL'
      sort by effective.value
  )
```

The second criterion of the “Get Base Indicator” statement is "Are diuretics or abnormal electrolyte or abnormal renal observations in prefetch". This criteria contains three main components which are the diuretics medication requests involved, the abnormal electrolyte observations involved, and the abnormal renal observations involved.

We apply the same approach to detect the diuretics medication requests involved by checking the existence of Aldosterone Antagonists and Loop Diuretics in the Medication Request. For the remaining components, a similar approach for the abnormal electrolyte observations and renal observations has been used to detect the abnormal level of Potassium, Magnesium, Calcium, and Renal.

Table 1 – The variety of resources tested in two use cases

Warfarin-NSAIDs	Digoxin-Cyclosporine
MedicationRequest	MedicationRequest
MedicationDispense	MedicationDispense
MedicationStatement	MedicationStatement
MedicationAdministration	MedicationAdministration
Patient	Patient
Encounter	Encounter
Condition	Observation

Evaluation

We prepared FHIR resources (Table 1) to cover all cases of two decision trees and tested around 170 FHIR resources in STU3 version for 21 different patients on 14 cases of Digoxin-Cyclosporine and 7 cases of Warfarin-NSAIDs decision tree to support the draft IG. We also performed the evaluation at the Connectathon held at the 32nd Annual Plenary & Working Group Meeting of HL7 held in September 2018.

The evaluation was done using Postman[6] and CDS Hooks Sandbox[7], a tool developed by CDS Hooks team demonstrate how CDS Hooks would work with an EHR system. In brief, the clinician enters the medication, e.g., Ketorolac Tromethamine 10 MG Oral Tablet, for the specific treatment, and the CDS Hooks Sandbox then invokes the "medication-prescribe" hook to send the request to the PDDI-CDS services endpoint (e.g. {baseUrl}/cds-services/warfarin-nsaids-cds). After the PDDI CDS service processes the request, the CDS Response cards are returned. Finally, the CDS Response cards is presented by the CDS Hooks Sandbox.

Discussion

We were able to implement a PDDI CDS service using CQL, FHIR, and CDS Hooks. This shows the feasibility and that CQL was sufficiently expressive to cover to realistic use cases. We suggested enhancements to CDS Hooks that would enable it to support PDDI CDS using the minimum information model, and the combination of tools indeed performed PDDI CDS as a

[6] https://www.getpostman.com

[7] https://sandbox.cds-hooks.org

Figure 4– The screenshot of CDS Hooks Sandbox tool to test our PDDI-CDS services through CDS Developer Panel

service for two patient use cases. One of the limitations is a lack of tools supported for CQL because the community who is using CQL and CQL itself is entirely new. Therefore, it takes a lot of time in debugging to find out the problem.

As described in the Concept section, we need two components including FHIR server and CQL engine to implement CDS services. There are many FHIR server implementations including HAPI FHIR, FHIR .NET API[8] and others. However, HAPI FHIR is the most stable and popular implementation. It has a well-written documentation and full FHIR version support. Apart from that, there are less choices for CQL engine and we chose CQL-Evaluation-Engine from Database Consulting Group. This implementation is based on CQF-Ruler open source developed by Database Consulting Group, and Level 1 implementation of the draft PDDI-CDS Implementation Guide developed by HL7 Clinical Decision Support Work Group.

The Level 1 Implementation uses a single CDS service call and response with the medication-prescribe hook. As a future work, we plan to implement Level 2 implementation which we are working with the CDS Hooks developers to define the new medication-select hook and clarify how DetectedIssues will be returned in the card responses.

To create the CQL artifacts, the CDS Authoring Tool[9] is a promising candidate which is a component of the CDS Connect project funded by the Agency for Healthcare Research and Quality (AHRQ). The CDS Authoring Tool introduces an interface for composing CDS logic step by step using simple forms and exporting it as CQL artifacts using the HL7 FHIR DSTU 2 data model.

Conclusions

Based on the PDDI-CDS IG, we successfully implemented PDDI CDS services supported the Level 1 implementation of PDDI-CDS IG. The implementation follows strictly the CDS Hooks specification which enables EHR system to interoperate and exchange the data by using FHIR STU3 standard to enhance the better clinical diagnostic decision making. In the future, we will focus on the Level 2 implementation to help advance the standards including the new mediation-select.

Acknowledgments

This work funded in part by grants from the United States Agency for Healthcare Research and Quality R01 LM011838, R21 HS023826 and R01 HS025984; and National Library of Medicine grants T15 LM007124 and R01LM011838.

References

[1] OBO Technical WG, Drug-drug Interaction and Drug-drug Interaction Evidence Ontology, (2018). http://www.obofoundry.org/ontology/dideo.html (accessed Nov 10, 2018).

[2] H. van der Sijs, J. Aarts, and et al., Turning Off Frequently Overridden Drug Alerts: Limited Opportunities for Doing It Safely, *J Am Med Inform Assoc*, **15** (2008), 439-448.

[3] R.T. Scheife, L.E. Hines, and et al., Consensus recommendations for systematic evaluation of drug-drug interaction evidence for clinical decision support, *Drug Saf,* **38** (2015), 197-206.

[4] T.H. Payne, L.E. Hines, and et al., Recommendations to improve the usability of drug-drug interaction clinical decision support alerts, *J Am Med Inform Assoc* **22** (2015), 1243-50.

[5] J.I. Wolfstadt, J.H. Gurwitz, and et al., The Effect of Computerized Physician Order Entry with Clinical Decision Support on the Rates of Adverse Drug Events: A Systematic Review, *J Gen Intern Med* **23** (2008), 451–458.

[6] A.M. Sirajuddin, J.A. Osheroff, et al., Implementation pearls from a new guidebook on improving medication use and outcomes with clinical decision support. Effective CDS is essential for addressing healthcare performance improvement imperatives, *J Healthc Inf Manag* **23** (2009), 38-45.

[7] K.W. Fung, J. Kapusnik-Uner, and et al., Comparison of three commercial knowledge bases for detection of drug-drug interactions in clinical decision support, *J Am Med Inform Assoc* **24** (2017), 806-812.

[8] M.Z. Herrero, I.B. Segura, and P. Martínez, Conceptual models of drug-drug interactions: A summary of recent efforts, *Knowledge-Based Systems* **114** (2016), 99-107.

[9] W3C Group, HCLS Drug-Drug Interaction, 2018. https://github.com/w3c/hcls-drug-drug-interaction/ (accessed Nov 10, 2018).

[10] M. Brochhausen, M. Herrero-Zazo, and et al., MPIO - A novel Minimum Potential drug-drug interaction Information Ontology implemented in OWL, *AMIA Informatics Summit*, San Francisco, CA. USA, 2018.

[11] HL7, HL7 IG: Decision Support Service, Release 1, (2018). http://www.hl7.org/implement/standards/product_brief.cfm?product_id=334 (accessed Nov 10, 2018).

[12] HL7, PDDI CDS Implementation Guide, (2018). http://hl7.org/fhir/uv/pddi/2018Sep (accessed Nov 10, 2018).

Address for correspondence
Oya Beyan, beyan@fit.fraunhofer.de.

[8] https://github.com/FirelyTeam/fhir-net-api

[9] https://cds.ahrq.gov/authoring/