On Implementing Clinical Decision Support: Achieving Scalability and Maintainability by Combining Business Rules and Ontologies.

Vipul Kashyap[a], Alfredo Morales[b], Tonya Hongsermeier[a]
[a]Clinical Informatics R&D, Partners HealthCare System, Wellesley, MA
[b]Cerebra, Inc., Carlsbad, CA
vkashyap1@partners.org

ABSTRACT

We present an approach and architecture for implementing scalable and maintainable clinical decision support at the Partners HealthCare System. The architecture integrates a ***business rules engine*** *that executes declarative if-then rules stored in a* ***rule-base*** *referencing objects and methods in a* ***business object model****. The rules engine executes object methods by invoking services implemented on the* ***clinical data repository****. Specialized inferences that support classification of data and instances into classes are identified and an approach to implement these inferences using an OWL based* ***ontology engine*** *is presented. Alternative representations of these specialized inferences as if-then rules or OWL axioms are explored and their impact on the scalability and maintenance of the system is presented. Architectural alternatives for integration of clinical decision support functionality with the invoking application and the underlying clinical data repository; and their associated trade-offs are discussed and presented.*

INTRODUCTION

Clinical care guidelines are important tools for reinforcing the adoption of best practices in clinical care and are intended to improve safety, quality and cost effectiveness[3] of patient care. As different payer agencies such as the Federal Government (through Medicare/Medicaid) and insurance agencies such as Blue Cross and Blue Shield move towards a pay for performance model, healthcare quality and patient outcome metrics, such as the JCAHO[1] and HEDIS[2] measures have come into focus. At Partners Healthcare, we seek to incorporate these measures within our clinical information systems.

Approaches for modeling and automation of clinical practice guidelines have been proposed over the years with different degrees of success. Some of them have contributed significantly to the state of the art: the Arden Syntax[4], EON[5], PRODIGY-3[6], PROforma[7], Asbru[8], GUIDE[9], Prestige[10] and GLIF3[11]. From an architectural viewpoint, GLIF3 deserves special discussion. The various steps in the GLIF3 guideline model have been delineated as[3]: *action and decision steps* to represent clinical actions and decisions; *patient state steps* to serve as entry points into a guideline; and *branch and synchronization steps* for modeling concurrency. In this paper, we present an approach for implementing clinical decision support that subscribe to GLIF3 architectural principles by using an industrial strength Business Rules Engine - iLOG[13] and an OWL[16] ontology engine, Cerebra[14].

We present an approach for architecting rule content that represents patient state encapsulated in classes and methods of a business object model. These classes and methods are referenced in a rule base containing a set of declarative if-then rules. The "if part" of a rule typically consists of boolean conditions on the patient state. The "then" part consists of actions such as updating the patient state, making clinical recommendations, and specifying medication orders, etc. We also propose further delineation of decision support logic into ***definitions*** and ***decisions***, where definitions correspond to characterization of patient states and classes; and decisions correspond to clinical recommendations and orders. Definitions are used in the context of assigning (or "classifying") a given patient to a particular state or class on the basis of her documented clinical profile. This approach leads to a modular architecture for decision support, and easier maintenance of rules in the face of changing definitions. Finally, integration of the clinical decision support component with the invoking application and the clinical data repository is also discussed. Different architectural alternatives are presented followed by a discussion of their advantages and disadvantages.

USE CASE

Consider the following guideline for lipid management suggested by the American Diabetes Association (ADA)[12]:

Lowering triglycerides and increasing HDL cholesterol with a fibrate are associated with a reduction in cardiovascular events in patients with clinical CVD, low HDL and near-normal levels of LDL (A). Lower triglycerides to <150 mg/dL (1.7 mmol/L) and raise HDL cholesterol to >40 mg/dL (1.15 mmol/L). In women, an HDL goal 10 mg/dL higher may be appropriate (C). Patient has CVD, triglycerides >150 and/or HDL<40 (for women HDL<50) and LDL levels are near normal.

The steps for implementing the above clinical guideline are:

- Create the *Business Object Model* that defines patient related classes and methods.
- Specify *Rules* to encode Decision Support logic.
- Delineate definitions characterizing patient states and classes and represent them in an *Ontology*.

BUSINESS OBJECT MODEL DESIGN

The business object model for the above guideline could be specified as follows:

```
Class Patient
method get_gender(): string;
method has_diabetes(): boolean;
method has_CVD(): boolean;
method get_last_triglycerides(): real;
method get_last_HDL(): real;
method get_last_LDL(): real;
```

The model describes patient state information by providing a class and set of methods that make patient state information available to the rules engine. The methods defined in the object model are executed by the rules engine which results in invocation of services on the clinical data repository for retrieval of patient data. These methods may also perform additional conditioning of data retrieved from the clinical data repository. This leads to interesting design choices for design and implementation of the business object model, which are discussed next.

Consider the methods `has_diabetes()` and `has_CVD()`, that determine whether a patient has diabetes or cardiovascular disease respectively. An alternative design would be to define a method `get_diseases()` that return a list of diseases which a patient suffers from. Whether the disease `diabetes` or `CVD` is on that list can be checked by the rule engine. This may be implemented by specifying an appropriate rule in the rule base. This simplifies the business object model, but introduces complexity in rule execution. The current design leverages optimized processing in the clinical data repository to determine whether a patient has diabetes or cardiovascular disease and at the same time reduces complexity in rule creation.

Consider the method `get_last_triglycerides()` that retrieves the last triglycerides reading for a patient from the clinical data repository. This approach assumes the existence of a service on the clinical data repository which would return the last triglycerides reading for a patient. In some cases the clinical data repository may return triglycerides readings for a patient. That may require additional computations to determine the "last" triglycerides reading. This may be implemented as a method in the business object model or as an additional rule in the rule base.

RULE BASE DESIGN

The business object model presented in the earlier section provides the vocabulary to specify various rules for implementing the clinical guideline. Consider the following rule specifications:

```
IF the_patient.get_gender() = "male"
AND the_patient.has_CVD() = "true"
AND the_patient.get_last_triglycerides()>=150
AND the_patient.get_last_HDL()<=40
AND the_patient.get_last_LDL()>=Value1
AND the_patient.get_last_LDL()<=Value2
THEN "order fibrate therapy"
```

The above rule represents the first part of the diabetes guideline. A special variable "the_patient" acts as a placeholder for the actual patient being evaluated. The definition of "near normal LDL" is modeled by assuming that the LDL reading of the patient lies between `Value1` and `Value2`. The next rule is similar to the above but has a higher threshold for HDL values for women and can be specified as follows:

```
IF the_patient.get_gender() = "female"
AND the_patient.has_CVD() = "true"
AND the_patient.get_last_triglycerides()>=150
AND the_patient.get_last_HDL()<=50
AND the_patient.get_last_LDL()>=Value1
AND the_patient.get_last_LDL()<=Value2
THEN "order fibrate therapy"
```

There are alternative ways of specifying these rules. For instance, the two rules above can be combined into one rule as follows:

```
IF the_patient.has_CVD() = "true"
AND the_patient.get_last_triglycerides()>=150
AND the_patient.get_last_LDL()>=Value1
AND the_patient.get_last_LDL()<=Value2
AND(  (the_patient.get_gender() = "male"
       AND the_patient.get_last_HDL()<=40)
    OR(the_patient.get_gender() = "female"
       AND the_patient.get_last_HDL()<=50))
THEN "order fibrate therapy"
```

This representation reduces the number of rules in the rule base which has a beneficial impact both on rule execution and maintenance. However, the rule specification itself has increased in complexity making it difficult for a knowledge author to understand and vet the rule for clinical validity effectively.

DEFINITIONS VS DECISIONS

Clinical decision support rules encode different types of inferences:

- Rule-based specifications of conditions that describe patient states and classes, for instance, "Diabetic Patient with Higher Risk for CVD" or characterize normal or near normal physiological patient states, for instance, "Patients with near normal values of LDL". These specifications are also called ***definitions***.
- Rule-based specifications that propose therapies, medications and referrals, for instance, prescribing fibrate therapy for a diabetic patient with higher risk for CVD. These specifications are called ***decisions***.

This observation provides us with an opportunity to further modularize our rule base by separation of the definition of a "Diabetic Patient with high risk of CVD", from decisions that are recommended once a patient is identified as belonging to that category.

The definitions in the rule base can be represented as follows:

```
IF the_patient.get_last_LDL()>=Value1
AND the_patient.get_last_LDL()<=Value2
THEN set the_patient.LDLcategory
          = "LDLNearNormal"

IF the_patient.has_CVD() = "true"
AND the_patient.get_last_triglycerides()>=150
AND  the_patient.get_LDLcategory()
                       = "LDLNearNormal"
AND the_patient.get_gender() = "male"
AND the_patient.get_last_HDL()<=40
THEN set the_patient.category
     = "DiabeticPatientWithHigherRiskOfCVD"

IF the_patient.has_CVD() = "true"
AND the_patient.get_last_triglycerides()>=150
AND the_patient.get_LDLcategory()
                       = "LDLNearNormal"
AND the_patient.get_gender() = "female"
AND the_patient.get_last_HDL()<=50
THEN set the_patient.category
     = "DiabeticPatientWithHigherRiskOfCVD"
```

The simplified rule base can now be represented as:

```
IF the_patient.get_category()
   = "DiabeticPatientWithHigherRiskofCVD"
THEN "order fibrate therapy"
```

The definitions of various patient states and classes can be represented as axioms in an ontology that could be executed by an OWL ontology engine, Cerebra[14]. At execution time, the ILOG rule engine can invoke a service that interacts with the Cerebra OWL engine to infer whether a particular patient belongs to a given class of patients, in this case, whether a patient is a diabetic patient with high risk of cardiovascular disease. The ontology of patient states and classes can be represented as follows:

```
Class Patient
ObjectProperty gender
ObjectProperty hasDisease
ObjectProperty lastTriglycerides
ObjectProperty lastLDL
ObjectProperty lastHDL

Class DiabeticPatient
≡ Patient ∩ ∃hasDisease.Diabetes

Class CVD
Class Diabetes
Class Disease
CVD ⊆ Disease
Diabetes ⊆ Disease

Class LDLNearNormal
Class HDLLessThan50
Class HDLLessThan40
Class TriglyceridesMoreThan150

Class DiabeticPatientWithHigherRiskofCVD
≡ DiabeticPatient ∩ ∃hasDisease.CVD ∩
∀lastTriglycerides.TriglyceridesMoreThan150
∩ ∀lastLDL.LDLNearNormal ∩
[[∀gender.{"male"}∩∀lastHDL.HDLLessThan40]∪
[∀gender.{"female"}∩∀lastHDL.HDLLessThan50]]
```

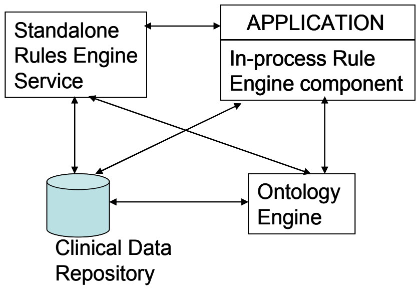
The specification of various definitions in the ontology is illustrated above. For simplicity, we have adopted a non-XML based notation although they will be implemented following the OWL specification[16]:

- The class `Patient` and properties `gender`, `hasDisease`, `lastTriglycerides`, `lastLDL` and `lastHDL` provide a framework for describing the patient. The class `DiabeticPatient` is a subclass of all patients that are known to suffer from diabetes. This is expressed using axiom following the Patient class definition.
- The class `Disease` represents various diseases and subclasses of interest, `CVD` and `Diabetes`.
- The classes `LDLNearNormal`, `HDLLessThan50`, `HDLLessThan40` and `TriglyceridedMoreThan150` represent ranges of values of normal LDL, HDL and Triglycerides respectively. Cerebra has implemented custom datatypes based on the OWL specifications[16], providing the ability to map XML Schema[17] datatypes to OWL Classes and supports range checking inferences on them.
- Finally, `DiabeticPatientWithHigherRiskofCVD` is defined using an axiom to characterize diabetic patients with higher risk of CVD. The representation of this axiom enables the rule author to simplify the rule base significantly (as only the rule in bold needs to be specified). The classification of a patient as being diabetic and

with high risk of CVD is now performed by the Cerebra Ontology Engine.

CLINICAL DECISION SUPPORT ARCHITECTURE

An architecture for implementing clinical decision support systems is illustrated below and consists of the following components:


Standalone Rules Engine Service
APPLICATION
In-process Rule Engine component
Ontology Engine
Clinical Data Repository

- **Clinical Data Repository:** The clinical data repository stores patient-related clinical data. External applications, the rule engine (via methods defined in the business object model) and the ontology engine retrieve patient data by invoking services implemented by the clinical data repository.
- **Standalone Rules Engine Service:** A standalone rules engine service is implemented using the ILOG business rules engine. On receiving a request, the service initializes a rule engine instance, loads the rule base and business object model. The rule engine service then executes methods in the business object model and performs rule based inferences. The results obtained will then be returned to the invoking application.
- **In-process Rule Engine Component:** This provides similar functionality to the rules engine service, except that the rule engine component is loaded in the same process space in which the application is executing.
- **Ontology Engine:** This will be implemented using the Cerebra Server. On receiving a request, the ontology engine performs classification inferences on patient data to determine if a patient belongs to a particular category, e.g., high risk patient.

We now revisit the design patterns discussed earlier and evaluate the trade-offs in the context of the architecture presented above.

1. In cases, where there is a lot of interaction between the invoking application and the decision support component, including it as an in-process component may reduce the time taken for execution as network latency between rule engine invocations will be minimized.
2. Caching of the patient state is likely to play a significant role in execution efficiency. For instance, in the case where specialized services check whether a patient has diabetes, is not available on the clinical data repository, the complete patient object will need to be populated so that the rule engine can check for existence of diabetes in the list of patient diseases. Efficient mechanisms to check, refresh and dispose cached patient state information will be required.
3. For large rule bases, the ability of the rule engine to leverage Rete Rule Matching computation[15] to rapidly identify rules that are likely to fire will be crucial.
4. Designing rule bases with a minimal set of rules will also be useful in speeding up rule engine execution. For example in the example discussed earlier, combining two rules into one may be helpful as it would result in one rule (as opposed to two) being loaded on the agenda.
5. Identification of a set of classification inferences that can be implemented by an ontology engine and invoked as a service from the rules engine offers significant potential for speeding up execution performance of the rules engine. A significant proportion of clinical decision support involves classification and this could result in reducing overhead on the rules engine and speeding up execution performance.

RULE AUTHORING AND MAINTENANCE CONSIDERATIONS

Some of the design patterns discussed earlier will have an impact when it comes to rules authoring and maintenance. We revisit some of the design patterns in this context.

1. The compactness of the business object model makes the job of maintenance easier. For instance, it's better to have a method for retrieving patient diseases rather than a large number of methods for checking the existence of a potentially large number of diseases a patient could have. On the other hand, this could increase the complexity of the rule base, should this checking be done via a specialized rule.
2. The ability to create a compact rule base by combining two or more rules into a single rule (as illustrated earlier) also makes the maintenance of the rule base easier. However, this could make the individual rules more complex and difficult to understand. This could impact the ease of rule re-use and editing.
3. The separation of definitions from decisions and their implementation in an ontology engine reduces the complexity of the rule base maintenance significantly. It may be noted that

the conditions that comprise a definition may appear multiple times in multiple rules in a rule base. Our approach enables the encapsulation of these conditions in a definition, for e.g., `DiabeticPatientWithHighRiskofCVD`. Thus all rules can now reference the class `DiabeticPatientWithHighRiskofCVD` which is defined and maintained in an ontology in the ontology engine. Whenever the definition of `DiabeticPatientWithHighRiskofCVD` changes, the changes can be isolated within the ontology engine and the rules that reference this definition can be easily identified.

The issue of rules maintenance is a very important one and decisions taken for modularization and re-design of the rule-base for execution performance are likely to be inter-dependent on each other.

CONCLUSIONS AND FUTURE WORK

We have presented an approach and architecture for implementing clinical decision support in a healthcare delivery system. An example clinical guideline was represented by designing a business object model that describes patient information and a rule base that makes inferences and suggests actions based on patient state. Furthermore, we propose delineation between definitions and decisions and the use of an ontology engine for performing classification inferences. This motivated an architecture for invocation of an ontology engine from a rules engine. A service oriented approach for retrieving information from the clinical data repository was presented. Alternative implementations of a rule engine component as an in-process component and as a stand alone service were also proposed. Different design decisions and trade-offs were discussed in the context of this architecture along with their impact on rule authoring and maintenance.

The work described in this paper is part of an ongoing project at Partners HealthCare to use an industrial strength business rules engine, ILOG and an ontology engine, Cerebra for implementing clinical decision support. Creation of a robust rules authoring and maintenance environment for rapid and consistent update of decision support knowledge is also being architected. We will investigate the following issues going forward:

- To what extent is it possible to isolate changes in definitions and the business object model from rules?
- What will be the impact of genomic and personalized medicine on clinical guidelines? Will our architecture be able to manage knowledge related to personalized medicine?
- Can semantic web technologies and reasoners based on OWL enable design of enhanced decision support and knowledge maintenance? Will this help support clinical decision support requirements for personalized medicine?

REFERENCES

1. http://www.jcaho.org
2. http://www.ncqa.org/Programs/HEDIS
3. Peleg, M, Boxwala AA, Tu S, Zeng Q, Ogunyemi O, Wang D, Patel VL, Greenes RA and Shortliffe EH, The Intermed Approach to Sharable Computer-Interpretable Guidelines: A Review. JAMIA, 2004;11(1):1-10.
4. Clinical Decision Support & Arden Syntax Technical Committee of HL7, inventor Arden Syntax for Medical Logic Systems, version 2.0 , USA, July 7, 1999.
5. Tu SW and Musen MA. A Flexible approach to Guideline Modeling. Proc. AMIA Syposium; 1999:420-424.
6. Sugden B, Purves IN, Booth N, Sowerby M. The PRODIGY Project – the Interactive Development of the Release One Model, Proc. AMIA Symposium; 1999:359-363.
7. Fox J and Rahmanzadeh A. Disseminating medical knowledge: the PROforma approach. Artif. Intell Med 1998;14:157-181
8. Shahar Y, Mikseh S and Johnson P. The Asgaard Project: A Task-Specific Framework for the Application and Critiquing of Time-Oriented Guidelines. Artif Intell Med 1998;14:29-51
9. Quaglini S, Stefanelli M, Lanzola G, Caporusso V and Panzarasa S. Felgible Guideline-based Patient Careflow Systems. Artif Intell Med 2001;22:65-80
10. Gordon C, Veloso M. Guidelines in Healthcare: The experience of the Prestige project: Medical Informatics Europe; Ljubljana, Slovenia; 1999:733-738
11. GLIF3 Technical Documentation. http://www.smi.stanford.edu/projects/intermed-web/guidelines/GLIF_TECH_SPEC.pdf
12. http://care.diabetesjournals.org/content/vol28/suppl_1/
13. http://www.ilog.com
14. http://www.cerebra.com
15. Forgy CL, Rete: A fast algorithm for the many pattern/many object matching problem. Artificial Intelligence, Vol 19(1), 1982.
16. http://www.w3.org/TR/owl-features/
17. http://www.w3.org/XML/Schema