Accuracy and performance evaluation of a clinical decision support system for laboratory result alerts

InSook Cho, SunTae Kim, JeongAh Kim, Kyu Seob Ha and Yoon Kim

aDepartment of Nursing, Inha University, Incheon, Korea; bDepartment of Computer Science and Engineering, Sogang University, Seoul, Korea; cDepartment of Computer Education, Kwandong University, Gangwon-do, Korea; dMood Disorder Questionnaire Clinic, Seoul National University, Bundang Hospital, Seoul, Korea; eInstitute of Health Policy and Management, Seoul National University, Seoul, Korea

(Received 7 September 2010; final version received 17 October 2010)

This study was designed to evaluate the accuracy and performance of a clinical decision support (CDS) architecture consisting of a knowledge engine and data interface adapter (DIA), which were developed as critical components for a clinical decision support system (CDSS). Accuracy and performance are the most essential quality attributes. CDSSs are becoming increasingly important in efforts to increase quality control in hospitals, but if a CDSS adds a new burden to the legacy system, the manager is unlikely to adopt it in the hospital information system. A laboratory alerting system was selected for evaluation, because of complex rules and the large amount of the data interface. Evaluation cases (N = 323,455) were captured from a tertiary care and teaching hospital over a 3-month period. We were able to confirm the accuracy, stability, and feasibility of the CDS application architecture and the knowledge engine, but we identified areas of the DIA that require improvement in order to obtain satisfactory performance.

Keywords: clinical decision support system; quality attribute; evaluation; accuracy; knowledge execution

1. Introduction

When used effectively, electronic health records (EHRs) can improve the quality of medical care. However, for maximum benefit, EHRs must be paired with a clinical decision support (CDS) system (CDSS) to influence clinician behavior effectively (Reichley et al. 2005) and impact healthcare processes and outcomes. Several issues must be resolved before CDSSs will be widely adopted, including difficulties translating medical knowledge and guidelines into a form usable by EHRs, the development of tools for doing so, and leveraging application of the system.

For several decades, the Arden Syntax (Clercq et al. 2004), Asbru (Shahar et al. 1998), Guideline Interchange Format version 3 (GLIF3; Boxwala et al. 2004), Guideline Elements Model (Shiffman et al. 2000), EON (MUSEN et al. 1996), Standards-Based Active Guideline Environment (SAG; Tu et al. 2007), PRODIGY (Purves et al. 1999), and PROforma (Fox et al. 1998) have been introduced as common shared models that facilitate direct interpretation or mapping to multiple implementation environments. SAGE, which was derived from previous works such as PROforma, GLIF3, EON, and PRODIGY, has advanced the state of the art by focusing on requirements that previous models had not met simultaneously, including (1) incorporation of workflow awareness, (2) adoption of information and terminology standards, (3) incorporation of simple flow-of-control standards, and (4) integration with a vendor clinical information system.

However, the SAGE execution engine is not currently available outside of the SAGE project. Thus, we developed a CDS application architecture, a knowledge engine, and an electronic medical record (EMR) data interface adapter (DIA) that can implement the SAGE formalism (Kim et al. 2008). In this article, we have examined that knowledge engine and the DIA in detail, and evaluated the clinical aspects of its performance by selecting laboratory alerts and converting the rules used in the laboratory alerting system into SAGE formalism. Most previously published studies used a run-time database lookup strategy based on an existing database structured query language (SQL) server platform and SQL code combined with a compiler (Iordache et al. 2001, Chen et al. 2002, Park et al. 2008). The proposed approach has practical advantages with regard to operational aspects. However, from the knowledge-sharing perspective, it causes dependencies on the local database – such
databases have not been standardized and are not readable by both humans and machines, which compromises interoperability among institutions.

It is important for physicians to be able to respond to notifications of critical laboratory test results in a timely and appropriate manner. Due to the importance of critical data that represent life-threatening situations, various types of automated alert systems have been used to prevent medical errors and to report critical laboratory data to physicians. Real-setting response times and signal-to-noise ratios are very important in these instances (Schedlbauer et al. 2009); we therefore focused on these two measures of system performance.

The rest of this article is organized as follows. Section 2 is the proposed architecture to be evaluated. Section 3 is the proposed evaluation method to validate the feasibility of architecture. The experimental results are explained in Section 4. Section 5 is the conclusion and suggestions for future work.

2. Implementation of engine and DIA

Knowledge-based platforms are services for supporting the medical management of clinical risk. Knowledge in medical areas is represented as up-to-date best scientific evidence from healthcare research into decision-making. Clinical guidelines in the medical field, which are the main components of evidence-based medicine, are systematically developed statements designed to assist practitioners and patients on decisions regarding appropriate healthcare for specific circumstances. Efforts to implement computer-interpretable guidelines in healthcare applications have been increased so that patient-specific guideline recommendations are available at the point of care. Implementing guidelines in computer-based decision support systems improves the acceptance and application of guidelines in daily care.

Many studies have described clinical guidelines in a reusable form and the execution of those guidelines to verify medical decisions (Goldberg et al. 2006, Kashyap et al. 2006) for medical information. These approaches, known as CDSSs, are intended to help prevent doctors from misdiagnosing patients and prescribing them medicines that may potentially cause problems. In order to reuse these guidelines in different hospitals, CDSSs should be built on an architecture that can cope with the heterogeneity over multiple clinical databases in data models, data structures, terminologies, and semantics (Sujansky and Altman 1996). Furthermore, the manual translation or creation of queries for a given EMR should be minimized, as these tasks are costly and prone to errors.

Diverse studies have investigated how to cope with the heterogeneous nature of existing clinical databases, most of which have introduced their own abstraction and mapping models that bridge the gap between the concepts in the guidelines and those in the clinical databases. Based on the mapping models, these previous works have proposed their own approaches that generate queries automatically. However, these studies have not examined the performance aspect of clinical data retrieval using the generated queries.

We have introduced and implemented a software architecture designed to execute the clinical guideline knowledge specified in the knowledge representation formalism of SAGE (Kim et al. 2009). Among the medical knowledge engines, SAGE (Tu et al. 2006) provides a particularly powerful knowledge representation and is a strong communication tool. The architecture proposed in (Kim et al. 2009) addresses run-time behaviors in the execution of the translated guidelines and provides data regarding verification of the conducted diagnoses. Figure 1 provides details of the software architecture and components of its workflow among components in detail.

The architecture depends upon the run-times of the converting and executing components. Converting time addresses a time that knowledge from SAGE is converted into the executable form used by the knowledge engine and DIA. During this time, knowledge authors specify clinical guidelines through SAGE and EMR data manager describes relationships between data part of SAGE and EMR. The data part of SAGE is represented in eVMR that stands for extendable virtual medical record in the figure. Finally, the knowledge is translated into the format that the knowledge engine (Kim et al. 2005) can execute and the queries are generated by Query Generator into the format for the query executor component (depicted in DIA in the figure) to access EMR data. The results are deployed in the place where the knowledge engine and DIA can read them.

Executing time denotes a time when the user’s diagnosis is verified automatically according to translated knowledge and queries by collaborating with Application, Knowledge Engine and DIA. After the application receives a request for verifying patient’s diagnosis, the DIA accesses the patient data in EMR from a hospital by executing the generated queries, and builds input data to the rule engine at first. Then, the input data is delivered to the knowledge engine to
verify user’s diagnosis in accordance with translated clinical guidelines.

2.1. Architectural tactics for increasing performance
This section introduces SW architectural tactics for increasing application performance. Performance of the application during execution time is divided into performance of application, knowledge engine, and DIA. Among the three components, the most influential component for the application performance is DIA, because DIA needs intensive network I/O to access the EMR database so that it must be one of the major causes of delay of system execution. Although all components can be allowed to deploy in a single machine, typically EMR database is not to be executed in the same process with the components, and it is distributed throughout the network in the general cases. In addition, the number of pieces of knowledge executed in the knowledge engine is fixed during the converting time so that it can be considered as a constant value for application performance. However, patient data that should be accessed by DIA may be huge. Thus, performance of DIA is the crucial driver to build SW architecture of the whole system in terms of system performance.

DIA carries out query execution to get patient data by accessing the EMR of hospital, and then sets the results of execution into the input model which is used as the input data of knowledge engine. In order to complete the function, DIA contains Query executor for executing the queries and Input Model Setter for setting the execution results in the input model template, as shown in Figure 2. For all queries, Query Executor executes a query and requests for Input Model Setter to set the result in the input model template.

In order to increase DIA performance, we adopted three architectural tactics: DB Connection Pool, Query queue, Pooled Thread, each of which is depicted in Figure 2 as a component. DB connection pool is a tactic that reuses database connections since they are created first. Because much time is consumed to make a new network connection to the data base, time can be saved by reusing the established connections through the DB connection pool. In addition, Query queue is a tactic to decrease query loading time, specified in a file on a hard disk. By accessing the query in the fast memory ever since it is used first, the loading time of queries can be decreased so that eventually it has a positive influence on application performance.

Similar to the DB connection pool, Pooled Thread is an architectural tactic for reusing created threads. In order to enable use of this tactic, steps that execute a single query and set the result to the input model are executed in a single thread, which is highlighted in the gray color of the figure. Executing these steps currently in multiple thread and revising them contribute to increase the performance of the system.
2.2. SW architectural support for achieving accuracy

Accuracy is one of the crucial quality attributes for a medical information system that handles critical information for a patient. Although accuracy is tightly related with the system functionality including correctness of knowledge authored in SAGE and patient data in EMR, architectural support also plays an important role to achieve accuracy of the system. The proposed CDSS system provides an approach to capturing input and output models in the application and storing them in the regression test oracle database for regression test. Figure 3 shows the architectural support for accuracy in the application.

In the architecture, the application has the KE Executor that requests the knowledge engine to conduct guideline checking for the input model and deliver it to the GUI component. It should be noted that the input model contains the patient data that is held by DIA. I/O Capturer component captures the input and output model and stores them in the test oracle database. Then, a knowledge author approves the models as a test case. Once the test cases are approved by the knowledge author, they can be reused to check the accuracy of the knowledge changes or system updates such as DIA and Knowledge Engine. It is meaningful to maintain the CDS system stably keeping accuracy after updating knowledge and the system itself. Whenever updating them, the accuracy of the system can be assured by replaying the knowledge execution process and comparing the results with stored output models.

2.3. Evaluation methods

This study was conducted at Bundang Hospital which provides a development and testing environment based on the system in real time service to medical stakeholders. This is a tertiary care and teaching hospital with 900 beds and has a computerized physician order entry system, EMRs, and a laboratory information system (LIS) with callback. Approximately 55,000 laboratory tests are performed daily, which includes tests ordered from the inpatient, outpatient, and emergency rooms.

We verified the accuracy of knowledge execution by applying a black-box testing mechanism in which knowledge engineers define the test case, which comprises an input test case, and the expected results. With this test case, our test oracle generates an input interface file to send to the knowledge engine, and compares the result from the knowledge engine with the expected results. The evaluated knowledge comprised ten alerting rules: nine based on the values of a single laboratory result, and one that detected changes in laboratory results over time. These rules were selected from a review of previous work (Park et al. 2008), expected prevalence of use, and potential morbidity associated with failure to perform appropriate laboratory monitoring by the CDSS at Bundang Hospital. A detailed description of the transformation and encoding of the rules into SAGE format was given in our previous work (Schedlbauer et al. 2009). Briefly, we used 18 concepts including concept qualifiers, and seven activity graphs.
We used a testing tool to simulate a laboratory alerting application that was programmed to trigger the system to retrieve data from a clinical data repository (CDR) every day. The CDR also received laboratory data from the LIS on a daily basis. The DIA requested the CDR data using standard terminology and mapped the data model used in the encoded rules with patient data from the CDR. Testing was conducted for 9 days in September 2008 using 323,445 retrospective laboratory results collected over a 3-month period.

We analyzed the tracking data that accumulated in the test-results database to evaluate the accuracy of each alert. The test-results database stored the data that were received from the CDR and the outputs created from the knowledge engine. Therefore, we were able to compare the input and output data of the engine as well as the failure cases that did not create any results. Figure 4 shows the transaction intervals measured to detect overall response time: (1) laboratory alerting system, (2) engine throughput, and (3) DIA output. The test was performed on a personal computer with a 1.86-GHz CPU, 1.5 GB of memory, and the Windows XP operating system.

3. Results
3.1. Alert volume and distribution
Based on the 323,445 laboratory results, 1650 alerts were created: 1164 (70%) for inpatients, 222 (13%) for outpatients, and 264 (16%) for emergency-room patients. The alert frequencies for inpatients were highest for falling hematocrit and hyperkalemia, at 764 (46.3%) and 203 (12.3%), respectively. The alert frequencies divided by the number of physicians in each department, corresponding to the amount of alerts a physician received per day, ranged from 1.0 for pediatrics to 6.0 for internal medicine, while the emergency room showed 2.9 alerts per day per physician. We were unable to adequately estimate the average alert frequency per physician in the outpatient setting due to daily variations in the number of physicians who saw patients.

The daily test distribution had a typical bell shape, with a peak point at around 10.00–11.00 a.m., except for Rh typing, which had a peak point at around 5.00–6.00 p.m. The test frequencies were highest for sodium, potassium, and white blood cell (WBC) count during the peak time period.

3.2. Alert accuracy
Accuracy refers to the completeness of LIS data mapping and correctness of an alert trigger based on the rule criteria. During testing, six points of the data mapping profile in the DIA were found and fixed. Several incorrect outputs caused by the converter transforming encoding knowledge into executable formats were also found and corrected. Through the
white- and black-box validation process, the accuracy rate reached 100%.

3.3. System performance

The overall system response time (interval ① in Figure 4) was 398.06 ± 799.95 ms (mean ± SD; range = 191.50–842.04 ms). Specifically, the rule for Rh typing consumed the largest amount of time and had the greatest variation. The run-time for the knowledge engine (interval ② subtracted from interval ①) was 51.90 ± 24.59 ms (range = 31.66–102.07 ms). Glucose rules consumed the most time (Figure 5).

The DIA took 346.16 ± 800.55 ms (range = 787.31–136.86 ms), and the Rh typing rule had the longest response time (Figure 6).

The peak time response was 475.23 ± 660.21 ms, except in the case of the Rh typing rule, which was typically used between 5.00 and 6.00 pm with a relatively low frequency and had the longest response time (Table 1).

According to the average response time analysis by clinical setting, inpatient and emergency cases exhibited similar overall and DIA times (Table 2). Engine times exhibited a consistent and stable turnaround, with an average of 50.0 ms regardless of department.

4. Discussion

It is important for physicians to respond to notifications of critical laboratory test results in a timely and appropriate manner because critical data may represent life-threatening situations. Therapeutic management by physicians of patients with critical data could thus serve as a valuable measure of laboratory outcomes (Howanitz et al. 2002). Specifically, the point-of-care reminders delivered when busy clinicians are reviewing or renewing medications in the hospital are useful, and their benefits have been reported previously (Kuperman et al. 1999). This study considered how such systems or services could be implemented and delivered widely.
In the healthcare industry, the knowledge assets underlying CDSSs are time-consuming, expensive to generate, voluminous, and subject to change. Thus, the ability to share and reuse this information once it is created would be highly advantageous (Kuperman et al. 1999). To facilitate sharing, the knowledge should be represented in standard form so that it can be disseminated and used widely; it also needs to be updated on a regular basis. An implementation environment is also important to realize knowledge sharing in the form of a CDSS. Therefore, in the present study we explored and evaluated the tools – CDS

Table 1. Average response time during the day at peak time by rule.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Mean frequency/day</th>
<th>Mean frequency of peak time/h</th>
<th>Mean (SD) overall system response (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sodium</td>
<td>563.79</td>
<td>101.78</td>
<td>621.57 (835.66)</td>
</tr>
<tr>
<td>Potassium</td>
<td>568.50</td>
<td>101.90</td>
<td>295.99 (532.43)</td>
</tr>
<tr>
<td>Glucose</td>
<td>488.15</td>
<td>74.19</td>
<td>632.11 (727.21)</td>
</tr>
<tr>
<td>Hematocrit</td>
<td>687.25</td>
<td>101.07</td>
<td>428.81 (807.52)</td>
</tr>
<tr>
<td>Rh typing</td>
<td>114.69</td>
<td>25.05</td>
<td>762.46 (1244.59)</td>
</tr>
<tr>
<td>CBC blast</td>
<td>438.19</td>
<td>87.36</td>
<td>237.81 (490.03)</td>
</tr>
<tr>
<td>WBC</td>
<td>685.58</td>
<td>101.26</td>
<td>586.57 (791.57)</td>
</tr>
</tbody>
</table>

Notes: a,b,c,d Groups with significant differences at the \( p < 0.05 \) level of multiple comparisons.

CBC, complete blood count; WBC, white blood cell count.

Table 2. Response times by clinical setting: DIA.

<table>
<thead>
<tr>
<th>Clinical department</th>
<th>( N )</th>
<th>Mean (SD) response time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Overall</td>
<td>Engine</td>
</tr>
<tr>
<td>Outpatient</td>
<td>122,949</td>
<td>197.36 (25,146)</td>
</tr>
<tr>
<td>Inpatient</td>
<td>149,282</td>
<td>626.22 (853.22)</td>
</tr>
<tr>
<td>Emergency</td>
<td>51,214</td>
<td>193.98 (247.97)</td>
</tr>
</tbody>
</table>

Figure 6. Average response time of the DIA by alerts.
architecture, knowledge engine, and DIA – developed by the authors in previous works, with regard to system performance. The rules used in the laboratory alerting system were in a simpler form than those of other CDS categories, so it was easier to implement and test using these.

With regard to alert volume and distribution, the system was heavily stressed at a specific time period during the day. Considering the work processes at hospitals, it is natural that tests are concentrated in the morning when ambulatory settings are open and work is starting for the day. The number of alerts that a physician is expected to receive per day by department was highest for internal medicine, at 6.0 alerts. Considering that most rules are relevant to internal medicine, the high number may be responsible for the alert fatigue found in previous studies as a cause of system failure. For example, if a physician already knows that a patient’s hematocrit is low or is expected to be low, repeated alerts would not be meaningful, and may even be disruptive. Potential users recommend that the rules should be more sophisticated, based upon each patient’s specific clinical context, and not just depend on a data value itself.

As for accuracy, at the beginning of our evaluation we found incorrect cases due to concept mismatches in condition expressions and LIS data. However, the rules were simple and explicit, so it was easy to refine the mappings and achieve 100% accuracy, as assured by black-box validation as a post-hoc approach. The reason for incorrect results was not the knowledge execution error, but rather an error injected during knowledge acquisition.

System performance gave feasible results in around 400 ms. During the peak time, the mean frequency was about 25–100/h, and the overall system response took 240–760 ms. However, the response time was sometimes more than 1 min. These outcomes were not unexpected for notification of laboratory results considering that the users are normally not standing at a computer monitor waiting for the results. Compared with a messaging time of an average of 15 s by pager and 10 s by mobile phone (according to the study of Chen et al. 2002), a response time of 240–760 ms represents acceptable performance. However, the response time varied widely depending upon the test, with some of this variation being attributable to the performance of the DIA. The role of the DIA was to map the concepts in knowledge representation with relevant data items in an EMR or LIS using a standard terminology code system. Thus, the DIA must pass several mapping steps during run-time, which requires time delays. (Goldberg et al. 2006) assessed the performance of a commercial engine for a CDS service, and also found that the delivery of data is the major bottleneck in rule service performance. In order to minimize the number of round trips between a rule service and an external repository, the rule service should be primed with a large swath of patient data. It is necessary to further investigate methods for bulk data retrieval in order to assure a scalable infrastructure for an enterprise CDSS.

We analyzed the data in detail by medical department, clinical setting (inpatient, outpatient, and emergency room), months, days, and times to examine the effects of other variables, but no typical patterns or trends were uncovered. In contrast to the DIA, the performance of the knowledge engine was quite stable and consistent, at approximately 50 ms.

System accuracy, as measured by signal-to-noise ratio, easily reached 100% due to the use of simple and explicit rules and system architecture. The response times measured by the overall system time, engine throughput time, and DIA throughput time were also acceptable for the domain of laboratory alerts; however, we detected opportunities to improve the DIA throughput time for other applications of the CDSS.

Acknowledgments

This study was supported by a grant of the Korea Healthcare Technology R&D Project, Ministry for Health, Welfare and Family Affairs, Republic of Korea (no. A050909).

References


