MEDINFO 2015: eHealth-enabled Health
I.N. Sarkar et al. (Eds.)

doi:10.3233/978-1-61499-564-7-270

Use of a Proven Framework for Computer Decision Support within the Intermountain Healthcare Network

R. Scott Evans[ab], James F. Lloyd[a], Kyle V. Johnson[a], Stephen Howe[a], Jacob S. Tripp[a]

[a]*Department of Medical Informatics, Intermountain Healthcare*
[b]*Department of Biomedical Informatics, University of Utah*

Abstract

Hospitalized patients in the U.S. do not always receive optimal care. In light of this, Computerized Decision Support (CDS) has been recommended to for the improvement of patient care. A number of methodologies, standards, and frameworks have been developed to facilitate the development and interoperability of computerized clinical guidelines and CDS logic. In addition, Health Information Exchange using Service-Oriented Architecture holds some promise to help realize that goal. We have used a framework at Intermountain Healthcare that employs familiar programming languages and technology to develop over 40 CDS applications during the past 13 years, which clinicians are dependent on each day. This paper describes the framework, technology, and CDS application development methods, while providing three distinct examples of applications that illustrate the need and use of the framework for patient care improvement. The main limitation of this framework is its dependence on point-to-point interfaces to access patient data. We look forward to the use of validated and accessible Service-Oriented Architecture to facilitate patient data access across diverse databases.

Keywords:

Computerized Decision Support; CDS; framework; Service-Oriented Architecture; SOA.

Introduction

Hospitalized patients expect to receive the best possible care. However, of 25 clinical conditions examined, patients in the U.S. only received the recommended processes for basic care an average of 54.9% of the time [1]. Use of information technology has been recommended to help improve patient care [2, 3]. Computerized Decision Support (CDS) has been defined as "any computer program designed to help health professionals make clinical decisions" [4]. Airline pilots cannot look out the cockpit window and determine their exact altitude, air speed and flight direction. On-board computer programs provide that information. Likewise, if pilots try to land without the landing gear down and locked, or numerous other potential problems are detected during flight, the on-board computers generate alerts notifying the pilots of the situation and danger. In a similar manner, CDS has been used to provide real-time information, reminders and alerts to healthcare providers over the past 40 years, and has been shown to improve patient care [5, 6]. For more than 20 years, a number of methodologies, standards, intelligent systems, syntaxes and frameworks have been developed to facilitate the development and interoperability of computerized clinical guidelines and CDS logic. Protégé, GLIF, Arden Syntax, Prodigy, SEBASTIAN, GELLO and SAGE have been successful in demonstrating the reconstruction of previous CDS logic and the use of standards and ontologies to facilitate CDS and computerized guideline interoperability [7-14]. Health Information Exchange (HIE), using a Service-Oriented Architecture (SOA), provides some additional capacity to realize the goal of data independence [15-18]. Recently, the Substitutable Medical Applications, Reusable Technologies (SMART) project has recommended the development of substitutable applications built around core electronic medical records (EMR) [19]. To date, using the Fast Healthcare Interoperability Resources (FHIR) data format standards for web-based application programming interfaces (API), SMART looks like the best candidate for API development [19]. However, HIE efforts are not ready to be used and historically, committee acceptance has been difficult to obtain [20]. For over 13 years, we have used a framework at Intermountain Healthcare that uses familiar programming languages and technology to develop over 40 CDS applications, which clinicians depend on each day. This paper describes the framework, technology and CDS application development methods and provides three different examples of applications that illustrate the need for, and how we use the framework to improve patient care.

Materials and Methods

Background

Intermountain Healthcare (IH) spans the U.S. states of Utah and Idaho, and is comprised of 22 hospitals and 197 clinics, in addition to urgent care centers, physician offices and home health.. A key feature of our independently developed hospital information system (HELP) is the integrated EMR that contains most clinical information. The coded data in the EMR facilitates the development and use of CDS applications to analyze patient data and constantly monitor patient care. Natural Language Processing (NLP) is also used to analyze non-coded dictated reports.

In 1995, development on the HELP^2 system was started and ambulatory data in addition to hospital data began to be entered into the EMR. HELP^2 was expected to replace HELP before 2005. Moreover, during the past 20 years, IH has partnered with four different EMR vendors to develop a new information system, which would replace HELP and HELP^2. The HELP system and two vendor partners' EMRs were built on a hierarchical data dictionary while HELP^2 and the other two vendor partners' EMRs were built on relational databases and different data dictionary coding. Thus, for over 15 years,

the needs of IH clinicians for new and updated CDS applications continued to grow while we were not sure which EMR or data dictionary would eventually be used at IH nor which clinical guideline, medical logic development or HIE would become the standard.

Framework Description

Currently, the EMR at IH contains data from HELP and HELP^2 (Figure 1). Eventually, all the data in the EMR will only be provided by Cerner, the current partner, as HELP and HELP^2 are replaced at all IH hospitals and clinics. Each night, all clinical and business data in the EMR are archived in the enterprise data warehouse (EDW). After the EDW has been updated, Extract, Transform and Load (ETL) jobs are activated by the time driver and use SQL to extract and populate numerous Oracle tables on the staging server. The staging server is maintained 24/7 and backed up each night. The staging server provides faster data access as opposed to collecting that data from the large, slower and non-24/7 EDW. The data on the staging server are stored in data marts oriented by specific clinical needs of decision support applications. The "time driver" (Table 1) is also used to activate over 40 Java based CDS applications that run from once a month to every 5 seconds. The enterprise encounter table contains the master list of all inpatient and outpatient encounters along with some demographic data. The first thing most of the CDS applications do is look in the enterprise encounter file for patients that match specific requirements specified in the logic. The logic in each of the CDS applications also has a specific list of needed patient data elements within specific time windows. An application may require a variety of different patient data elements that could have been stored in any inpatient or outpatient facility from a number of years back in time or just during the current encounter. Historical data from other facilities is retrieved from the tables on the staging server, data for monthly reports are retrieved from the EDW, and current encounter data are retrieved from the EMR. A "data driver" monitors the EMR and is activated by data entry to the system. The data driver then runs specific medical logic modules based on the data being stored. If the medical logic module detects an action that needs to be made, an alert is stored in the Alert File on the IH network. Additionally, there exist data types not stored in the EMR. Electrocardiograms, echocardiograms, pulmonary function tests and data from the IH Health Plans are stored in different databases but also on the IH network.

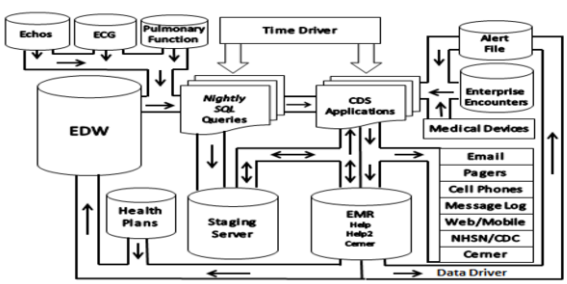

Figure 1 - Framework used for clinical decision support applications at Intermountain Healthcare. NHSN-National Healthcare Safety Network, CDC-Center for Diseases Control and Prevention.

Medical Devices Interface

Some CDS applications also need access to data from medical devices on the framework such as ventilators, infusion pumps, bedside and portable monitors, etc. Collection of data from medical devices and stored in tables on the staging server or the EMR is straightforward if the device has an interface that allows data to be transmitted to an external device. That interface can be analog or digital. If it is analog, the data must be converted from an analog signal to a digital interpretation of the signal. Many common analog-to-digital (A/D) converters are available to perform that function. If the data comes from the medical device in digital format, it may be in a number of different formats, e.g., binary or ASCII. Binary data are generally decoded to form some kind of human understandable format so that they can be stored as coded data. If the data are in ASCII, it may follow several standard protocols, e.g. HL7, XML, or it may be in a manufacturer's proprietary format, which may also need to be decoded to extract the desired components from the data. Hardware interfaces on the manufacturer's devices range from 2 wires for an analog output, to RS232 interfaces for simple serial data transmission and wired Ethernet connections, or a wireless interface including 802.11 (WiFi) or 802.15.4 (Zigbee).

Through the use of this framework, the logic in the CDS applications has access to all data within the IH network and not just a single hospital's EMR. Some data stored in the staging server are not needed until a patient is re-admitted or scheduled for a clinic or home health visit the next day. Actions from the CDS applications can be stored in the EMR or include alerts that can be sent to email addresses, pagers/cell phones, the Message Log within the HELP^2 EMR, Web and mobile devices, the National Healthcare Safety Network at the Centers for Diseases Control and Prevention, or through an API or service to a vendor. Data in the EMR from HELP^2 is accessed through Oracle SQL queries imbedded in Java code while HELP data are extracted using a client/server system of components developed in-house.

CDS Application Development

The CDS applications we have developed on the framework range from fairly simple alerts such as sending primary care physicians daily alerts in the HELP^2 "Message Log" stating that ECG results are finalized to a complex web-based version of the Antibiotic Assistant program [21]. We have found Java to work well and provide us with all the methods we need for simple to complex CDS logic (Table 1). We use Eclipse as the integrated development environment (IDE) for the CDS application development (Eclipse Foundation, Inc., Ottawa, Ontario, Canada). In addition to being free and open source, it provides all the development and testing tools we need; source code editor, intelligent code completion, build automation, documentation generator and especially debugging tools. For complex applications, we need a symbolic debugger that provides the ability to step through code, set break points, break when variables are changed, modify variables and rerun the program from specific lines of code. SQL embedded within the Java code is used to extract the needed data elements from relational databases through Java database connectivity. Java classes also handle the HELP data extraction interface or point-to-point interfaces for the electrocardiogram, echocardiogram and pulmonary function databases and medical devices.

We use JBoss middleware to write to queues and buffer data sent to the EMR. While we use in-house developed services to display alerts to our HELP^2 Message Log, we use common methods to provide most alerts and reports to the users. Simple mail transfer protocol (SMTP) is used to send alerts to

pagers and cell phones and lengthy text reports are sent via email. Reports are encoded using XML to send monthly antibiotic use data to the Center for Diseases Control and Prevention. JavaScript is also used to display CDS derived output on the web.

Table 1 – List of CDS tasks and common methods used within the framework.

CDS Task	Common Method
Core programming language	Java
Integrated development environment: source code editor, intelligent code completion, build automation, documentation generator, symbolic debugger	Eclipse
Time driver	Windows Scheduler
Relational data access	SQL embedded within Java and Java Database Connectivity
Non-relational data access	Point-to-point developed Java classes
Messages within the HELP[2] EMR	In-house developed Java classes
Pager, cell phone and email alerts	Simple Mail Transfer Protocol
Medical device access and data storage	RS232 interfaces, analog-to-digital converters, HL7, XML, wifi, JBoss
Antimicrobial use reports to Center for Diseases and Control	XML
Web-based report generation	JavaScript and HTML within Java

Application Examples

Ventilator disconnection: In 2004, we developed a system to monitor critical ventilator events by using the framework [22]. We collect data from mechanical ventilators every 5 seconds and look for evidence of a disconnection. Whenever an event is identified, the system takes control of every computer in the patient's intensive care unit and generates an enhanced audio and visual alert indicating there is a critical ventilator event and identifies the room number. Once the alert is acknowledged or the event is corrected, all the computers are restored back to the pre-alert status and/or application state. That CDS application was first installed in the Shock/trauma intensive care unit (ICU) at LDS Hospital. Today, the ventilator disconnection alerts are installed in 13 ICUs at 6 different IH hospitals. The audio and video alerts are so distinct and annoying that all medical personnel quickly respond to get the alerts turned off. Data from the past 10 years show that the present critical ventilator alarm times are below durations that are likely to be dangerous. In addition, now that data is collected on all ventilator disconnections, our respiratory therapy departments have been able to identify risk factors such as where ventilators are placed in relation to heat and air conditioning vents which can cause water accumulation in the tubing and restrict airflow. This CDS example illustrates the need for data access with the encounter data, medical devices, EMR, the time driver and use of an IDE for complex code development.

Venous thromboembolism (VTE) high-risk alerts: Routine use of evidence-based guidelines for VTE prophylaxis is uncommon due to the difficulty in integrating them into routine patient care**.** We developed a CDS tool which captures sufficient data to use a risk prediction model to identify patients at high risk for VTE [23]. This CDS application illustrates the need for encounter, historical and current patient data from throughout the enterprise. Patients are screened each day of hospital admission at all 22 IH hospitals and their risk of VTE is determined. Pharmacists are sent an email message listing each high-risk patient and which risk factors they have. Patients at high-risk are then checked to see if they are on appropriate VTE chemoprophylaxis including the drug, dosage, route and interval. If not, 65 hospitalists get a page if their patient is identified. Patient age, new surgery, bed-rest, cancer, height and weight information are collected from the EMR. Previous cancer, previous VTE, previous surgery within 90 days, hypercoagulability and hormone replacement and oral contraception use are collected each night from the EDW and stored in the staging server. The time driver activates the application at 7am each day. We found this computerized tool to identify patients at high risk for VTE with a sensitivity of 98% and positive predictive value of 99%. As of January 2014, high and low risk alerts are sent for every patient to the nurse-charting program in the HELP EMR to meet the new Center for Medicare and Medicaid Service mandate in the U.S.

Antibiotic Assistant: The most complex CDS application we have developed is the antibiotic assistant [21]. Most of the data are accessed from the EMR except for previous microbiology, laboratory, radiology and pathology data from other hospitals during the previous 6 weeks. The application displays all the information physicians need to be aware of before they order antimicrobial agents or are checking the infection status of their patients. The most complex part is the suggestion of which antimicrobial(s) along with the dosage, route, interval and duration to use. The application was originally developed using Tandem Application Language and a symbolic debugger on HELP. With the eventual sunsetting of HELP, we were able to duplicate the application using Java and JavaScript so it can run on HELP[2] or any other system that can render web-based applications such as most vendor EMRs. Based on the complexity of the logic and number of interdependent Java classes in this application, it is used to illustrate the need for a symbolic debugger. It would not have been possible to develop, maintain or add new features to this CDS application without the functionality provided by a symbolic debugger - an ability provided by both Java and Eclipse.

Discussion

The CDS framework and application development methods described in this paper have served us well for the past 13 years, and have allowed us to meet the diverse information needs of many different types of clinicians and medical domains. The majority of CDS applications that we have developed are a) time-driven, b) run in the background, and c) send alerts or email reports at programed intervals in time designated by clinician workflow and patient care. These applications are well aligned with the four predictors of improved clinical practice: 1) automatic provision of decision support as part of clinician workflow, 2) provision of recommendations rather than just assessments, 3) provision of decision support at the time and location of decision making, and 4) computer based decision support [6].

At present, there is no widely accepted, standard approach for representing CDS intervention types and their structural components including alerts and reminders, order sets, infobuttons, documentation templates/forms, and relevant data presentation [24]. Many gaps and challenges remain, including the complexity of many standards, limited availability of accessible knowledge resources, and lack of tooling and other resources to enable the adoption of existing standards [25]. Moreover, many question the "outsourcing" of CDS to potentially imperfect external agencies [26].

While system-agnostic CDS development tools hold great potential to facilitate CDS implementation, little has been described regarding their benefits and challenges (26). A benefit of using Java and SQL is that they are common programming languages taught within most computer science departments. Thus, no other special knowledge or training is required to learn how to use a new CDS development tool and syntax by the programmers. Our experience is that most clinicians are extremely busy providing and looking for ways to improve patient care and don't have the time for or an interest in learning how to program CDS applications. Also, use of Java classes and libraries provide an easy way to share and reuse the same developed and tested logic from one application to another. While not quite true, the Java write once, run anywhere java virtual machine does facilitate the transfer of CDS applications to many different platforms. We do not imply that this framework ought to replace others, nor do we predict that it will outperform any of the other CDS development methods available. Rather, we contribute this framework as a viable option for other systems, given its high functionality in application development and reduction of testing time at IH.
Use of this framework has taught us that the ability of CDS to improve patient care depends on reliable access to the needed patient data. Often, those data need to be derived in real time from the EMR or medical devices, and not the EDW. We are fortunate to have access to most patient data using this framework we have built at IH. Other institutions can use these same programming and development methods we describe to create new CDS applications using the data to which they have access.

The main constraint of our framework is its limiting accessibility to EHR patient data, by way of medical device and database point-to-point interface. Accessing patient data across multiple EHRs has been a desired but largely unattained aim of clinical informatics, especially in commercial EHR systems (24). A potential opportunity for enabling CDS across multiple EHRs is to leverage vendor-supported SOA development and APIs. A recent report by JASON, an independent group of scientists who advise the U.S. federal government on technology issues, states that to achieve the goal of improving health outcomes, Stage 3 Meaningful Use requirements should be defined such that they enable the creation of an entrepreneurial space across the entire health data enterprise and suggest that EHR software vendors should be required to develop and publish APIs for EMRs for third-party software developers (http://healthit.gov/sites/default/files/ptp-13700hhs_white.pdf).

SMART's approach is to use FHIR, a HL7 draft standard describing data formats and elements and an API, to help drive down healthcare costs, support standards, accommodate differences in workflow, and especially foster competition in the market to help promote "innovation" in healthcare [19]. To date, we have used services provided by Cerner to access data for two of the clinical applications that run on our framework. Cerner's upper management has publically stated they support the need for vendors to provide SOA and especially use SMART on FHIR. If so, when a SOA is finalized and in use, we would be able to replace the current point-to-point database specific services and continue to be innovative without the need to make any changes to our current framework, technology or CDS logic.

Conclusion

We have used the described framework, which has employed familiar programming languages and technology in the development over 40 clinician dependent CDS applications during the past 13 years. The main limitation of this framework is the dependence on point-to-point interfaces to access patient data. We look forward to when we can use validated and accessible SOA to facilitate patient data access across diverse databases.

References

[1] McGlynn EA, Asch SM, Adams J, Keesey J, Hicks J, DeCristofaro A, et al. The quality of health care delivered to adults in the United States. The New England journal of medicine. 2003 Jun 26;348(26):2635-45.

[2] Medicine Io. Crossing the quality Chasm: a new health system for the 21st century. National Academy Press. 2001;Washington DC

[3] Executive Office of the President PsCoAoS, technology. a. Report to the President: Realizing the Full Potential of Health Information Technology to Improve Healthcare for Americans: the Path Forward. 2010:http://www.whitehouse.gov/sites/default/files/microsites/ostp/pcast-health-itreportpdf.

[4] Shortliffe EH. Computer programs to support clinical decision making. Jama. 1987 Jul 3;258(1):61-6.

[5] Hunt DL, Haynes RB, Hanna SE, Smith K. Effects of computer-based clinical decision support systems on physician performance and patient outcomes: a systematic review. Jama. 1998 Oct 21;280(15):1339-46.

[6] Kawamoto K, Houlihan CA, Balas EA, Lobach DF. Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success. Bmj. 2005 Apr 2;330(7494):765.

[7] Musen MA, Gennari JH, Wong WW. A rational reconstruction of INTERNIST-I using PROTEGE-II. Proceedings / the Annual Symposium on Computer Application [sic] in Medical Care Symposium on Computer Applications in Medical Care. 1995:289-93.

[8] Peleg M, Boxwala AA, Bernstam E, Tu S, Greenes RA, Shortliffe EH. Sharable representation of clinical guidelines in GLIF: relationship to the Arden Syntax. Journal of biomedical informatics. 2001 Jun;34(3):170-81.

[9] Pryor TA, Hripcsak G. The Arden syntax for medical logic modules. International journal of clinical monitoring and computing. 1993 Nov;10(4):215-24.

[10] Johnson PD, Tu S, Booth N, Sugden B, Purves IN. Using scenarios in chronic disease management guidelines for primary care. Proceedings / AMIA Annual Symposium AMIA Symposium. 2000:389-93.

[11] Borbolla D, Otero C, Lobach DF, Kawamoto K, Gomez Saldano AM, Staccia G, et al. Implementation of a clinical decision support system using a service model: results of a feasibility study. Studies in health technology and informatics. 2010;160(Pt 2):816-20.

[12] Jenders RA, Del Fiol G, Kawamoto K, Sailors RM. Standards in clinical decision support: activities in health level seven. AMIA Annual Symposium proceedings / AMIA Symposium AMIA Symposium. 2008:1244-5.

[13] Park JY, Musen MA. VM-in-Protege: a study of software reuse. Studies in health technology and informatics. 1998;52 Pt 1:644-8.

[14] Tu SW, Musen MA, Shankar R, Campbell J, Hrabak K, McClay J, et al. Modeling guidelines for integration into clinical workflow. Studies in health technology and informatics. 2004;107(Pt 1):174-8.

[15] Loya SR, Kawamoto K, Chatwin C, Huser V. Service oriented architecture for clinical decision support: a systematic review and future directions. Journal of medical systems. 2014 Dec;38(12):140.

[16] Kawamoto K, Jacobs J, Welch BM, Huser V, Paterno MD, Del Fiol G, et al. Clinical information system services and capabilities desired for scalable, standards-based, service-oriented decision support: consensus assessment of the Health Level 7 clinical decision support Work Group. AMIA Annual Symposium proceedings / AMIA Symposium AMIA Symposium. 2012;2012:446-55.

[17] Kawamoto K, Hongsermeier T, Wright A, Lewis J, Bell DS, Middleton B. Key principles for a national clinical decision support knowledge sharing framework: synthesis of insights from leading subject matter experts. Journal of the American Medical Informatics Association : JAMIA. 2013 Jan 1;20(1):199-207.

[18] Kawamoto K, Lobach DF. Proposal for fulfilling strategic objectives of the U.S. Roadmap for national action on clinical decision support through a service-oriented architecture leveraging HL7 services. Journal of the American Medical Informatics Association : JAMIA. 2007 Mar-Apr;14(2):146-55.

[19] Mandl KD, Mandel JC, Murphy SN, Bernstam EV, Ramoni RL, Kreda DA, et al. The SMART Platform: early experience enabling substitutable applications for electronic health records. Journal of the American Medical Informatics Association : JAMIA. 2012 Jul-Aug;19(4):597-603.

[20] Mandl KD, Kohane IS. No small change for the health information economy. The New England journal of medicine. 2009 Mar 26;360(13):1278-81.

[21] Evans RS, Pestotnik SL, Classen DC, Clemmer TP, Weaver LK, Orme JF, Jr., et al. A computer-assisted management program for antibiotics and other antiinfective agents. The New England journal of medicine. 1998 Jan 22;338(4):232-8.

[22] Evans RS, Johnson KV, Flint VB, Kinder T, Lyon CR, Hawley WL, et al. Enhanced notification of critical ventilator events. Journal of the American Medical Informatics Association : JAMIA. 2005 Nov-Dec;12(6):589-95.

[23] Evans RS, Lloyd JF, Aston VT, Woller SC, Tripp JS, Elliott CG, et al. Computer surveillance of patients at high risk for and with venous thromboembolism. AMIA Annual Symposium proceedings / AMIA Symposium AMIA Symposium. 2010;2010:217-21.

[24] Zhou L, Hongsermeier T, Boxwala A, Lewis J, Kawamoto K, Maviglia S, et al. Structured representation for core elements of common clinical decision support interventions to facilitate knowledge sharing. Studies in health technology and informatics. 2013;192:195-9.

[25] Kawamoto K, Del Fiol G, Lobach DF, Jenders RA. Standards for scalable clinical decision support: need, current and emerging standards, gaps, and proposal for progress. The open medical informatics journal. 2010;4:235-44.

[26] Nadkarni PM, Miller RA. Service-oriented architecture in medical software: promises and perils. Journal of the American Medical Informatics Association : JAMIA. 2007 Mar-Apr;14(2):244-6.

Address for correspondence
R. Scott Evans, M.S., Ph.D., FACMI
Dept. Medical Informatics, LDS Hospital
8th Avenue & C Street
Salt Lake City, Utah, 84143 USA.
rscott.evans@imail.org (801) 408-3029