A Modular Framework for Clinical Decision Support Systems: Medical Device Plug-and-Play is Critical

Williams M, Wu F, Kazanzides P, Brady K, Fackler J

Johns Hopkins University, Baltimore MD 21218

Abstract

This paper describes the design and initial implementation of a modular framework for Clinical Decision Support Systems and highlights the need for medical device plug-and-play standards. The software handles the tasks of data acquisition and validation, visualization, and treatment management in order to enable the development of protocol guideline modules as "plug-ins" to the framework. The system utilizes an asynchronous data-driven design to support real-time information flow and user interaction. All components of the framework are modular and easily extendible, allowing for new data sources, visualization methods, and protocols to be inserted. The system is configured by assigning each protocol a manager that handles decision communication with the rest of the framework. A set of classes has been created to allow communication between the different modules along with persistence of all data, decisions, and treatments to a database. The initial prototype is a Clinical Decision Support System focusing on the treatment of Traumatic Brain Injury.

Keywords

Clinical Decision Support System; modular framework; visualization; Traumatic Brain Injury; plug-and-play

Background

Clinical Decision Support Systems (CDSS) are fifty years old (Ledley and Lusted 1959). Examples of CDSS range from the laminated pocket cards that outline cardiopulmonary resuscitation guidelines but more typically are paper or computer-based order-sets as well as reminder systems (e.g, a reminder that gentamicin doses be trimmed in a patient with a rising creatinine, or activated protein C be considered based on a multi-factorial score).

Relevant to medical devices and requirements for device interactivity, a specific form of CDSS, closed-loop control of medical devices, has an even older history. Almost 60 years ago, a paper was published describing the “robot anesthetist” describing earlier work using the electroencephalogram driving either ether or sodium thiopental (HAWARD 1952), (MAYO, BICKFORD and FAULCONER 1950). A comprehensive literature review of closed-loop anesthesia was published in 1992 (O'Hara, Bogen and Noordergraaf 1992) and literally hundreds of articles have been since published (particularly in the domain of mechanical ventilation (Tehrani and Roum 2008).

Two meta-analyses of CDSS recently reported remarkably similar findings. Garg et al searched multiple databases through September 2004 and found 100 studies of decision support systems meeting their rigorous entry criteria. Fifty-two of those studies measured one or more patient outcomes. Studies were evaluated for methodological quality and for study characteristics that predicted success (defined as at least a 50% improvement in the measured outcome) (Garg, et al. 2005). Only seven of 52 studies that attempted to examine patient outcomes showed an impact of CDSS. Overall, two decision support system characteristics were associated with an impact. First, studies more often showed an effect when the authors of the CDSS were authors of the report. The only extensible finding of this meta-analysis was decision support systems that automatically prompt users (compared to systems requiring user activation) were associated with improved outcomes. The second meta-analysis examined 70 studies for 22 CDSS features the authors believed important for successfully improving provider behavior (Kawamoto, et al. 2005). A multiple logistic regression analysis identified four of these features as independent predictors of improved provider behavior: 1) automatic provision of decision support as part of clinician workflow, 2) provision of recommendations rather than just assessment, 3) provision of decision support at the time and location of decision making, 4) and computer based decision support.

A major impediment to the extensibility of CDSS (beyond their largely unproved clinical efficacy (Garg, et al. 2005)), is the lack of widely implemented data standards. Even efforts to share CDSS between sites running the same major health care IT company's system can not simply "import" a rule set. In the domain of medical devices, CDSS require hard-coded connections between input signals and output controls.

The goal of the project described is to create a modular framework for a Clinical Decision Support System to facilitate the development of guideline protocol "plug-ins". Instead of focusing on creating a new method for defining guidelines or representing clinical knowledge such as in (Achour, et al. 2001), we address the issues of data acquisition and validation, visualization, and treatment management. The basic concept is to develop a software framework that allows users to concentrate on implementing guidelines and protocols, with our system handling the data, management, and display aspects. The initial prototype is focused on supporting the treatment of Traumatic Brain Injury (TBI) based on internationally agreed algorithms (Brain Trauma Foundation, American Association of Neurological Surgeons and Congress of Neurological Surgeons 2007) as a sample application of the framework. As much of the data inputs are necessarily from medical devices (e.g. physiological monitors) and in its future closed-loop form the therapeutic devices are also medical devices, this project presents a complex use-case for identification of medical device plug-and-play efforts.

One of the key components of the software framework is the acquisition and fusion of medical data from multiple sources. These sources include bedside monitoring devices, hospital electronic medical record databases, and manually entered information from clinical staff. In the development and testing of the TBI CDSS software, the data is acquired from the physiological monitors (e.g. heart rate and blood pressure), stand-alone monitors (e.g. oxygen saturation and end-tidal carbon dioxide), and therapeutic devices (e.g. ventilators and infusion pumps). Given the current lack of device data standards, we were forced to acquire real-time (5 second interval) data only from the devices on the monitoring network and it involved writing a module to continuously poll a set of text files generated by a third party solution (BedMaster, Excel Medical Electronics, Jupiter, FL) and then extract and reformat this data, finally sending it to the TBI

CDSS framework. If plug-and-play standards were implemented, such a convoluted (and therefore risky) set of work-arounds would not be necessary. The current lack of a universal interface for medical device I/O poses a significant difficulty to the continued development and use of our software framework. Every time a new device needs to be supported, a module designed for communication based on that device's protocol and standards must be written. This roadblock would be removed with the implementation of a common communication interface for all medical monitoring devices.

Besides monitoring device connectivity issues, access to lab results and the patient's medical record also pose an issue. Currently, the electronic medical record module in the framework must continuously query the hospital's databases for updated information. Each piece of information requested requires a separate query tailored specifically for the hospital's database. This makes deployment to medical centers using a different electronic medical record system difficult, as a new module specific to their implementation must be written. A common standard for electronic medical records, lab results, and methods to access this data would greatly enhance the range of environments into which our software could be deployed.

System Overview

The framework utilizes a modular design to enable expansion and scalability. Each module has an interface that defines which methods it supports. In addition, there are a number of classes that handle communication between the different modules (these are explained in the sections where they are relevant, and are loosely based on those used by the PROforma system (Sutton and Fox 2003)). The framework can be viewed as three different areas: data collection and processing, protocol and treatment management, and visualization (see Figure 1). The communication between these three areas could easily be extended to include wireless or ad-hoc network capabilities. Data collection and routing consists of DataSource and Validator modules, ProtocolManagers handle interaction with the Protocol modules that perform the medical data analysis and decision support, and the ProtocolDisplay and FeedbackManager modules interact with the user. As previously mentioned, the framework is designed to make any data necessary available to the guideline protocol modules, and then display the decisions of these protocol modules through a graphical user interface that the user can interact with to provide feedback to the system.

We chose an asynchronous data-driven design to support the real-time goals of our requirements, similar to the approach taken in (Van Den Bossche, et al. 2008). This allows each piece of physiological data or user input to be handled and received by the protocol modules as soon as it is available, along with allowing the recommendations of the protocols to be available to the user as frequently as possible, which is a key component of CDSS design. The framework is event based, following a push methodology (as opposed to a pull) where data is pushed to the protocol modules, which in turn push their decision outputs back to the framework. This choice was made in an attempt to remove any timing or thread requirements from the protocol modules in an effort to simplify them.

Therefore, to add a protocol module to the framework, it only has to be configured to listen for data using the required interfaces, and then to output its decisions in the specified format, as the framework handles timing and initialization. The software is written in Java. The following subsections discuss the different modules and communications mechanisms that the framework is built upon.

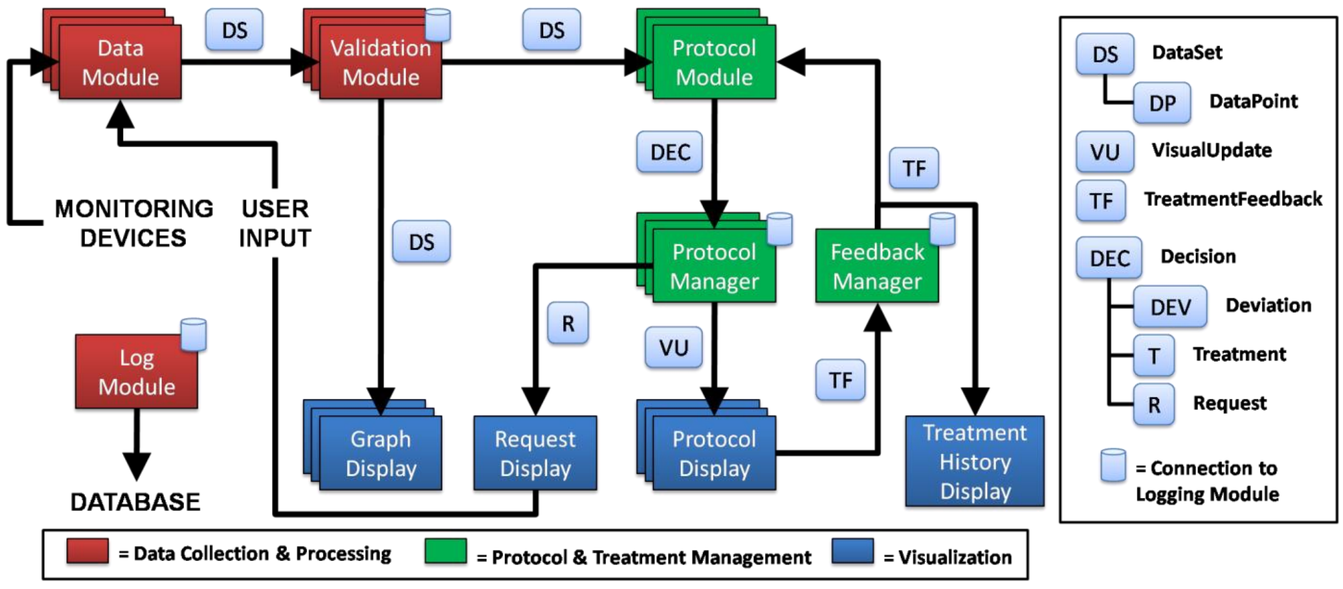


Figure 1: System Block Diagram

Data Sources & Validation

These modules interact with data sources, such as medical monitoring devices, patient record databases, and the clinical users of the system and transform the acquired data into formats that are recognizable by the rest of the framework. These classes are defined by the DataModule interface. DataModule objects generate DataSet objects, which are containers for the general DataPoint objects that hold different types of physiological data. The DataSet class is the primary mechanism for delivering information to the Protocol Modules, and it contains methods that allow its consumer to extract the different DataPoints that it contains. Each DataPoint consists of a source tag, a timestamp, the name of the data value it contains, and the actual data value itself. The relationship between DataSet and DataPoint is one-to-many (i.e., one DataSet can contain multiple DataPoints). Each DataModule can be configured to optionally forward the DataSet it generates to a Validator, which checks that the data it receives is within pre-configured boundaries to help prevent erroneous data from disrupting the rest of the system. The Validator then forwards this error-checked data to other modules in the framework.

Protocol Managers

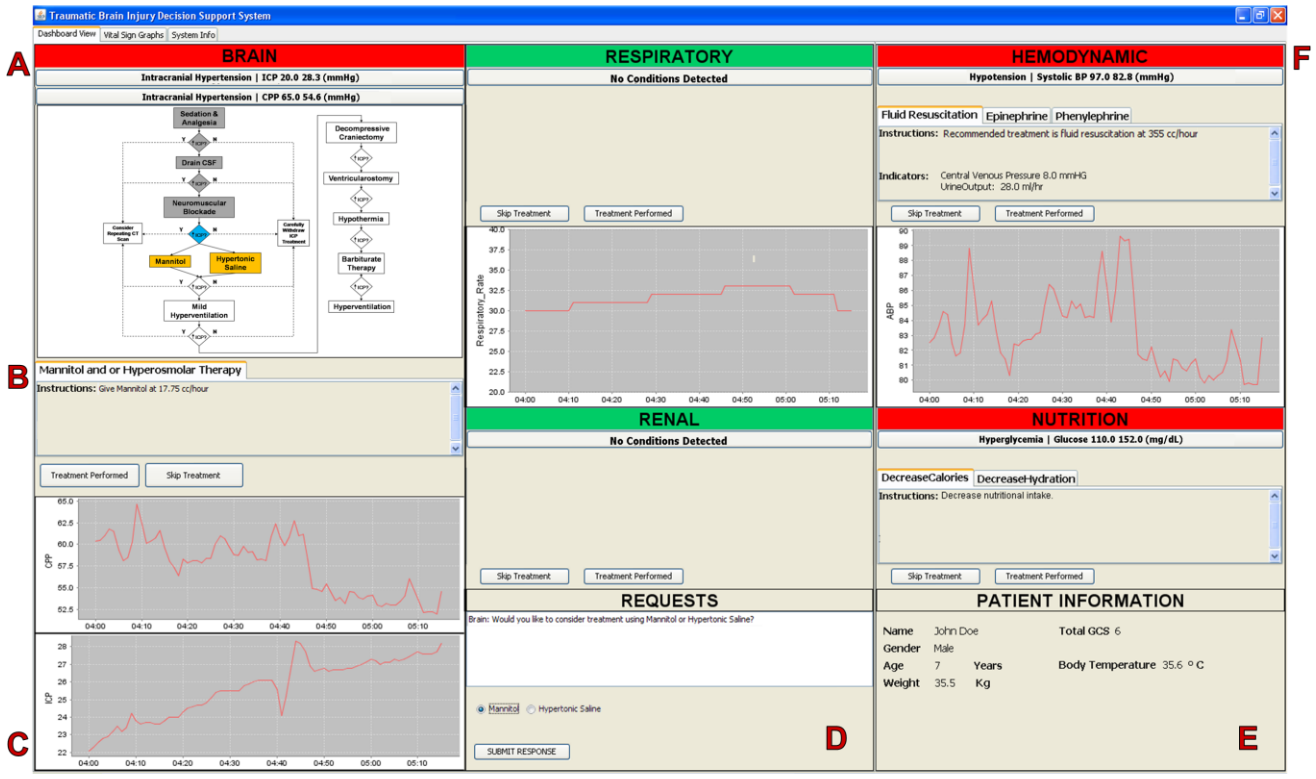
Each ProtocolModule forwards the Decision objects it generates to its assigned ProtocolManager. These managers are responsible for handling the Decision and then forwarding its contents to the rest of the framework, allowing there to be a single point of communication between a Protocol Module and the rest of the framework. Each Decision object contains Deviations, Conditions, Treatments, and Requests. A Deviation is generated by a protocol when a physiological variable is outside of the guideline range (e.g. Intracranial Pressure > 20 mmHg). The Deviation contains the name of the physiological variable, the actual value, the guideline value that was violated, and a timestamp. A Condition is the medical condition that is indicated by one or more Deviations (e.g. Intracranial Hypertension). A Treatment is the medical procedure that should be taken to alleviate a given Condition. Therefore, each Condition has one or more Deviations that triggered it, and one or more Treatments that will remedy it. The ProtocolManager internally keeps track of what Conditions, Deviations, and Treatments are currently suggested, along with any pending Requests. The Request class is used by a ProtocolModule to request any additional information that cannot be determined from the currently available data. A Request contains a source tag, a timestamp, the question to present to the user, a list of response options, and the name of the variable to fill with the user's response. An example of a Request would be "Does the patient have a skull fracture?" with responses being either yes or no. If the response list is left empty, then a text field is provided for the user. The ProtocolManager forwards any active Requests to the RequestDisplay, and a VisualUpdate is generated to tell the corresponding ProtocolDisplay module what to update. The VisualUpdate class contains lists of which Conditions, Deviations, and Treatments to add and/or remove from the ProtocolDisplay (making it easier to develop displays).

Visual Modules

There are several different types of visual modules that allow the user and the system to interact. The first is the ProtocolDisplay, which provides a visual representation of the Deviations, Conditions, and Treatments that a Protocol generates, along with allowing the user to input their feedback about the suggested Treatments, specifically whether they were performed or not (see Figure 2). This feedback is sent in the form of a TreatmentFeedback object, which is handled by the FeedbackManager discussed in the next section. The default ProtocolDisplay can be extended to provide more in depth features, such as a flowchart representation of a specific protocol. Each ProtocolDisplay is tied to the corresponding ProtocolManager of the ProtocolModule that it represents. Another visual module is the TreatmentHistoryDisplay, which is a timeline that gives the user a quick and simple overview of all the Treatments that have been performed on the patient during this session. The third type of visual module is the GraphModule, which is a real time dynamic graph of one or more physiological vitals. (In the future we plan to incorporate guideline, raw, and average values into the graph also). The GraphModule provides the user a history of important physiological values to reference when viewing the suggested treatments. Finally, the RequestDisplay visual module presents any pending information requests to the user. The user provided data is treated as another DataSource.

Feedback Manager

The FeedbackManager collects and records all the treatment feedback from the user, such as if/when a treatment was performed. The ProtocolDisplay modules send TreatmentFeedback objects to the manager based on user input. The TreatmentFeedback class contains the Treatment in question, a timestamp, a tag indicating if the treatment was performed or skipped, and an optional explanation section the user can fill in. The FeedbackManager forwards these feedback results to the ProtocolModules and the TreatmentHistoryDisplay, along with keeping an internal record.



Intracranial Hypertension | ICP 20.0 28.3 (mmHg)

Intracranial Hypertension | CPP 65.0 54.6 (mmHg)

Instructions: Give Mannitol at 17.75 cc/hour

No Conditions Detected

Hypotension | Systolic BP 97.0 82.8 (mmHg)

Instructions: Recommended treatment is fluid resuscitation at 355 cc/hour

Indicators: Central Venous Pressure 8.0 mmHG
UrineOutput: 28.0 ml/hr

No Conditions Detected

Hyperglycemia | Glucose 110.0 152.0 (mg/dL)

Instructions: Decrease nutritional intake.

Brain: Would you like to consider treatment using Mannitol or Hypertonic Saline?

[✓ Mannitol] Hypertonic Saline

Name	John Doe	Total GCS	6
Gender	Male		
Age	7 Years	Body Temperature	35.6 °C
Weight	35.5 Kg		

Figure 2: Graphical User Interface
This figure highlights some of the visualization and user interaction features of the framework. The title bars for each protocol are green if no conditions are detected and red otherwise. The buttons directly below each title bar show currently detected conditions. (A) The custom Brain ProtocolDisplay including the treatment flowchart. (B) The currently suggested treatment based on the Brain protocol, along with buttons to allow the user to choose direct the treatment flow. (C) Vital sign graphs. (D) The RequestDisplay allowing the user to choose which treatment path to take. (E) General patient information display area. (F) The default ProtocolDisplay for the Hemodynamic protocol.

Logging Subsystem

The primary function of the LogModule is to persist all necessary values for record keeping purposes. Each ValidationModule uses the log to save any DataPoints that it determines are erroneous, and can be configured to save all DataPoints that pass through it in the event that the real time physiological data is not being simultaneously recorded on another system. All of the ProtocolManagers record every new Deviation, Condition, Treatment, and Request that they receive in a Decision. The FeedbackManager records all TreatmentFeedback objects it receives. The logging subsystem is implemented using Hibernate and a MySQL database. A future goal is to be able to replay the treatment history of a patient using this framework for both analytical and training purposes, i.e. “flight data recorder” concept.

Traumatic Brain Injury Application

Two additional DataModules were written besides the standard user input module: one to query data from the hospital’s Electronic Medical Record database, and one to interface with the BedMaster software that integrates output from multiple bedside monitoring devices. The BedMaster software provides real time physiological value monitoring by pulling data at five second intervals. The traumatic brain injury guidelines (Brain Trauma Foundation, American Association of Neurological Surgeons and Congress of Neurological Surgeons 2007) were adapted into five separate ProtocolModules, focused on the brain, hemodynamic, respiratory, renal, and nutrition aspects of a patient. The development and functioning of these protocol modules was previously presented in depth (Wu, et al. 2009). The base ProtocolDisplay module was extended to support the sequenced treatment pattern from the brain protocol, while the default display was used for the other four protocols. The screenshot in Figure 2 details some of the important elements of the visualization modules and how they interact with the user and the rest of the framework.

Conclusions and Future Work

This paper presented the initial progress on a framework for developing a Clinical Decision Support System, with an example application focused on treating Traumatic Brain Injury. The focus of this framework is to enable guideline protocols to be written as plug-ins that take advantage of the data, treatment, and visual management provided by the framework. The framework is built on a modular design that allows for easy expansion to include new data sources, protocol guidelines, and visual interfaces. Programmers perform the initial configuration of connecting DataSources to provide information to ProtocolModules, connecting ProtocolManagers to handle the decision outputs of each guideline protocol module, and connecting the available VisualDisplay modules to interact with the user. Future work includes continued testing on a larger variety of patient data and medical conditions, along with clinical trials with studies of patient outcomes. Another goal is the development of additional protocol modules and to expand into areas other than TBI.

However, without implementation of medical device plug-and-play standards, the ability for other investigators to replicate our efforts is impossible. As such, progress in the development of CDSS and deployment will be impossible. This project provides the community a robust and challenging use-case as these standards efforts continue.

Reference

Achour, S L, M Dojat, C Rieux, P Bierling, and E Lepage. "A UMLS-based knowledge acquisition tool for rule-based clinical decision support system development." *Journal of the American Medical Informatics Association.* 8, no. 4 (Jan 2001): 351-60.

Brain Trauma Foundation, American Association of Neurological Surgeons, and Congress of Neurological Surgeons. "Guidelines for the management of severe traumatic brain injury." *Journal of neurotrauma* 24 Suppl 1 (Dec 2007): S1-106.

Garg, Amit X, et al. "Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: a systematic review."*Journal of the American Medical Association* 293, no. 10 (Mar 2005): 1223-38.

Hayward, L R C. "The robot anaesthetist; an introduction to the automatic control of anaesthesia by means of an electro-encephalographic intermediary." *Medical world* 76, no. 23 (Aug 1952): 624-6.

Kawamoto, Kensaku, Caitlin A Houlihan, E Andrew Balas, and David F Lobach. "Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success." *BMJ (Clinical research ed)* 330, no. 7494 (Apr 2005): 765.

Ledley, R, and L Lusted. "The Use of Electronic Computers to Aid in Medical Diagnosis." *Proceedings of the IRE*, Dec 1959.

Mayo, C W, R G Bickford, and A Faulconer. "Electroencephalographically controlled anesthesia in abdominal surgery." *Journal of the American Medical Association* 144, no. 13 (Nov 1950): 1081-3.

O'Hara, D A, D K Bogen, and A Noordergraaf. "The use of computers for controlling the delivery of anesthesia." *Anesthesiology* 77, no. 3 (Aug 1992): 563-81.

Sutton, David R, and John Fox. "The syntax and semantics of the PROforma guideline modeling language." *Journal of the American Medical Informatics Association : JAMIA* 10, no. 5 (Jan 2003): 433-43.

Tehrani, Fleur T, and James H Roum. "Intelligent decision support systems for mechanical ventilation." *Artificial Intelligence in Medicine* 44, no. 3 (Nov 2008): 171-82.

Van Den Bossche, Bruno, Sofie Van Hoecke, Chris Danneels, Johan Decruyenaere, Bart Dhoedt, and Filip De Turck. "Design of a JAIN SLEE/ESB-based platform for routing medical data in the ICU." *Computer methods and programs in biomedicine* 91, no. 3 (Sep 2008): 265-77.

Wu, F, M, Kazanzides, P Williams, K Brady, and J Fackler. "A Modular Clinical Decision Support System: Clinical Prototype Extensible into Multiple Clinical Settings ." *Pervasive Health.* London 2009.