Journal of Clinical Monitoring and Computing (2008) 22:189–198
DOI: 10.1007/s10877-008-9124-1

A KNOWLEDGE AUTHORING TOOL FOR CLINICAL DECISION SUPPORT

Dustin Dunsmuir[1], Jeremy Daniels, BAsc, EIT[1], Christopher Brouse, BAsc[2], Simon Ford, MB ChB, MRCP, FRCA[1] and J. Mark Ansermino, MBBCh, MMed(Info), FRCPC[1,3]

Dunsmuir D, Daniels J, Brouse C, Ford S, Ansermino JM. A knowledge authoring tool for clinical decision support.
J Clin Monit Comput 2008; 22:189–198

ABSTRACT. Anesthesiologists in the operating room are unable to constantly monitor all data generated by physiological monitors. They are further distracted by clinical and educational tasks. An *expert system* would ideally provide assistance to the anesthesiologist in this data-rich environment. Clinical monitoring expert systems have not been widely adopted, as traditional methods of *knowledge encoding* require both expert medical and programming skills, making knowledge acquisition difficult. A software application was developed for use as a *knowledge authoring tool* for physiological monitoring. This application enables clinicians to create *knowledge rules* without the need of a *knowledge engineer* or programmer. These *rules* are designed to provide clinical diagnosis, explanations and treatment advice for optimal patient care to the clinician in real time. By intelligently combining data from physiological monitors and demographical data sources the expert system can use these *rules* to assist in monitoring the patient. The knowledge authoring process is simplified by limiting connective relationships between *rules*. The application is designed to allow open collaboration between communities of clinicians to build a library of *rules* for clinical use. This design provides clinicians with a system for parameter surveillance and expert advice with a transparent pathway of reasoning. A usability evaluation demonstrated that anesthesiologists can rapidly develop useful rules for use in a predefined clinical scenario.

KEY WORDS. Decision support, Expert system, Situation awareness, Usability, Anesthesia, Monitoring, Knowledge resources.

INTRODUCTION

In the operating room, the anesthesiologist is required to manage various responsibilities simultaneously: the patient, physiological monitors and audio alarms; patient fluid and drug administration, and student education. Although the anesthesiologist carefully prioritizes these tasks, they may still exceed the information processing capacity of even highly trained, focused clinicians [1]. Technology such as an expert system can convey more precise information about a patient and potentially improve vigilance, standardize clinical protocols, enhance situational awareness and reduce errors in anesthetic practice [2].

From the [1]Department of Anesthesiology, Pharmacology and Therapeutics, The University of British Columbia, Vancouver, Canada; [2]Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada; [3]Department of Pediatric Anesthesia, British Columbia Children's Hospital, 4480 Oak Street, Vancouver, Canada V6H 3V4.

Received 23 January 2008. Accepted for publication 23 April 2008.

Address correspondence to J. M. Ansermino, Department of Pediatric Anesthesia, British Columbia Children's Hospital, 4480 Oak Street, Vancouver, Canada V6H 3V4.
E-mail: anserminos@yahoo.ca

Computers have the capacity to monitor large volumes of diverse data rapidly while on average, humans are only able to monitor a maximum of seven different parameters at any given time [3]. Expert systems have the potential to improve clinician performance by accurately executing repetitive tasks, to which humans are ill-suited, such as physiological parameter analysis and surveillance. Additionally, expert systems can be used to standardize clinical guidelines, while providing memory aids to the clinician.

Expert systems have been developed to aid clinicians with their decision-making with respect to patient care in various fields of medicine. Unfortunately, few of these systems have been adopted into clinical practice. Reasons for this lack of adoption are numerous and have been recently reviewed [4]. For example, some expert systems are not based on the best available knowledge while others have been created with a focus on technology rather than their true purpose of aiding the decision-making process of the clinician.

Cognitive aids have recently been developed to cue physicians to recall previously learned information and to help clinicians adhere to established protocols [5]. For example, an aid developed by the Veterans Health Administration's National Centre for Patient Safety to help anesthesiologists manage rare, high-mortality adverse events, was reported to be helpful in both emergency and non-emergency situations [6]. How an aid is integrated in the anesthetic emergency response process is more important than how frequently the aid is used [7]. These cognitive aids would ideally be integrated electronically into the anesthesiologist's workflow and be triggered automatically when adverse events occur.

Clinician satisfaction and trust in a particular system are essential to the success of any practical expert system. It is then essential that the system be programmed to display explanations of all the reasoning processes to the clinician. Additionally, the encoded expert knowledge must evolve over time while being continually sanctioned by the professional community.

The traditional process of gathering expert clinician knowledge and subsequently encoding it in an expert system can involve extensive, time-consuming interviews and complex programming. Artificial intelligence techniques using knowledge-based *machine learning*, *neural networks*, *Bayesian networks* or *fuzzy logic* are some methods used for creating expert systems [8]. These methods can only be implemented by individuals with substantial computer science training and experience. Thus there are few expert clinicians who can directly translate clinical knowledge into a computerized format. Instead, development must involve two people, a knowledge engineer (programming the knowledge into the system) and the expert clinician, but there is often a communication breakdown between the two experts. A simpler method of knowledge encoding is required; one which does not require a knowledge engineer.

Table 1. Considerations in the design of a knowledge authoring tool

1. The *rule* structure should be easily understood by any clinician
2. The process should be intuitive and allow for open collaboration
3. All decisions should be visible and sanctioned by the user while functioning as memory aids
4. Any willing clinician with minimal computer skills should be able to contribute to the knowledge base

We have designed a knowledge authoring tool which allows clinicians to easily encode their knowledge for use in a *rule* based expert system. The expert system, with the aid of the *decision support engine*, uses the knowledge encapsulated in the rules to provide clinicians with advice in real time. The decision support engine contains a *knowledge base*, defined as a set of *rules* holding expert knowledge. The clinician creates *rules* for the knowledge base and continually updates them when new knowledge is available. Individuals with significant and sufficient medical knowledge (users) can use the knowledge authoring tool to build on the knowledge base, which is then provided to the real time decision support engine. Four essential attributes of the tool were considered as listed in Table 1.

BACKGROUND

In previous work we have developed *trend detection algorithms* [9–11] that have been used to extract key features from the stream of physiological data produced at the bedside. These features (subtle trend changes) provide context relevant information. To overcome the distraction of the high frequency and low importance of individual features we chose to combine the individual features in single variables using a set of rules. These rules included individual patient demographics and information on the anesthetic technique. This has potential to reduce the amount of information required to be communicated to the anesthesiologist. It soon became evident that a tool was required to support rule development by offering an easy method for clinicians to create the knowledge base. Traditional methods for the knowledge base creation, through interviews between clinicians and programmers, are cumbersome and time consuming. A simple and easy-to-use knowledge authoring tool was required: a software program that clinical experts could use to encode their knowledge without the need for a programmer.

Expert systems are well established for use in the prevention of errors in the aviation and atomic energy industries [12]. For example, extensive research using *human reliability analysis* on human reaction response in nuclear power plant environments has shown humans to be more reliable with the use of decision support tools [13]. A knowledge-based approach to monitoring and treating patients undergoing anesthesia for cardiac surgery has been described [14]. This system combines measurements of vital signs with information on recent medication administration to build an intelligent alarm system for patient monitoring during cardiac anesthesia. A decision support system used to identify patient conditions during surgery (such as light anesthesia or unstable blood pressure) and alert the clinician to potential critical conditions has been described [15].

InCare is an example of a successful *rule* based patient monitoring alarm system [16]. This system was limited to detecting only four conditions. When these conditions were detected, explanations of the conditions were stored in a log file, however the authors suggested a more comprehensive explanation displayed directly on the user interface would be an improvement.

The SmartCare methodology for automating clinical guidelines using a combined knowledge engineer and expert systems technique has been described [17]. Evaluation of a SmartCare application that automated control of a mechanical patient ventilator was shown to reduce the amount of time needed for patient mechanical ventilation. The application controlled the ventilation rate but could be easily overridden by a user at any given moment. This application has been integrated into a modern patient ventilator.

DESIGN CONSIDERATIONS

Rule-based

Our knowledge authoring tool was initially modeled using the Web Ontology Language (OWL) and the Protégé 2000 plug-in program [18]. The logical relationships of this tool, although extensively flexible, were too complicated to be efficiently understood by the majority of clinicians. Instead a *normative* rule-based system was chosen to represent *static decision-making*. Static decision-making means the *rules* in the system do not automatically change. Given the same set of inputs, the same outputs will always be returned by the system. The knowledge base consists of a set of *rules* and each *rule* is designed as an if-then statement representing expert knowledge. Each *rule* contains a list of *patterns* and an *outcome*. Each *pattern* is a statement about a data parameter and each will be true or false depending on the value of the data parameter. The *outcome* is a descriptor of patient psychological status that exists given that all the *rule's patterns* are true. For an example, a *rule* contains a single *pattern*: ''The concentration of end-tidal carbon dioxide $EtCO_2$ is found between 6 kPa and 20 kPa'' and an *outcome*: ''Hypercapnia: too much carbon dioxide in the patient's blood''. If the decision support engine detects a value of $EtCO_2$ that is in this range then the *pattern* statement is true and the *rule* is said to fire. This results in the hypercapnia *outcome* being displayed to the user and added to the database of the system. As this *outcome* is now present in the database, other *patterns* may use it. For example, the *pattern:* ''hypercapnia exists'' is true if the hypercapnia *outcome* is present in the system. This reuse of an *outcome* is the concept of *rule* chaining which allows *rules* to build on each other and removes redundant *patterns* from the system.

User interface

The *graphical user interface (GUI)* of the application is designed to simplify the process of *rule* creation (Figure 1). The GUI is modeled after the *rule* structure (a list of *patterns* and an *outcome*) to guide a user through the *rule* building process. Elements of the GUI related to *patterns*, *outcomes*, or entire *rules* are color-coded and grouped together. To successfully create a *rule*, a user is not required to have knowledge of the *syntax* behind *rule* coding. They need only fill in each part of the GUI. This task involves making selections from drop down boxes, clicking buttons, and entering numerical values or text descriptions, all of which are more user-friendly for the average clinician than input of complex computer code.

When designing a software program, it as important to enable the user to recognize, rather than recall, the terminology they are required to use. In this application the user is able to select from a list of demographic and physiological parameters, rather than being forced to recall the required parameters for making *patterns*. These lists of parameters are loaded from an *XML* file each time the application is initiated. Users can edit the XML file with any text editor and they can learn the format of this file from the help pages of the application. For example, the user may wish to add additional parameters to the default parameter list in this file. Also included in this file, is the range of values allowed for each parameter. For example, in Figure 1, the user can only enter a range of heart rate between 20 and 250 beats per minute. These GUI constraints allow the application to translate the *rules* that a user has created into code that conforms to the style and logic of the *rule* syntax.

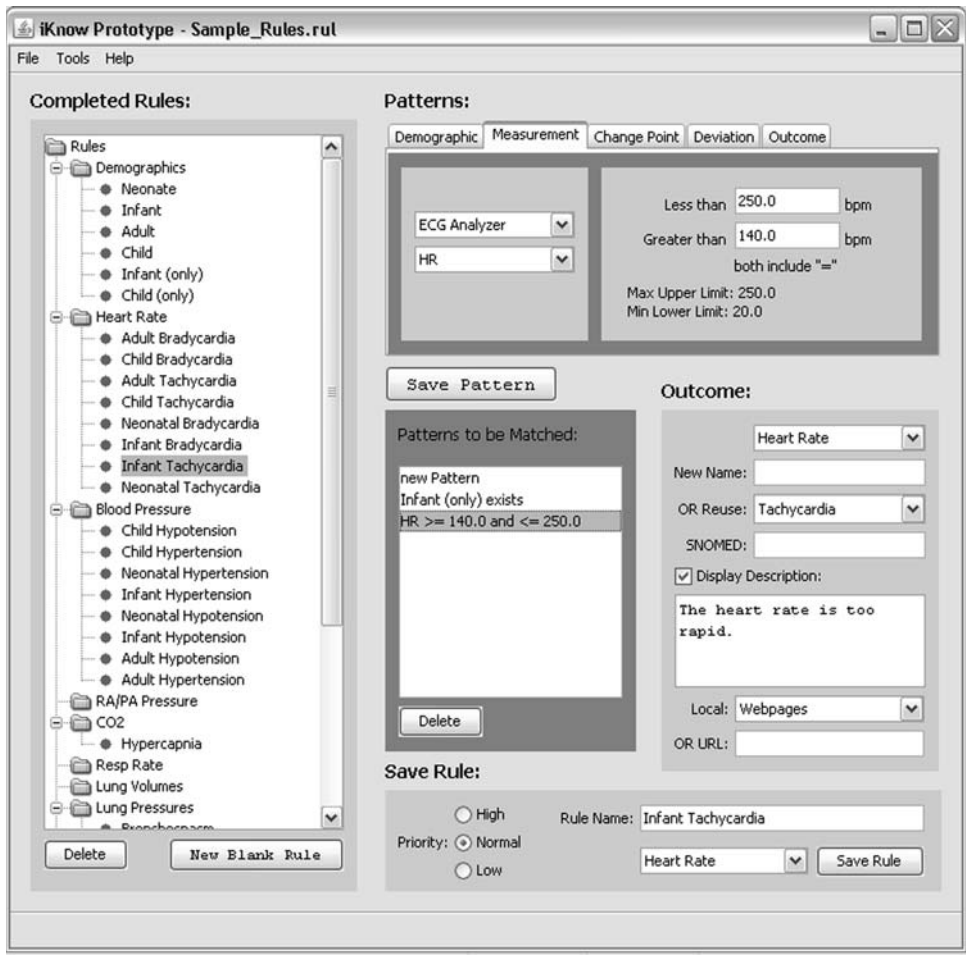

Fig. 1. The graphical user interface of the knowledge authoring tool.

Visibility

A visualization tool has been developed that allows users to see the integration of multiple levels of *rules*. To see the visualization for a *rule* or *outcome* the user selects the *rule* or *outcome*. Opening the *rule* visualization window shows the hierarchy of the *rule* or *outcome* (Figure 2). This is represented in a tree structure, where each node (rectangle in the tree) is a *rule* or *outcome*. The top node of the tree represents the selected *rule* or *outcome*. The user may also expand this window to show all the *patterns* within each *rule*.

Visualization of the *rule* and *outcome* hierarchy allows the user to recognize potential inconsistencies and correct the *rule* structure. For example, leaf nodes are those elements of the tree with no nodes connected below them (all nodes in the bottom row in Figure 2). Then the presence of a leaf node that is an *outcome* indicates an inconsistency in the *rule* structure, since it implies that no *rule* ends with this *outcome*. The visualization tool acts as a confirmation of the mental model and logic relationships between any *rules* and *outcomes* created by the user.

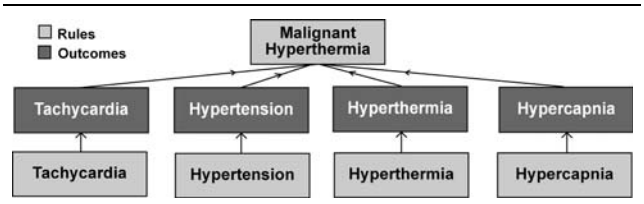

Fig. 2. Part of the visualization tree for Malignant Hyperthermia in an infant.

Rule testing

The application includes a test plug-in, which simulates the real time decision support engine. This plug-in allows the user to test *rules* during the rule development process.

A spreadsheet file containing clinical data arranged in tab delimited columns can be used to test *rules*. Data in the spreadsheet can be edited to test the integrity of the logic and ensure *rules* fire when they should.

Collaboration

In the design of the application, it was important to incorporate a user-friendly method of combining *rule* sets. A collaborative development of the knowledge base by various medical communities, such as anesthesiologists, is anticipated. To facilitate *interoperability* of this system and widespread adoption, the programming language *Java* was chosen. Java enables portability across operating systems and hardware platforms. Users from different medical communities may easily merge separate *rule* files to provide a wider range of knowledge to the decision support engine.

Merging is the process of adding an external set of *rules* to the current set of *rules*. To maintain integrity, any additional *rule* may not have the same name as any of the pre-existing *rules*. However, to establish connections between each set of *rules*, an *outcome* with the same name may appear in both sets of *rules*. If an *outcome* is in both *rule* sets, the user can choose which of the two *outcome* definitions they wish to keep. For example, assume user A created *rule* 1 containing the *outcome* Hypercapnia with a description ''Too much carbon dioxide in the blood'', while user B has created a *rule* 2 with the *outcome* Hypercapnia and description ''End tidal carbon dioxide too high''. Now assume user A wants to merge user B's *rules* into the current rule set. The user will be prompted to choose which of the two descriptions to use for *outcome* Hypercapnia. The chosen description will be used as the description for this outcome throughout the new set of *rules*, while the alternative description will be discarded. Thus, this Hypercapnia *outcome* will now be the *outcome* for both *rules* A and B. This functionality allows convenient merging of *rule* sets while ensuring standardization of outcome descriptions across the *rule* set.

SYSTEM DESCRIPTION

Data model

The ISO/IEEE 11073 International Standard Health Informatics Domain Information Model [19] and Nomenclature [20] were used to model the current application data input and output. The Medical Package Object defined in the ISO/IEEE 11073 Domain Information Model is designed to offer an organized structure for health information. The Medical Package contains a Virtual Medical Device (VMD) that contains all the input and output data elements. This VMD has several Channel Objects. Each Channel Object is modeled after a physiological monitor and contains a set of physiological parameters. These parameters are each stored as a Metric Object, which contains a set of values and time stamps. For example, the Blood Pressure Analyzer Channel contains Metrics for systolic (NIBPsys), diastolic (NIBPdia) and mean (NIBPmean) non-invasive blood pressures. Additionally, a Custom Channel Object contains a Custom Metric object for the results of the trend algorithms.

Rule engine environment

The *open source* JBoss Rules engine, Drools, was chosen, which has an enhanced implementation of an *object-oriented* modification of the Rete algorithm [21]. The Rete algorithm, invented by Dr. Charles Forgy, is a *pattern* matching algorithm that optimizes the sequence of *rule* selection in rule engines. *Working memory* is the component of a *rule* engine which stores the data that are tested by the *rules*. The working memory of the Drools engine is designed with the capability to store Java objects. This allows the *patterns* in the *rules* to be directly tested on the values of these objects. The Drools engine has been implemented in the test plug-in (built into the authoring tool) and in the real-time decision support engine.

Rule structure

A *pattern* is a statement about data that can be verified by the decision support system as either true or false. Clinicians can create five different types of *patterns*:

1. Demographic *patterns* define value ranges for *static parameters*, entered into the decision support engine by a clinician or collected from an electronic data record from the Hospital Information System. (e.g. The patient is less than 1 year of age.)
2. Measurement *patterns* define limits for physiological parameters, generated by monitors in the operating room. (e.g. The heart rate is greater than 140 beats per minute.)
3. *Change point patterns* define the existence and direction of change points determined from the trend detection algorithms [9]. (e.g. The expert system found an increasing change point for the heart rate.)
4. Deviation *patterns* define the existence of changing physiological parameter values over time. To create one of these *patterns* a user selects a parameter and then a multiple of the *standard deviation* (1, 1.5, 2, 2.5, or 3)

and a time frame (5, 10, 15, 20, 30 or 60 min). When running the rules, the mean value of the parameter is computed over the time frame. The standard deviation of this parameter is also computed over the time frame and multiplied by the chosen multiple. Then it is added to and subtracted from the mean value to define a range where the current value is most likely to be found. This *pattern* is then determined to be true if the value is in fact above or below the defined range. (e.g. The current heart rate value is above its standard deviation multiplied by 2, computed from the last 10 min of heart rate data.)

5. *Outcome patterns* define the existence of *outcomes* from previous *rules* in the decision support engine. (e.g. The patient had tachycardia (*outcome* of a previous *rule* with a measurement *pattern*).)

If all *patterns* in a *rule* are found to be true, the *rule's outcome* will then be displayed to the decision support engine user and added to the working memory. Additionally, the *patterns* of the *rule* and the data values which made these *patterns* true are displayed (explanation feature). An *outcome pattern* is true if the *pattern's outcome* exists in the working memory (if a *rule* with this *outcome* has fired). Multiple *rules* may contain the same *outcome* and a *rule* may contain multiple *outcome patterns*. This reuse of *outcomes* is called *rule* chaining, and is structured as shown in Figure 3.

An example of *rule* chaining is shown in Figure 4. A *rule* is created to standardize the varying definitions of Child. This is desirable if there are several *rules* that apply only to children. If this Child rule is created, then the user may

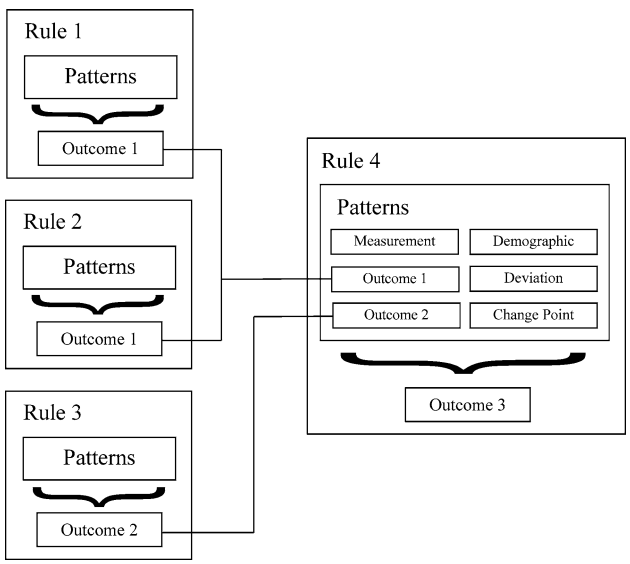

Fig. 3. Rules 1 and 2 contain outcome 1 while rule 4 contains both outcomes 1 and 2 and results in outcome 3.

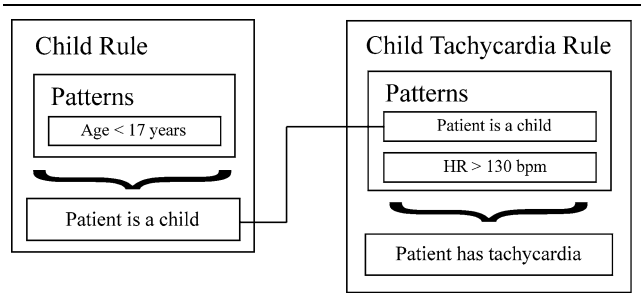

Fig. 4. A child rule defines a child outcome that can be used in any rule specific to a child.

select the Child *outcome* for use in each child specific *rule* rather than redefining the child age *pattern* for each *rule*. The Child *rule* will be amenable to possible future changes in definition. If the standard definition of a child were to change, only this *rule* would be required to be altered to update the complete *rule* set.

Control mechanisms in the decision support engine

The *rules* are saved as Java objects so they may be easily opened by the decision support engine or reopened by the knowledge authoring application. In the decision support engine, these *rules* are then translated into the Drools language format. Specific copies, called *instances*, of a simple Java class called Fact are used to store each piece of data that enters the decision support engine. Prior to running all the *rules*, the decision support engine retrieves data (e.g. patient age, weight, mode of ventilation) from a user interface, clinical information system, clinical monitor interface, or hospital information system. The decision support engine then creates a Fact about each of these data. During the running of the *rules*, real time data are retrieved from the physiological monitors in the operating room. The decision support engine then repeats the following process every time new data are present (usually every 5 s):

1. Create the rule engine's working memory.
2. Assert into working memory, each Fact in the list of patient demographics.
3. Assert the current time in working memory.
4. Create Facts from all current parameter values and assert each in working memory.
5. Run all the *rules* on all information asserted which results in any *outcomes* from fired *rules* appearing in an *outcome* table with the time listed.
6. Destroy working memory.

During each case, the clinician may view additional information for each *outcome* that has been displayed, such as a link to more information (this information can

include a differential diagnosis or treatment protocol), and an explanation of the displayed *outcome* consisting of the data and *rules* supporting it.

Data quality

The most important requirement for high quality decision support and user adoption is high quality data input into the decision support engine. Physiological monitors in current clinical use have high incidences of artifacts. The many causes of artifacts have been reviewed [22]. In our current real time system we use a multistage approach to improve the quality of the data. The monitor hardware carries out filtering of most signals with averaging over 5 s or more. The proposed authoring tool design uses the averaged data values. Each trend value is also interpreted in the temporal context in which it was recorded. Our trend detection algorithms remove artifacts using individual or combinations of a number of filtering techniques including the Kalman filter, exponentially weighted moving average (EWMA) and a median filter [9, 10]. The Kalman filter has been shown to achieve optimal performance at removing nonlinear noise [22].

Electrocautery noise is another significant cause of poor quality data. Our expert system uses an electrocautery noise detector which provides an estimate of the quality of the ECG-based heart rate trend. When the ECG is clean, we use the heart rate derived from the ECG. When the ECG is noisy, we use the heart rate derived from the pulse oximeter or arterial blood pressure.

Explanation feature

A clear explanation to the user, of why each *outcome* was chosen, was considered essential with respect to the design of the decision support engine to support clinical adoption. This explanation feature allows the user to understand the reasoning of the rules used in the decision support engine. This explanation consists of a hierarchy of the raw input data, *patterns* that were true, and each *rule* in the rule chain that led to the *outcome*. The use of our user-friendly application to create the *rules*, rather than a more traditional coding approach, allows peers to collaboratively create the *rules* and explanations. The simplicity of the *rule* logic is essential in allowing the user to understand the *rule* reasoning.

EVALUATION OF USABILITY

Method

Following institution ethics approval and informed consent a usability study of the prototype application was undertaken. Anesthesiologists' with no prior exposure to the knowledge authoring tool were asked to develop rules for simulated clinical events. A pre-study questionnaire was used to collect demographic information.

Each subject received a tutorial (less than 5 min) on the purpose, layout and functions of the application. This web based tutorial is packaged in the application along with additional help pages. Next, the subject was given the task of creating a specific set of four knowledge *rules*. These *rules* were:

Rule 1: Infant.
Rule 2: Tachycardia (in an infant).
Rule 3: High BIS value.
Rule 4: Light Anesthesia (in an infant) using *outcomes* from *rules* 2 and 3.

After a time limit of 30 min, or after the task was complete, a modified (to remove non-applicable questions) Post-Study System Usability Questionnaire (PSSUQ) was used to record feedback on usability of the software program [23].

Table 2. Median rule creation times in minutes

Rule	Median time (range)
Infant	2:34 (0:56–5:29)
Tachycardia (in an infant)	3:05 (1:26–14:31)
High BIS value	2:31 (1:21–3:52)
Light anesthesia (in an infant)	5:28 (1:02–11:58)

RESULTS

Ten anesthesiologists at the British Columbia Children's Hospital were recruited. Of the 10 subjects, five were in the age range 30–39 years, with one below and four above. The subjects had an average of 9.5 years experience in anesthesiology. All subjects used computers every day, and eight subjects considered themselves to have average computer skills while the other two reported above average skills.

Despite recent introduction to the application, all subjects completed the four *rules* within a range of 8–29 min, with the median completion time of 13 min (broken down by *rule* in Table 2). During the task, subjects were permitted to ask questions (of the application

Table 3. PSSUQ scale

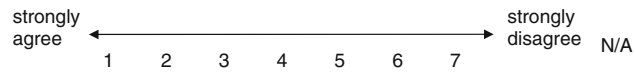

Table 4. Sample from PSSUQ results

Statement	Median rating	Sample related comment
It was easy to learn to use this system	3.5	''Help sheet quite useful''
I believe I could become productive quickly using this system	2	''Quick learning curve''
The organization of the information on the system screens was clear	3	''The layout is nice but the boxes need to be labeled more intuitively''
Overall, I am satisfied with this system	3	''When using a system like this, I think when merging to be able to drag + drop rules into the new rule/outcome would be useful''

developer). A range of 0–5 questions were asked, with a median of one question. Using the modified PSSUQ, each subject was asked to rate 16 statements on a 7-point scale (Table 3). The median rating across all subjects and statements in the questionnaire was 3, reflecting positively on the application's usability. Within the questionnaire, subjects were also able to write comments and suggestions about their user experience with the application (Table 4), and this feedback is currently being incorporated to improve usability of future releases of the software.

Future plans

Based on feedback from our usability study, the application graphical user interface will be updated to make *rule* creation more intuitive. The following are examples of future changes:

1. Make useful features more visible. For example, move *outcome pattern* creation to a panel separate from other *pattern* creation panels.
2. Avoid redundant information entry. For example, for any *rule* we will allow the user to reuse the *outcome's* name as the *rule's* name, without having to retype it (becomes the default value). A *rule* and the resulting *outcome* may have the same name.

The visualization tool (Figure 2) will be expanded to include an editing functionality. Users will be able to *drag and drop rules* and *outcomes* using the visualization tool.

While the binary nature of each *outcome* (present or not present) is limiting, this can be partially resolved through graduated labeling (e.g. mild, moderate or severe) of the *outcomes*. A more elegant method to remove this limitation may be to use a mathematical representation of each *outcome*. The binary nature of *outcomes* lacks information on how the decision support engine defines certainty of the *outcome's* existence. In the future, we plan to add degrees of likelihood with fuzzy logic and/or probabilities using Bayesian algorithms to the *outcomes*, with the challenge of maintaining a simple to use system.

In future releases users will be encouraged to connect a SNOMED code [24] with each *outcome*. This update will further standardize the *outcomes* provided by the decision support engine.

CONCLUSION

The adoption of expert systems into everyday clinical practice can be significantly enhanced by allowing a group of users to encapsulate their knowledge in a simple set of *rules*. This will allow users to continually add new knowledge and adapt the system to change with medical knowledge. The application *rule* structure and simple interface allows users to intuitively create and understand the knowledge they are expressing. This software application is simple yet can encapsulate complex chains of knowledge. The program encourages collaboration among medical experts and provides a platform for implementing clinical guidelines and standardizing treatment protocols. A clear explanation of each *outcome* allows clinicians to trust the advice provided by the decision support system. A simple knowledge authoring tool provides an important step to the introduction of expert systems into clinical monitoring. This early prototype will unleash the potential for the processing power of computers to improve clinical care and reduce severe adverse events.

This project was supported by the Canadian Institutes of Health Research. We would like to thank all the experts who have assisted and advised us with developing this application. We would also like to thank the volunteers who participated in the usability evaluation for their ideas and feedback and Elizabeth Cheu for reviewing and editing the manuscript.

GLOSSARY

Bayesian Networks – A technique that is based on the relative probability of an event given the probabilities of associated events in the network; employs Bayes' theorem.
Change point – A significant point of change in a physiological parameter found by using trend detection algorithms.
Decision support engine – An expert system that assists and potentially enhances a human's ability to make decisions.
Drag and drop – In a computer graphical user interface, the process of clicking an object and then holding down and dragging it to another location before releasing.
Expert system – A software-based system that integrates a mass of information based on rules or processing performed within the software program to supply expert knowledge about a specific field.
Fuzzy logic – Reasoning methodology producing a definite conclusion based upon vague, ambiguous, imprecise, noisy or missing input information.
Graphical user interface (GUI) – A user interface (part of the program the user interacts with) which contains the graphic elements: icons, text, labels, buttons, etc.
Human reliability analysis – The study of the probability that a human will correctly perform a task and those factors related to this probability.
Instance – In terms of Java, a specific object of a Java class.
Interoperability – The ability to communicate and operate with different hardware and software systems.
Java – An object-oriented programming language.
Knowledge base – The encoded knowledge for an expert system. In a rule-based expert system, a knowledge base incorporates definitions of attributes and rules along with control information; a store of factual and heuristic data.
Knowledge encoding – The process of building a knowledge base through encoding the human expert knowledge in the computer language being used.
Knowledge engineer – The person encoding the knowledge into the knowledge base.
Knowledge rule – A rule written in the language of the knowledge base, which is used by a decision support system or expert system to analyze data and make decisions.
Machine learning – A method of artificial intelligence in which patterns are found within the data to enable the application to slowly learn how different pieces of data are interconnected.
Neural network – A method of artificial intelligence which is used to solve tasks through a network of simple processing units, model similar to biological neuron networks.
Object-oriented – Design methodology that breaks down problems into objects rather than procedures.
Open source – Any project whose source code is made available for use or modifications as users or developers see fit.
Standard deviation – The measure of the spread of a parameter's values.
Static decision-making – Decision making that does not change. Given the same input, the same decision will always be made.
Static parameter – A parameter whose values remains known and unchanged throughout a process.
Syntax – In computer programming, the conforming rules of the code, which must be followed for the code to be valid in the computer language used.
Trend detection algorithm – A computer process which identifies changing trends of the physiological parameters being monitored. Significant trend changes in a parameter are recorded as change points.
Ventilatory events – A change in patient's ventilation, outside defined normal limits, within anesthesia.
Working Memory – This is where all the facts in the decision support engine are located.
XML – A general purpose mark-up language used as the format for configuration files.

REFERENCES

1. Gaba DM, Fish KJ, Howard SK. Crisis management in anesthesiology, Churchill Livingstone: New York, 1994.
2. Ansermino JM. Patient safety: technology and design can help. B C Med J 2006; 48(7): 339–341.
3. Miller GA. The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychol Rev 1956; 101(2): 343–352.
4. Liu J, Wyatt J, Altman D. Decision tools in health care: focus on the problem, not the solution. BMC Med Inform Decis Mak 2006; 6: 4 .
5. Hales BM, Pronovost PJ. The checklist—a tool for error management and performance improvement. J Crit Care 2006; 21: 231–235.
6. Neily J, DeRosier JM, Mills PD, Bishop MJ, Weeks WB, Bagian JP. Awareness and use of a cognitive aid for anesthesiology. Jt Comm J Qual Patient Saf 2007; 33: 502–511.
7. Harrison TK, Manser T, Howard SK, Gaba DM. Use of cognitive aids in a simulated anesthetic crisis. Anesth Analg 2006; 103: 551–556.
8. Imhoff M, Kuhls S. Alarm algorithms in critical care monitoring. Anesth Analg 2006; 102(5): 1525–1537.
9. Yang P, Dumont G, Ansermino JM. Adaptive change detection in heart rate trend monitoring in anesthetized children. IEEE Trans Biomed Eng 2006; 53(11): 2211–2219.
10. Yang P, Dumont G, Lim J, Ansermino JM. Adaptive change point detection for respiratory variables. In: Proceedings of the EMBS 27th Annual International Conference, Shanghai, China, Sep 1–4, 2005; 780–783.
11. Yang P, Dumont G, Ansermino JM. An adaptive cusum test based on a hidden semi-Markov model for change detection in non-invasive mean blood pressure trend. In: Proceedings of the EMBS 28th Annual International Conference, New York City, USA, Aug 30–Sep 3, 2006; 3395–3398.

12. Nabeshima K, Suzudo T, Seker S, Ayaz E, Barutcu B, Turkcan E, et al. On-line neuro-expert monitoring system for borssele nuclear power plant. Prog Nucl Energy 2003; 43(1): 397–404(8).
13. Swain AD, Guttmann HE. Handbook of human-reliability analysis with emphasis on nuclear power plant applications, Sandia National Labs: Albuquerque (US-NM), 1983.
14. Schecke T, Langen M, Popp HJ, Rau G, Käsmacher H, Kalff G. Knowledge-based decision support for patient monitoring in cardioanesthesia. J Clin Monit Comput 1993; 9(1): 1–11.
15. Krol M, Reich DL. Development of a decision support system to assist anesthesiologists in operating room. J Med Syst 2000; 24(3): 141–146.
16. Sukuvaara T, Makivirta A, Kari A, Koski E. An intelligent intensive care alarming system. Proc Comput Cardiol 1989; 225–228.
17. Mersmann S, Dojat M. SmartCare – automated clinical guidelines in critical care. In: Mantara RL, Saitta L, eds, 16th European Conference on Artificial Intelligence, Valencia, Spain, Aug 22–27, 2004; 745–749.
18. Knublauch H, Musen MA, Rector AL. Editing description logic ontologies with the protégé OWL plugin. In: Haarslev V, Möller R, eds, Proceedings of the International Workshop on Description Logics, June 6–8, 2004, Whistler, BC, Canada. Central Europe: CEUR Workshop Proceedings; 2004.
19. IEEE-SA Standards Board. Health informatics—point-of-care medical device communication—part 10201: domain information model (1st ed.). ISO/IEEE: USA, 2004.
20. IEEE-SA Standards Board.. Health informatics—point-of-care medical device communication—part 10101: nomenclature (1st ed.). ISO/IEEE: USA, 2004.
21. Drools Documentation. Accessed online at http://downloads.jboss.com/drools/docs/4.0.3.15993.GA/html_single/index.html Nov 9, 2007.
22. Takla G, Petre JH, Doyle DJ, Horibe M, Gopakumaran B. The problem of artifacts in patient monitor data during surgery: a clinical and methodological review. Anesth Analg 2006; 103: 1196–1204.
23. Lewis JR. IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. Int J Hum Comput Interact 1995; 7: 57–58.
24. Elevitch FR. SNOMED CT: electronic health record enhances anesthesia patient safety. J Am Assoc Nurse Anesth 2005; 73(5): 361–366.