Combining Visual Block Programming and Graph Manipulation for Clinical Alert Rule Building

Dave Krebs
Department of Computer Science
University of Pittsburgh.
Pittsburgh, PA 15213, USA
djk37@pitt.edu

Alex Conrad
Department of Computer Science
University of Pittsburgh.
Pittsburgh, PA 15213, USA
apc31@pitt.edu

Jingtao Wang
Department of Computer Science
University of Pittsburgh.
Pittsburgh, PA 15213, USA
jingtaow@cs.pitt.edu

CHI'12, May 5–10, 2012, Austin, Texas, USA.
ACM 978-1-4503-1016-1/12/05.

Abstract
In this paper, we present MARBLS (Medical Alert Rule BuiLding System) – a visual end-user programming environment to facilitate the design and testing of clinical alert rules. MARBLS enables a two-way, synchronized visual *rule workspace* and visual *query explorer*. Clinical rules can be built by docking relevant block components in the *rule workspace*, by directly manipulating graphs in the *visual query explorer*, or a combination of both. In a pilot study with five healthcare experts, we found MARBLS is easy to learn and users can build rules efficiently and discover errors in existing rules quickly with the visual environment provided by MARBLS.

Author Keywords
Visual Language; End User Programming; Health Care; Clinical Monitoring Rules.

ACM Classification Keywords
H5.2 [Information interfaces and presentation]: User Interfaces - Graphical user interfaces; J.3 [Computer Applications] Life and Medical Sciences - Health, Medical Information Systems.

General Terms

Design, Human Factors.

Introduction

Hospitals nowadays usually deploy various clinical decision support systems and tools [2] aimed to help healthcare experts to monitor patients, alert on the occurrences of various adverse events [1], or send reminders assuring the adherence to various clinical guidelines. In majority of existing systems, the knowledge about what to detect and when to alert is defined in terms of if-then (or condition-action) rules. For example, the following is a rule for detecting a patient who is at risk of the heparin-induced thrombocytopenia (HIT) [14], a condition that may lead to thrombosis and even to death if not handled promptly:

If a patient is on heparin, dalteparin, or enoxaparin and the most recent platelet count is low (<100k) or the platelet count dropped more than 50% within last seven days, then send an email alert to the attending clinical pharmacist.

According to our contextual inquires with physicians, The monitoring subsystem in a typical hospital setting usually maintains a few hundred active rules. These rules are usually written in a special rule-definition language. If rules are simple, the task of writing them for an expert is usually easy. However, when rules and the conditions they detect become more complex, writing them accurately and testing them thoughly in the given language can become a very challenging process. In fact, it is not uncommon that the physicians have to work closely with programmers to accurately encode them in the system. In addition, physicians usually do not design all the rules from scratch, they build new rules from an existing commercial rule base, and rules could also evolve alone the time. All these challenges demand effective and intuitive to use interfaces for rule building.

To address these challenges, we present the design, implementation and a pilot evaluation of MARBLS (Medical Alert Rule BuiLding System) (figure 1), to explore whether a visual block based end-user programming environment could help physicians and practitioners to design clinical alert rules more intuitively and reliably, and whether such a visual programming environment could help physicians to discover different types of errors in a program effectively.

Related Work

There is no industrial standard on clinical alert rule building at this moment. The rule building tools and interfaces available depend on the actual provider of hospital information systems (HIS). Common HIS providers include Cerner, Epic, MEDITECH, and McKesson etc. In a Cerner based solution, programmers need to learn to write alert rules in a proprietary program language named Cerner CCL [3]. The primary programming environment for designing alert rules is EKM Editor, which is a classic IDE for text based programming, with features such as keywords highlighting and code reusing via programming templates.

Adverse Event Detection

The idea of monitoring and detecting adverse events is not limited to healthcare [1]. Adverse event detection is an important problem in diverse application domains

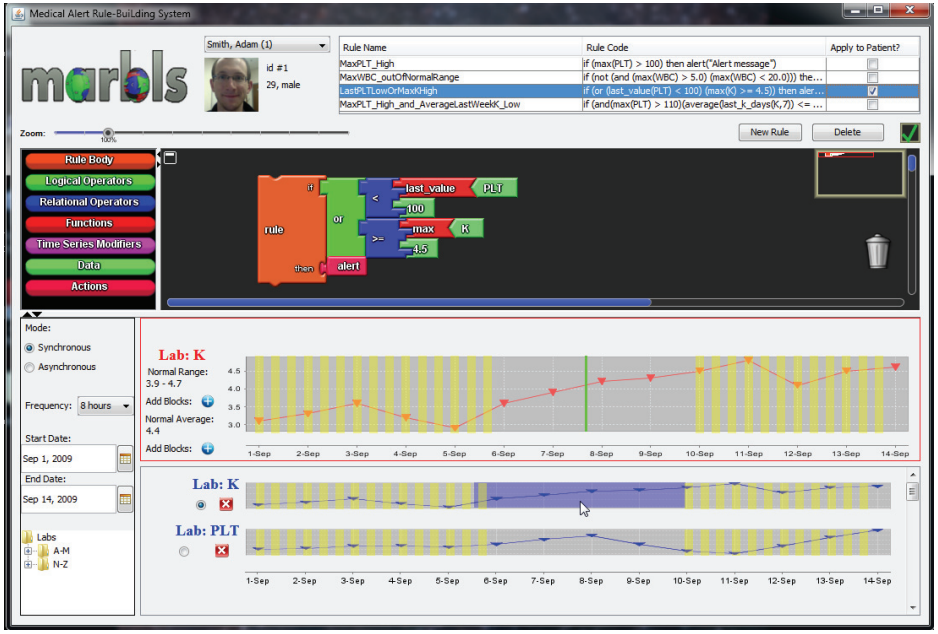

Figure 1. The primary interface of MARBLS. Top-right: commonly used alert rules. Middle: the interactive *rule workspace*. Bottom: the *visual query explorer* (showing simulated data).

such as manufacturing, finance, space exploration, etc. [4]. However, the use of automatic algorithms for adverse event detection is relatively limited in healthcare due to special restrictions in protecting patient data and the high reliability needed. We expect that rule based alert systems and automatic algorithm based adverse event detection could co-exist and complement each other in the future.

End User Programming

The design of MARBLS is motivated by previous work on Blocked based Visual Programming (BVP). The idea of BVP has been widely used by systems such as KidSim [5], Scratch [12], and Google App Inventor. Such kind of visual environments have been proven to have a low learning threshold for kids and non-programmers. Recent research by Harvey and Mönig [8] has also shown that advanced programming concepts such as recursion, lambda operator and object oriented programming can be easily extended in Block based programming environments, hence enabling a high ceiling even for "computer scientists".

The current application of EUP in healthcare environment is limited, used mainly in creating customizable medical report forms [11].

Novel Interfaces and Visualizations in Healthcare

Researchers have designed various visualization tools to help physicians explore and make sense of patient records. PatternFinder by Fails and Karlson el al [7] allows physicians to query patient records by cascading a set of Event Boxes. The LifeFlow project [15] can generate a tree-based, overview of medical event sequences. In TimeSearcher [9], Hochheiser and Shneiderman invented novel on-chart direct-manipulation techniques such as TimeBox and Angular Query to capture patterns from a large number of time series data. These tools can help physicians locate interesting patient data, hence inspire the creation of new rules, but they cannot be used to create, test and simulate alert rules directly, especially for medication related rules.

The Design of MARBLS

The architecture of MARBLS is shown in Figure 2. The system consists of three main components: the

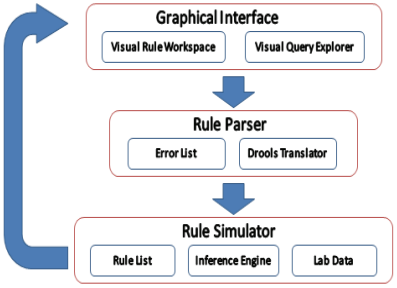

Figure 2. The system diagram of MARBLS.

Graphical Interface, the Rule Parser, and the Rule Simulator.

The Graphical Interface

One major research contribution of MARBLS is that it maintains two-way, synchronized connections between the *rule workspace* and the visual *query explorer* – e.g. operations applied in one view are able to generate and update properties in the other view, and vice-versa. At this time, MARBLS supports six categories of blocks: body blocks (used to connect the **if**-conditions and the **then**-actions together), operator blocks (including logical operators such as "and", "or" etc and relational operators such as "<=", ">" etc), function blocks (such as "min", "max", "average" etc), time-series-modifier blocks (e.g. "last-three-days"), lab observational data blocks (e.g. platelets level, a.k.a. PLT; glucose level, a.k.a. GLU), and action blocks (such as activating an alert message). These six categories of blocks are expressive enough to cover most of the real world rules we have investigated. The constraints provided by the particular shapes of the "plugs" and "sockets" on each block ensure that blocks can only be attached together in a style that leads to a syntactically-valid rule. Similar approaches that relay on interlocking "jigsaw puzzle" pieces have been shown to be effective for non-technical users to complete programming tasks [10]. Zooming and panning operations are supported for building more complex rules.

MARBLS also supports building and updating rules in the *rule workspace* by direct graph manipulations operating such as selecting and lassoing in the *visual query explorer*. One function of the *visual query explorer* is to automatically display relevant data to the user during rule-building activities. The information that is displayed is patient-specific and general lab data. A time series graph of lab values is given as the patient-specific data for a particular lab, while a normal range for the lab and a normal average value are given as the general data. This data is displayed in response to a user action: when the user inserts a lab piece into the *rule workspace* (e.g. the glucose lab piece, GLU), the *visual query explorer* will automatically add a time-series graph for glucose, along with the general data for glucose, to its current contents. In this way, actions in the rule workspace generate data in the visual query explorer.

The *visual query explorer* can also display triggered alert signals in real time once a syntactically correct rule is constructed in the rule workspace.

The Rule Parser

The *Rule Parser* accepts the raw output from visual Rule Workspace, parses it, reports errors when necessary and translates the rule into the format accepted by the *Rule Simulator*.

Implementation

MARBLS is implemented in Java. OpenBlocks [13] and JFreeChart have been adopted for block manipulation and charting support. We use Java Compiler Compiler (JavaCC) to create the rule parser component in use. Drools [6] has been employed to build the rule simulator.

Pilot User Study

We conducted a pilot user study to help evaluate our system. The pilot study consisted of six parts (1. Pre-study questionnaire; 2. Tutorial; 3. Free trial and Q&A;

4. Rule building task; 5. Error detection task; 6. Closing questionnaire). It took around one hour to complete.

In the rule building task, a participant was given English descriptions of two rules and was asked to build these rules in MARBLS. In the error detection task, each participant was given the English description of six rules, along with the actual implementations, three in the block condition and three in the string condition (i.e. defining rules via a style similar to that of high level programming languages with more "reader friendly" key words). An error was intentionally inserted into each implemented rule. The cause of errors and the complexity of rules were comparable in both conditions, but rules used were not exactly the same, the order of the two conditions was randomized.

Five subjects (3 male and 2 female), 35 – 53 years old, participated in the experiment. All the subjects hold Doctor of Medicine (MD) degrees and three of them have experiences with clinical alert rules. A within subject design was employed.

Results

For the rule building task, all the five subjects, although get exposed to MARBLS for the first time, could complete both rule building tasks and build correct rules. Given that a rule can be built via the block based rule workspace alone or a combination of rule workspace and visual query explorer, four of the five users consistently used the visual query explorer to assist in the construction of both rules. The average completion time for constructing the first rule was 240 seconds (SD = 50.5 sec), and the average completion time for the second rule was 515 seconds (SD = 580.8 sec). Please note that users were not instructed to complete rule building tasks as fast as possible. Consequently, users approached this task with a relaxed and exploratory attitude. Because of the small sample size and the exploratory nature of the study, there was no significant correlation between template usage and rule completion time. However, in future work we plan to conduct a comprehensive user study to examine the relationships between our various interaction mechanisms and rule completion time.

For the error detection task, one user didn't complete it because he indicated extra interest in MARBLS and spent longer than usual time in the Q&A stage and didn't have time to complete the error correction task. The average completion time for locating errors in the block-based tasks was 8.7 seconds (SD = 9.1 seconds), while for the string-based tasks, the average time was 18.9 seconds (SD = 6.9 seconds). All users showed a significant improvement in detection times for the second and third block-based error-detection tasks over the first task of this kind.

The overall feedback from the subjects was highly positive. Subjects found MARBLS not only intuitive to learn, but also "*Fun*" to use. They felt the visual affordance of a block based programming environment fit nicely with the requirements of rule building: e.g. "*In the block language, the wrong type of argument cannot be used because it does not 'click in.'*". "*The stepwise approach of the blocks is useful to the novice user and allows the building of complex rules…*". "*The visualization block building is very useful in that it builds a part of the rule in one step…*".

Subjects enjoyed the convenience introduced by the visual query explorer – "*… using direct manipulation of*

objects for rule construction is great for non-technical as well as technical users."

The subjects also identified usability problems in the current implementation. For example, our current "and" function block only has two docking sites, so the users feel slightly clumsy to implement a three-way "and" logic via two cascaded "and" blocks. The subjects also feel that docking some frequently connected blocks together every time could become repetitive from time to time:

"*I think having several templates will facilitate rule building. For example, relational operators always require a value, so these blocks can be combined*."

Conclusion

We have presented MARBLS (Medical Alert Rule BuiLding System), a visual end-user programming environment for designing clinical alert rules. MARBLS combines both Block based Visual Programming (BVP) and direct graph manipulation to lower the threshold of clinical alert rule building and testing for healthcare experts. In a pilot study with five domain experts, we discovered MARBLS is easy to learn; users can build rules efficiently and discover errors in existing rules quickly with the visual environment provided by MARBLS.

References

[1] Bates, D., Evans, R., et al, Detecting Adverse Events Using Information, Technology, *Journal of AMIA*, Vol. 10, No. 2, Mar / Apr 2003.

[2] Berner, E., ed. Clinical Decision Support Systems: Theory and Practice. New York, NY: Springer, 2007.

[3] Cerner CCL, http://en.wikipedia.org/wiki/Cerner_CCL

[4] Chandola, V., Banerjee, A., Anomaly Detection: A Survey, *ACM Computing Surveys*, Vol. 41, No. 3, Article 15.

[5] Cypher, A., Smith, D., KidSim: End User Programming of Simulations. In *Proc CHI 1995*.

[6] Drools Business Logic Engine, http://www.jboss.org/drools

[7] Fails, J., Karlson, A., Shahamat, L., et al, A Visual Interface for Multivariate Temporal Data: Finding Patterns of Events over Time, In *Proc VAST 2006*.

[8] Harvey, B., Mönig, J., Bringing "No Ceiling" to Scratch: Can One Language Serve Kids and Computer Scientists?, In *Proc Constructionism 2010*.

[9] Hochheiser, H., Shneiderman, B. Dynamic query tools for time-series data sets: Timebox widgets for interactive exploration. In *Proc Infovis 2004*.

[10] Horn, M., Solovey, E., et al. Comparing the use of tangible and graphical programming languages for informal science education. In *Proc of CHI 2009*.

[11] Orrick, E., Electronic Medical Records–Building Encounter forms. In *Proc the 2nd Workshop on End-User Software Engineering, in Conjunction with CHI 2006*.

[12] Resnick, M., Maloney, J., et al. Scratch: Programming for All. In *Comm of the ACM*, November 2009.

[13] Roque, R., OpenBlocks : an extendable framework for graphical block programming systems, Master thesis, Dept. of EECS, MIT, 2007.

[14] Warkentin T., Heparin-induced thrombocytopenia: pathogenesis and management. In *British Journal of Haematology*; 121:535-555, 2003

[15] Wongsuphasawat, K., Gomez, J., LifeFlow: Visualizing an Overview of Event Sequences, In *Proc CHI 2011*.