A Programmable Rules Engine to Provide Clinical Decision Support Using HTML Forms

John Heusinkveld, MD, Antoine Geissbuhler, MD, David Sheshelidze, MD, Randolph Miller, MD
Division of Biomedical Informatics, Vanderbilt University Medical Center, Nashville, TN

ABSTRACT

The authors have developed a simple method for specifying rules to be applied to information on HTML forms. This approach allows clinical experts, who lack the programming expertise needed to write CGI scripts, to construct and maintain domain-specific knowledge and ordering capabilities within WizOrder, the order-entry and decision support system used at Vanderbilt Hospital. The clinical knowledge base maintainers use HTML editors to create forms and spreadsheet programs for rule entry. A test environment has been developed which uses Netscape to display forms; the production environment displays forms using an embedded browser.

INTRODUCTION

In order to bring useful decision-support capabilities to the clinician, a Provider Order Entry (POE) system must draw on the knowledge of experts in a wide variety of medical specialties and allied health professions, ranging from hematologists to nutritionists to social workers [1]. The designer of such a system is therefore confronted with the problem of providing a scheme for maintaining this diverse knowledge base; ideally, the system's knowledge about each field should be maintained and updated by experts in that field, with minimal assistance from those responsible for maintaining the core software [2].

Since few clinical experts are also programmers, a means must be found to allow non-programmers to format and present complex, patient-specific information to system end-users. The expert also needs to be able to specify rules for how the system is to present and utilize clinical information for the user. The emergence of HTML as an international standard for Web pages and submission forms has provided a large part of the solution to this problem. Since many tools are available for constructing and editing HTML documents, field experts do not even need to know HTML in order to construct informative documents and even create forms to solicit information from the user. However, an additional mechanism is needed to tell the system how to operate on the data. This includes importing of dynamic patient-specific data into "static" Web forms, and after a form is completed by the user, applying bounds checking, completeness checking, and post-processing rules. CGI scripts can perform these tasks, but writing a CGI script requires knowledge of a programming language such as C or Perl.

Vanderbilt University Medical Center makes extensive use of HTML as a means for presenting and obtaining information to and from users of its POE system, known as WizOrder. The original OS2 version of the WizOrder client incorporated a simple Web browser [3], which was capable of displaying documents and forms in an HTML-subset; the newer (1998-1999) Java-based WizOrder client ("JavaWiz") incorporates ICE, a commercially available class which implements a browser for HTML 3.0.

DESCRIPTION OF RULE SCRIPTING LANGUAGE

In the new environment, the processing of a clinical form containing business rules consists of two phases: the INIT phase, in which specified fields are set to their initial values according to rules that relate "blank" fields to environment variables, and the EDIT phase, in which values form forms completed by WizOrder users are processed with values and actions returned to the "calling" WizOrder program. All rules are specified as belonging to the INIT or EDIT phases.

A rule consists of a phase descriptor, a command, a variable with optional type specifier, an operator, an operand, and an optional connector. The phase descriptor can be either INIT or EDIT. The command can be any of the values listed in table 1. The variable can be any input field named in the form, and the operator can be "=", "<>", >", "<", ">=", "<=", or "TO". The value field contains a string or numeric constant. Connectors include THEN, AND, OR, AND_NOT, and OR_NOT; this field may also be blank A simple example would be as follows:

```
INIT: SET form_title TO "Hello World"
```

This rule belongs to the INIT phase and is therefore applied before the page is displayed, causing a variable named form_title in the relevant HTML form to be initialized to the value "Hello World". A more complex example would be:

```
EDIT: IF input_field = "quit" THEN
        EXIT
```

This rule is applied after a form has been submitted, and states that if the variable input_field contains the value "quit" (presumably because the user entered that value), the processing of that form is finished.

Several scopes for variables are defined. "Local" variables correspond to form fields whose values are not saved between sessions. Persistence variables can be stored and retrieved in different sessions and are divided into two subclasses: patient-specific (PSPV) and user-specific (USPV). PSPVs are tied to the patient identifier and retain their stored values across all sessions relating to one specific patient. USPVs are tied to the user ID and retain their stored values across all sessions involving one specific user. A further class of variable, PUSPV, is tied both to the patient identifier and the user ID and is visible across all sessions involving that patient and that user. Finally, environment variables contain information imported from other systems, such as demographic or laboratory data.

APPLICATION OF METHODOLOGY: BLOOD-PRODUCT ORDERING

One of the early uses of this technology involves enforcing the institutional policies on blood product transfusions. These policies are formulated by the blood bank and Blood Products Committee in accordance with national guidelines for the therapeutic use of blood products. The first version of the rules and forms have been developed by the authors in order to pilot initial development and testing of the approach. At a later time, without knowledge of any programming language, an individual at the blood bank can create or modify an HTML form that will be displayed every time a given blood product is ordered; the physician ordering the blood product fills in such required information as the reason for the transfusing. The individual at the blood bank also uses the simple rules language to specify how the system interprets the contents of these fields. Figure 1 contains a screenshot showing a portion of the form used to order Red Blood Cells (RBCs).

To illustrate the use of the rules language, a fragment of the HTML source code for the page is included in Figure 2, which causes the first input field ("RBC Units to CROSSMATCH/RESERVE:") to be displayed.

By selecting one of the checkboxes, the user assigns a value to the variable PSPV.rbc_adult_crossreser_amt. The rules associated with this page include the following:

```
EDIT:
  IF PSPV.rbc_adult_crossreser_amt >20
THEN DISPLAY WARNING
  "To order more than 20 units of RBCs,
   call the Blood bank for approval"
```

This will cause the page to be redisplayed with the specified warning if the user attempts to order more than 20 units of RBCs. Similar forms and rules are being developed for the ordering of platelets and plasma.

Rules are stored in a tabular form for easy parsing by the program that applies them to the information submitted in HTML forms. An internal symbol table is constructed from the variables in the form; variables designated as PSPVs can be placed in long-term storage so that the next time a given form is displayed in the context of the same patient, the fields on the form will be initialized to their stored values. This feature prevents clinician users from having to repeatedly select the same reason to order the same product. For example, in the case of a patient with ongoing blood loss, the justification for the transfusion ("acute blood loss > 15% of estimated blood volume") need only be selected for the initial transfusion orders; for subsequent orders for the same patient, this will be the default value.

The end result of the interaction between the user and the HTML form is one or more orders which are confirmed by the POE and then released to the relevant hospital units and ancillary services. In the case of the blood products order form, orders may be generated to reserve, crossmatch and transfuse blood products. The use of the form and rules guarantees that all of the required information is provided, and allows easy updating as rules and requirements change.

CREATION OF A TEST ENVIRONMENT FOR RULE DEVELOPMENT

In order to facilitate rule and form development, a test environment was created which utilizes the form-processing modules developed for the WizOrder client. Because the Web browser window implemented within WizOrder is not available in the test environment, forms are displayed using the Netscape browser.

When the test environment starts up, it loads a template for the HTML form to be displayed, and, from a separate file, the rules relating to that form. The rules engine control logic sets the fields in the form to their initial values by applying all the rules with the INIT phase descriptor. Netscape™ is then launched as a separate process to display the form. The user is then able to interact with the form in the Netscape environment; when the user submits the form, it is sent to a socket to which the rules engine control logic is listening, arriving as raw HTTP data which is parsed by the control logic. The values of all variables in the form are extracted and placed in the internal symbol table. After the rules with the EDIT phase descriptor have been applied, a new form is constructed and returned to the browser. This test environment allows forms and rules to be developed and tested on workstations that do not have the WizOrder client installed.

DISCUSSION

The problem of maintaining a distributed knowledge base is by no means unique to the WizOrder system at Vanderbilt. For example, the OPADE system, developed in Europe for outpatient prescribing, incorporates several knowledge bases on several different servers including a drug information server, a patient information server, and a thesaurus server. The drug information server a implements a simple language for specifying rules about prescriptions for the drugs in its knowledge base [4].

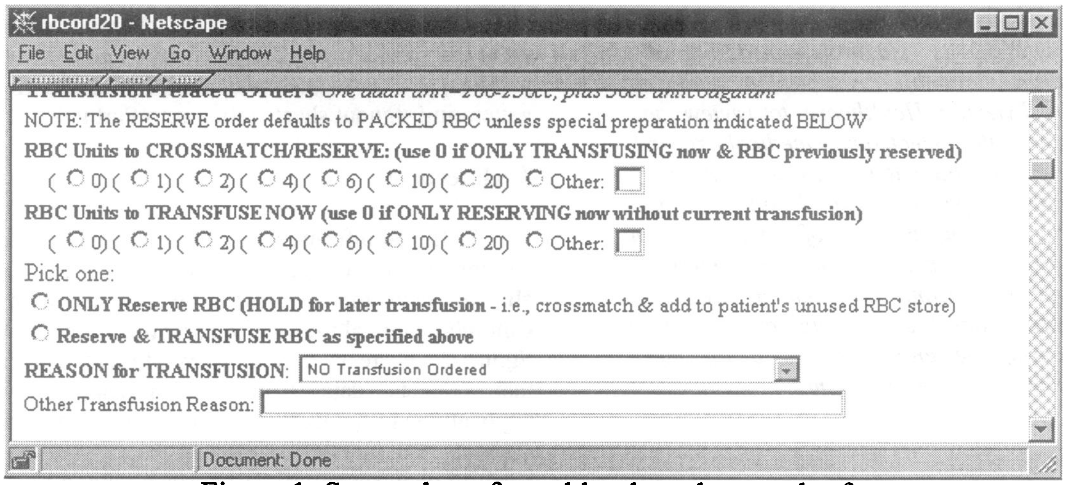
The approach taken here, utilizing HTML forms for static information and a simple rule language for specifying dynamic relationships between different pieces of information, makes no assumptions about what type of information is to be stored and is therefore generalizable to a variety of distributed knowledge base applications. It could be argued that this rule language is in fact a programming language, and that the claim that it obviates the need for clinical domain experts to learn a programming language is therefore inaccurate; however, preliminary experience at Vanderbilt indicates that the amount of time needed to master the rule language is significantly less than that required to learn C or Perl. It seems likely that, in time, commercial products will become available that can perform the same task; however, until a standard emerges, the approach described here will provide an interim solution.

References

1. Geissbuhler A, Miller RA. A new approach to the implementation of direct care-provider order entry. Proc AMIA Annu Fall Symp. 1996; 689-93
2. Geissbuhler A, Miller R. Distributing the knowledge maintenance for a clinical decision support system: the "knowledge library" model. Submitted to AMIA Annu Fall Symp. 1999.
3. Geissbuhler A, Grande JF. Embedding a Web-browser in an order entry system to improve the distributed maintenance of decision-support resources. Proc AMIA Annu Fall Symp. 1997; 939
4. De Zegher I, Venot A, Milstein C et al. OPADE: optimization of drug prescription using advanced informatics. Comput Methods Programs Biomed. 1994; 131-6.

DISPLAY: displays a message. Messages are inserted at the top of the next HTML form to be displayed.
EXIT: exits the current VGR. 11/17: add type=RESTORE (similar to THIS) and **RESTART** (reruns the INIT phase)
IF: evaluates a condition; execution of next rule is contigent upon condition evaluating to true.
LOAD: loads a new VGR or a new form. Local symbols represent the current VGR (current_VGR) and form (current_form).
MAP: maps a local variable to an environment variable. Environment variables cannot be changed directly.
SET: sets the value of a variable
STORE: force a permanent storage of a set of persistent variables.
USE: declares that a persistent variable is going to be used as a variable. If the variable does not exist yet, it is set to FALSE. Changes to variables are saved using the STORE command

Table 1: Scripting Language Commands



NOTE: The RESERVE order defaults to PACKED RBC unless special preparation indicated BELOW
RBC Units to CROSSMATCH/RESERVE: (use 0 if ONLY TRANSFUSING now & RBC previously reserved)
(○ 0)(○ 1)(○ 2)(○ 4)(○ 6)(○ 10)(○ 20) ○ Other: ☐
RBC Units to TRANSFUSE NOW (use 0 if ONLY RESERVING now without current transfusion)
(○ 0)(○ 1)(○ 2)(○ 4)(○ 6)(○ 10)(○ 20) ○ Other: ☐
Pick one:
○ **ONLY Reserve RBC (HOLD for later transfusion** - i.e., crossmatch & add to patient's unused RBC store)
○ **Reserve & TRANSFUSE RBC as specified above**
REASON for TRANSFUSION: NO Transfusion Ordered
Other Transfusion Reason:
Document: Done

Figure 1: Screenshot of part blood products order form

```
<b>RBC Units to CROSSMATCH/RESERVE: (use 0 if
    ONLY TRANSFUSING now &amp; RBC previously reserved)</b></font>
<br>
<font size="2">&nbsp;&nbsp;&nbsp;&nbsp; (
<input type="radio" name="PSPV.rbc_adult_crossreser_amt" value="0">0)
(<input type="radio" name="PSPV.rbc_adult_crossreser_amt" value="1">1)
(<input type="radio" name="PSPV.rbc_adult_crossreser_amt" value="2">2)
(<input type="radio" name="PSPV.rbc_adult_crossreser_amt" value="4">4)
(<input type="radio" name="PSPV.rbc_adult_crossreser_amt" value="6">6)
(<input type="radio" name="PSPV.rbc_adult_crossreser_amt"
  value="10">10)
(<input type="radio" name="PSPV.rbc_adult_crossreser_amt"
  value="20">20)&nbsp;
(<input type="radio" name="PSPV.rbc_adult_crossreser_amt"
  value="other">)Other:&nbsp;
<input type="text" size="2" maxlength="2"
  name="PSPV.rbc_adult_crossreser_text">
```

Figure 2: HTML for part of blood products order form