ED 475 858 EC 309 541

AUTHOR	Smith, Jesse
TITLE	Behavioral Consultant Application. PRISM Project Technical Report.
INSTITUTION	North Dakota Center for Persons with Disabilities, Minot.
SPONS AGENCY	Special Education Programs (ED/OSERS), Washington, DC.
REPORT NO	PRISM-2
PUB DATE	2002-06-00
NOTE	5p.
CONTRACT	H324M000086
AVAILABLE FROM	North Dakota Center for Persons with Disabilities, Minot State University, 500 University Ave. West, Minot, ND 58707. Tel: 701-858-3260; Web site: http://www.msuprism.org.
PUB TYPE	Reports - Descriptive (141)
EDRS PRICE	EDRS Price MF01/PC01 Plus Postage.
DESCRIPTORS	Reflective Teaching; *Behavior Problems; *Classroom Techniques; *Distance Education; Elementary Secondary Education; *Inservice Teacher Education; Internet; *Online Courses; *Peer Teaching; Teaching Models; World Wide Web
IDENTIFIERS	*North Dakota

ABSTRACT

This brief paper describes the Peer Coaching Rural In-Service Model (PRISM) Behavioral Consultant (PBC) program, an online tool for teachers that provides advice on handling simple classroom behavior problems. PBC's advice is based on a series of rules and expressions used by the computer program to make inferences and eliminate inappropriate suggestions. The paper describes the principal technologies involved and then reviews the application's logic. The knowledge base resides in a series of normalized database tables and each table is described in some detail. An overview of the application describes the PBC directory structure and how each of the application files functions. (DB)

PRISM Project Technical Report #2

PRISM Behavioral Consultant Application

Jesse Smith • June 2002

Preparation of this paper was supported in part by a grant from the US Department of Education (#H324M000086) to Minot State University. The opinions expressed here are those of the authors and do not necessarily reflect the policy or opinion of the U.S. Department of Education.

This document is available in alternative formats upon request.

GENERAL OVERVIEW

The PRISM Behavioral Consultant (PBC) is an online tool for teachers. PBC asks questions and provides advice on how to handle simple classroom behavior problems. The PBC uses an adaptive reasoning technique that selects appropriate questions and uses each teacher's answers to determine the best advice in a constrained advice universe. Applications that reason in this manner are also known as expert systems.

PBC's advice is based on a series of rules and expressions that are used by a computer program to make inferences and eliminate inappropriate suggestions. Although the basic process used to arrive at a conclusion is transparent to the user, the apparent simplicity masks the intensive knowledge base operations necessary to achieve a reasonable adaptive outcome. The next section describes PBC's current technologies. The remaining sections provide a detailed description of the application's logic.

TECHNOLOGY OVERVIEW

The PBC works over the Internet by using a back-end database, and Active Server Pages (ASP). It operates on a dual processor server equipped with Windows 2000 Server and SQL Server 2000. The amount of available bandwidth at any given time is around 1270 KB (or the equivalent of a T1 line). A short list PBC's principal technologies is shown below.

Operating System	Windows 2000 Server
Database Server	SQL Server 2000
Scripting Technology	Active Server Pages
Web Server	IIS 5.0

A dual processor server or above is recommended for PBC, but is not necessary. A server running at 850 mhz or above could adequately perform the necessary operations in a timely manner. SQL server software is recommended for performance reasons. Using MS Access or MySQL is not recommended because the use of either one of these databases will introduce significant performance penalties. SQL server software also provides extensive replication control. Since SQL Server software only operates in the Windows environment, Windows 2000 is the best operating system choice to date. Windows 2000 has 'rock solid' stability, and is highly rated by several different companies. ASP and the IIS Web server are great choices for implementing the system over the Internet. ASP supports dynamic calls, and has a sub-language to specifically handle databases known as Active Data Objects or ADO. IIS Web server version 5.0's performance and threading capabilities make it ideal for PBC. The next several sections will describe the tables comprising the PBC knowledgebase and knowledge structures.

PBC KNOWLEDGEBASE OVERVIEW

PBC's knowledgebase resides in a series of normalized database tables. There are 11 tables that make up the PBC knowledgebase. The following sections describe each table, their fields, and each field's purpose in the table.

North Dakota Center for Persons with Disabilities
NDCPD
A University Center of Excellence • Minot State University

NORTH DAKOTA CENTER FOR PERSON WITH DISABILITIES
Minot State University
Minot, ND

Cache_tbl. The cache table records the value assigned to each expression, and it shows how the system assigned that particular value. Staff used this table to analyze PBC's recommendations. Fields in the cache table include:

- *Cach_id* is an auto id field for the table. It gives each record in the table a unique id.
- *Expression* is associated with a concept used in the PBC's rules, facts, or questions. Although an expression might be assigned a value by the answer to a question it is not a shorthand name or reference for a question because it could be assigned a value by a knowledgebase rule or fact.
- *Value* holds the values associated with each expression. The value could come from a rule, fact, or question.
- *Cf* exists to give PBC the ability to accommodate facts, users' answers, and rule-based inferences that are entered with less than 100% certainty. The numeric value in the CF field is determined by a fact, rule or a direct user response. If the question being asked has a certain answer, the CF rating for this value will be 100%. As the value in the CF field goes down, our confidence in that value for this expression goes down.
- *Howcome* is a field that records how the value and its associated CF was derived (fact, rule, question). If a rule inferred the expression value, <expression name>[Rule #]<rule number> is recorded in the field. An example of a string sequence using this format could be "behavior Rule # 2." If a fact was used to infer a value "behavior FACT # 3" might be in the howcome field. If the value was determined by user query, the CF the text string "Because you said so" is recorded in this field.
- *Consult_id* contains the unique number assigned to each consultation.

Consult2_tbl. The consult table records unique information associated with each consultation. Fields in the consult table include:

- *Consult_id* contains the unique number assigned to each consultation.
- *User_id* contains the unique number assigned to each user.
- Time is a time-date field that contains the date and time of the beginning of each consultation.
- Out is a time-date field that contains the date and time of the end of each consultation.

Expression_tbl. The expression table contains fields that hold the critical attributes of each expression used by the knowledgebase. Fields in the expression table include:

- *Exp_id* contains the unique number assigned to each expression.
- *ˋlame* contains the English name of the expression.
- *Mult* is an integer field that identifies an expression as either single-valued or multi-valued. An integer value of 0 indicates the expression can only have one possible value at 100% cf. A value of 1 denotes an expression that can have more that one value at 100% cf .

Fact_tbl. The fact table holds information describing the critical attributes of each knowledgebase fact. Fields in the fact table include:

- *Fact_id* contains the unique number assigned to each fact.
- *Exp_id* contains the id of the expression associated with the fact.
- *Conclusions* is a varchar field containing a value for an expression associated with the exp_id.
- *Cf* is a numeric field that contains an integer value describing the fact-based certainty that the value in the conclusions field is actually the proper value for the expression associated with the id in the Exp_id field.

Getin_tbl. This table contains information associated with gaining access to the system. Fields in getin_tbl include:

- Access_id holds the unique id for an access record. This id is not used anywhere else in the PBC database, but having a unique id for every record in a table is important.
- *User_id* holds the unique id for each user.
- *Login* holds the user's PBC login.
- *Pass* holds the user's PBC password.
- *User_type* is an integer field that is associated with user type.
  A value of 0 is assigned to administrators, 1 is assigned to PRISM staff and 2 is used for all other authorized PBC users.

Quest_tbl. The question table holds the information that ties questions to expressions and display mechanisms. The fields in quest_tbl include:

- *Q_id* holds a unique id for each question and is used to tie entries in the Quest_tbl to entries in Rb_item_tbl and Text_item_tbl.
- *Exp_id* contains the expression id used to connect questions to expressions.
- *Type* is an integer value. 0 represents a radio button question, and 1 represents a text question.

Rb_item_tbl. This table holds the critical attributes of all radio button questions. A radio button question is a question that lets PBC users choose from among multiple, fixed, answers.

- *Rb_id* holds each radio button question's unique id.
- *Q_id* contains the question id and ties radio button entries to the question entries in quest_tbl.
- *Verbiage* contains the question's text.
- *H_v* is an integer value. 0 specifies that the answers for a radio button question will appear in a vertical format and 1 specifies a question whose answers are shown horizontally.
- *Choices* holds the answers associated with each radio button question. This field contains one text string in which the '~' character is used as a delimiter separating each choice. **'Yes~Probably~Probably Not~No'**, is an example of a string in this field.

Rule_tbl. The rule table defines each rule in the knowledgebase. This is a static table that is sometimes checked when PBC's inference engine seeks a value resolve an expression.

- *Rule_id* contains a unique id for each rule in the table.
- *Exp_id* holds the id of the expression concluded by this rule. Exp_id ties the record to Exp_tbl.
- *Stuff* contains a text string that uses delimiters to specify the conditions of the rule. All entries in stuff follow this syntax: **<expression>[˜]<relational operator>[˜]<expression response or answer> [˜]<certainty factor>[˜]<Boolean operator>[| |]<expression> [˜]<relational operator>[˜]<expression response or answer>[˜]<certainty factor>[˜][last][| |].**
- *Conclusions* contains the value assigned to the expression of interest if the rule succeeds.

School_tbl. The School table describes registered PRISM schools. Each user is tied to a school.

- *School_id* contains each school's unique id.
- *Name* holds the name of the school.
- *Address* holds the address of the school.
- *Phone* holds the phone of the school.
- *Contact_person* holds the name of the main contact person at the school.
- *City* holds the name of the school's city.
- *State* holds the name of the school's state.
- *Zip* holds the school's zip code.
- *Need_to-contact* has an integer value. 0 indicates the contact person at the school does not need to be contacted. A value of 1 indicates PRISM staff need to call the school's contact person to discuss authorization.
- *Date_contacted* holds the date the school was notified that their access to PBC was authorized.

Text_item_tbl. The Text_tbl holds the critical attributes of all text questions.

- *Text_id* holds a unique id each text question.
- *Q_id* contains the question id and ties text entries to the question entries in quest_tbl.
- *Verbiage* contains the question's text.
- *Box_h* contains an integer value that is actually the HTML height attribute for the <textbox> tag.
- *Box_w* contains an integer value that is actually the HTML width attribute for the <textbox> tag.

User_tbl. The user table holds records describing PBC's registered users.

- *User_id* holds a unique id assigned to each user record.
- *School_id* contains the school id of the user's school.
- *F_name* holds user's first name.
- *L_name* holds user's last name.
- *Needs_help* is an integer field that indicates the help status of each user. An integer value of 0 means that no help is needed. An integer value of 1 means that the user has requested information about his/her login or password.

PBC APPLICATION OVERVIEW

PBC's inference engine uses 17 ASP files and 9 ASP subroutines (modularized ASP code segments). An additional 7 ASP files make up the user membership process and administrative portions of the application. There are also 10 conclusion files. When a recommendation is made, these file are displayed to the user as a Web page. The next section of this paper discusses PBC's directory structure and application files.

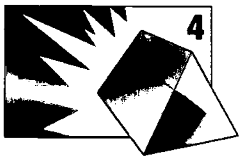
PBC Directory Structure.

I. Expert is the main PBC directory. All PBC application calls for program files are made from this directory. This directory is a subdirectory of the PRISM Web site.
   A. Subs is the directory from which PBC's subroutines are called.
      1. Input is the program file that handles the process of restoring or 'rolling back' a consultation to a previous point. Input is initiated when the user triggers the 'OOPS' function from the PBC user menu. It allows the user to "go back" and answer a question or questions, with different answers that may result in a different recommendation.
   B. Modules contains the web pages for the PBC's recommendations.

Application files: Behind the curtain.

A brief explanation of how each PBC component follows.

- *Login.asp* generates PBC's first user screen. This Web page consists of a welcome message, a login text box, and a password text box.
- *Login.resp* is the response file that handles the user input (login, and password), from the login.asp page. Login.resp takes these values and checks them to see if they are consistent with any of the entries in the getin_tbl's login and pass fields. If a match for both input values is found, the user is granted access to the web page generated by startresp.asp . If there is no match for at least one of these values found in the table, they are returned to the login.asp page with a message telling them they were not authenticated, and asking them to try again or get assistance.
- *Startresp.asp* is used after successful user verification. This file takes the user id, and looks up the school_id. The file then adds a consultation record to the Consult2_tbl and sets up some session variables (variables that can be easily transferred across Web pages). Operation of the application is then transferred to the file goget.asp.



- *Goget.asp* – The goget.asp file is the heart of PBC's inference engine. Goget.asp manages entries in PBC's cache_tbl. Cache_tbl records inferred values for each expression sought by the system during a consultation. Cache_tbl also has a field that contains a record of how each entry was generated (i.e. from facts, rules, or by asking the user). This file also determines the confidence factor associated with each inferred value. Goget.asp is called iteratively and may literally be called thousands of times during a single consultation. The goget.asp determines values to expressions by following four steps. First, it will check in cache_tbl to see if an expression value has already been resolved. Then it will call fact_main.asp, if fact_main.asp does not resolve the expression value, goget.asp calls rules_main. If system rules can not resolve an expression's value, goget.asp queries the user by calling quest_main.asp.
- *Fact_main.asp* is called from goget.asp when an expression being sought by the PBC has an unresolved value. Fact_tbl is queried by this file to see if it holds a fact that assigned a value to the expression of interest. If an expression value is found, insert_cache.asp is called to update cache_tbl. If the certainty factor of the value is less than 100, the expression is still considered to be unresolved and fact_main will be called again to see if fact_tbl contains additional expression-related facts.
- *Rules_main.asp* is called from goget.asp when an expression being sought by the PBC has an unresolved value. Rules_main.asp queries rules_tbl to see if it holds a rule that assigns a value to the expression of interest. Because many expression-related rules can be exhausted before reaching a conclusion with 100% certainty, several calls to rules_main.asp can take place from goget.asp. If an expression value is found, insert_cache.asp is called to update cache_tbl. Rules_main.asp calls to two subroutines (checkrule.asp, and firerule.asp) as it works through each expression-related rule.
- *Checkrule.asp* checks the Cache table to see if the rule has been used before, and if it has, an integer rule status value is generated as a session variable. A failed rule is given a rule status value of 1. If the certainty factor for a rule's success in satisfying a conclusion is greater than 1% and less than 20%, the status is 2. The status value of 3 is assigned to rules generating a certainty factor between 20% and 99% and 4 is assigned to rules that have concluded an expression certainty factor value of 100%.
- *Firerule.asp* 'fires' a rule to see if can find a value for the expression of interest. Since each rule can be associated with multiple expressions, each expression for that rule must be evaluated. Furthermore, each expression may be associated with multiple rules. The expressions associated with a rule are combined in a session string variable called ruledat. Ruledat, is then parsed or separated so that each expression can be checked individually. Once a rule has satisfied an expression, control is returned back to the rules_main.asp file (where it was called from). Rules_main.asp calls an important subroutine stripseek.asp (see below), and then returns control to goget.asp. The cycle the PBC uses in determining a recommendation is either satisfied or starts over for a new question (see above – goget.asp).
- *Quest_main.asp* Quest_main.asp file is called from goget.asp when an expression can not be resolved by using facts and rules. It is the inference engine's "last resort." Questmain.asp manages the tasks associated with question display. Quest_main.asp determines whether the question will be a radio button or text by looking up the type field value in the quest_tbl. If it is a 0, then the file text_display.asp is called, and if the value is a 1, then the file rb_display.asp is called.
- *Insert_Cache.asp creates records in cache_tbl whenever the inference engine resolves a value for an expression or when the inference engine determines that the value of an expression is unknown (based on facts, rules, and user input). The expression as well as its value, certainty factor, and method of determination (fact, rule, or question) are recorded.*
- *Stripseek.asp* is a subroutine called from goget.asp, rules_main.asp, and fact_main.asp. It is usually called just after the Cache table is updated. The session string variable, "seeking", is parsed and reconstructed. The new string generated by this subroutine removes any expressions that have been resolved. Remaining expressions are still sought by goget.asp. Once every expression has been satisfied, this string becomes empty or null. When goget.asp goes to check this string and it is empty, the PBC knows that a recommendation can be made for the consultation. The recommendation is determined by goget.asp by going through each record in the Cache table for the consultation and checking for the highest certainty factor for each expression. The expression with the highest certainty factor becomes the recommendation. The recommendation file tied to the expression is called and displayed to the user. The Cache table is updated to reflect the recommendation made to the user, and the consultation is finished.