RULE BASED SYSTEMS APPLIED TO ONLINE SURVEYS

Paulo Urbano
Faculdade de Ciências da Universidade de Lisboa
C6, Piso 3, Campo Grande 1749-16, Lisboa, Portugal
E-pub@di.fc.ul.pt

David Rodrigues
Instituto Superior das Ciências do Trabalho e da Empresa
Av. das Forças Armadas 1649-026 Lisboa, Portugal
m4467@iscte.pt

ABSTRACT

We describe in this paper De.:Sid, an Intelligent Online Survey program based on rules. We have incorporated three rule knowledge bases in a standard online survey architecture, that are used to control three vital components on a survey: (1) the dependencies between questions and answers, i.e. the structure and survey branching logic, (2) the decision regarding the selection of the next question to be asked and (3) the inconsistency detection between different answers. These rule based components allow us to escape to a predetermined question sequence, achieving flexibility and adaptability to the users answers; not only that, they enhance usability allowing an easy navigation along the different survey questions and the possibility at any moment to backtrack and revise the answers without loosing global coherence, finally there is an explicit treatment of inconsistent situations by exposing and explaining them and inviting the user to revise its answers. De.:Sid benefits from the qualities generally associated with rules: (1) separation from other system elements (database, middleware and web graphical user interfaces) allowing its explicit management, reusability and independent modification, (2) externalization which allows everyone to know and understand the decision making process, and finally (3) to enable easy and quick rule changes tracing the change impact.

KEYWORDS

Rule-based Systems, Intelligent Online Survey, Knowledge base.

1. INTRODUCTION

In the context of a scientific project we had to create an online survey. This survey was directed towards the enterprise world and the goal was to study the impact of design as a strategic resource. When we started we had two possible choices: use one of the commercial online survey tools (SmartMonkey, Ideascale, Questionpro, Questionform, etc..) that allow us to build and manage online surveys, or (2) build our own system. After analyzing the existent tools we choose the second option: to build our own system. The reasons behind our final decision were the limitations we found on those tools: the most relevant of them is that they force a predetermined question sequence, which is not sophisticated enough. In general, these systems, that use very similar techniques, the branching logic that manages the dependencies between questions and answers is not separated from the order of question presentation. We can exemplify with a typical ramification rule: if question 17 has answer = “yes” then jump to question 45, otherwise continue with question 18. This promiscuity between these two aspects is not ideal in case we want to change our survey. If we want to modify a question identification number or the skip and branching logic, we have to propagate the modifications by hand to other survey sections (the questions affected by the modification). Besides that, there is a compromise with the question ordering. The fact that a certain question is valid because a certain answer was saved should not imply that it should be the next question to be presented. We think that the branching logic and question selection should be independent and clearly separated to improve usability and the ability to easily modify the survey.

The fact that we wish to express the questions ramification logic does not inhibit us from establishing more sophisticated criteria regarding the question selection process with implication in the presentation order of questions. With the commercial systems we cannot select a question based on the particular answer history of a particular user. It is not also possible to express the preference for questions that were presented less often—note that this imply the possibility to postpone an answer and to present the same not yet answered question several times. Also the fact that a user has just logged in can be a source of information for choosing next question—we suppose here that the survey can be answered along different sessions. In resume, we want that the survey narrative (order of questions presentation) may adapt to user answer profiles and avoid having a predetermined narrative prepared offline.

One of our goals was also to provide the functionality to define the combination of inconsistent answers, warning the users about those situations and explaining them, inviting them to revise their incoherent answers, without inhibiting them from continuing to answer the survey. Moreover, it would also be interesting in terms of usability that users could freely navigate along the different questions and that they could answer postponed questions or revise answers whenever they wish, without loosing the survey coherence.

We also arrived to the conclusion that it would be important that the knowledge about the survey's skip and branching logic, inconsistency check and next question selection criteria should be separated in independent modules. We would have externalization of knowledge and decisive criteria used on surveys and the possibility to refine and improve them without touching other system components. The use of rule knowledge bases (Jackson—1997) seems to be the natural tool to represent the structure, inconsistency relations between answers and question selection criteria. Moreover, rules independency, modularity and expressiveness enable easy and quick modification.

"Rule-based Systems represent knowledge in terms of multiple rules that specify what should be concluded in different situations. A rule-based system consists of IF-THEN rules, facts and an interpreter that controls which rule is invoked depending on the facts in working memory." (Giarratano and Riley—2004). There are two types of interpreting rules: forward chaining and backward chaining. A forward chaining inference engine is data-driven and starts with a set of facts and tries to draw new facts by applying the rules until no more rules are applied. The backward chaining rule is goal-driven, i.e., tries to prove a certain hypothesis and uses the rules in order to prove it (Debenham—1998). Rule-base systems have several good properties: modularity which enables incremental development, externalization and explanation facilities by exposing the inference trace, and they may represent and model human knowledge (Giarratano and Riley—2004). Recently we face an excitement on rule based systems with the new application and research area of Business Rules Management (Halle—2001,Grahan—2006)

From the point of view of who designs the questionnaire, D.:Sid architecture provides three knowledge bases to be filled in by rules specific to the questionnaire in question. In the first one, we can define the structure and branching logic of the survey; in the second one, we may define the criteria regarding the selection of the next question to be posed to the user, which we think is the most innovative aspect of our work and, in the third one, we may express which combination of answers are declared to be inconsistent along with the respective inconsistencies explanations to be presented to the user. We have in fact the possibility not only to define the knowledge regarding these three aspects of the survey but also the ability to revise and improve or adapt them without touching the other architecture elements, due to the independence of these knowledge bases.

Rules were defined in CLIPS, a multi-paradigm programming language developed by NASA that provides support for rule-based, object-oriented and procedural programming. We have just used the rule-based facet of Clips. The access to CLIPS is made by a PHP extension named PHLIPS, which functions as an interface between the web application and the rule engine. We have the power to express in our rules whatever is possible using CLIPS rules and so we do not have a restricted syntax and semantics besides CLIPS. CLIPS has a forward chaining rule engine.

In the next section we give background information pertinent to the motivation and design of an intelligent online survey program, specially we will focus on the typical ways to deal with survey's skip and branching logic. In section 3 we present our main contributions, in particular how the online survey D.:Sid allows a free navigation along questions, how we can design rules to select the next question to be posed based on the user answer history and its abilities to provide backtracking and answer revision without loosing integrity and how to express and treat inconsistency; In sections 2, 3, 4 and 5 we will describe respectively the dependency between questions and answers, the bactracking process allowing answer revision, the

decision rules to decide next question to be posed and inconsistency detection. In section 6 we will describe the architecture of D.:Sid, its different components and organization. Before completing, we will focus in the interaction between the web server and the rule system controller and finally we present our conclusions and point future directions for possible improvements.

2. SKIP AND BRANCHING LOGIC

One of the most important commercial online survey characteristics is the skip and branching logic. Skip and branching logic allows that users do not have to see the whole survey and skip some of the questions, following customized survey paths. For example, it is only meaningful to ask to which countries an enterprise exports its products in case it sells products to foreign countries. This is the role of skip and branching logic, to create different sequence of questions depending on the user answer's profile.

In general, the existent commercial software systems for creating web surveys provide the ability to define the skip and branching logic of questionnaires. We can declare, for example, that if the answer to question numbered 18 was "yes" then jump to question 45 or skip directly to question 46 independently of the answer to question 38. The skip and branching functionalities do not allow to skip or to branch into a previous question, which makes sense in order to avoid circular surveys. These systems allow only the specification of a skip or branching question in case of a particular answer. This is positive because it allows the survey customization to users: highly sensitive and adapted to the nature of answers. But note that the way the skip and branching logic is declared forces the designer to establish a predetermined and fixed questionnaire narrative. There is clearly a strong promiscuity between the dependencies between questions and the order in the question sequence as shown by the following examples: question 18 is always posed after answer to 17 or question 25 is always posed after an answer "no" to 19.

Consider now that we want to introduce a new question and modify the questions identification numbers, or that we want do select randomly any question from a specific group of questions. Or that we know that question 19 makes only sense after a particular answer to question 15 but we do not want to select it immediately because it is hard to answer and perhaps we would like to put a lighter one. Or even suppose that we want to allow the user to postpone his answers to certain questions but continue the survey. All these examples show that these systems are not flexible enough and that we need to separate what is the domain of dependencies between questions and answers (the survey structure) and the next question selection process, in order to change easily the questionnaire narrative, i.e, the questions sequence. We are in face of two related processes, because the dependencies between questions and answers define which questions are valid and can be asked but the influence should stop here without a precise compromise on the question ordering. We will speak about the question selection process later in section 3.

This way, we need a way to express the dependencies between questions and answers without any compromise what so ever with the order of questions achieving a clear separation between the skip and branching logic and the questions presentation order. We have rules that express the dependencies between questions and answers, but in opposition with standard online survey systems we can have not only but several questions that depend on a particular combination of answers or absence of answers. As an illustration we show a rule translated in natural language:

Rule1: ***if*** *there is answer 2a* ***or*** *2b to question 2* ***and*** *answer 15a* ***or*** *15b to question 15* ***then*** *the set of questions {18,19,20,21,22} is valid.*

We can also have facts that assert that questions from 1 to 15 are unconditionally valid ones and can therefore be selected to any user. These are non-customized questions that will be presented to any kind of user, like for example, the name, the address, etc.

The survey structure is created without any compromise regarding the precise question order—in principle any question form the valid set can be posed. After answer 2a to question 2 and answer 15b to question 15 the survey is not obliged to "jump" to question 18 or any another from the valid set. What the rule expresses are the conditions for enabling the selection of a particular set of questions, which will be eventually selected except in case the user interrupts the survey or revise any of the answers to question 2 and 15, inhibiting these questions.

In resume, we have a set of rules that express the dependencies between questions and answers and which establish at each moment the survey territory, i.e. the set of questions that may be posed. In the survey territory we can find questions already with answers, questions that were posed but not answered and never selected questions. From a different perspective, these rules inhibit some questions that are invalid and thus cannot appear in the questionnaire. Of course they can also inhibit answers to questions that were presented to the user but due to a revision process are not meaningful (valid) anymore.

The inference engine that controls the dependencies knowledge base, made of facts and rules, calculates which are the valid questions, updating the survey territory in accordance with the user answer profile.

3. BACKTRACKING FOR ANSWER REVISION AND ITS CONSEQUENCES

One of the critics to the online survey systems is the impossibility to navigate freely along the questionnaire, to choose and reanalyze already answered questions and to revise their answers. We think that is very practical to have the functionality of jumping to any question but only if the question is valid and it if was already presented to the user. We think that user navigation freedom should be given but with some limits: we should not allow the user to anticipate questions and jump to a valid question never presented to him. The question selection module has the responsibility to choose the right never posed question. So, to improve usability, in our system the user can postpone an answer due, for example, to the absence of data and he may return to that same question later to fill it. Users can even return to a particular answer of a question and he may revise that answer. We have to be careful because answer revision can be problematic since changes in the survey answers may implicate the modification of the survey landscape: it may invalidate some questions and validate others, i.e. it may change the ramification state of the inquiry. We can think on a situation where a certain question had its answer revised and that question is a ramification node. The previous answer has taken the survey into a particular branch and path, which is going to be changed by the new answer.

So whenever there the user answers to a particular question and that answer is new (is really new or is a revised answer) the chain of dependencies between questions and answers is recalculated to determine which questions are valid and invalid. The module described in the previous section is executed whenever there is a new answer and the survey coherence is always guaranteed. Even if there are answers to questions that become invalid due to a revision process, these answers are kept but are no longer relevant. We just keep these no longer valid answers just in case there will be a new revision that will support them again.

An alternative solution, more sophisticated and efficient would consist in using a Truth Maintenance System (TMS) (Doyle-1977) that would recalculate just the questions that are directly or indirectly implicated by the new answer. For that to happen the Inference engine or rule controller must be active during the different login sessions, keeping the state of the dependencies between active questions and answers and that is not possible in our system because of the way PHLIPS interacts with the web server. In our case, the inference engine is restarted every time the uses answers or postpones an answer and that means that the dependencies chain has to be recalculated in a brute force way. Our solution is admissible for surveys with small dimension and it's really the case: our questionnaire has 53 questions. David Crighton in the ERA4 (Crighton-2005) used a TMS in order to backtrack and revise a questionnaire for an online Electronic Referee Assistant.

4. CHECKING AND TREATING INCONSISTENCIES

It is normal to ask questions in a survey in which the answers can be inconsistent. Questions are created by different individuals and it is not always neither possible nor desirable to "filter" the survey in order to avoid obtaining inconsistent answers. Therefore, we need a module that checks for inconsistent answers. We need also to expose, modify and evolve our knowledge about what information is not compatible. A rule base system was again the ideal tool to achieve that goal. It allows to (1) separate inconsistency knowledge making it independent from data and from programming code; (2) turns it explicit and understandable and (3) it is easy to modify without disturbing the other survey architecture components. In fact, the three big rule

system qualities. We give two rule examples from the design survey. The rules are written in CLIPS but were translated to natural language for better comprehension.

InconsistencyRule1: If the enterprise does not look to other competitors (answer to question 27) and know the contribution of design in the market leaders (answer to question 29) then there is an inconsistency between rules 27 and 29.

InconsistencyRule2: If the enterprise is pro-active (answer to question 18) and does not pay attention to both market and consumer (answer to question 28) then there is an inconsistency between questions 18 and 28.

In our case we have decided to warn the user as soon as an inconsistency appears for the first time. Each warning is associated with an explanation for the inconsistency—they can jump into the questions involved and solve the inconsistency revising the involved answers. But if they wish they can persist with their answers. So, the users are warned about the different inconsistencies but may continue answering their surveys—questions involved in inconsistencies are marked. Just before closing their surveys users with inconsistencies are again warned and can revise their answers or decide to maintain them.

5. SURVEY NARRATIVE: RULES TO SELECT THE NEXT QUESTION TO BE POSED

We will describe now the next question selection module. Remember that the question chosen have to be valid and can be completely new or an old question that was left unanswered. Note that validity is assessed by the rules discussed in section 2. As we told before we want flexibility in what concerns the design of question sequence (survey narrative) and adaptability concerning users answer profiles. We want to escape to a monolithic narrative completely defined a priori. To accomplish this goal we will again use a set of rules that express the criteria used to select the next question to be presented to the user. The criteria can be whatever the questionnaire designers wish. Again, our idea is to profit from the good qualities of rules: externalization, modularity, explanation and position for change.

The next rule selection can depend on a set of elements like the existence of certain answers to certain questions, the fact that the user has just made login, the different qualities of questions in terms of difficulty or the particular user history. We are going to present two rule examples that we have used in D.:Sid, translated from CLIPS to natural language.

SelectionRule1: If the user has just made login and it is the first login then choose question 1

SelectionRule5: If the user has just made login and it is not the first login then choose a valid question that was presented before but not answered and different from the last presented but was presented less than three times

Note that these rules imply that answers can be postponed and that we count the number of times they were presented without being answered. They also imply that we know if the user has just made a login and that we register the number of logins. We have also to register which was the precedent question. All this information can be accessed and used in D.:Sid.

We could easily rewrite the first selection rule in such a way that any of the 5 first questions could be selected.

NewSelectionRule1: If the user has just made login and it is the first login then choose randomly one of the questions in set {1,2,3,4,5}

We could also, for instance, classify the questions as “hard” or “light” questions and define rules that try to avoid presenting two consecutive hard questions. Or we could try to present the hard questions in the

middle of the questionnaire avoiding presenting hard questions both in the beginning and in the end of the survey as it is normally suggested.

The advantages of using narrative rules are obvious: everybody can know them and can easily change and refine them in an independent way. The advantages of separation branching logic rules and narrative rules seem also obvious: improving flexibility, usability and adaptability.

6. ONLINE SURVEY ARCHITECTURE

Our first option was a web application survey because we thought it would be (1) easier for users; (2) easier for us to compile and treat information and (3) we could also trace and follow the users history, as they were answering their surveys. The decision to design a web application forced us to think in several modules. From several possibilities we decided to use a stack LAMP (Linux, Apache, MySQL e PHP) for project support and development. We implemented a BackOffice for data base management and a mailing system to communicate with survey potential users. We installed and configured an electronic mail server—we choose Postfix. The survey was developed in PHP, and we used the extension/module PHLIPS to mediate between PHP and CLIPS.

The architecture diagram is depicted in figure 1.

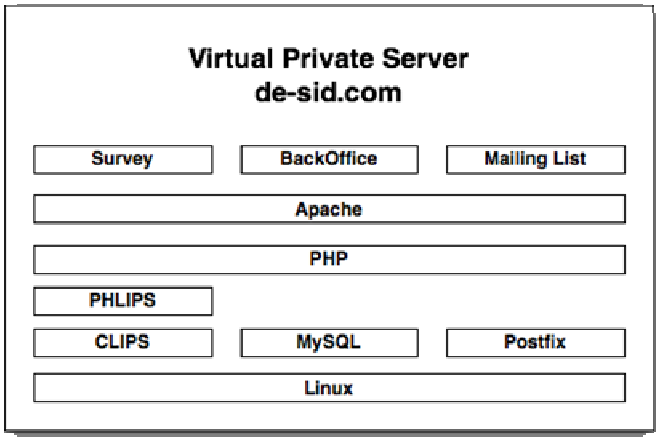

Figure 1 – De-Sid Implementation Diagram

7. INTERACTION WITH THE RULE BASED SYSTEM

We have seen in the last section that the PHLIPS module is the interface between the Web application (the online survey) and the inference engine (CLIPS). The interaction between theses two levels is done whenever it is necessary to decide which question should be presented to the user. This happens when the user makes login, postpones an answer to a question or saves an answer. So whenever one of these situations occurs it is necessary to send the answer profile to the inference engine and it will load the three knowledge bases and do its job that culminates in the next question selection and in inconsistencies detection. This task can be summarized in the following sequence of actions:

1. The user executes some action that changes his answer profile or makes a login
2. The rule engine is started
3. The 3 knowledge bases are loaded.
4. The user answer profile is sent to the rule engine as a set of facts.

5. The dependencies module is executed.
6. The inconsistency module is executed.
7. The process of next question selection is executed
8. The output of the rule engine is treated by the web application. This output is composed of (1) valid questions, (2) sets of inconsistent answers and (3) the next question to be presented to the user.

Data persistence is assured by a MySQL database in the BackOffice of the Web application. The rule engine is activated each time there is a call to decide what is the next question to be presented to the user. This means that all user survey data must be passed from the MySQL to CLIPS through PHLIPS. It would not be possible to execute several rule engine instances, waiting for users actions and so our option to restart CLIPS engine every time.

8. CONCLUSION AND FUTURE WORK

We have applied Artificial Intelligence techniques, namely rule-based systems to online survey systems with the goal of improving their usability, transparency, flexibility, user adaptability and also their ability to modification. We have used three rule knowledge bases for expressing the knowledge associated with three important components of survey systems: the dependencies between questions and answers (skip and branching logic), inconsistency detection and the decision regarding the questions selection (survey narrative). We think that online survey systems benefit from the introduction and separation of these modules, from their independence, and from the externalization of all criteria involved in the survey process.

We have applied the rule-based systems to a specific survey directed to enterprises in order to study the impact of design, but the architecture can be generalized to any survey. Our conclusions are based in our own experiences with D.:Sid. We have not yet evaluated our system in a real setting because we have just launched the survey to a population of 15 enterprises and soon we will launch it for a wider population (around 1500 enterprises). After the survey is online we want to collect feedback from user in order to evolve and refine the right question selection rules that can help users finish their questionnaires with success.

We want to incorporate in the future the possibility to improve our brute force interaction with the rule engine, saving state and incorporating a Truth Maintenance System in order to improve our backtracking and revision process.

We have the project of applying multi-agent systems to online survey systems. The idea is to have a survey controlled by agents responsible to present and control the survey to each user; they would at certain points in time syncronize with the central server and report the results. Each agent would manage a particular survey in a local way.

AKNOWLEDGMENTS

We would like to thank all members of De.:Sid (http://www.de-sid.com).
We thank Science and Technology Foundation (FCT) that funded the research project with the reference PTDC/AUR/70607/2006 and the title “Design as a company’s strategic resource: a study of the impacts of Design”.

REFERENCES

Halle, B. von ,2002. *Business Rules Applied*. New York: Wiley.

Crighton D., 2005. Adding a Truth Maintenance System to ERA, the Electronic Referee Assistant, to allow backtracking. Master of Science, University of Edimburgh.

Debenham, J., K. 1989. *Knowledge Systems Design.* Prentice Hall.

Doyle, J. 1977. Truth Maintenance Systems for problem solving. *5th International Joint Conference on Artificial Intelligence,* 1977.

Giarratano, J., Riley, G., 2004. *Expert Systems: principles and programming*. Course Technology.

Jackson, P. 1998. *Introduction to Expert Systems.* Addison Wesley.

Graham, I. 2006. *Business Rules Management and Service Oriented Architecture: a pattern language.* New York: Wiley.

Ross, R. G. 1997. *The Business Rule Book.* (2nd ed.). Houston: Business Rule Solutions, LLC.

CLIPS *Expert System Shell* <http://www.ghg.net/clips/CLIPS.html>.

IdeaScale <http://www.ideascale.com/>

PHLIPS *PHP extension for CLIPS.* <http://phlips.sourceforge.net/>

QuestionPro <http://www.questionpro.com/>

Questionform <http://questionform.com/>

SurveyMonkey <http://www.surveymonkey.com/>