Advantages of Using Rules in online SURVEYS

Paulo Urbano
LabMag - Faculdade de Ciências da Universidade de Lisboa
pub@di.fc.ul.pt

Abstract

In this paper we present *De.:SID*, a rule-based Intelligent Online Survey program. 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, 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 answers to different questions. These rule-based components allow us to escape a predetermined question sequence, achieving flexibility and adaptability to the user's answers; besides, they enhance usability allowing an easy navigation along the different survey questions and the possibility to backtrack and revise the answers, at any moment, without loosing global coherence. There is an explicit treatment of inconsistent situations by exposing them and inviting the user to revise his answers. *De.:SID* benefits from the qualities that are generally associated with rule-based systems: (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) easy modification of rules that are modular and can be easily deleted, inserted or changed.

Keywords: Rule-based Systems, Intelligent Online Survey, Knowledge base, Expert Systems, Inconsistency Detection.

Resumo

Neste artigo, descrevemos o *De.:SID*, uma aplicação dos Sistemas Baseados em Regras a um programa de inquérito "online" inteligente. Incorporámos no programa três bases de conhecimento baseadas em regras que controlam três componentes vitais num inquérito: (1) as dependências entre as várias perguntas e as respostas, ou seja a própria estrutura e a lógica de ramificação do inquérito; (2) a decisão quanto à selecção da próxima pergunta a ser colocada, e (3) a detecção de inconsistências entre as várias respostas. Estes componentes permitem que não se imponha uma ordem pré-determinada na colocação das perguntas, dotando o sistema de flexibilidade e capacidade de adaptação às respostas do utilizador; permitem também uma fácil e livre navegabilidade ao longo das várias perguntas e a possibilidade da revisão das respostas sem perder a coerência do inquérito. As situações de inconsistência nas respostas podem ser representadas, expondo as respostas envolvidas e convidando o utilizador a rever uma dessas respostas. *De.:SID* beneficia das qualidades que estão normalmente associadas ao uso de regras: (1) a separação e independência face aos restante componentes do sistema, (2) a exposição e a transparência dos processos de tomada de decisões associados ao inquérito, e, (3) a rápida e fácil mudança das regras devido à sua modularidade.

Palavras chave: Sistemas Baseados em Regras, Inquéritos “Online” Inteligentes, Bases de Conhecimento, Sistemas Periciais, Detecção de Inconsistências.

1 Introduction

In the context of the scientific research project De.:SID [Romão et al. 2007] we had to create an online survey. This survey was directed to the enterprise world and the goal was to study the impact of design as a strategic resource in Portuguese manufacturing industry. When we started we had two possible choices: use one of the commercial online survey tools that allow us to build and manage online surveys, or (2) build our own system. Online survey tools provide the resources to design survey questionnaires and enable you to collect, organize and analyze survey results all in one single web browser. The respondents complete the surveys over the web and store the results on their computers. There are a variety of self- or full-service online survey tools available. After analyzing the existent tools [NPowerNY n.d., Marra and Bogue 2006] we chose the second option: to build our own system [Urbano and Rodrigues 2008]. The reasons behind our final decision were the limitations we found on those tools: the most relevant of them being that the skip and branching logic that manages the dependencies between questions is not clearly separated from the process of question selection. This issue was vital to us because we wanted to make dynamic questionnaires that were adapted to the respondents. We can exemplify with a typical linear ramification rule: if question 17 has an answer = “yes” then jump to question 45, otherwise continue with question 18. The rule above illustrates very well the confusion between these two aspects. The fact that a certain question is eligible because a certain answer was selected should not imply that it should be the next question to be presented. We think that branching logic and question selection should be independent and clearly separated to improve usability, customization, questionnaire adaptation to users and also the ability to easily modify the survey. Even online survey tools that allow complex and versatile conditional branching, like WWW Survey Assistant [S-Ware 1996-1998] Websurveyor [WebSurveyor Corporation 2002] Sawtooth [Sawtooth Technologies 2002] or OnQ [Pargas et al. 2003] suffer from this limitation. For example, with OnQ users can form logical subgroups of questions called question blocks and specify the order in which the question blocks should be presented, by creating Boolean expression-controlled transitions between pairs of blocks. OnQ uses Deterministic Finite Automata [Linz 1996] to model transitioning between blocks.

The fact that we wish to express the questions ramification logic should not inhibit us from establishing more sophisticated criteria regarding the question selection process with implications in the questions presentation order. With the commercial systems, mentioned above, we cannot select a question based on the particular user answer history. It is not also possible to express the preference for questions that were presented less often—note that this implies the possibility to postpone an answer and to present the same question, not yet answered, several times. Also the fact that a user has just logged in can be a source of information for selecting a question—we suppose here that the survey can be answered along different sessions. We want that the survey narrative (order of presented questions) may adapt to the user answer history and avoid having a predetermined narrative prepared offline. Some of the available tools allow question randomization but they do not allow the survey author to define explicit criteria for the next question selection that can take into account the user answer history. It is the case of OnQ, Sawtooth Websensus, WWW Survey Assistant and Hosted Survey [Hostedware Corporation 2002].

One of our goals was also to provide the functionality of defining the combination of inconsistent answers, warning the user about those situations, describing the answers involved and inviting him to revise his incoherent answers, without inhibiting him from continuing to answer the survey. Moreover, it would be also 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 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 make explicit (externalize) all knowledge and decisive criteria used on surveys and we would also able to refine and change 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 dependencies between different questions, inconsistency relations between answers and next question selection criteria. Moreover, rule independency, modularity and expressiveness enable easy and quick modification.

Rule-based Systems have been extensively used in applications like bank loans authorization, credit card emission and fraud detection, medical diagnosis, etc. [Winston and Prendergast 1984, Edwards and Connel 1989, Forsyth 1984, Harmon 1985, Waterman, 1986]. "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 a set of facts and IF-THEN rules, 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 have been facing an enthusiasm on rule-base systems with the new application and research area of Business Rules Management [Ross 1997, Halle 2001, Graham 2006].

From the point of view of who designs the questionnaire, *De.:SID* architecture provides three knowledge bases to be filled in by rules specific to the questionnaire in question. In the first one, we define the structure and branching logic of the survey; in the second one, we 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 express which combination of answers are declared to be inconsistent along with the respective inconsistency warnings to be presented to the user. We have in fact the possibility not only to define 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.

The three rule knowledge bases were defined in CLIPS [Riley 2001], 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. PHP is used as the web server-side scripting environment and the access to CLIPS is made by a PHP extension named PHLIPS [The PHLIPS Project 2004, Yang 2006], which functions as an interface between the web application and the rule engine. Using CLIPS there is no limitation in terms of what we can express but demands competence on CLIPS programming. CLIPS uses 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. We will focus on the typical ways to deal with survey's branching logic. In section 3, we present how the online survey *De.:SID* allows a free navigation along questions enabling answer revision. In section 4, we describe the rule-based expert system we have designed to cope with the selection of the next question to be presented to the user based on the user answer history. In section 5, we will describe the module designed to deal with answer inconsistencies. Finally, we present our conclusions and point future directions for possible improvements.

2 Branching Logic

One of the most important commercial online survey characteristics is the skip and branching logic. Skip and branching logic allow that a user does not have to see the whole survey and skip some of the questions as they only make sense after particular answers have been given. By providing only one question at a time in successive order, the questionnaire will pose only relevant questions associated with the chosen answer, following different 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, that is, to create different sequences of questions depending on the user answer's history.

In general, the existing 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. But note that the way the skip and branching logic is declared forces the designer to establish a predetermined and fixed questionnaire narrative, which has a decision tree structure. 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 "yes" to 17 or question 25 is always posed after an answer "no" to 19.

Suppose now that users are able to skip answers and come back to them later. When should these unanswered questions be posed to the users? It will be hard to make a decision tree cope with every possible situation. In case the decision tree shares some parts, (where the same questions appear in different paths of the tree), it would be also hard to decide which question to pose after a certain answer because there is a context dependency. Anyhow, we may just want to express dependencies between questions and without any commitment whatsoever with the precise order of questions. We know that question 19 makes only sense after a particular answer to question 15 but we do not want to select it immediately after the answer to question 15 was asserted because perhaps there is some higher priority eligible question. All these examples show that these systems are not flexible enough and that we need to separate what is the domain of representation and maintenance of dependencies between questions (the survey structure) and the next question selection process. We are faced with two related processes, because the dependencies between questions define which questions are valid or enabled but the relationship should stop here without a precise compromise on the question ordering. In this section we will deal with survey structure declaration. We will speak later about the question selection process, in section 4.

This way, we need a way to express the dependencies between questions and answers. We have facts and rules that express the dependencies between questions. From now on we will use the notion of valid question to define an eligible question. A valid question is a question that can be posed to the survey user.

It is not our intention in this paper to go deep into the CLIPS code, but anyhow we are going to show some CLIPS facts and rules. To indicate that a question is valid, a CLIPS fact is asserted in the working memory: `(question QuestionID)`. If question with ID 25 is valid we should find the following fact (question 25) in the working memory. In order to register answers we have facts with the template `(answer Field Answer QuestionID)`. Note that we can have answered questions that were valid before but that are not valid anymore, due to some answer revision.

There are questions whose validity is independent of any answer and thus we do not need any rules to declare that they are valid, we just declare them valid using CLIPS facts. For example, we can declare, using facts, that questions 1 to 4 are unconditionally valid ones. These are questions that will be presented to any kind of user, like for example, asking the name, the address, etc. For that, we use the following `deffacts` construct that asserts the initial facts after a `reset` operation.

```
(deffacts InitialQuestions
  (question 1)
  (question 2)
  (question 3)
  (question 4))
```

There are other questions that are only eligible (valid) in presence of a certain combination of answers to currently valid questions. In that case, we express the conditions for validity as rules where the rule consequent is rule validity and rule conditions are the prerequisites for validity. Using rules we have the possibility of declaring whatever we wish to define validity and we do not want to commit to any formal definition of validity. Note that we can have several questions whose validity depends on a particular combination of answers or absence of answers. It is usual in surveys to have some sections composed of several questions that only make sense only in case there is a combination of certain 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* ***and*** *question 2 is valid* ***and*** *question 15 is valid* ***then*** *all the questions in the set {18,19,20,24,26} are valid.*

Translated in CLIPS this rule will be the following:

```
    (defrule rule1
        (question 2) (question 15)
        (or (answer “1” 2a 2) (answer “1” 2b 2))
        (or (answer “1” 15a 15) (answer “1” 15b 15))
  =>    (assert (question 18) (question 19) (question 20) (question 24) (question26)))
```

Let’s see the versatility of our rules. We can say, for example, that a certain question is valid whenever there is an answer to a particular valid question (any answer).

*Rule5: **if** there is any answer to question 15 **and** question 15 is valid **then** question 51 is valid.*

We can also say, for example, that a certain question is valid whenever there is not a particular answer to a particular valid question.

*Rule8: **if** there is an answer to question 15 different from A **and** question 15 is valid **then** question 51 is valid.*

There are questions that should be asked in sequence. So, we may want to express some strong dependency between certain questions. For example, if we want to ask if a certain company exports (question 10) and after we want to know to which countries it exports (question 11), we may declare that these two questions are strongly related and that question 10 should be followed by question 11, in case a certain answer was asserted. We are not forcing that question 11 always follows question 10, but we are expressing the strong dependency and that the module responsible for question selection can take it into account or not.

To express this special kind of dependencies we have to assert CLIPS facts like:

```
    (dependency 51 “1” "yes" 52)
```

meaning, in this case, that question 52 should follow question 51 in case the answer to field “1” was “yes”.

Even so, you should note that this is just information regarding a strong dependency between a certain answer and a question. The survey structure is created without any compromise regarding the precise question order—in principle, any question from the valid set can be posed. What the dependency rules express are the conditions for enabling the selection of a particular set of questions, which will be eventually selected except in the case the user interrupts the survey or revises any of the answers changing the validity state of questions.

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 valid questions that may be posed. In the survey territory we can find questions already with answers, questions that were posed but not answered and not yet selected questions. Outside of this territory are the invalid questions. From a different perspective, 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 (they were valid and were selected) but that due to a revision process are not valid anymore). We will speak about revision in the next section.

After some user action, the inference engine that controls the dependencies knowledge base, which composed by facts and rules, infers which questions are valid, by updating the survey territory in accordance with the user answer history.

3 Answer Revision

It's essential to navigate freely along the questionnaire, to choose questions that were not answered, to choose and reanalyze questions that were already answered and to review their answers. We think that user's freedom of navigation should be given but within certain limits: we should not allow the user to anticipate questions and jump to a valid question that was never presented to him. The question selection module has the responsibility to choose the next question to be posed. So, to improve usability, the user can postpone an answer that was due, for instance, to the absence of data and return to that same question later to fill it in. Therefore, the user may return to a particular answer of a question that was left unanswered or review a particular answer. This way, he can choose to jump to a valid question that was presented before.

We have to be careful because answer reviews can be problematic since changes in the answers survey may imply the modification of the survey landscape: it may invalidate some questions and validate others, i.e. it may change the survey ramification state. For instance, we can think of a situation where a certain question had its answer reviewed and some questions depend on that answer. 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 the user answers a particular question and that answer is new (is really new or a reviewed answer) the chain of dependencies between questions is recalculated to determine which questions are valid and invalid. The module described in the previous section is executed whenever a new answer is inserted, and so the survey coherence is always guaranteed. Even if there are answers to questions that became invalid due to a review process, these answers are kept but no longer relevant. We keep these 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 only recalculate 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 questions but 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 user answers or postpones an answer which means that the dependencies chain has to be recalculated in a brute force way.

Our solution is admissible for surveys with small dimension, which it is really the case: our questionnaire has 53 questions. David Crighton in the ERA4 [Crighton 2005] used a TMS in order to backtrack and review a questionnaire for an online Electronic Referee Assistant.

4 Next Question Selection

We will describe now the next question selection module. Remember that if the question chosen has not been answered yet, it has to be valid and can only be one that was never posed before or an old question that was left unanswered. Note that validity is assessed by the rules discussed in section 2. As mentioned before, we want flexibility in what it concerns the design of question sequence (survey narrative) and the adaptability to the user answers history. We want to avoid a monolithic narrative that is completely defined a priori. To accomplish this goal we will again use rules to express the criteria to be used to select the next question to be presented to the user. The criteria can be very simple or very complex, whatever is the wish of the questionnaire designers. Again, our idea is to take advantage of the good qualities of rule-based systems: externalization, modularity and position for change.

In the online applications survey, available in the market, a question is simply selected because a particular answer was asserted. We think that here is other type of information important for question selection besides the presence of certain combinations of answers, like the number of times a question was presented to the user and left unanswered; the fact that the user has just logged in; if the user has just logged in, it can be also important to know if it was the first login. The fact that there are strong dependencies between questions and answers can be also taken into account. In case the last posed question was left unanswered, we do not want to select it again immediately, so it is important to register the last question presented to the user. The survey application must collect this information, which can be used by the question selection rules.

Note that we have several rules competing for the selection, i.e., several different rules may have their antecedents satisfied and can fire. We assign different priorities to the rules in order to solve the conflict between different activated rules. This way, we have assigned priorities to the selection rules whenever there is an intersection between the rules conditions.

We are going to present an example of a question selection rule that we have used in *De.:SID*, translated from CLIPS to natural language.

SelRule1 with priority 100: If the user has just made login and it is the first login then choose question 1

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

NewSelRule1 with priority 100: If the user has just logged in and it is the first login then choose randomly one of the questions in set {1,2,3,4,5}

We show next a more complex rule:

SelectionRule5 with priority 10: If the user has just logged in and it is not the first login then choose a valid question that was presented before but not yet answered and which is different from the last presented one but was presented less than three times

Using the information about the strong dependencies presented in section 3, we can declare the following rule

SelectionRule4 with priority 80: If there is a valid question that depends strongly on the answer to the last question that was presented then select it

So if the last question was question 51 and the answer to 51 is "1" and there is a fact stating that (dependency 51 "1" 52) and if question 52 is valid then 52 is selected.

In surveys we can find questions that can be quickly answered and others that demand more time. Thus, we could also, for instance, classify the questions as "hard" or "light" 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.

5 Checking Inconsistencies

It is possible to ask questions in a survey in which inconsistent answers can coexist — 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 (1) to separate inconsistency knowledge making it independent from data and from programming code; (2) to make it explicit and understandable and (3) to make it easy to modify without disturbing the other components of the survey architecture. In fact, the three rule system qualities. We give two rule examples from the *De.:SID* design survey. The rules are written in CLIPS but were translated to natural language for better understanding.

InconsistencyRule1: If the enterprise does not look to other competitors (answer to question 27) and knows 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 a justification for the inconsistency—the answers involved appear on the screen, users can jump into any of the questions whose answers are inconsistent and review them. 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 review their answers or decide to maintain them. The way we deal with inconsistencies have been subject to some criticism by marketing specialists, during the presentation of [Urbano and Rodrigues 2008] saying that we should never warn the user about their inconsistencies and that inconsistency treatment should be made off-line. In either case, we think that detecting inconsistencies is an important issue and we want to underline that rules are a very natural way to express and detect inconsistent answers. That information can also direct the question selection module to present a question that is related to some inconsistency. Other criticism was related with the possibilities of different answer interpretation, so an inconsistency for us is not an inconsistency from the user point of view. This is the main reason we did not force users to solve inconsistencies due to their subjective nature.

6 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 knowledge bases for expressing knowledge associated with three important components of survey systems: the dependencies between questions and answers (skip and branching logic), inconsistency detection and the decisions regarding question 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 on Portuguese Manufacturing Industry, but the architecture can be generalized to any survey. One natural extension of our work can be the integration with an Online Survey Tool, to provide it with these new functionalities. We have to consider that survey authors do not have to know how to program in CLIPS and we should think in a user-friendly CLIPS rules translation system.

In the future, we also want to incorporate a Truth Maintenance System in order to improve our backtracking and revision process.

Acknowledgments

We would like to thank all members of De.:SID (http://desid.fa.utl.pt). We thank the Science and Technology Foundation (FCT) that funded the research project with reference PTDC/AUR/70607/2006 and the title “Design as a company’s strategic resource: a study of the impacts of Design”. I would like to thank also the anonymous reviewers and to João Balsa for their valuable text revision and suggestions.

References

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. Proceedings of the 5th International Joint Conference on Artificial Intelligence, 1977.

Edwards, A., Connel, A.A.D. 1989. Expert Systems in Accounting. Herdfordshire, UK: Prentice Hall International (UK)

Forsyth, R. 1984. Expert Systems: Principles and Case Studies. London Chapman and Hall Computing.

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

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

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

Harmon, P., King, D., 1985. Expert Systems: Artificial Intelligence in Business. New York: Wiley.

Hostedware Corporation. 2002. Hosted Survey. http://www.hostedsurvey.com

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

Linz, P. 1996. An Introduction to Formal Languages and Automata, 2nd edition. D.C. Heath and Company, Lexington, Massachusetts

Marra, R. M., Bogue, B. 2006. A Critical Assessment of Online Survey Tools. In Proceedings of the 2006 WEPAN Conference, WEPAN-Women in Engineering Programs and Advocates Network .

NPowerNY (nd). Online Survey Tools. Retreived 13 of September 2008 from http://www. austinfix.net/seminar/files/survey/guide+to+online+survey+tools.pdf.

Pargas, R.P., Witte, J.C., Brand, L. 2003. OnQ: An authoring Tool for Dynamic Online Surveys. In proceedings of ITCC'03, International Conference on Information Technology: Computers and Communications. IEEE Computer Society.

Riley, G. 2001. CLIPS: An Expert System Building Tool. In Proceedings of the Technology 2001Conference, San Jose, CA, December.

Romão, Luís; Almendra, Rita; Urbano, Paulo; Dias, Eduardo Afonso; Barata, José Monteiro; Nevado, Pedro; Marcelino, Rui; Dias, Joana Afonso; Gomes, Francisco Ferreira. 2007. An online survey's design to capture portuguese companies' perspective of design, in Proceedings 7ª Conferência DEFSA (Design Education Forum of Southern Africa), Internationall Design Education Conference 2007, FLUX - Design Education in a Changing World, Cape Península University of Technology, October 3-5.

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

S-Ware, 1996-1998. WWWSurveyAssistant. http://or.psychology.dal.ca/~wcs/hidden/ home.html.

Sawtooth Technologies 2002. Sawtooth http://www.sawtooth.com.

The PHLIPS Project 2004. http://phlips.sourceforge.net

Urbano, P, Rodrigues, D. 2008. RuleBased Systems Applied to Online Survey, Proceedings of IADIS International Conference www-Internet 2008, Freiburg, October 12-15.

Waterman, D. A.1986. A Guide to Expert Systems. Reading, MA: Addison-Wesley.

Winston, P. H., Prendergast, K.A. (editors) 1984. The AI Business: Commercial Use of Artificial Intelligence. Cambridge, MA. The MIT Press.

Yang, H.H. Forrester, D. Harris, C. 2007. A PHLIPS-based expert system for genealogy search. In the Proceedings of SoutheastCon, 2007. IEEE .