

Comparative analysis among open source and commercial software for the development of electronic questionnaires for statistical surveys

Laura Capparucci¹, Massimiliano Degortes¹, Marco Landriscina¹,
Manuela Murgia¹

¹ISTAT, e-mail: capparuc@istat.it, degortes@istat.it, malandri@istat.it,
murgia@istat.it

Abstract

This study is aimed at evaluating the capabilities of open source software to satisfy technical and methodological requirements for the development of electronic questionnaires for statistical surveys. In other words, to understand whether they could offer a reduction in license costs still guaranteeing high levels of efficiency and usability quite similar to those offered by commercial software. To this aim, some open source software were compared with Blaise, which is a commercial software used by many National Statistics Institutes (NSIs) since its functionalities and characteristics can be considered standards to be observed by software for CAI applications. The comparative analysis consisted in developing, with the selected open source software, a prototype of an electronic questionnaire containing a minimum set of functionalities that a CAI questionnaire should carry out. As the selected software were not able to satisfy this minimum set, we designed and implemented a possible strategy to develop, in-house, an ad-hoc product based on XM (*eXtensible Markup Language*)L technology, AJAX infrastructure and Relational Database Theory.

Keywords: Open source software, CAI questionnaires, XML

1. Introduction

This paper can be divided in two parts. The first one (chapters 2-5) describes weakness and strengths of existing open source/free software in implementing a questionnaire prototype. The second one (chapters 6-7) is about a new strategy aimed at overcoming the limitations encountered with the analysed open source products and the description of the elements it consists of.

2. The prototype of an electronic questionnaire

The comparison of the selected open source software consisted in evaluating their efficiency and usability for the implementation of an electronic questionnaire prototype. We designed this prototype as a “container” of a minimum set of characteristics, all possible types of questions and checking rules, that could be met in a CATI questionnaire (scheduling issue not considered). Why CATI? The answer can

fonts of characters have been used according to the role played by the text (instructions, warnings, questions wording, on-line helps etc.). This prototype was designed according to Blaise potentialities (Blaise can implement any of them) and we were interested in understanding what open source software were able to do.

3. The software selection process

To select open source or free software we used the following two criteria: *i*) the experience of usage by other NSIs or other organizations carrying out statistical surveys; *ii*) the availability and accessibility of documentation. We already knew some software satisfying these two criteria: **CSPro** developed and used by the Bureau of Census and **Lime Survey** used in Istat for experimental projects (Cianchetta R. et al. (2005) and “*Road accidents survey*”). To get other open source software we firstly made a small “e-research” among the main NSIs asking them if they were using or intended to use them. Due to the negative answers we had at our disposal no experienced software. Then we made a web recognition and, among a very huge amount of software, we chose the following ones: **Mod_Survey** developed by Mid Sweden University – Department of Information Technology and Media; **JCaif**, developed by Micheal Hall, hosted in the Sourceforge.net. Therefore our study will focus on the following products: **CSPro**, **Mode_Survey**, **LimeSurvey** and **JCaif**.

4. Technical details about the selected software

Before describing the potentialities of the above mentioned software, it will be useful to mention their main features without going into too many technical details.

Software	Developed by	Release (used)	Platform	Data storage
CSPro	Serpro S.A., ORC Macro International and U.S. Census Bureau	3.3	Windows	ASCII file
Mod_Survey	Mid Sweden University – Dep. of Information Technology & Media	3.2.5	Apache Server	ASCII file, PostgreSQL, Oracle, MySQL
LimeSurvey	LimeSurvey Project Team	1.72	Linux or Windows	MySQL
JCaif/Sonar	Micheal Hall	0.7	----	----

CSPro (<http://www.2010census.biz/ipc/www/cspro/index.html>) is a free software used for entering, editing and tabulating data from surveys and censuses. It was used also in Istat for some international co-operation projects among which the “*Agricultural Census 2006*” in Albany. It supports stand-alone applications while **CSProX** manages multi-users data capturing but it is a commercial product. Documentation is very detailed and full of examples that make it possible to easily cover eventual lacks by directly writing lines of code alongside the offered standard functions. It provides other functions to process data as it was thought as a package able to support the survey process from data collection to the data processing phase.

Mod_Survey (<http://www.modsurvey.org>) is a “mod_perl” module for Apache that links the powerfulness of the programming language Perl with the Apache Server’s functions. It manages files containing questionnaires described in XML (*eXtensible Markup Language*) based tag notation. It is a software for web questionnaires that allows data capturing, data validation, data storage and the data export. It offers a detailed and well organised documentation together with a well designed website where it is possible to find many answers to specific problems. It is developed by Mid Sweden University that has worked on it since 1988. Among the experience of its use we mention the implementation of the questionnaire for the “*Longitudinal survey on university degrees in 2007-2008*” of the Italian University of Padua.

LimeSurvey (<http://www.limesurvey.org>), formerly called PHPSurveyor, is written in PHP language. It is quite easy to use and requires few developer skills. It allows the implementation of simple web questionnaires clicking on pre-defined functions. It divides the survey in groups (sections of the questionnaire) displaying, in general, a group per page. The navigation of the questionnaire is managed by two buttons: “next question” and “previous question”. It can export data in many formats like Excel, SPSS, etc.. Documentation is available on the web in several languages.

Sonar/JCaif (<http://sourceforge.net/projects/jcaif>) is an XML-like language that represents an interface for the usage of the JCaif package. JCaif is a set of Java classes that implement: different question formats, flow control and data display and saving. To build surveys, Sonar uses Javascript to make Java calls to the JCaif library. It is a well designed product but it lacks of accessible documentation and has a limited use in surveys. Therefore we couldn’t adopt it for the implementation of our prototype.

5. The implementation of the prototype

Before describing advantages and shortcomings of each selected software, it is important to remind that the evaluation is bounded to our knowledge and to our experience in developing complex questionnaires. Needless to say, all software would fit in case of simple survey questionnaires.

The first product analysed is **CSPro** that turned out to be the only software, among those analysed, able to implement all types of questions included in the prototype and to perform the on-line coding of textual variables. Besides, it offers a quite good layout management both in terms of text’s colours and fonts and of video screen appearance. At the same time we observed some shortcomings. The first is a general lack of standardised functions for the error management during the data capturing phase: while it is possible to perform a complete checking and correction of data at the end of data collection by means of a batch procedure, during the data collection itself only a limited set of this kind of functions are available. For example, the ‘range control’ function is missing and must be specified, with lines of code, each time a quantitative variable is used: statistical surveys measure a lot of quantitative phenomena and range controls are hard checks to limit keying errors. Besides, it is not possible to customise the content of an error message and its window doesn’t list all variables involved in the error but the last, with potential problems for questionnaire navigation. For what concerns list variables, it is possible, during the data collection, to key a value not included in the pre-defined values set (after its confirmation in a

warning window) and this might introduce too many keying errors. Referring to multi-choice questions, there isn't a standardised function that, given a certain number of answer alternatives, limits the maximum number of respondent's answers. Therefore in this case it is necessary to write lines of code to set this type of constraint. Another feature, that could represent a shortcoming, is that missing values are the default ones for any variables and there isn't a standardised function to state that a variable must be answered. This approach is correct for Data-Entry or Web techniques but not for the other CAI modes since it might increase partial non response rates. There are also some limits in the questionnaire navigation in that it is not possible to pose on a certain variable unless passing upon all previous ones and "page down" and "page up" keys cannot be used to run down and up the questionnaire's pages. Finally, a last shortcoming: all branches filled in during the interview are stored, also those that should have been blanked according to routing instructions. This means that data belonging to mutually exclusive branches might be present on the final database. Needless to say, this is reasonable for data entry but it is useless for the other CAI techniques whose aim is to eliminate branch's errors during the data collection.

The second product analysed is **Mod_Survey** that is able to provide the majority of questions' types and also multi-paging for the administration of different branches. Besides, it offers an easy implementation of the item rotation for matrix variables and a good layout management in terms of text's colours and fonts. For what concerns the types of questions, matrix of quantitative variables, on-line coding and fields with data format can't be implemented (release 3.2.5). We also found a bit complex the multi-paging structure if it has to be used for complex questionnaires. In fact, it is necessary to create as many pages as the number of filter questions and then list all pages, that form the questionnaire, in the last one. This will apply also for branches formed by only one question. Creating different pages is also required for texts' customisation since the information that makes the text dynamic needs to be stored and the storing can only be done using different pages. Finally, it lacks of standardised control functions. The existing ones are managed on server side and customised controls can be inserted on client side using Javascript, but we found this solution not so flexible and easy to implement in case of complex questionnaires. A good feature of Mode_Survey is the tag <CUSTOM> for the definition of completely customised questions using HTML and Javascript languages¹: any kind of question formats and, above all, checks on data can be added according to the survey's needs. This feature, together with the goodness of Mod_Survey, made us thinking to use it as the open source software to develop Istat questionnaires: Mod_Survey would have provided question formats, some native checks and branching rules; Istat would have integrated them with control functions on data. Unfortunately, this strategy turned out not feasible for complex questionnaires due to different reasons: 1) native checks are performed on a server side once the form (questionnaire) is filled in. This means that errors can't be solved in real time during the administration of the questionnaire but only after the form is sent to the server. This can generate "traffic" problems on the LAN in case of complex questionnaires since data go back and forth from clients to server many times. Besides, if the form consists of only one page, errors are shown from the first one met with a possible backward navigation of the entire questionnaire;

¹ For completeness, it has to be said that Mod_Survey uses an XML based language which is translated into HTML by the browser.

2) Javascript language can help in performing checks on a client side, but the nature of the native checks of Mod_Survey (performed on server side) implies their re-writing. Therefore we can't talk about integration and hence customisation of checks but of a mere substitution or duplication of the native ones; 3) using Mod_Survey as a generator of HTML code is not feasible since the HTML deriving from the translation is not well structured and doesn't exploit at best the potentialities of the code itself; 4) the amount of HTML and Javascript code to be used to add missing control functions would have definitely outnumbered the lines of code produced by Mod_Survey thus lowering the advantages of its use. Therefore, although Mod_Survey can't be used for Istat's aims, it surely highlighted the great potentialities of XML technologies and helped us in designing a new strategy that will be described in chapter 6.

The last software analysed is **LimeSurvey** that offers many tools to create simple web questionnaires, like the managing of branching, the use of different types of questions, the use of quotas etc. It offers several levels of customisation: at a survey level, it is possible to choose the title, to write an introductory or welcome message, to make available some information about the survey administrator, to choose the survey language, to customise the page layout, etc.; at groups of questions level, it is possible to set the title (example: Section 1: Household information) and, if necessary, to change the order of questions administration; at a single question level, it is possible (mandatory) to define an identification code, to write, question's text and to display a remark (e.g. the range of possible values). Questions' formats can be chosen among a wide list of question types. For list questions, it is possible and easy to use the randomization of the answer alternatives. On the other side we found some limits, the greatest of which was the difficulty of adding missing features by writing line of codes. This is because customisation requires both a good knowledge of PHP language and the understanding of where to write lines of codes to make them recognisable and executable by the product. Consequently adding code to implement our prototype would have meant to write another product. Besides, it is not possible to use loop of questions and the on-line coding is quite difficult and complex to be used since it is not possible to read external data. Besides, branching management is almost limited: *i)* only the logical operator "AND" can be used to link different questions involved in the condition since the "OR" operator applies only among different values of the same variable; *ii)* it is necessary to repeat skipping rule for each question belonging to the branch. Finally, for what concerns checking rules we noticed a lack of standardised control functions² and a poor error messages management: they can be managed using "boiler plate" questions (i.e questions not to be answered) and then conditioning their visualisation, but this is hard to apply for complex questionnaires.

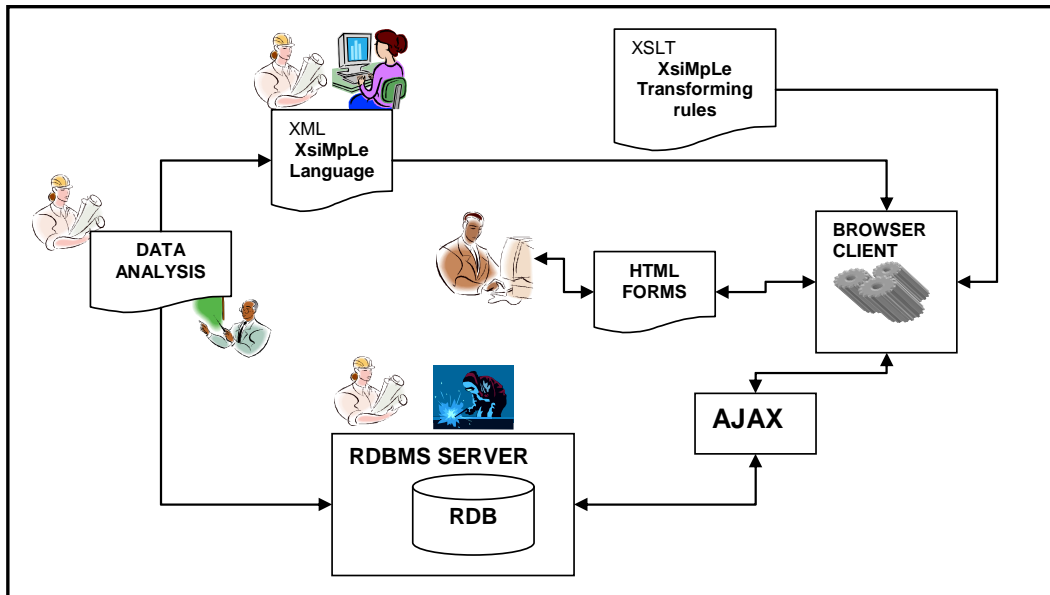
6. An ad-hoc product based on XML technology

The experience done allows us to say that the analysed products, especially the web based ones, can be used to implement quite simple questionnaires. In general, they all lack of standardised control functions and for the web-based software there is still need to improve the client and server 'dialogue' to perform checks on a client side rather than on a server side. These elements make them unsuitable for Istat

² It is possible to control keyed value using a very "peculiar" syntax. (example of a range control 0 to 15: `"/^([0-9] | [1] [0-5])$/`).

questionnaires that are generally very complex and contain a lot of checks. Therefore, if developed with the available web-based products, it could be possible to encounter problems of data exchange between client-server if the dialogue between them is based on a classical approach. We then tried to design a possible strategy as described in the following picture.

A possible strategy for an Istat open-source data capturing phase



Generally speaking, it consists of many different open source technologies to make a home made product. We made a search in the open source world to collect different tools into a package having all the features we need. The main idea was to use an ad-hoc XML language associated with an XSLT (*eXtensible Stylesheet Language Transformations*) document to be used as a sort of compiler for the XML language. Browsers, like Explorer are, in fact, able to carry out the translation step from XML to HTML via the XSLT document³: the browser will be the questionnaire user interface. The result of this strategy is the home made product, called *XsiMpLe*. The reason why we chose this strategy can be found in the need of a very simple language that, on one side, makes the programmer's job easier avoiding the massive use of HTML formats (quite a number of graphic symbols like inverted commas, parenthesis, apostrophes, etc.) and, on the other side, can be used also by no IT experts that have at least a knowledge in text editors and browsers. The language is able to create HTML forms and to use generalised Javascript functions to check the keyed data. This will assure that some typical frequent checks will be done in real time, i.e. on client side with no delay. Besides, since client side checks are only a first step in the control process, a relational database (RDB) is used to perform structural checks and to assure data integrity and data constraints. The package is a client-server web application: the data collection, carried out through the questionnaire implemented with *XsiMpLe*, can be considered the client process while the data storage and the data retrieving the server process. Client-server communication is based on *AJAX* framework. Besides, in order to assure our goals, the package - *XsiMpLe* + *AJAX* + *RDBMS* - will be used together with some methods for questionnaire design: since filling in a questionnaire is not so

³ We still need to verify this feature among the latest releases of open source browsers. In the meantime we are using open source programs that supply this function.

different from populating a database, we can apply techniques for RDB design for the questionnaire design itself. Therefore we chose the entity-relationship model (E/R model) that can be used not only to analyse and to structure the questionnaire (Buratta V. et al.,1989) but also to design the entire RDB structure: data tables, data links and data constrains. Obviously questionnaire and data analysis with relational methods is introductory to all the other activities. Once data tables, data links and data constraints are well understood, the form creation (web questionnaire page) and the DB creation can be carried out at the same time during the questionnaire design phase. In this way we can get a big aid in implementing the electronic questionnaire through *XsiMpLe* and, most importantly, a correct design for the RDB underneath the data.

6.1 A brief description of *XsiMpLe*

Example of a form created with *XsiMpLe*

The screenshot shows a web form titled "EMPLOYMENT" with the following fields: "WORK PLACE" (text input with "ISTAT"), "JOB" (radio button with "1" selected, and a list: 1. Clerk, 2. Executive, 3. Manager), "STARTING FROM (dd-mm-yyyy)" (date input with "01", "10", "2000"), and "HOW MANY YEARS DID YOU WORK THERE?" (text input with "3"). A JavaScript error dialog box is overlaid on the form, titled "[Applicazione JavaScript]", with a warning icon and the message: "Please verify inserted data, you told me you worked there for 3 years, but you began working there in the 01-10-2000". The words "verify" and "01-10-2000" are circled in red in the original image.

The above picture is an example of a questionnaire implemented with *XsiMpLe*. The small window contains an error message whose text is standardised according to the values of the two variables involved in the error (“Starting from” - “How many years did you work there?”). This check is done on a client-side using a Javascript function. The *XsiMpLe* code to create this page is shown in the following pictures.

Example of *XsiMpLe* code

```

136 *****datablock2*****
137 <datablock tablename="employment"> <title>EMPLOYMENT</title>
138 <data>
139 <question>WORK PLACE</question>
140 <field>
141 <name>companyName</name><tablename>employment</tablename>
142 <default></default>
143 <type>text</type>
144 <length>20</length>
145 <onexit>refuse('companyName')</onexit>
146 </field>
147 </data>
148 <data>
149 <question>JOB</question>
150 <field>
151 <name>job</name><tablename>employment</tablename>
152 <default></default>
153 <type>text</type>
154 <length>1</length>
155 <onexit>range(1,3,'job')</onexit>
156 </field>

```


XsiMple is able to implement all questions' types included in the questionnaire prototype. Therefore, with respect to the other web-based products it guarantees the management of many question formats and, in addition, allows to use standardised control functions on a client side without the need of experiencing Javascript. The final aim of this strategy is to create a tool suitable for any kind of data collection mode. Any lack in control functions on a client side will be solved on a server side using the functionalities of the RDB and AJAX to facilitate the client-server communication. We prepared a small application that simulates a data capturing process. It consists of a questionnaire and the relative database (MySQL) with consistency controls in it. Each time a question is filled in, data are sent to the RDB and in case of data corruption the RDB sends an error message that is captured by *XsiMple* and then managed by the respondent/interviewer. By the time this paper is written there still the need to improve the questionnaire navigation as well as to test the application's performance by measuring the response time in case of many users filling in the questionnaire. This will be done using some simulation programs.

7. Conclusions

At the moment there aren't open source/free product able to implement complex electronic questionnaires often used by NSIs. Till now commercial software like Blaise still represent the best solution to this issue. Anyway open source products are becoming more and more flexible and powerful and will be soon able to guarantee high quality levels and saving of money. This is an important aspect in public organisations because of the saving of money imposed by local governments (Barcaroli 2008) and even more in small companies that can have access to sophisticated tools before not affordable. The strategy we proposed can represent a solution that could well balance these two aspects. We believed in this since it is based on new open source technologies that permit to put in practise a data collection approach that was difficult to adopt till now due to the high cost and low flexibility of the commercial IT software. Nowadays, open source database (RDBMS) and operative systems together with web technologies – HTML, XML, XSLT, browser, web services - allow to implement in an easy and efficient way a quite complex data collection architecture. They permit to simplify the complexity and heterogeneity – in terms of instruments and professional profiles - that are typical of this survey phase that includes analysis, collection, control, storage and transmission of data.

References

- Barcaroli G. (2008) Il processo di produzione dell'informazione statistica e l'opzione open source, *Istat workshop: il software per la statistica ufficiale: dai sistemi*
- Buratta V., Sabbadini L.L., Fortunato E. (1989) Manuale di tecniche di indagine - il questionario: progettazione, redazione e verifica, *Note e relazioni* Vol.2, n.1
- Cianchetta R., Pagliuca D. (2005) Soluzioni Open Source per il software generalizzato in Istat: il caso di PHPSurveyor, *Collana Documenti Istat* n.17
- House, C. C. (1985) Questionnaire design with Computer Assisted Telephone Interviews, *Journal of Official Statistics*, Vol.1, n.2, pp. 209-219