QuON
A Generic Platform for the Collation and Sharing of Web Survey Data

David Paul, Mark Wallis, Frans Henskens and Kim Nolan
Distributed Computing Research Group, University of Newcastle, Callaghan, Australia
David.Paul@uon.edu.au

Keywords: Survey : Data Collection : Software

Abstract: QuON is an open-source system that allows researchers to design and conduct Web-based surveys. It was created to overcome limitations in existing survey systems, and is especially concerned with conditional branching, user and group management, and the generation of metadata to allow optional sharing and external discovery of collected survey data. This paper describes the current QuON software system, its features, and future directions.

1 INTRODUCTION

Surveys and questionnaires are a vital component of many research projects (Ader et al., 2008). They can be used for various different purposes, such as providing a benchmark of a population's current attitudes or behaviours, presenting information to respondents in a relevant manner, or even to evaluate an individual's understanding of a particular topic or issue. Surveys can be conducted via face-to-face personal interviews, via the telephone, through paper forms sent in the mail, using custom software, or, using Web forms, over the Internet.

Surveys conducted over the Internet provide many advantages to the other survey types. Like paper forms, Web surveys do not require a researcher to be actively involved in the collection of responses after the survey has been designed and published, which reduces overall cost. However, like personal and telephone interviews, it is possible for the survey to change dynamically based on a respondent's previous answers. For example, a Web system can automatically skip questions asking for more details if the respondent has indicated that a particular set of questions is not relevant for that individual. Further, such customisation in Web forms can be automatic, removing the risk of human error in determining which questions should be asked to a particular respondent. While custom (non-Web-based) survey software also enjoys these advantages, the software must be installed on the respondent's computer. The fact that most computer (and tablet/smartphone) users already have access to a Web browser lowers the barrier of entry, making it easier for Web surveys to reach a larger audience.

Thus, many research projects choose the Web as their medium for conducting a survey. While it would be possible to manually create the HTML forms required to conduct each new survey, various platforms have been developed to make the creation of Web surveys easier. However, while numerous different frameworks exist, none has all of the features necessary to conduct all kinds of research surveys.

This paper introduces QuON, a generic survey platform designed for managing and collating scientific Web surveys, especially in the area of health behaviour. Section 2 describes existing survey systems and some of their limitations. Section 3 then describes the QuON framework, which has been designed to overcome these limitations, to improve support for scientific surveys. The current unique features of QuON are described in Section 4. Finally, Section 5 concludes the paper, describing QuON's advantages over existing survey systems for scientific surveys and indicating on-going and future work.

2 EXISTING SYSTEMS

At a high level, systems designed to facilitate Web surveys can be split into two groups: online services, such as (FluidSurveys, 2013; SurveyMonkey, 2012); or stand-alone applications, such as (Digivey Survey Center, 2012; LimeSurvey, 2013; Webmyne Systems Inc., 2013). Some systems also offer both online and stand-alone versions (Checkbox Survey, 2013).

For scientific surveys, especially surveys involving sensitive information such as medical details, online services are not always appropriate. This is especially true when ethics approvals do not allow third-parties to store the information for a researcher; only local systems can be used in these cases. Since QuON is designed primarily for scientific surveys, this section will concentrate on survey systems that can be installed on local infrastructure.

One of the complications with survey software is that surveys are not always linear. For example, in many paper surveys there are instructions similar to "If 'No', go to Question 7". Web surveys do not need to display these instructions; the system can automatically skip certain questions based on previous responses. However, in order to support such operation, it is necessary to specify this branching logic when the survey is defined. Different systems allow this information to be entered in different ways.

One way to allow selective display of questions, supported by (Checkbox Survey, 2013; Digivey Survey Center, 2012; LimeSurvey, 2013; SurveyMonkey, 2012), is to have a condition for each question, such that the question is only displayed if the condition is true. The problem with this approach is that, if the researcher wants to skip a group of questions together, then the necessary condition must be entered once for each question. Thus, any change in the condition logic would need to be reentered in multiple places, which can be tedious and error-prone.

Another option to allow skipping of questions is to include, with each question definition, an optional condition that specifies which question to display next, based on the response given to the current question. Such a technique is used by systems such as (Digivey Survey Center, 2012; SurveyMonkey, 2012). However, the condition for these systems can depend only on the value of the current question. This makes it impossible, for example, to display a question that discriminates between respondents, then display some common questions, and then branch based on the discriminatory question. Other systems, such as (Checkbox Survey, 2013) do allow more complex conditions that are based on questions other than the current one, but the branch logic is still part of a question, meaning that multiple questions cannot reference the same conditions. It is possible to overcome these limitations by displaying any discriminatory question after the common questions, but this changes the survey's design. Another option is to define the common questions or branching logic multiple times, once for each of the discriminated types, but any change to the common questions would then require each copy to be altered.

User management is also important when conducting surveys. Often, it is useful to collect anonymous information, where the researcher does not know the respondent's identity. Other times, only authorised users should have access to complete a particular survey, so a username and password is required. In other cases, such as when only known respondents should be able to participate but passwords would place too great a burden on the respondent, it would be useful to assign an identifier to a respondent without a password. The individual can be identified for the current, or any future, survey, without being required to enter authentication information. This is particularly useful when combined with custom URLs generated for each respondent that include the identification information automatically. While current systems do support some combination of anonymous and authorised access, we have no knowledge of systems that support identified but unauthenticated respondents.

Another consideration, for scientific surveys in particular, is publishing the fact that data has been collected. While particular studies, based on collected data, can be published in journals or conferences, there is currently no easy way for researchers to publish information about existing datasets, and thus it is difficult for others to discover any survey responses that have already been collected. No known existing survey software publishes metadata about the surveys that have been conducted. However, if researchers could publish such metadata to a registry, then other researchers could dynamically discover the dataset and may be able to produce further results without any extra data collection. Any ethics requirements could still be enforced, since only information that the dataset exists, and how to request access to it, is released with the metadata. However, discovery of the metadata would allow new researchers to examine all requirements necessary to gain access to the dataset, and to determine whether it is better to access the existing data, or whether new data should be collected.

QuON is a system designed to overcome the above limitations of existing survey software. It can be installed on a local system, meaning that no collected data is held by a third party. It also supports complex branching to allow customised surveys, different user types to meet researcher requirements, and the ability to publish survey metadata to allow external researchers to discover the collected datasets.

3 QUON FRAMEWORK

QuON is a Web application built using the CakePHP platform (Cake Software Foundation, 2012). It uses a MySQL database (Oracle, 2012) to store both definitions of the surveys in the system and the answers collected for each survey. When a researcher logs in to the system he or she can either create a new survey or edit an existing survey. When a survey is properly defined, the researcher can publish the survey, allowing respondents the ability to access and complete the survey over the Web.

To create a survey, a researcher specifies a number of survey objects and indicates the order in which they should be presented when a respondent is completing the survey. Survey objects can either be *questions*, which request a response from the respondent, *calculations*, which evaluate an expression that may rely on a respondent's previous answers, or *branches*, which provide a novel way to customise the order in which survey objects are presented to an individual respondent. Each survey object is given a name, which is unique for the survey. Researchers enter attributes for each survey object, which specify options for how the object should be processed or displayed. Figure 1, for example, shows the researcher's and respondent's views of a question object.

*Question*s present the user with a prompt, and record a response from the user. QuON automatically records information such as the time the question is presented to a respondent, the time the respondent provides a response to the question, and the value of the response entered. The framework has out-of-the box support for many common question types, and is extensible to allow easy development of new question types.

The default question types supported by QuON are:

Informational Displays text, pictures, and/or video to the respondent and waits for an acknowledgement that the respondent is ready to continue.

Text Allows the user to enter or edit a text response and waits until the user requests the next survey object. A researcher can specify a regular expression that the response must match before the respondent is allowed to continue.

Checkbox Displays a list of check boxes, allowing the respondent to select as many options as are appropriate. Limits can be placed on the number of options that can be selected, and there is configurable support for entering "other" values (displaying a text box to allow the respondent to enter more details), or for selecting "none of the above". Once the respondent has selected the applicable items, clicking "next" causes the system to store the response and proceed to the next survey object.

Radio Button Similar to a Checkbox, but the respondent can only select one option. Once an option (which could be "other" or "none of the above") is selected, clicking "next" moves the system to the next survey object.

Button Option Similar to a Radio Button, but the options are displayed as buttons that proceed to the next survey object as soon as they are selected, rather than requiring the respondent to confirm their selection is complete.

Drop Down Presents a drop-down list to the respondent, allowing a single option to be selected.

Calendar Presents a calendar allowing the respondent to specify a date. Input can be restricted to a particular date range, and the answer can be stored as either a fixed date (with a configurable date format), or as the difference from a specified (by the researcher) date.

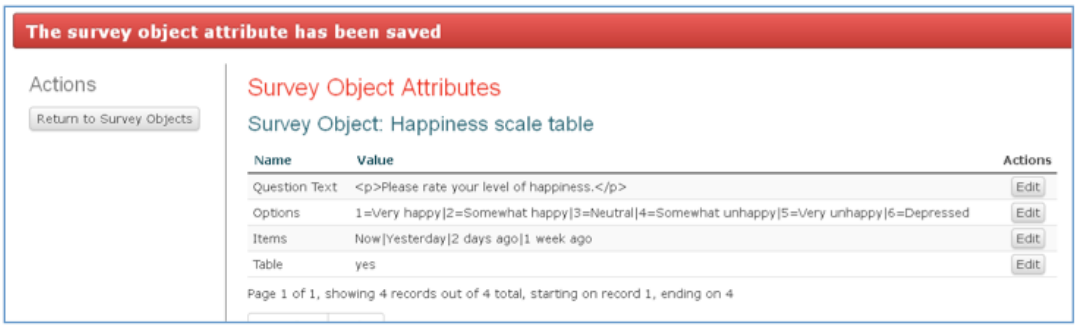
Rank Order Displays a list of items that the respondent can rank. The researcher can specify the minimum and maximum number of options that should be ranked.

Distribution Of Points Displays a list of items and specifies a number of points that the respondent can assign. The respondent assigns points to each item, and the system ensures the total number of points are within the required range.

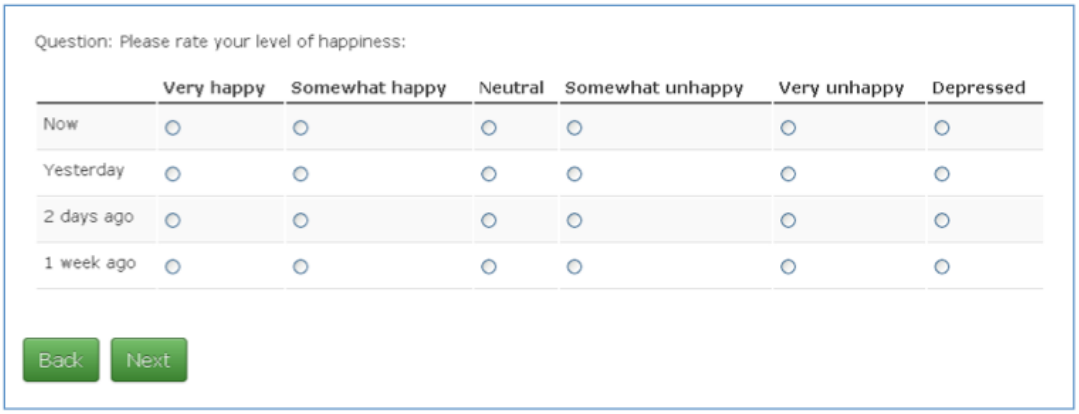
Likert Scale Presents a list of options, allowing the respondent to select a value on a scale for each item. The researcher specifies both the options and the scale that are used.

Each question type is specified in a CakePHP Helper consisting of an array that specifies the attributes of a question that can be customised by a researcher, a function that specifies how the question should be displayed to a respondent, a function that specifies how a response should be validated, and a function that specifies how the value of any response can be stored in a way that ensures the answer is reproducible if the question is displayed again. Thus, to support a new question type, a developer simply specifies a new Helper with the appropriate details.

As well as the above question types, QuON also supports *calculation*s based on answers to previous questions. The researcher can specify any mathematical expression supported by PHP, and question names surrounded by square brackets are replaced by the values of the answers stored by the system for



(a) Researcher's view



(b) Respondent's view

Figure 1: Different views of a Likert Scale question survey object.

the current respondent. The value of the calculation can be displayed to the respondent, or simply stored so the researcher can access the value when viewing the results of a survey. For example, if the survey includes questions about the respondent's mass (in a question named "mass") and height (in a question named "height"), a calculation object could be used to display the respondent's body mass index by setting its formula attribute to $[mass]/([height]*[height])$.

QuON also supports *branch* objects to alter the flow of a survey. By default, after a survey object has been processed, the system moves to the next survey object in the ordering specified by the researcher. Branches allow conditional logic to alter how the next survey object is selected. Branch objects use a similar syntax to calculation objects to allow specification of a PHP boolean expression that can include a respondent's previous answers, or any previous calculations. Branch objects also include the name of the survey object to jump to if the expression is true, and the name of the survey object to jump to if the expression is false. For example, if a question named "hasEmail" asks "Do you have an email address?", a branch object with the condition $[hasEmail == 'yes']$ could cause the system to skip a question asking for the respondent's email address if the respondent does not have one.

As well as specifying the survey objects to be included in a survey, the researcher must also specify the order in which the objects are to be processed. When a respondent begins a survey, the first survey object from the specified ordering is processed and the survey continues until the last survey object has completed. The different survey objects are processed as follows:

Question Depending on the question type, the appropriate Helper's display method is called, and the system waits for the respondent to enter a response. When a response is received, it is passed through the Helper's validation method. If the validation fails, an error is presented to the respondent and the question redisplayed. Once validation is completed successfully, the response is serialised and stored in the database. The system then moves to the next survey object in the order specified by the researcher.

Calculation The abstract mathematical expression associated with the calculation is parsed and any reference to previous questions is replaced with the value of the answer to that question entered

by the current respondent. The expression is then evaluated and the result stored in the database. If the researcher has indicated the calculated value should be displayed to the respondent, a display page is presented. The system then moves to the next survey object.

Branch The abstract boolean expression associated with the branch is parsed and any reference to previous questions is replaced with the value of the answer to that question entered by the current respondent. The expression is then evaluated and the system jumps to the survey object with the name specified by the branch object as the appropriate destination for a true or false result.

Once a researcher has specified the various survey objects and the order they should appear in the survey, the researcher can publish the survey, making it available to respondents. When a survey is published, the researcher is given a URL to be shared with potential respondents. When a respondent accesses the URL, the first survey object is presented (after any necessary authentication; see Section 4 for the different authentication types supported by QuON). The system processes this first object as specified above, and the respondent answers questions to progress through the survey.

On completion of the survey, the respondent is presented with a "thank you" page. The researcher can then access the information collected from that respondent either directly through a Web browser, or by downloading a CSV file, with all respondents' answers, that can be processed by local software tools.

4 CURRENT FEATURES

The QuON framework already offers advantages over many existing survey systems. The branching logic, in particular, is extremely powerful. Most alternatives require either the same condition to be entered for multiple questions, or questions to be entered multiple times, to support the same level of flexibility as offered by QuON. However, QuON also includes some other features that make it unique.

One of the limitations of existing survey systems is the lack of support for identified, but not authenticated respondents. QuON overcomes this limitation by supporting the following respondent types:

Anonymous The system does not request any information from the respondent before presenting the first survey object.

Identified The system requests a name that uniquely identifies the respondent. The respondent can only begin the survey if the entered name matches an expected respondent for the system.

Auto-identified The system requests a name that uniquely identifies the respondent. The respondent can begin the survey by entering any name that has not already been entered by a respondent. This mode can be used to allow anybody to complete a survey, but then to restrict access so that only respondents can access any followup surveys.

Authenticated The system requests a name and password that uniquely identifies the respondent. The respondent can only begin the survey if the entered name and password match an expected respondent for the system.

QuON also supports the publication of metadata to a registry to allow publication of a dataset. Researchers can enter information such as a description of the data collected, who the contact person for the data is, and any restrictions on the access or use of the dataset collected for a particular survey. When ready, the researcher can then publish this information to a ReDBox (Queensland Cyber Infrastructure Foundation, 2012) repository using a standard based on the ISO 2146:2010 standard for Information and documentation – Registry services for libraries and related organisations. Other researchers can then discover the dataset through ReDBox, and may be able to avoid collecting similar data by gaining access to what has already been collected.

5 FUTURE WORK

QuON is a platform for the creation and administration of surveys over the Web. It aims to overcome some of the limitations of existing survey systems for research applications, especially in areas such as health where ethics implications make the use of online services difficult. An open source implementation of QuON is available from `http://code.google.com/p/quon/`.

QuON takes a unique approach to branching in surveys, making it much easier to alter the flow of a survey as it is being completed. This approach is based on the idea of survey objects, which, in QuON, can be either Questions, Calculations, or Branches. Researchers then order the survey objects for each survey, and the system's processing rules ensure the specified order is followed. Branch objects allow complex orderings, such as skipping optional questions, or jumping back to an earlier part of a survey, based on boolean expressions that can include any

previous answers entered by the respondent. Further, separating branch objects from question objects allows the same conditional logic to be used from multiple areas in the survey, removing the duplication that is necessary in other survey systems.

QuON also supports the export of metadata to facilitate the discovery of collected data by other researchers. This can enhance the usefulness of any survey by allowing further research to be conducted on any collected data, rather than requiring similar data to be recollected by the new researcher. The system does not automatically grant access to the collected data, it simply allows external researchers to discover that the data exist. The metadata also provides further information, such as contact details, so the external researcher can request access to the actual data.

QuON allows multiple users in a group to work on a survey. In this way, multiple researchers can access the same survey. The system does not currently allow the sharing of survey objects between surveys or groups, but support for this is planned in future releases. Such a feature would allow the creation of a pool of questions that could be pulled in to new surveys without having to be completely redeveloped.

A QuON system has recently been deployed for the Health Behaviour Research Group at the University of Newcastle, Australia. The experiences of the researchers in this group will be used to evaluate the current state of the system and further QuON development. For example, a feature currently being implemented allows templated feedback sheets to be generated after a respondent completes a survey, allowing researchers to download documents that are customised based on the answers given by the respondent.

REFERENCES

Ader, H., Mellenbergh, G., and Hand, D. (2008). *Advising on Research Methods*. Johannes van Kessel Publishing.

Cake Software Foundation (2012). *CakePHP Cookbook*. Cake Software Foundation, 2.x edition.

Checkbox Survey (2013). Checkbox Survey: Survey Software. *http://www.checkbox.com*, Accessed 2013-01-11.

Digivey Survey Center (2012). Digivey Plus collects web survey data online. *http://www.digivey.com/digiveyplus.htm*, Accessed 2013-01-11.

FluidSurveys (2013). Online Survey Software — Create Your Survey In Minutes. *http://fluidsurveys.com*, Accessed 2013-01-11.

LimeSurvey (2013). LimeSurvey - the free & open source survey software tool ! *http://www.limesurvey.org*, Accessed 2013-01-11.

Oracle (2012). *MySQL Reference Manual*. Oracle, 5.5 edition.

Queensland Cyber Infrastructure Foundation (2012). ReDBox - Mint. *http://www.redboxresearchdata.com.au*, Accessed 2013-01-11.

SurveyMonkey (2012). Surveymonkey: Free online survey software & questionnaire tool. *http://www.surveymonkey.com*, Accessed 2013-01-11.

Webmyne Systems Inc. (2013). Survey software — magicsurveytool.com. *http://www.magicsurveytool.com/survey-software/*, Accessed 2013-01-11.