Using Service-oriented Architectures for Online Surveys in *Coast*

Thomas M. Prinz, Raphael Bernhardt, Linda Gräfe, Jan Plötner, and Anja Vetterlein
Course Evaluation Service
Friedrich Schiller University Jena
Jena, Germany
e-mail: {Thomas.Prinz, Raphael.Bernhardt, Linda.Graefe, Jan.Ploetner, Anja.Vetterlein}@uni-jena.de

***Abstract*—Electronic surveys are used in empirical sciences in many cases to reach a large and global crowd. Such surveys can be separated into five big phases (e. g., planning and data collection), where each phase belongs to each other. Since the interdependencies between the phases are sometimes complex, it is helpful to have a software system, which supports each phase of a survey. There already exist such systems, which cover together all of the phases. However, the focus is on the implementation of the survey rather than on its evaluation and reporting. This was one reason to build a new survey and report tool, *Coast*. The first version of *Coast*, however, has some disadvantages regarding its monolithic architecture — a new version of *Coast* had to be developed. This paper presents the new version of *Coast* as a practical result and example of service-oriented systems. The architecture of *Coast* is built on concepts of service orientation and model-driven software development. Furthermore, the paper explains the big design decisions during development, describes parts of the used meta-models, and shows some practical problems during the development of service-oriented programs and how they are solved in *Coast*.**

Keywords–Service Orientation; Architecture; Coast; Survey; Model-Driven Software Development.

I. Introduction

Surveys are indispensable in empirical sciences (for example in psychology and sociology). In many cases, it is profitable to use an *electronic* or *online* survey to reach a large and locally dispersed crowd (e. g., [1]). An electronic survey can be separated in different phases, e. g., survey implementation, data collection, and reporting [2]. Each of the phases is interwoven with the other. For example, the reporting phase needs the specific questions asked in the survey as well as the different variables and scales from the implementation phase. In turn, the collected data can only be interpreted by knowing the scales and variables they belong to. So, it is very helpful to have a software system, which supports each of the phases of a survey.

Such software tools are available on the market, e. g., *EvaSys* [3], *Unipark* [4], *LimeSurvey* [5], *KwikSurveys* [6], and *SurveyMonkey* [7]. Most of them cover parts of all phases of a survey. Almost all tools provide only a predefined or simple evaluation and reporting, since the evaluation and reporting phase are very individual processes and can become quite complex. For further analyses (e. g., multivariate analyses) exceeding the standard repertoire, these tools offer the download of the raw data instead. This, however, disrupts the link between the data and the other phases of a survey — the individual evaluation becomes a time consuming and cumbersome task even when the link to the other phases can be simulated with further provided information.

This fact was one of the main reasons of the *Course Evaluation Center* at the Friedrich Schiller University Jena in the year 2009 to build its own survey and report tool, called *Coast*. It focuses on the integration of the analysis and report phases for advanced analyses. Although it is in successful use for the evaluation of the university, the first version of *Coast* has its disadvantages. The disadvantages resulted from varying requirements, wrong design decisions, and surveys, which getting complexer as the architecture can handle. A monolithic architecture emerged [8], which makes it hard and risky to implement changes on the business logic.

As a result, a new version of the *Coast* application is currently under development. This new version removes the disadvantages from the first version by clarifying the architecture and modules as well as the model of *questionnaire*s and reports. In this context, a *questionnaire* refers to all the questions, scales, etc. being necessary to collect the data.

In this short paper, we present parts of our tool *Coast* as an example of a service-oriented software. Furthermore, we give a brief overview about the phases of an electronic survey and already existing survey solutions in Section II. In Section III, we show the architecture of our system as a practical example and explain some design decisions as well as the meta-model of questionnaires used by *Coast*. Eventually, we conclude this paper with a short outlook into future work in Section IV.

II. State of the Art

In the following, we explain the five big phases of an electronic survey and give a brief overview about existing solutions for creating and carrying out online surveys.

A. Phases of an Electronic Survey

A survey can be separated into five big phases [2]:

1) Planning, design, and implementation,
2) Data collection,
3) Data preparation,
4) Data analysis, and
5) Reporting.

The first phase, *Planning, design, and implementation*, includes *planning* on what you want to ask, *deciding* how to ask these questions and how to measure concepts of interest, and *specifying* who will be surveyed (the *population*). Furthermore, the survey will be implemented. That

 ISBN: 978-1-61208-606-4

implementation is used in the second phase to survey the population. This phase is called *Data collection*. Since the collected data may comprise malformed, incomplete, or obviously wrong records, the data has to be cleaned and prepared (*Data preparation* phase) to be used in a *Data analysis*, the fourth phase, afterwards. In this phase, the collected data is analysed in order to answer the questions from the first phase. The results of the evaluation as well as general information about the survey are finally summarized in detail in a report. The report is the outcome of the last phase, *Reporting*.

B. Other Survey Tools

Obviously, *Coast* is not the only survey tool available on the market supporting the phases of a survey. There are well-established tools with a lot of features like *EvaSys* [3], *Unipark* [4], *LimeSurvey* [5], *KwikSurveys* [6], and *SurveyMonkey* [7]. All of them cover parts of the previously introduced five phases of an electronic survey. They provide an implementation of the questionnaire by simply drag and drop the survey questions on a paper-like sheet in almost all products. The resulting questionnaire can be used for the data collection subsequently. In the case of *EvaSys*, it is possible to produce paper-based questionnaires too and scan them to collect the results.

The phases of data preparation, data analysis, and reporting are not clearly separated in most of the tools and, therefore, merge smoothly. All the tools have the possibility to export the collected data for further investigations in external tools (e.g., *IBM SPSS* [9], *Excel* [10], or *R* [11]). Furthermore, they provide standardized reports, which include the questions of the questionnaires, frequencies, significance tests, and mean values, among other things.

III. The System Architecture of *Coast*

The tools mentioned above show a clear focus on the development of the survey instead of its fine-granular and complex data analysis and reporting. As mentioned before, this was one reason to build an own survey tool *Coast*.

The first version of *Coast* had a monolithic software architecture caused by historical growing and changing requirements. Therefore, it showed the typical symptoms of monolithic software systems, e. g., changes on the business logic were difficult and risked the instability of the system. It is well-known that such architectures are some of the ”worst” forms of architectures established in practice [8].

As the questionnaires handled by *Coast* became more complex, the usage of the first *Coast* version was not longer possible. However, the tools introduced in Section II-B cannot handle our complex questionnaires without bigger changes on the questionnaire structure (and questions asked in the survey). Since we use longitudinal analyses [12] (which require equal questionnaires during each data collection), it is necessary to keep the existing structure intact. Therefore, it is *not* possible to use a different survey tool. This makes it necessary to transfer our tool *Coast* into a different architecture. We explain our decisions for the main architectural approaches in the following:

As mentioned in Section II-A, making an electronic survey is an almost well-defined process. The same holds true with the modelling of questionnaires. It is our goal for *Coast* to offer a proper, controlled, and realistic modelling and development of questionnaires, i. e., we want to provide a *domain-specific language* (DSL) to model questionnaires. As a result of this approach, the focus lies on the *meta-model* of a questionnaire, whose instance is in turn the result of the modelling. The focus on the meta-model of questionnaires makes it possible to derive different kinds of surveys from the same model: For example, surveys represented online on PC, on paper, or on smart phones. Therefore, we follow a *Model driven software development* (MDSD) approach [8].

By using a MDSD architecture, the models are transferred into, e. g., runnable surveys, by using different functions. Since these functions could be long running and computational expensive tasks, they should not affect, e. g., the data collection of a survey. That means, the functions should *scale* and be physical *separable*. With *Service-oriented architectures* (SOA) [8] such a separation and scalability is easy and, therefore, a good choice.

In general, *Coast* is a typical enterprise application consisting of a client-side, a database, and a server-side. Its architecture is built around the meta-model for questionnaires and reports (as illustrated in Figure 1). Furthermore, the architecture can be separated into the five phases of a survey (cf. Section II-A). Subsequently, the phase architecture of Figure 1 will be explained.

The phase of planning, designing and implementing surveys is solved with a *Survey designer* on the client-side. It is a web tool, which communicates via services to transmit changes on the model of the client- to the server-side. *Designer services* help to commit these changes from the designer to the model.

As mentioned before, different implementations of surveys can be derived originating from the *questionnaire model* via different *Survey generator services*. For example, this could be *paper-based surveys* or *online surveys*. These generated surveys can be used to collect the data subsequently. For the online surveys, there is also a *database*, in which the collected data is stored.

Besides different derivable surveys, the *report model* can be derived from the questionnaire model as well. This is done by a *Transformation service*. It transforms the questionnaire structures and transfers the texts (e. g., formulations of questions) to a new report model. Furthermore, the collected data from the surveys are verified based on the questionnaire model by *Preparation services*. During this verification process, malformed and incorrect data is identified. Therefore, the preparation services belongs to the data preparation phase.

The verified and collected data is used for the data analysis. The data is analysed with the functionality of the statistical programming language *R* via *Analysis and report services*. Therefore, complex calculations, like tests of significance, filters, or regressions, can be applied on the data. To structure the report and use the evaluated data, an *Analysis and report designer* is available. Since the report exists as model, it can be also transformed to different kinds of ”physical” reports, e. g., *online* or *paper-based reports*. This is done by the use of different *Report generator services*

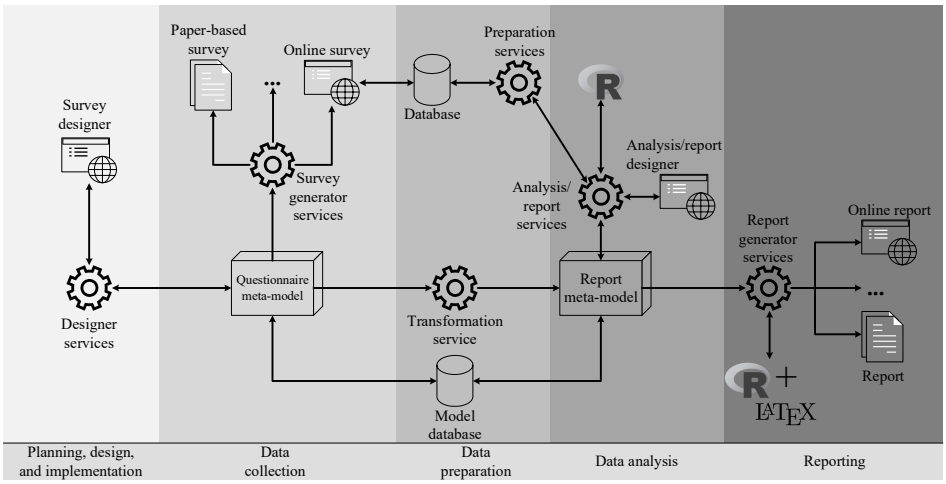

Figure 1. The architecture of the *Coast* system regarding the survey phases.

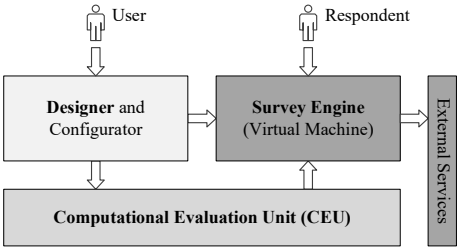

Figure 2. The component architecture of the *Coast* system.

and the inclusion of *R* and *LaTeX*.

Since the important parts of the architecture follow ongoing research in the context of psychoinformatics and compiler construction, they have to be flexible, simple, and evolutionarily growable. These requirements are complied by the use of the principles of service orientation and *microservices* [13]. Using (micro)services requires the *componentization* of the application into subsystems and modules (services).

In general, *Coast* disintegrates into three bigger subsystems: *Designer and Configurator* (in the following simply referred to *Designer*), *Survey Engine*, and *Computational Evaluation Unit* (CEU, illustrated in Figure 2). The *Designer* is the main application, in which the surveys will be defined, evaluated, and bundled in a report. In the *Survey Engine*, the surveys are conducted and the collected data is stored. The last subsystem CEU performs calculations for the analysis.

Each module of *Coast* is defined as a service with its own interface. If the functionality of the service is shared with other subsystems or a user, it will be provided as a *RESTful* web interface. Otherwise, it is a simple programming interface for reasons of performance. The arrows in Figure 2 explain how the subsystems interact with each other.

The *CEU* does not have a user interface as shown in the figure. It is a fully *computational* service only used for statistical calculations. It is based on the scripting language *R*, which is perfect for analysing big data information and to apply multivariate analysis methods. A *Coast* specific *R* library is supplied by an *openCPU* [14] server as a RESTful web service.

In contrast to the *CEU*, the subsystems *Designer* as well as the *Survey Engine* have a user interface. In the *Designer*, the user configures its surveys and transfers them to the *Survey Engine* where a participant can fill in the questionnaires — together, both subsystems follow an old computer science principle: Compilation and execution. Since the engine is separated from the designer, it is possible to have multiple *Survey Engine* service instances on different physical systems. As a result, the survey conduct becomes scalable. For this, the compiler (*Designer*) stores the survey in an intermediate representation (*IR*) called *liQuid* [15] and, afterwards, this IR will be transferred to the *Survey Engine*, which executes it. To guarantee equal data structures on both subsystem, they share the same modules.

It has become best practice during the *Coast* development to maintain each module (not only the subsystems) as a separate project and to build up the subsystems based on them. This approach helped us to generalize modules, to define proper interfaces, to get loosely coupled functionality, and to reuse code whenever possible.

For example, one module defines our meta-model for questionnaires. It is natural to think about questionnaires as sheets of paper with well-formulated questions. However, electronic surveys offer more ways to think of questionnaires than it is traditionally done in state-of-the-art tools. Instead, *Coast* treats questionnaires as program-like constructs. That means, in *Coast*, questionnaires have a well-defined start point, they have questions (*items*), whose order can be defined by acyclic graphs, and the order of these items depends on different conditions. In other words, questionnaires are acyclic *control flow graphs*.

Figure 3 shows the simplified meta-model of questionnaires used in *Coast*. It consists of the *questionnaire* with *edges* and *items*. Each edge has one item as *source* and one

 ISBN: 978-1-61208-606-4

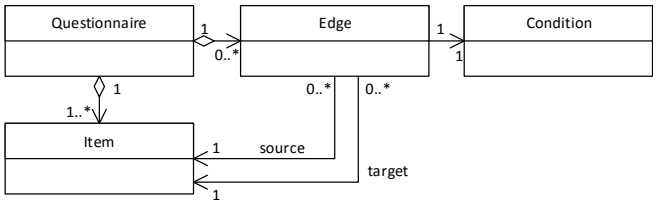

Figure 3. The meta-model of a questionnaire.

as *target* as well as a condition, which defines when this edge should be followed during the data collection.

First benefits of using such a control flow graph-based meta model of questionnaires were shown in previous work [15]. For example, we have a questionnaire, which is used to survey alumni of the university. Since all alumni have an individual career, the questionnaire became very complex with many so-called *adaptive* paths and variables. The visualization of the questionnaire model as a graph helped us to keep the overview of all paths and variables as well as to maintain the central theme. Furthermore, it is possible, to apply the complete palette of static and dynamic analyses of compiler construction, e. g., asking the same item multiple times on the same path.

The fine-grained meta-models for the definition of questionnaires and reports result in big models of sometimes hundreds to thousands objects. If a user wants to modify such a model within its web application, all those objects have to be taken from the database and have to be sent to the client. Sometimes, this took more than 30 seconds — which is unacceptable. Therefore, the pattern of *lazy loading* [16] objects was used. This was done by replacing each association of objects with a symbolic link, e. g., with an id and the corresponding object class. When the client application tries to access an associated object, the system loads the required object from the server automatically if it was not already loaded before. Nevertheless, to allow a straight forward modelling and programming, the concept of promises [17] was extended to an ordinary if-then-construct making the programming of asynchronous applications easy.

Certainly, there are some issues on *Coast* and its architecture. For example, sometimes, the performance of the system is slower than in the first monolithic application. This comes from the increased overhead by using a service-oriented architecture. Furthermore, there are sometimes a lot of messages being transferred between the different services. Especially, the communication between the client and the server during the design of a questionnaire is verbose. Although there are these weaknesses on the current design, the chosen one has shown its benefits so far.

IV. Conclusion and Outlook

This short paper explained the benefits of having a software in empirical sciences, which allows the control of all survey phases. As an example of such a tool, we introduced our tool *Coast*, which focuses on the analysis and report phases of a survey. Using the example of *Coast*, we explained our design decisions, why we used a model-driven software and a service-oriented architecture. Furthermore, parts of its architecture were presented as practical examples.

Since the *Coast* system is currently in an alpha version, *Coast* is unpublished up to now. In future versions, it should be available to everyone via the web. For this purpose, the application has to reach a stable stage and some of the concepts have to be extended.

References

[1] V. M. Sue and L. A. Ritter, Conducting Online Surveys, 2nd ed. Los Angeles, USA: SAGE Publications, 2012.

[2] J. Reinecke, Handbuch Methoden der empirischen Sozialforschung. Wiesbaden, Germany: Springer, 2014, vol. 1, ch. Grundlagen der standardisierten Befragung, pp. 601–617.

[3] Electric Paper Ltd., "Survey Automation Software - EvaSys and EvaExam," Website, available: http://en.evasys.de/main/home.html, retrieved: January, 2018.

[4] QuestBack GmbH, "Startseite — Unipark," Website, available: https://www.unipark.com/en/, retrieved: January, 2018.

[5] LimeSurvey GmbH, "LimeSurvey: the online survey tool - open source surveys," Website, available: https://www.limesurvey.org/, retrieved: January, 2018.

[6] Problem Free Ltd., "KwikSurveys: Free online survey & questionnaire tool," Website, available: https://kwiksurveys.com/, retrieved: January, 2018.

[7] SurveyMonkey, "SurveyMonkey: The Worlds Most Popular Free Online Survey Tool," Website, available: https://www.surveymonkey.com/, retrieved: January, 2018.

[8] O. Vogel, I. Arnold, A. Chughtai, E. Ihler, T. Kehrer, U. Mehlig, and U. Zdun, Software-Architektur: Grundlagen - Konzepte - Praxis, 2nd ed. Heidelberg, Germany: Springer, 2009.

[9] IBM United Kingdom Limited, "IBM SPSS Statistics - Overview - United Kingdom," Website, available: https://www.ibm.com/uk-en/marketplace/spss-statistics, retrieved: January, 2018.

[10] Microsoft, "Microsoft Excel 2016, Download Spreadsheet software — XLS XLSX," Website, available: https://products.office.com/en-gb/excel, retrieved: January, 2018.

[11] The R Foundation, "R: The R Project for Statistical Computing," Website, available: https://www.r-project.org/, retrieved: January, 2018.

[12] T. Mika and M. Stegmann, Handbuch Methoden der empirischen Sozialforschung. Wiesbaden, Germany: Springer, 2014, vol. 1, ch. Längsschnittanalyse, pp. 1077–1087.

[13] J. Lewis and M. Fowler, "Microservices — A definition of this new architectural term," martinfowler.com, Online publication, Mar. 2014, available: https://martinfowler.com/articles/microservices.html#footnote-etymology, retrieved: January, 2018.

[14] J. Ooms, "The OpenCPU System: Towards a Universal Interface for Scientific Computing through Separation of Concerns," Computing Research Repository (CoRR), vol. abs/1406.4806, 2014, pp. 1–23.

[15] T. M. Prinz, L. Gräfe, J. Plötner, and A. Vetterlein, "Statische Aanalysen von Online-Befragungen mit der Programmiersprache *liQuid* (Static Aanalysis of Online Surveys with the Help of the Programming Language *liQuid*)," in Proceedings 19. Kolloquium Programmiersprachen und Grundlagen der Programmierung, KPS 2017, Weimar, Germany, pp. 59–70, September 25–27, 2017.

[16] M. Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee, and R. Stafford, Patterns of Enterprise Application Architecture, 1st ed., M. Fowler, Ed. Boston, USA: Addison Wesley, 2003.

[17] B. Cavalier and D. Denicola, Promises/A+ Promise Specification, Open Access, Promises/A+ organization Std. 1.1.1, 2014, available: https://promisesaplus.com/, retrieved: January, 2018.

 ISBN: 978-1-61208-606-4