Please cite as: *K Stol & B Fitzgerald (2014) Research Protocol for a Case Study of Crowdsourcing Software Development. Available from: http://staff.lero.ie/stol/publications, University of Limerick.*

Research Protocol for a Case Study of Crowdsourcing Software Development

Klaas-Jan Stol
Lero—the Irish Software Engineering Research Centre
University of Limerick, Ireland
klaas-jan.stol@lero.ie

Brian Fitzgerald
Lero—the Irish Software Engineering Research Centre
University of Limerick, Ireland
bf@ul.ie

ABSTRACT
Crowdsourcing is an emerging topic within software engineering research. This report presents the protocol for our case study of crowdsourcing at a multi-national company. The findings of the case study are presented in a paper in the proceedings of the 36th International Conference on Software Engineering (2014) (see ref. [37]). This protocol presents additional details that provide more insight regarding the background, design and execution of our study. The research design can also be used for replicating the case study so as to be able to more easily compare different case studies.

Categories and Subject Descriptors
K.6.3 [**Software Management**]: Software development, Software process; D.2.8 [**Software Engineering**]: Management—*Programming teams*; K.4.3 [**Organizational Impacts**]: Computer-supported collaborative work

General Terms
Management, Human Factors, Theory

Keywords
Crowdsourcing software development, case study, protocol, empirical study, epistemology, philosophical stance, research tradition

1. INTRODUCTION
Crowdsourcing is gaining significant attention in the software engineering research [4, 28]. Crowdsourcing has been suggested as a useful approach in GUI testing [13], performance testing [28] and even as a means to recruit participants in empirical studies of software engineering [38]. There is an increasing level of attention to social interactions and networks within software engineering research [3], and 'crowds' are an important aspect of this [6]. We are in particular concerned with using crowds as an alternative form of sourcing, contrasting it with other forms such as open-sourcing [1], inner-sourcing [35] and traditional software outsourcing. In other words, how can a crowd, or 'unknown workforce' effectively contribute to the development of a software system?

Much research has focused on general-purpose crowdsourcing platforms such as Amazon's Mechanical Turk (AMT) [19]. However, very little research exists on crowdsourcing software development, in contrast to the topic of crowdsourcing in a more general sense. We argue that there is significant potential in software development through crowdsourcing, but that much research is needed to better understand how to optimally do this. This report presents the study protocol for our case study on crowdsourcing software development [37], and is structured as follows. Section 2 presents the research goal and method used for the study. Section 3 discusses the development of the theoretical framework prior to the field-work for this study. A brief summary of the framework is also presented. Section 4 discusses data collection for the study. Section 5 discusses data analysis methods, and also the process of reporting our study. Section 6 discusses validity issues of the study and how these were mitigated. Section 7 briefly summarises this report.

2. RESEARCH GOAL AND METHOD
The goal of our study was *to investigate crowdsourcing in a software development context from a crowd-sourcing customer perspective, to better understand this process and the challenges associated with it.*

Since crowdsourcing in the context of software development is an emerging topic with very little in-depth research, we decided to conduct an industry case study. In a crowdsourcing scenario, there are three distinct stakeholders – the crowdsourcing customer, the crowdsourcing workers and the crowdsourcing platform provider. While there have been a small number of studies focusing on crowdsourcing platforms and crowdsourcing workers, to our knowledge there have been no studies to date of crowdsourcing software development from a customer perspective.

Case study research is a highly appropriate method to explore contemporary phenomena within a real-world setting, where it is difficult to draw clear boundaries between the subject of study and the context within which it takes place [42]. Furthermore, case study research is very useful to answer "how" and "why" questions. Given that we were interested in achieving an understanding of *how* crowdsourcing software development works in practice, we deemed case study method an appropriate choice.

Other common research methods in software engineering are surveys and controlled experiments, both of which offer a number of benefits over case study research. Surveys, for instance tend to result in findings that have a higher degree of generalizability, whereas controlled experiments offer the ability to quantitatively study relationships (e.g., causal) between different constructs relating to, for instance, project success. However, given that the state of research on crowdsourcing software development is still in its nascent phase (i.e. no body of published literature on organizations that use crowdsourcing as a strategy for software development, limiting the opportunity to conduct field surveys), we chose to conduct an industry case study.

The case study method has a number of design options:

- *Single* versus *multiple*;

- *Embedded* versus *holistic*;

Our case study was designed as a *holistic, single-case study*. By adding additional case studies at different organizations (for which this protocol could offer guidance), our study could be extended to a multiple-case study, facilitating a comparative analysis between the results of the different cases. This is one direction for future work.

Furthermore, our case study was **holistic** as opposed to embedded. This means that the case was the unit of analysis; in an embedded case study design, a case could contain different units of analysis.

Another characterization of case studies that can be made is whether they are [30]:

- **Descriptive** – portray the current status of a situation or a certain phenomenon;
- **Exploratory** – seek new insights; generate ideas and hypotheses for further research;
- **Explanatory** – seeking to explain a situation, possibly by identifying causal relationships between constructs;
- **Improving** – attempting to improve a certain aspect of a studied topic or phenomenon.

We position our case study as **exploratory**, as it sought to generate new insights. This is different from descriptive case studies, which tend to be used to illustrate certain events and their specific context.

One consideration in case study design is the ontological and epistemological stance that researchers take. These stances relate to what a researcher considers to be 'knowledge,' and how that knowledge should be acquired. The debate on epistemology has been particularly strong in the information systems (IS) field [15], but has hardly attracted any interest from software engineering (SE) researchers, and mostly left implicit. The philosophical basis underpinning of an empirical study affects the assumptions made in a study as well as how a study is designed. A lack of understanding of the assumptions underpinning a study design may therefore limit a reader's appreciation of the study's findings.

This tension can arise from a fundamental mismatch between the assumptions of a researcher conducting the work on the one hand, and readers who have a different set of assumptions regarding the nature of knowledge. While there a variety of epistemological stances, we illustrate this with the two stances best known, namely positivism and interpretivism. Some assumptions of positivist researchers are [15]:

- There is a single objective truth that can be discovered independent of an individual's cognition;
- Complexity can be resolved by reductionism;
- Focus on quantification and measurement.

On the other hand, some assumptions of the interpretivist stance are:

- Multiple realities exist as subjective constructions of the mind;
- A study's findings emerge from interaction between a researcher and a research situation;
- Focus on "thick descriptions" to incorporate natural context.

These different beliefs about what constitutes "knowledge" will influence a researcher's choice of research methods (e.g., data collection). Furthermore, they also affect a reader's evaluation criteria and expectations of a research report. A positivist reader expecting a controlled experiment that measures a number of constructs (representing a simplified yet "valid" view of the real-world) in order to identify a causal relationship, is likely to be disappointed when confronted with an interpretivist study that presents 'thick' descriptions of a certain phenomenon in which no causal relationship is identified nor tested.

The debate is often characterized (and oversimplified) as being 'hard' versus 'soft,' with the positivist stance representing the former, and interpretivism representing the latter. Positivist research could roughly be characterized as being quantitative, based on reductionism, constructs, propositions and hypotheses, whereas interpretivist research depends on "thick descriptions," multiple world views using qualitative methods. The software engineering research community has increasingly adopted qualitative approaches in the last 15 years or so, but there has so far been no consideration of the epistemological underpinnings that are usually associated with such qualitative approaches. Consequently, qualitative studies may still be criticized for "lacking control" for example, while interpretivist researchers may not necessarily aim at "generating truth" but rather at making interpretations *"available in the 'consultable record' "* [41].

There are numerous reference works on case study methodology, and there are different types of case study research. Yin [42], for instance, takes an implicit positivist stance and consequently his discussion of case study research features activities that are typically found in the positivist school. For example, his guidance includes the development of *propositions*, a recommendation that has been echoed by software engineering researchers as well [40]. Other hints include the use of terms such as "internal validity," "external validity," and "reliability," terms which make sense in quantitative studies where a model or theory is constructed and hypotheses are tested. In studies of a more qualitative nature, however, these terms are not necessarily applicable. For instance, whereas in a positivist research philosophy there is much attention for identifying causal relationships between constructs, the goal of qualitative studies is often to gain an understanding, rather than testing a hypothesis. Thus, the term "internal validity" has little meaning in qualitative studies.

Besides this postivist approach to conducting case studies, there is also an interpretivist approach. Walsham discusses how case studies in the information systems (IS) field can be conducted in the interpretivist philosophy [41].

A common concern about case study methodology is a "lack of control" as would be present in, for instance, controlled experiments (as the term implies). However, the goal of case studies is not to manipulate, or control behavior [42]; in fact, given that the boundaries between the phenomenon being studied and the context in which that phenomenon is embedded are *"not clearly evident"* [42].

Braa and Vidgen identified three research outcomes, namely that of **prediction**, **understanding** and **change** [7]. The desired research outcome will therefore dictate the choice of the research strategy and method. For instance, when the targeted research outcome is the ability to make a confident prediction, one would choose a controlled experiment where relationships between a number of constructs (which together represent a reductionist view of the real world) are studied. If, on the other hand, the goal is to make a change, one would be wiser to adopt the action research method. The aim of our study was to gain an understanding of crowdsourcing software development, and thus we adopted, what Braa and Vidgen called a "soft" case approach. These three research outcomes together represent a framework to position different research methods – as shown in Figure 1.

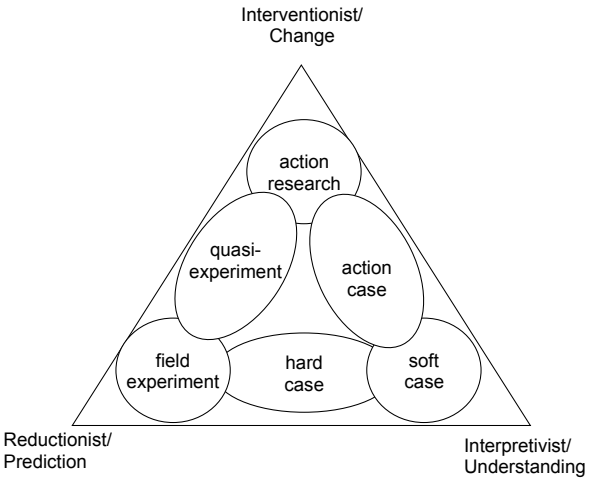

Figure 1. Research outcomes and research methods. Adapted from Braa and Vidgen [7].

The framework proposed by Braa and Vidgen represents one view of how research methods are linked to research goals—numerous other classifications of research methods have been proposed (see for example Runkel and McGrath's discussion on this topic [31]). They consider techniques such as surveys and interviews to be orthogonal to the methods in the figure. For instance, surveys could be designed with an aim to achieve statistical generalizability (i.e., a high degree of external validity), and as such they would fall within the reductionist/prediction corner of the figure. Alternatively, surveys that collect qualitative data that are subsequently analyzed using qualitative analysis techniques (as opposed to quantitative/statistical techniques) fit better in the interpretivist/understanding corner of the figure.

3. THEORETICAL FRAMEWORK

Prior to the field work we conducted a comprehensive literature review to identify key topics that have received attention within the crowdsourcing literature. Together these topics form a framework, which established the boundaries of our study [33].

3.1 Development of the Framework

Our first step in developing an understanding of the crowdsourcing phenomenon was the study of some of the seminal works in this area [8, 17, 18]. This established a common vocabulary and understanding among the researchers about the concept of crowdsourcing.

Our literature review was a traditional one, as opposed to a systematic literature review (SLR), which has become a common approach in software engineering research. SLRs have a number of associated benefits, such as repeatability and the ability to be more exhaustive than traditional reviews. Consequently, SLRs are suitable for conducting extensive surveys of a research field and to develop a taxonomy, for instance. However, the goal of our literature review was not to develop such a taxonomy that captures the whole crowdsourcing research field, nor was our goal to identify as many studies as possible, as would be desirable when reviewing empirical studies so as to synthesize all the evidence pertaining to a certain research question. Rather, given the exploratory goal of our study, we sought to identify a set of key concerns that would be important considerations given the nature of the crowdsourcing phenomenon, and which would be of particular importance in a software engineering context.

The literature review thus started with a number of searches in search engines and digital libraries (Google Scholar, IEEE Xplore, ACM Digital Library). Through an iterative approach we identified a set of papers which grew to a collection of 160 papers. Based on an analysis of the papers' titles and abstracts, we identified a number of candidate topics that we deemed of interest. We then read a number of papers in each category in more detail to capture some of the key insights that were presented. As we became increasingly immersed in the crowdsourcing literature, we reflected on whether the themes were (a) relevant to software engineering, (b) how parsimonious they were, and (c) whether they represented a sufficiently complete set of topics that were relevant to our study. We revisited our initial choice of themes making a few small changes to finalize our framework (e.g., the theme of "intellectual property" now includes the topic of "knowledge").

3.2 Key Concerns in Crowdsourcing Software Development

Our theoretical framework consists of six themes, which are described in detail in [37]. A brief description of each theme follows in sections 3.1.1 to 3.1.6.

3.2.1 Task decomposition

Development of a significant software systems cannot be done by a single person in a crowd. In order to benefit from a potentially large crowd, the system should be split up into many small pieces that can be developed in parallel by different developers in the crowd. This raises an age-old question in software engineering, namely, how should the system be decomposed into smaller modules without causing problems in putting them back together once they are developed [20, 21, 22]. Common questions in software engineering within the scope of decomposition relate to assumptions, interfaces and dependencies. While dependencies are an important consideration in decomposing a system, managing these dependencies through coordination and communication is part of a second theme, discussed next.

3.2.2 Coordination and Communication

While task decomposition is mainly concerned with the question of how to decompose a system to be developed into manageable chunks of work, coordination is concerned with the process of managing the dependencies between these activities [26]. Coordination is important to ensure that activities are performed in a timely fashion and that together they achieve the ultimate goal of building a system. To achieve this, communication is needed between the developers and the customer.

3.2.3 Planning and Scheduling

With crowdsourcing, timely delivery of software implementations becomes much more uncertain than when development is done in-house, or in 'normal' outsourcing scenarios where delivery is subject to a negotiated contract. One potential benefit of crowdsourcing is a quicker delivery as the work can be split up in smaller tasks which can then be executed in parallel. On the other hand, given that crowdsourcing competitions cannot really be expedited once the deadline is set, it is not possible to intervene to achieve faster delivery. Therefore, important questions in crowdsourcing software development are related to how a timely delivery of a software project can be guaranteed when portions are crowdsourced to an unknown workforce.

3.2.4 Quality Assurance

Some crowdsourcing advocates claim that, given a large number of submissions (from a large enough crowd), the resulting output will be of high quality [5, 32], thus addressing a key concern in software engineering. Also, similar to Linus's Law, namely that given a sufficiently large group of people, there is bound to be someone who knows how to fix a certain defect, a similar line of thinking would argue that there is a wide variety of expertise available in the crowd. In other words, whatever the software development task at hand, there is bound to be someone who has sufficient domain expertise to provide a solution to a given software development task.

3.2.5 Knowledge and Intellectual Property

Software development is a knowledge-intensive task, and knowledge sharing and management plays an important part throughout the software development lifecycle [2]. A key difference between in-house development and traditional outsourcing scenarios on the one hand, and crowdsourcing on the other hand is that the latter is characterized by a possibly continuous turnover of workers [11].

3.2.6 Motivation and Remuneration

Motivation and remuneration are topics that have received significant attention in the crowdsourcing literature [9, 12, 14, 16]. Crowdsourcing tasks on platforms such as Amazon's Mechanical Turk, sometimes referred to as 'micro-tasks,' tend to be very short in duration, and only a small remuneration is offered for those, usually less than one US dollar [19]. As software development tasks are much more complex, one can no longer speak of micro-tasks as they tend to be interdependent, long in duration (days/weeks as opposed to seconds/minutes), and requiring a great deal of cognitive effort. Therefore, remuneration for such complex tasks must be significantly higher than micro-tasks. An important consideration for a crowdsourcing customer is to decide on an appropriate remuneration that will attract sufficient participants to a crowdsourcing contest. Furthermore, participants who have a lot of experience with crowdsourcing may have a significant advantage over those who are inexperienced, in that they may be more proficient with a platform, and thus may be more likely to win a crowdsourcing contest. Whether or not this puts off inexperienced participants, to the extent of 'scaring them away' would be a concern for a crowdsourcing customer as this reduces participation and may affect the number of solutions offered.

4. DATA COLLECTION

This section presents the data collection techniques we applied, namely interviews and documentation study. For the interviews, we purposive selected key informants that would be able to provide us with useful information.

Lethbridge et al. present a taxonomy of data collection methods for field studies in software engineering research, and distinguished three levels of engagement [25]. These 'first,' 'second,' and 'third' degree types of data collection methods are categorized according to the degree of human contact that is required. These three degrees of interaction are:

- **First degree**: requires direct access to a participant population; e.g. interviews.
- **Second degree**: requires access to participants' environment but no direct access to participants themselves is necessary; e.g., observation.
- **Third degree**: requires access to work artifacts only, e.g. source code and documentation; e.g., document analysis.

We applied methods of the first degree and third degree. In particular, we conducted a number of semi-structured interviews (first degree), and we studied documentation available from the case company as well as data that were available from the crowdsourcing platform that the case study was using (third degree). We discuss data collection through interviews and documentation study in order due to the inherent linear structure of this report. However, the two modes of data collection happened in parallel, in an alternating fashion.

4.1 Interviews

4.1.1 Interview Design

Prior to conducting the field study, i.e., interviews, we developed an interview guide based on the framework identified in Section 3. Furthermore, we also received a short report from the case company that outlined a short description of the crowdsourced project as well as names of a number of the key persons involved (see also Section 4.2). The interview guide was used primarily for the first round of interviews. After analysis of the initial interviews (see also Section 5) we identified further questions to clarify details that had not become clear initially, or to validate some of our assumptions. Subsequent interview sessions helped in answering any outstanding questions.

4.1.2 Selection of Participants

The choice of selection of participants is an important decision in the design of a research study. In selecting participants we applied the principle of purposive sampling, where it is more important to include informants who are closely involved who can offer rich insights, rather than to necessarily identify a certain number of participants.

Participants were selected based on their level of involvement with the crowdsourcing initiative at the case company. We conducted interviews with the following participants:

- Divisional CTO
- Software architect
- Software development manager
- Program manager
- Project manager

4.1.3 Execution of the Interviews

Prior to these visits one of the involved researchers had an exploratory discussion with the divisional CTO so as to set the goal and scope of the research. We conducted the interviews on site during three company visits and two teleconference calls. In total, we spoke to five people. The on-site interviews were conducted in three half-day workshops, and resulted in seven hours of interviews that were transcribed for further analysis.

The extensive time period of eight months allowed us to analyze the data during data collection. While this does not constitute a *longitudinal* case study whereby there is a monitoring of events and trends over a significant amount of time or a larger number of data collections, we do consider eight months to be sufficient for establishing an in-depth analysis of the case at hand. As soon as interviews were transcribed they could be analyzed. Through an iterative process of data collection and analysis, we could focus our follow-up questions very specifically on issues that we were not yet clear about.

4.2 Supporting Documentation

In addition to the semi-structured interviews, we also gathered data through study of documentation that was available. Specifically, the case company had written a short report with a

number of problems they were facing, as well as a brief description of the project and key persons that were involved. This document was used to identify the informants for the interviews. This report also helped us in gaining an understanding of the domain that the company were working in, so as to get a grasp of the terminology, which would help in conducting the interviews. Furthermore, we inspected the specification documents for the various crowdsourcing contests. This form of data source triangulation permitted us to cross-check facts, figures and findings; this is a general recommendation to establish a study's dependability.

5. DATA ANALYSIS AND REPORTING

The collected data were analyzed using qualitative techniques described by Seaman [34]. All interviews were transcribed, resulting in approximately 112 pages of text (A4 format, 10 points font, single line-spacing). The analysis consisted of coding the transcripts using the six themes of our framework (see Section 3) as seed categories. The transcripts were analyzed in parallel by both authors and several analytical memos were written. The memos established an audit trail of the analysis, and facilitated a process of peer debriefing for the researchers.

Of key importance is that the analysis results in findings that correctly reflect the insights and opinions of the participants. In order to address this we applied the tactic of member checking. We sent several drafts of our paper to the interviewees so as to ensure that our report correctly reflected the participants' intended answers and insights.

6. VALIDITY OF THE STUDY

All research studies are limited in one or more aspects and have associated threats to validity. Our study is no exception. In this section we discuss the validity, or *trustworthiness*, of our study. While a number of tactics and elements in our study design (e.g., member checking, audit trail) were already presented as part of the study protocol (see Sections 4 and 5), we will briefly reiterate these practices as part of our discussion of the validity issues below.

A standard set of of validity criteria are internal validity, external validity, reliability and objectivity. Given that we collected mostly qualitative data we felt it was more appropriate to use an alternative set of validity criteria that are more suitable to consider these issues for such a qualitative study. We have previously used these criteria in a multiple-case study of inner-sourcing, a topic somewhat related to crowdsourcing [35]. These alternative criteria are credibility, transferability, dependability and confirmability. These are discussed below.

6.1 Credibility

Credibility is concerned with the extent to which we can have confidence in the findings, answering the question: *How plausible are the findings?* This question can be posed with respect to the two components of our study, namely the framework development and the empirical study.

Regarding the framework development, we should consider the question as to how our research design established the credibility of our framework comprising six key concerns (see Section 3). As we outlined in Section 3, these concerns were identified through a literature review following a process that should be characterized as "traditional," as opposed to a "systematic" literature review. Our literature review considered a significant number of papers, which gave us confidence that we had identified the key issues that are important in crowdsourcing software development. Furthermore, through a process of peer-debriefing, we discussed these six concerns extensively. Indeed, as the original set of concerns was discussed and one of us posited some additional concerns as being important, the other researcher played 'devil's advocate' by critically gauging their importance [10]. Thus, we argue that the final set of key concerns came from a sufficiently rigorous process of reviewing the literature and deliberation among the two researchers involved.

With respect to the credibility of the empirical study, we wish to cite Leininger, who wrote that *"crediblity refers to the truth as known, experienced, or deeply felt by the people being studied (emic or local) and interpreted from the findings with co-participant evidence as the 'real world,' or the truth in reality"* [24]. One recommended approach to ensure that findings are indeed "experienced" or "felt by" the participants of a study is to adopt a tactic of member checking. We sent several preliminary versions of our study report to the interviewees to solicit feedback. This resulted in further clarifications that we subsequently incorporated into our report.

6.2 Transferability

Transferability refers to the extent to which findings of a study can be applied in other settings. This answers the question: *To what extent are our findings relevant in other cases of crowdsourcing?*

Clearly, our case study was a singular one, and no *statistical* generalizations can be drawn from this. However, other forms of generalizations exist [23]. Walsham identifies a set of four alternative types of generalization [41]:

1. Development of concepts
2. Generation of theory
3. Drawing of specific implications
4. Contribution of rich insights

Each of these types of generalizations are present in our study. Firstly, we developed a number of concepts, namely our theoretical framework that consists of six key concerns in crowdsourcing software development. These concepts will be concerns for any customer who will be crowdsourcing software development.

Secondly, while our study does not result in a fully developed theory, we do set forth some propositions—a *theory fragment* [36]. For instance, we found that crowdsourcing software development is more useful in self-contained and independent tasks, as opposed to complex software components that exhibit a high degree of interdependencies.

Additionally, we were able to draw a number of specific implications. For instance, our findings suggest that the development process followed by the crowdsourcing platform is inherently a waterfall one. The immediate implication of this is that the company will encounter challenges when trying to synchronize this with their agile development process.

Furthermore, through "thick" descriptions our study contributes rich insights that can be useful to researchers who are studying crowdsourcing software development, as well as customers who may plan to embark on a crowdsourcing initiative.

The main purpose of our case study was to present an in-depth investigation of how crowdsourcing software development is done. The selection of the case study organization is based on *purposive sampling* [29], i.e., the selection is based on a researcher's judgment as to the suitability of a participant (in this case, an organization).

In case studies it is important to present sufficient context to understand an organization's constraints and behavior. To enable readers to gauge the extent to which findings could be useful in other contexts, we aimed at presenting "thick" descriptions, whose purpose is to *"create verisimilitude, statements that produce for the readers the feeling that they have experienced, or could experience, the events being described in a study"* [10].

6.3 Dependability

Dependability refers to the extent to which the data are "stable," and addresses the issue of how reliable the findings are, and whether any variance in those findings can be traced and explained. In short: to what extent can a researcher depend on the correctness of his or her findings?

We applied a number of common tactics to establish dependability. Firstly, we triangulated across different data sources: (i) we conducted a number of in-depth interviews with key informants; (ii) we studied documentation that was provided by the company (including details of the contests that were run on TopCoder) (iii) we retrieved data from the TopCoder platform (using the contest details mentioned above). Through this form of triangulation, we were able to confirm findings, or where issues remained unclear we requested clarification in further interviews.

Another tactic is that of establishing an audit trail. In our study, this trail consists of the original source material of the interviews, transcriptions, memos and spreadsheets used during analysis, and the final paper that presents the results. The audit trail helps to establish dependability, as it allows *"another investigator to follow the cognitive development of a project as it developed"* [27, p.24]. Thus, as we both were involved in this research progressed in independent data analysis, we were able to follow each other's cognitive processes and confirm that any interpretations and presentation of the results were in agreement.

6.4 Confirmability

A study's confirmability considers the neutrality aspect of a study, i.e., can findings be confirmed by others. Different investigators, and indeed, participants, may have different experiences or impressions in a study. These 'multiple realities' [39] may diverge as a result of, for instance, a researcher's subjective understanding of the topic under study. The attitudes of the participants involved in a study may also have an impact.

To address this concern, we employed member-checking (as mentioned above) by sending preliminary versions of our report to the informants of our study. This helps in assessing the extent to which we correctly captured the informants' insights and experiences. Furthermore, the triangulation of data sources (interviews, document study, contest data from the crowdsourcing platform), and triangulation of investigators (i.e., two researchers involved) are also recommended practices to establish a study's confirmability. Peer-debriefing, which we used in establishing the credibility of our framework, was also used in the empirical phase of our case study—the two researchers discussed the findings at great length, both in face-to-face meetings and in analytical memos that were exchanged.

7. SUMMARY

In this document we presented a protocol for conducting case studies on crowdsourcing software development. Developing a protocol prior to conducting data collection in the field is a recommended approach in case study research, as it helps to establish a focus of the topic under study as well as a vehicle to reach agreement among researchers. Furthermore, by making the protocol available, readers who are interested in the research methodology used for a particular study can inspect this protocol so as to become confident that a sound and rigorous approach was used. Additionally, given the nascent state of research on crowdsourcing software development, other researchers can use and/or adapt this protocol to use in future and/or replication studies.

8. ACKNOWLEDGMENTS

We are grateful to the participants in our study for their time and insights. This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero—the Irish Software Engineering Research Centre (www.lero.ie).

9. CHANGE LOG

- V1.0 – First complete draft.
- V1.1 – removed dangling reference. Completed a reference.

10. REFERENCES

[1] Ågerfalk, P.J. and Fitzgerald, B. 2008. Outsourcing to an unknown worforce: Exploring opensourcing as a global sourcing strategy, *MIS Quarterly*, 32, 2.

[2] Aurum, A., Jeffery, R., Wohlin, C. and Handzic, M. 2003. *Managing Software Engineering Knowledge*, Springer.

[3] Begel, A., Bosch, J. and Storey, M.A. 2013. Social Networking Meets Software Development: Perspectives from GitHub, MSDN, Stack Exchange, and TopCoder, *IEEE Software*, 30, 1.

[4] Begel, A., Herbsleb, J.D. and Storey, M.-A. 2012. The Future of Collaborative Software Development. *Proc. Computer Supported Cooperative Work*.

[5] Bonabeau, E. 2009. Decisions 2.0: The Power of Collective Intelligence, *MIT Sloan Manage Rev*, 50, 2, 45-52.

[6] Bozzon, A., Brambilla, M., Ceri, S., Silvestri, M. and Vesci, G. 2013. Choosing the right crowd: Expert finding in social networks. *EDBT/ICDT*.

[7] Braa, K. and Vidgen, R.T. 1999. Interpretation, intervention and reduction in the organizational laboratory: a framework for in-context information systems research, *Information and Organization*, 9, 1, 25-47.

[8] Brabham, D.C. 2013. *Crowdsourcing*, MIT Press.

[9] Chandler, D. and Kapelner, A. 2013. Breaking monotony with meaning: Motivation in crowdsourcing markets, *Journal of Economic Behavior & Organization*, 90, 123-133.

[10] Creswell, J.W. and Miller, D.L. 2000. Determining Validity in Qualitative Inquiry, *Theory into Practice*, 39, 3, 124130.

[11] Dabbish, L., Farzan, R., Kraut, R. and Postmes, T. 2012. Fresh Faces in the Crowd: Turnover, Identity, and Commitment in Online Groups. *Proc. CSCW*. ACM.

[12] DiPalantino, D. and Vojnovic, M. 2009. Crowdsourcing and all-pay auctions. *Proc. 10th ACM Conf. Electronic Commerce*.

[13] Dolstra, E., Vliegendhart, R. and Pouwelse, J. 2013. Crowdsourcing gui tests. *6th International Conference on Software Testing, Verification and Validation*.

[14] Faridani, S., Hartmann, B. and Ipeirotis, P.G. 2011. What's the Right Price? Pricing Tasks for Finishing on Time. *Proc. AAAI Workshop on Human Computation*.

[15] Fitzgerald, B. and Howcroft, D. 1998. Towards dissolution of the IS research debate: from polarization to polarity, *Journal of Information Technology*, 13, 313-326.

[16] Horton, J.J. and Chilton, L.B. 2010. The Labor Economics of Paid Crowdsourcing. *Proc. Conf. Electronic Commerce*.

[17] Howe, J. 2006. The Rise of Crowdsourcing, *Wired*, 14.

[18] Howe, J. 2008. *Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business*, Crown Business.

[19] Ipeirotis, P.G. 2010. Analyzing the Amazon Mechanical Turk marketplace, *XRDS*, 17, 2, 16-21.

[20] Ipeirotis, P.G. and Paritosh, P.K. 2011. Managing Crowdsourced Human Computation. *WWW*.

[21] Kittur, A., Smus, B., Khamkar, S. and Kraut, R.E. 2011. CrowdForge: Crowdsourcing Complex Work. *Proc. ACM Symposium on User Interface Software and Technology.*

[22] Kulkarni, A., Can, M. and Hartmann, B. 2012. Collaboratively Crowdsourcing Workflows with Turkomatic. *Proc. Computer-Supported Cooperative Work.*

[23] Lee, A.S. and Baskerville, R.L. 2003. Generalizing Generalizability in Information Systems Research, *Information Systems Research*, 14, 3, 221-243.

[24] Leininger, M. 1994. Criteria and Critique, in: J.M. Morse (Ed.) Critical Issues in Qualitative Research Methods, SAGE Publications.

[25] Lethbridge, T.C., Sim, S.E. and Singer, J. Studying Software Engineers: Data Collection Techniques for Software Field Studies, *Empirical Software Engineering*, 10, 311-341.

[26] Malone, T.W. and Crowston, K. 1994. The Interdisciplinary Study of Coordination, *ACM Comput Surv*, 26, 1.

[27] Morse, J.M. 1994. "Emerging From the Data": The Cognitive Processes of Analysis in Qualitative Inquiry, in: J.M. Morse (Ed.) Critical Issues in Qualitative Research Methods, SAGE Publications.

[28] Musson, R., Richards, J., Fisher, D., Bird, C., Bussone, B. and Ganguly, S. 2013. Leveraging the crowd: how 48,000 users helped improve Lync performance, *IEEE Softw.*, 30, 4.

[29] Robson, C. 2002. *Real World Research*, 2nd ed., Blackwell Publishers.

[30] Runeson, P., Höst, M., Rainer, A. and Regnell, B. 2012. *Case Study Research in Software Engineering: Guidelines and Examples*, Wiley.

[31] Runkel, P.J. and McGrath, J.E. 1972. *Research on Human Behavior: A Systematic Guide to Method*, Holt, Rinehart & Winston, New York.

[32] Schenk, E. and Guittard, C. 2009. Crowdsourcing: What can be outsourced to the crowd, and why?

[33] Schwarz, A., Mehta, M., Johnson, N. and Chin, W.W. 2007. Understanding Frameworks and Reviews: A Commentary to Assist us in Moving Our Field Forward by Analyzing Our Past, *Database Adv Inform Syst*, 38, 3.

[34] Seaman, C. 1999. Qualitative Methods in Empirical Studies of Software Engineering, *IEEE Trans Softw Eng*, 24, 4.

[35] Stol, K., Avgeriou, P., Babar, M.A., Lucas, Y. and Fitzgerald, B. 2014. Key Factors for Adopting Inner Source, *ACM Transactions on Software Engineering and Methodology*, 23, 2.

[36] Stol, K. and Fitzgerald, B. 2013. Uncovering Theories in Software Engineering. *2nd SEMAT Workshop on a General Theory of Software Engineering*. IEEE.

[37] Stol, K. and Fitzgerald, B. 2014. Two's Company, Three's a Crowd: A Case Study of Crowdsourcing Software Development. *36th International Conference on Software Engineering (ICSE)*. ACM.

[38] Stolee, K.T. and Elbaum, S. 2010. Exploring the use of crowdsourcing to support empirical studies in software engineering. *ESEM*.

[39] Swanson, J.M. and Chapman, L. 1994. Inside the Black Box: Theoretical and Methodological Issues in Conducting Evaluation Research Using a Qualitative Approach, in: J.M. Morse (Ed.) Critical Issues in Qualitative Research Methods, SAGE Publications.

[40] Verner, J.M., Sampson, J., Tosic, V., Abu Bakar, N.A. and Kitchenham, B.A. 2009. Guidelines for Industrially-Based Multiple Case Studies in Software Engineering. *Third International Conference on Research Challenges in Information Science*.

[41] Walsham, G. 1995. Interpretive case studies in IS research: nature and method, *European Journal of Information Systems*, 4, 74-81.

[42] Yin, R.K. 2003. *Case Study Research*, 3rd ed., SAGE.