Software developers need help too! Developing a methodology to analyse cognitive dimension-based feedback on usability

Article *in* Behaviour and Information Technology · December 2019

DOI: 10.1080/0144929X.2019.1705393

3 authors:

Chamila Wijayarathna
UNSW - Canberra
12 PUBLICATIONS 32 CITATIONS
SEE PROFILE

Marthie Grobler
The Commonwealth Scientific and Industrial Research Organisation
107 PUBLICATIONS 276 CITATIONS
SEE PROFILE

Nalin Asanka Gamagedara Arachchilage
University of New South Wales, UNSW Canberra at ADFA
82 PUBLICATIONS 837 CITATIONS
SEE PROFILE

Software Developers need Help too!

Developing a Methodology to Analyse Cognitive Dimension Based Feedback on Usability

Chamila Wijayarathna *(c.diwelwattagamage@student.unsw.edu.au)*[1, 2], Marthie Grobler[2], and Nalin A. G. Arachchilage[1, 3]

[1]School of Engineering and Information Technology, University of New South Wales-Canberra, ACT, Australia

[2]CSIRO's Data61, Victoria, Australia

[3]Optus La Trobe Cyber Security Research Hub, Department of Computer Science and IT, School of Engineering and Mathematical Sciences, La Trobe University, Victoria, Australia

ABSTRACT

Software developers use various methods to evaluate usability and identify usability issues exist in systems they develop. Cognitive Dimensions Framework (CDF) based usability evaluation is one of the popular usability evaluation methods. It uses an open-ended questionnaire to collect qualitative feedback from users after using a system. To identify usability issues, evaluators should analyse this qualitative feedback. However, the approach to follow when performing this analysis is not explored in detail. We conducted a systematic literature review and reviewed 70 studies that used various CDF questionnaires for usability evaluations and investigated how those studies have analysed CDF questionnaire responses to identify usability issues. This revealed five methods that previous research has used for data analysis and four methods for identifying usability issues from CDF questionnaire responses. We applied the results of the literature review to develop a methodology and a set of guidelines to analyse qualitative feedback collected via a CDF questionnaire that targets evaluating security APIs. We tested the developed guidelines by conducting an empirical investigation. The results of the experiment revealed that using the proposed guidelines helps to identify significantly more usability issues with a higher validity.

Keywords

Usability; Usability Testing; Cognitive Dimensions Framework; Qualitative Data Analysis

1. INTRODUCTION

Lack of system usability negatively affects the experience of people who use those systems in various ways [36, 37, 58]. It not only reduces their efficiency and satisfaction of using the system (e.g. user will take more time to complete a task using an unusable system, user will be frustrated after using an unusable system), but can drive people to unintended harmful behaviours (e.g. enter an unexpected data item to a data field, use an easy to guess password) that could make the systems vulnerable to cyber-attacks [3]. Not only systems that are used by end-users, tools, libraries and Application Programming Interfaces (APIs) that are used by programmers also need to be usable in order to allow them to use them efficiently, effectively and correctly. While usability issues of the system can give bad experience to individual users, mistakes that are lead by usability issues of programming tools can introduce bugs and security vulnerabilities into applications that would affect all users who use those applications. Therefore, it is important to ensure that systems as well as programming tools achieve high usability before they are used by people. To this end, software developers test the usability of systems they develop and fix usability issues that exist in them before the system is used by the intended user [58, 59].

There are numerous methods and tools that software developers can use to evaluate the usability and identify usability issues of software systems. The Cognitive Dimensions Framework (CDF) is a commonly used tool for usability evaluations of different types of systems such as visual notations [6, 8, 10, 50], programming languages [13, 14, 75, 81], APIs [20, 33, 32, 64], etc. It describes a vocabulary designed to capture the cognitively relevant aspects of a product that have an influence on how people use the system [40, 41]. Even though cognitive dimensions were originally proposed as a discussion tool to discuss the usability of systems [41], later it has been used to evaluate the usability of systems as well [6, 8, 13, 20, 33, 32, 48, 64]. The work reported in this paper focuses on a commonly used CDF based usability evaluation conducted using questionnaires based on the CDF.

In this usability evaluation method, users of the system under evaluation provide their feedback on the system by answering a questionnaire that is based on the CDF [6, 48, 87]. They might provide their feedback based on their previous experience on the system under evaluation [6, 48] or they may have to complete some specified tasks to gain experience with the system in order to provide their feedback [20, 87, 88]. Most commonly used CDF questionnaires, including the one proposed by Blackwell and Green [6], are open-ended in nature. Therefore, those require evaluators to identify usability issues of the evaluated system by performing some form of analysis on qualitative questionnaire responses. However, previous research does not clearly describe how this analysis is done in order to extract usability issues from the data collected through various CDF questionnaires [7, 14, 50, 82]. It is important that there is a clear set of guidelines for evaluators (in a industry context, evaluators will be the people who develop or test the product) to identify usability issues from questionnaire responses that they gather.

Therefore, in this research, we investigated how software developers should analyse qualitative responses gathered from CDF questionnaire based usability evaluations to identify usability issues. To achieve this, we first conducted a systematic literature review to identify how various researchers have analysed qualitative responses that are collected through CDF questionnaires and how they have identified usability issues. Based on the results, we proposed a method with a set of guidelines for software developers to follow when using CDF questionnaires. For the purpose of this study, the developed set of guidelines would focus on the CDF based questionnaire proposed by Wijayarathna et al. [86] to evaluate the usability of security APIs. Finally, we empirically investigated the proposed guidelines by conducting a between subject study with software developers and Computer Science (CS)/Information Technology(IT) students. In the experiment, two groups of participants were tasked with listing usability issues that a programmer has encountered by analysing the response that the programmer has provided for the security API version of CDF questionnaire. One group was given to use the proposed set of guidelines and the other group was not.

The results of the study revealed that participants who used the proposed set of guidelines identified and reported more issues with a higher validity compared to participants who did not use the guidelines. Results also revealed interesting insights about how the proposed set of guidelines should be further improved in order to enhance the effectiveness of using them to identify usability issues of security APIs.

The rest of the paper is structured as follows. Section 2 provides background information about various usability evaluation methodologies used for evaluating the usability of software systems. Section 3 presents the systematic literature review we conducted and results we obtained. Section 4 presents how we developed a set of guidelines based on the results of the systematic literature review. Then, Section 5 presents the empirical evaluation we conducted to evaluate the proposed guidelines and Section 6 presents results we gathered. Section 7 discusses the overall findings, Section 8 discusses the limitations of the study and finally, section 9 concludes the paper.

2. BACKGROUND

This section intends to provide some background of usability evaluations and the CDF questionnaire based usability evaluation method, which is the focus of this research. Usability evaluations are mainly aimed at finding usability problems/issues in an existing design and then fixing those issues to improve the usability of the design [59]. Numerous methodologies have been used to achieve this objective:

- **Heuristic evaluation**: This is an informal usability evaluation method [59]. In this method, one or more usability specialists inspect a product to see if it adheres to a set of usability principles or heuristics [57, 60]. Nielsen highlighted this method as a discount usability evaluation methodology, which is low cost and less time consuming. However, Petrie and Power [63] informed that heuristic evaluation tends to identify large number of false positives and gives high emphsis to less severe issues.

- **Cognitive walkthroughs**: In this method, a set of experts simulate the user's cognitive procedure as they interact with an interface of a product while accomplishing a specified goal. It evaluates if the user's actions can be assumed to lead to the next correct action [5, 65]. This method can be applied with a prototype of the system without a working product. However, since experts are involved, they may find it hard to take the role of an inexpert user and therefore, come up with a large number of false positives and less severe usability issues.

- **Usability walkthroughs**: This method is a two phase evaluation method. In the first phase, participants explore the product on their own, as in the heuristic evaluation. In the second phase, they complete a set of tasks in a given order while finding usability issues of the product [49, 70]. Sears [70] informed that this method identifies less amount of severe problems compared to other methods.

- **Heuristic walkthroughs**: This is a method proposed to contain benefits of heuristic evaluation, cognitive walkthrough and usability walkthrough methods. This is a two phase process, where evaluators will use a set of tasks, usability heuristics and a set of "thought-focusing" questions to explore a system and identify usability issues [70]. This method is reported to identify serious issues with a limited number of evaluators as well as to identify less false positives [70].

- **Cognitive Dimensions Framework (CDF)** : This is a vocabulary designed to capture the cognitively relevant aspects of a structure that have an influence on how people use it [40, 41]. Earlier, this was used as a discussion tool to discuss usability of programming languages and visual programming notations [41]. Later CDF was adapted to use in empirical usability evaluations. Researchers have used questionnaires based on the CDF to evaluate the usability and identify usability issues of different systems [6, 48, 82].

- **Empirical usability testing** : This refers to evaluating a system by testing it with representative users. During a test, participants will try to complete typical tasks using the system while observers watch, listen and take notes. The goal is to identify any usability problems, collect qualitative and quantitative data and determine the participant's satisfaction with the product [55]. Testing the usability of a system with actual users of the systems incurs a higher cost, but identifies most severe issues with a higher validity [63].

The research reported in this paper focuses on the usability evaluation methodology based on CDF questionnaires. One of the early research studies that used this approach is by Kadoda et al. [48], where they used this approach to identify usability issues of educational theorem provers. Later, Blackwell and Green [6] enhanced this methodology by proposing a generic questionnaire to use across usability evaluations of different systems. Since then, this method has been widely adopted for usability evaluations [6, 8, 13, 20, 48] and different variations of this questionnaire have been proposed [20, 82, 86].

This method can be conducted with actual users of the system, which helps the evaluators to identify actual issues of the system that the users would encounter [63, 87]. The use of a questionnaire based on CDF makes the interaction between the evaluator and the user easier, as it specifies usability aspects and areas of the system that the evaluator has to question from the users [6]. Furthermore, when using such a questionnaire, the participant programmer does most of the work, except the questionnaire designing [6]. Therefore, the involvement of evaluators and API designers will be minimal and API developers/testers who are not experts in usability can use this method to evaluate the usability of their APIs.

3. SYSTEMATIC LITERATURE REVIEW

The objective of conducting a systematic literature review was to identify how previous research have analysed and identified usability issues from qualitative responses they collected using the open-ended CDF based questionnaires. Therefore, we needed to review literature that used CDF based open-ended questionnaires. This systematic literature review was conducted in June and July months of year 2018.

3.1 Procedure

3.1.1 Paper search strategy

We decided to use the snowballing approach [92] to search for papers that used CDF based questionnaires for experiments. We started the search with the two papers that introduced the CDF based questionnaire approach [6, 48], assuming research that use a CDF based questionnaire would cite at least one of these papers. Then we used forward snowballing [92] where we used Google Scholar[1] to identify papers that cited the two papers we selected as our starting set [6, 48]. We could identify 125 unique papers that have cited at least one of those two papers.

Furthermore, while reading the papers for data extraction (data extraction is described later in this section), we searched for papers that use the CDF based questionnaire by looking at references of reviewed papers (backward snowballing [92]). This revealed eight new papers and only seven of them were available to access.

3.1.2 Study selection criteria

Thereafter, we reviewed those 132 papers by reading abstracts and sections in the body of the paper where each paper refers to at least one of the initial two papers. In this step, we tried to identify whether or not a paper has cited Kadoda et al. [48] or Blackwell and Green [6] referring to the usability evaluation approach based on CDF questionnaires that those papers have proposed and used. From the 125 papers reviewed in the first iteration, we short-listed 61 papers that cited one of the two papers of the starting set by referring to the usability evaluation approach they introduced. The rest of the papers cited the initial two papers referring to claims made on those two papers other than the proposed usability evaluation methodology. Those papers have not used the CDF based questionnaire approach for a usability evaluation. The seven papers reviewed in the second iteration also found to have used CDF questionnaires.

3.1.3 Data extraction

Thereafter, we conducted the data extraction from the short-listed papers by reviewing them to find answers to the following questions.

1. Did the research use a CDF based questionnaire?
2. What version of the CDF based questionnaire did the research use?
3. Did the CDF based questionnaire that the research used contain open-ended questions?
4. For what purpose did the research use a CDF based questionnaire?
5. What did the research evaluate using a CDF based questionnaire?
6. How were they analysing questionnaire responses?
7. What conclusions did they make by using the a CDF based questionnaire?
8. Did they identify usability issues of the evaluated product using a CDF based questionnaire?
9. If they did, how did they identify issues from questionnaire responses?
10. What format did they use for reporting usability issues?

When performing the data extraction, the texts from papers that described their questionnaire response analysis process and usability issue identification proces were extracted and recorded.

3.1.4 Data summarisation and analysis

The descriptions of questionnaire response analysis process and usability issue identification process were analysed and summarised to identify similarities and differences among different papers. Based on this, common techniques used for questionnaire response analysis and issue identification were identified. For example, when summarising questionnaire response analysis methods, papers that reported data analysis method were categorised into several categories based on the similarities of the methods that were used.

Even though quality assessment is another step of conducting systematic literature reviews [51], this review did not perform this step. The quality assessment step is mainly important when reviewing the results obtained by literature

[1]scholar.google.com.au

that is being reviewed [51]. Since this review focused on exploring the methodology that has been used in the reviewed literature, quality assessment was not performed in this review. We attempted to identify many available methods as possible in order to investigate them. Figure 1 presents a summary of the entire procedure followed in the systematic literature review.

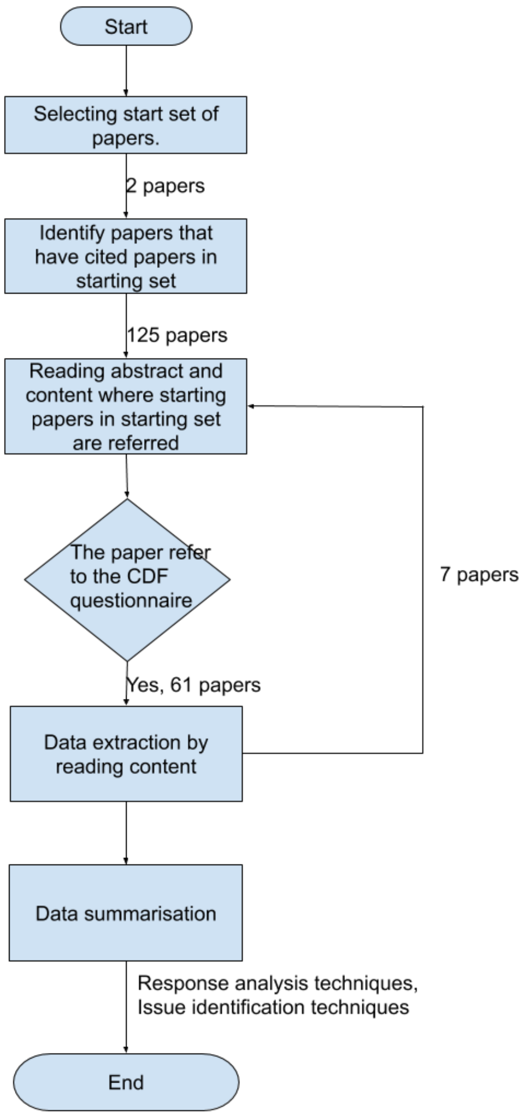
Start
Selecting start set of papers.
2 papers
Identify papers that have cited papers in starting set
125 papers
Reading abstract and content where starting papers in starting set are referred
The paper refer to the CDF questionnaire
7 papers
Yes, 61 papers
Data extraction by reading content
Data summarisation
Response analysis techniques, Issue identification techniques
End

Figure 1: Systematic literature review process

3.2 Results

A total of 70 publications (Appendix D) were reviewed in this systematic literature review (starting set-2, 1st iteration-61, 2nd iteration-7). Figure 2 shows the number of papers we reviewed based on the publication year. This suggests that researchers have referred to and used the usability evaluation approach that use CDF questionnaires throughout the last 18 years since it was first used.

Table 1 shows the number of publications we reviewed based on the publication type. Furthermore, Appendix D presents the particular venue of each paper that we reviewed. The highest number of papers that were reviewed have been published at the "psychology of programming interest group conference" (12 papers). The table also reveals that the CDF based usability evaluation is a topic of interest among many communities and researchers, considering the number of different venues that this literature is published. In this work, when we are mentioning papers reviewed, that includes all literature we have considered including dissertations, reports and book chapters.

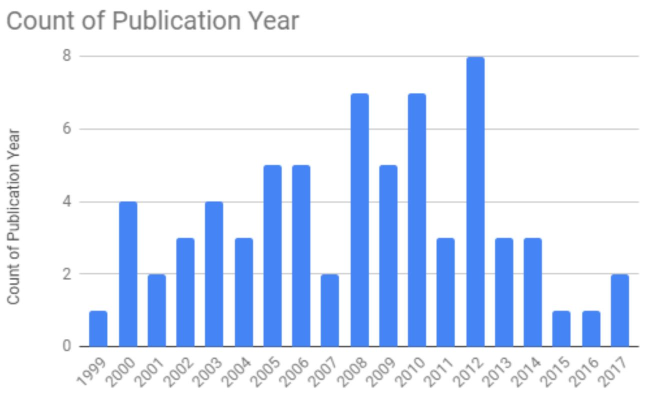

Figure 2: Publication year statistics of the papers reviewed

Table 1: Publication type statistics of the publications reviewed

Publication Type	Number of Papers Reviewed
Conference paper	35
Journal paper	17
PhD dissertations	9
Masters dissertations	1
Technical report	1
Book chapter	1
Unknown	2

From the 70 papers we reviewed, 60 papers had used a CDF based questionnaire for the usability evaluation. Those 60 papers reported 63 different studies that used CDF based questionnaires. Table 2 shows the details of different versions of CDF questionnaires used for these experiments.

From the 60 papers that used a CDF based questionnaire, 46 studies used questionnaires that contained open-ended questions and six papers used questionnaires that only contained close ended questions. Eight papers did not clearly mention whether or not their questionnaire contained any open-ended questions.

We only present the details regarding the data analysis of 54 papers in which either mentioned that the questionnaire they used contained open-ended questions or we could not determine conclusively whether their questionnaire contained open-ended questions or not (we pertinently excluded the papers that only include close ended questions). From these

Table 2: Questionnaire versions used for different studies

Questionnaire version	Number of studies that used this version
Version developed by Blackwell and Green [6].	19
Developed their own version	10
A modified version of the questionnaire developed by Blackwell and Green [6]	24
A translation of Blackwell and Green [6]'s questionnaire	1
Semi structured interviews based on Blackwell and Green [6]'s questionnaire	7
Semi structured interviews based on Blackwell and Green [6]'s questionnaire and Clarke [20]'s questionnaire	1
Likert scale questionnaire based on Clarke [20]'s CDF	1

54 papers, only 13 papers described how they analysed data collected through the questionnaire they used.

3.2.1 *Qualitative data analysis*

While the data analysis techniques followed in these 13 papers are mostly different from each other, we categorised them into five main categories by considering similarities and differences between those techniques.

Five papers analysed data by categorising them into different information categories, which is similar to the commonly used qualitative data analysis technique known as **thematic analysis**[13, 30, 31, 34, 91]. While some papers just mentioned that they analysed responses using thematic analysis [13, 30], other papers described in more detail how they analysed data they collected from their CDF questionnaire [31, 34, 91]. Wingrave [91] used the multiple pass affinity diagramming process [4] and developed four themes, which are representation, structuring thought, designability and learnability. Duignan [34] analysed CDF questionnaire responses by searching for patterns where research data interrelates to theory and literature. They used a coding tool called "TAMS analyser" [84] and notated data with a set of prior theory driven codes that included the 14 cognitive dimensions. Datta [31] also analysed their data using a technique similar to thematic analysis. Using NVivo, they auto-coded questionnaire responses into cognitive dimensions and then themes of each node were explored. This resulted in three main themes, which are usability issues, suggestions for improvement in the current tools, and suggestions for new features.

Three studies analysed open-ended questionnaire responses by **identifying positive, negative and equivocal comments** [6, 32, 33]. Blackwell and Green [6] mentioned that they identified these three types of comments that experiment participants made while responding to the CDF based questionnaire. Diprose et al. [32, 33] presented more details about the process. Initially they have classified responses into positive, equivocal or negative, and then they have classified each identified comment as general and specific. They mention that while general comments indicated whether or not the notation was acceptable with respect to a particular dimension, specific responses showed how specific usability features perform against a particular dimension.

Two studies used the **grounded theory approach** for the data analysis [66, 67]. Both studies followed similar methods for data analysis. First, they applied open coding where the responses were examined for objects of interest based on their research questions. They used *microanalysis* [76] for this process. The analysis focused on identifying major themes or categories and how often they emerged in the data under varying conditions. After completing the open coding, an axial coding process was conducted to form a higher level of abstraction by identifying relationships between categories based on their properties. They have used CDF and the International Organization for Standardization (ISO)'s usability criteria [45, 46] for identification of categories and properties.

Blackwell and Green [6] further considered **specific criticisms of the system** when analysing responses collected from the CDF based questionnaire. They suggested that "*the usability of a system is best measured not by general expressions of satisfaction, but by the number of specific criticisms that are made by questionnaire respondents*". Furthermore, when analysing questionnaire responses, they considered a usability issue as an aspect where the respondent identified some specific aspect of the system that can be improved.

Stead [74] also described how he analysed qualitative responses he collected. He mentioned that he only **used quotes from qualitative responses to support results** he collected through the likert scale based quantitative data. He did not use responses for the open-ended questions to form any new results.

3.2.2 *Identification of usability issues*

From the 54 papers that we considered for investigating the qualitative data analysis process, 20 papers have used responses that they collected to identify usability issues of the system that was evaluated using the CDF. Other papers have used questionnaire responses for different purposes such as,

- Comparing views of different types of users about a product [47].
- Giving a usability rating to a product [8, 13, 14, 52, 81, 82].
- Discussing usability of a product [50].
- Identifying user's reflection of the product [90].
- Identifying how certain trends in the field have affected the product [34].
- Comparing different products and methodologies [9, 61].
- Identifying user's impression of a product [30].

From the 20 papers that identified usability issues of a product, only seven papers described the process they followed to identifying usability issues from qualitative responses they collected. These seven papers used four main criteria for identifying usability issues.

1. Aspects that can be improved or caused problems [6, 43, 53].
2. Concerning comments by participants [83, 91].
3. Comments where the product under evaluation has under-performed compared to other similar products [48].
4. User responses that falls into the category called usability issues when using thematic analysis [31].

Even in these seven papers, very little information were provided on how usability issues were identified using the responses gathered from the open-ended questionnaire.

3.3 Discussion

Even though the papers that were reviewed in this section did not provide much insights about data analysis techniques and issue identification techniques that were used, it is important to discuss the strengths and weaknesses of the techniques that were identified. Five studies employed approaches similar to thematic analysis to perform the data analysis. Having a pre-defined set of categories to map the data was the main similarity identified in those studies. Having such pre-defined set of categories would provide a guidance for the analyst to perform the data analysis and therefore, would allow people with less experience in data analysis and usability to perform it. Especially, CDF can be accommodated to use as the set of categories in the analysis.

Identifying positive, negative and equivocal comments can also be listed as a way of thematic analysis, where positive, negative and equivocal will be the pre-defined themes. This method also includes one of the usability issue identification method that were identified, which is 'concerning comments by participants'. This is also a method that can be easily followed with a less guidance. However, due to the lack of use of the CDF in this method, it might limit the power of this method and will require the analyst to have an in-depth knowledge about the CDF to use this method.

Grounded theory approach is a method commonly used by researchers for qualitative data analysis [76]. Even though there are many guidelines available for performing this method, it might be hard for people such as API developers who are not familiar with this methodology. Therefore, evaluators might need more training in order to use this method.

Considering specific criticisms of the system is also a straight forward method that can be easily followed with a less guidance. This method includes one of the usability issue identification that were identified, which is aspects that can be improved or cause problems. However, since specific criticism might not be the only useful data that are available in questionnaire responses, this method would limit the results that can be obtained from the analysis. Similar to identifying positive, negative and equivocal comments, due to the lack of use of the CDF in this it method, will require the analyst to have an in-depth knowledge about the CDF to effectively use this method.

3.4 Summary

This section discussed the systematic literature review we conducted to explore existing techniques that are available to analyse CDF questionnaire responses and identify usability issues. However, after reviewing 70 papers presenting research that used CDF questionnaires, we could not identify an established method that have been used in this process. The results of the literature review highlighted the lack of guidance available to analyse CDF questionnaire responses and stress the need for a guidance to perform this process. Therefore, we developed a method with a set of guidelines to assist usability evaluators to analyse CDF questionnaire responses.

4. DEVELOPING A METHOD TO ANALYSE CDF QUESTIONNAIRE RESPONSES

From the results of the systematic literature review, we can clearly see that there is no established method to identify usability issues from the qualitative responses that are collected using a CDF questionnaire. Even though researchers can analyse CDF questionnaire responses using their qualitative data analysis knowledge, software developers may not carry the same skill set to do that when using the CDF to evaluate the usability in a software development environment. Therefore, in this section, we propose a method with a set of guidelines that software developers can follow when performing this. As some specifics of the proposed method would vary on the particular CDF that is being used, for the purpose of this research, here onward we are focusing on the version of CDF questionnaire proposed by Wijayarathna et al. [86] to evaluate the usability of security APIs. This version of CDF consists of 15 dimensions that describe the usability of security APIs, which are abstraction level, learning style, working framework, work-step unit, progressive evaluation, premature commitment, penetrability, API elaboration, API viscosity, consistency, role expressiveness, domain correspondence, hard to misuse, end-user protection and testability [86, 89].

4.1 Initial development

We conducted the systematic literature review in order to acquire existing knowledge on the field for developing the set of guidelines. However, the papers we reviewed only revealed a little amount of information on how they analysed CDF questionnaire responses. We used the collected information in developing a guideline document that guides experimenters to identify usability issues. When developing the approach, several methods that were identified from the systematic literature review were taken into account, which are 'thematic analysis', 'identifying positive, negative and equivocal comments' and 'considering specific criticisms of the system'. The guidelines document was developed as a thematic code book [12], that is used in the thematic analysis.

A thematic code book would define useful types of information that a response can contain and guide the analyst to locate those information. For each information category, the code book would contain a code that consists of a label, description, rules for inclusion, rules for exclusion and examples [12]. When analysing a CDF questionnaire response, the evaluator should refer the developed guidelines document (i.e. the thematic code book). It will point evaluators to identify useful information (such as usability issues) through rules for inclusion, rules for exclusion and examples.

4.1.1 *Development procedure*

From the CDF, we know that questionnaire responses would contain information about a security API related to the 15 cognitive dimensions [6, 31, 86]. The previous research that used thematic analysis has also used cognitive dimensions as the themes in their analysis [31, 34]. Furthermore, questionnaire responses can contain information about positive aspects of the API as well as usability issues of the API [6, 32, 33]. Therefore, the guidebook we developed contained 30 main categories (i.e. Issues of Abstraction Level, Positives of Abstraction Level, Issues of Learning Style, Positives of Learning Style, etc.).

CDF provided the basis for developing descriptions for each information category [15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 86]. Following are some example descriptions of information categories.

- Issues of Abstraction Level - The minimum and maximum levels of abstraction exposed by the API are not adhering to the minimum and maximum levels usable/expected by a targeted developer.
- Positives of Abstraction Level - The minimum and maximum levels of abstraction exposed by the API follows the minimum and maximum levels usable/expected by a targeted developer.
- Issues of Learning Style - Issues that make it difficult for programmers to obtain the minimum understanding about the API that is required to use the API to achieve a goal.

Rules for inclusion and exclusion were added to provide guidance on what should be identified as a useful piece of information (i.e. a usability issue or a positive aspect) related to each category and what should not. When developing rules for inclusion and exclusion, we used two sources. CDF itself provided guidance on what would be a usability issue and what would be a positive aspect of the API with respect to each cognitive dimension [15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 86]. Furthermore, we used results of an experiment that used the particular CDF questionnaire [86] where 11 programmers who used the Java Secure Socket Extension (JSSE) API provided their feedback through the questionnaire [89]. We asked two coders who are familiar with cognitive dimensions to identify usability issues and positive usability aspects of those questionnaire responses. They independently analysed the collected data and identified issues and positives of the API. Cohen's kappa calculation for inter coder agreement [28] was 0.818. Once both of them came up with their own results, they discussed the issues and positives that were identified by only one of them and developed a final results that both of them agree with. We used their final result and included rules for inclusion and exclusion into developed guidelines that would identify usability issues and positive aspects of the API they identified. When including rules for inclusion and exclusion identified in above two criteria, it was attempted to encourage the evaluator to identify positive and negative comments provided by the respondent as well as to identify specific criticisms of the system following the observations of the systematic literature review. Table 3 shows rules for inclusion and exclusion related to the category 'Issues of Abstraction Level'. Table 4 shows rules for inclusion and exclusion related to the category 'Positives of Abstraction Level'.

Finally, we included examples for each category that would provide more help for experimenters to identify what should be identified as a usability issue and what should not be, and also what should be identified as a positive usability aspect of the API and what should not be. We included examples in general terms as in table 3 and table 4.

Table 3: Guidelines for "Issues of Abstraction Level" in the initial version.

Label	Issues of Abstraction Level
Description	The minimum and maximum levels of abstraction exposed by the API are not adhering to the minimum and maximum levels usable/expected by a targeted developer.
Rules to Inclusion	Developer expects to accomplish an individual goal by using only one component of the API where developer needs to use more than one component of the API to accomplish the goal. Developer expects to accomplish an individual goal by using more than one components of the API where API has provided all the functionalities required to accomplish the goal in a single component. Components of the API are too low level compared to what programmer expects. Components of the API are too high level compared to what programmer expects. Identify any components of the API that are mentioned as too high level or too low level. Developer suggests to include extra classes/methods.
Rules to Exclusion	Components of the API are too high level, but it made it easy to use the API. Components of the API are too low level, but it made it easy to use the API.
Examples	Developers expect to achieve goal X by using only 1 class, but API requires them to use more than 1 class. Method parameters for the method X are too low level. API should provide a method A that provide the functionality X.

Table 4: Guidelines for "Positives of Abstraction Level" in the initial version.

Label	Positives of Abstraction Level
Description	The minimum and maximum levels of abstraction exposed by the API follows the minimum and maximum levels usable/expected by a targeted developer.
Rules to Inclusion	The number of components of the API that a developer has to use to achieve a goal meets the number of components they expect to use. Abstraction level of the components of the API are neither low level or high level from what programmer expects. Identify any components of the API that are mentioned to have ideal abstraction levels. A component of the API had higher or lower abstraction level than what programmer expected which made it easier than expected to achieve the goal.
Rules to Exclusion	
Examples	Number of classes required to achieve a task meets programmer expectations. Abstraction level of the class A meets the expectations of programmers. API provides high level functionality which makes it easy to use.

4.1.2 *Outcome of the initial development*

The aforementioned process resulted in a guidelines document that contained guidelines to identify 30 different information categories. Table 3 and Table 4 show the guidelines to analyse CDF questionnaire responses and identify usability issues of abstraction level dimension and positive aspects of abstraction level dimension. The complete set of developed guidelines can be accessed at `https://bit.ly/2nlZodt`.

We developed the aforementioned set of guidelines by reviewing previous research and by referring to results obtained by researchers while using the cognitive dimensions questionnaire. Hence, the developed set of guidelines are in a researcher focused language and might still be difficult for software developers to understand and interpret. Therefore, we decided to convert the this version of the guidelines into a developer understandable version.

4.2 Converting the guidelines into a developer understandable version

4.2.1 *Procedure*

In a software development environment, the main objective of conducting a usability evaluation is identifying usability issues [59] and not the strengths of their product. Identifying usability issues allow them to enhance usability of the product by fixing them. Therefore, when converting the developed guidelines into a developer understandable language, we first removed the sections of the guidelines that are corresponding to identifying positives and only kept sections related to identifying usability issues.

Thereafter, four independent software developers were recruited by the convenience sampling method [38, 42, 54, 68] to review the set of guidelines (i.e. sections related to identifying usability issues). They were provided with the set of guidelines developed in Section 4.1 (only including usability issue categories) and requested to provide their feedback on what sections and rules they find difficult to interpret. After they reviewed the guidelines, they provided their comments on the guidelines and provided their suggestions to improve it.

4.2.2 *Results*

They highlighted sections where the language was not comprehensible to them and suggested to use terms that are more familiar to software developers in those sections, so the idea would be articulated more effectively to software developers. Furthermore, they suggested to add more examples for some sections of the document. The suggestions they provided were discussed with them in order to identify how the set of guidelines should be improved. They also mentioned aspects of the current version of the guidelines that they liked. For example, while they mentioned that they would prefer to have more examples, they stated that they liked the available examples and the way those have been presented. Based on their feedback, the developed set of guidelines was improved, which resulted in the developer focused version of the guidelines. Table 5 shows a sample information category from the developer focused version of the guidelines document. The improved version of the set of guidelines can be accessed at `https://bit.ly/2nnMhZr`.

4.3 Sample use of the developed set of guidelines

Let us assume a programmer has provided following answers in the security API version of CDF questionnaire in order to understand how an evaluator should use the developed set of guidelines.

Q1 : Were you able to implement the core functionality of the task you completed using only one class from the API or more than one class?

A : More than one class was needed.

Q2 : Did you expect to use only one class or more than one class to implement the core functionality of the task you completed?

A : I expected to use more than one class.

Q3 : How would you describe your experiences with respect to the types of classes that you worked with while completing the task?

A : They were too low level.

Q4 : Explain.

A : Classes I had to use for implementing the login functionality are too low level.

Table 5: Guidelines for "Issues of Abstraction Level" in the developer focused version.

Label	Issues of abstraction level
Description	The minimum and maximum levels of abstraction exposed by the API are not adhering to the minimum and maximum levels usable/expected by a targeted programmer.
What is a usability issue?	Programmer expects to accomplish a particular goal by using only one component of the API. However, programmer needs to use more than one component of the API to accomplish the goal. Programmer expects to accomplish a particular goal by using more than one component of the API. However, API has provided all the functionalities required to accomplish the goal in a single component. A particular component of the API is mentioned as too high level or too low level than what the programmer expects. Programmer mentions in general that components of the API are too high level or too low level than what they expect. Programmer suggests including extra classes/methods.
What is not a usability issue?	Components of the API are high level than what the programmer expected. However, being high level made it easy to use the API. Components of the API are low level than what the programmer expected. However, being low level made it easy to use the API.
Example issues	Developers expect to achieve a goal X by using only one class, but API requires them to use more than one class. Method parameters for the method X are too low level. API should provide a separate method that provide the functionality X.

When using the developed set of guidelines, the evaluator have to check the guideline document to see if there are any "what is a usability issue?" rules that can be applied to the responses provided by the programmer. For example, the rule "components of the API are too low level compared to what programmer expects" of the "issues of abstraction level" category can be applied to the answer of Q3 and Q4 where programmer has said that the classes of the API that are required to implement the login functionality are too low level. Then the evaluator has to check the answer with "what is not a usability issue" rules of the "issues of abstraction level" category to see if there are any rules that disqualify this from been recognised as an issue of abstraction level. Since there are no "what is not a usability issue" rules that are applied to this category, the evaluator can record "classes of the API required to implement a login functionality are too low level" as an issue of abstraction level of the API by following the usability issues format presented in the second example.

5. EMPIRICALLY EVALUATING THE GUIDELINES

We conducted a between subject empirical investigation to evaluate whether the developer focused version of the proposed guidelines served its purpose (i.e. whether or not the proposed guidelines helped software developers to identify usability issues from qualitative responses for the security API version of CDF questionnaire). The experiment mainly asked two groups of participants (who are either software developers or computer science students) to identify usability issues that some programmers have encountered by analysing their responses to the security API version of CDF questionnaire. While participants in the experimental group followed the proposed guidelines for identifying usability issues, participants in the control group performed this without the help of the developed guidelines. Then, we evaluated whether participants have correctly identified all usability issues that programmers have encountered by comparing the identified issues with a reference issue list.

The study described here has been approved by the Human Research Ethics Committee of our university.

5.1 Sample questionnaire responses

To evaluate the proposed set of guidelines, we used CDF questionnaire responses collected in an experiment to evaluate the usability of the Google authentication API [85].

In this experiment, ten participants implemented Google sign-in and sign-out functionalities into a web application using the Google authentication API. The experimenters provided participants with a simple web application that consists of an index page and a login page. They asked participants to implement sign-in and sign-out functionalities, so that the index page and the data in it can only be accessed after successfully signing into a Google account. Once each participant completed the task, they provided their feedback on the API using the security API version of CDF questionnaire.

The version of CDF questionnaire completed by the participants [86] contained a separate section with questions for each cognitive dimension [86]. Therefore, for each cognitive dimension, we selected a corresponding question-answer set from the participant who provided the most amount of information according to the experimenters.

5.2 Generating the reference usability issue set

The reference set is the list of issues that have actually been encountered by each programmer who responded to the security API version of CDF questionnaire and can be extracted from their responses. This list is used to evaluate the correctness of issues that participants of our experiment report.

To create a reference issues list, we asked two researchers who are familiar with the CDF and usability evaluations to analyse sample questionnaire responses we selected and identify usability issues of the API. They independently analysed the questionnaire responses where they identified usability issues that programmers have encountered with the help of the CDF. Once this was completed, they discussed issues that only one of them had identified. Based on the discussion, they came up with a final issue set that both of them could agree with. Table 6 shows details of the reference issue set. For some dimensions, we had to use questionnaire responses that did not indicate any issues as none of the programmers who answered the security API version of CDF questionnaire had encountered any issues related to those dimensions.

5.3 Participant recruitment

The experiment was conducted via Google forms so participants could participate remotely. We used two main methods to recruit participants for the study. We posted advertisements in Facebook groups, Twitter and LinkedIn where we can reach software engineers and CS/IT students. We also invited GitHub users who have made their email address publicly available by sending them an invitation email (Statistics related to the GitHub recruitment are available in Appendix A). In advertisements and invitations, we requested participation in a 30-60 minutes study and offered a $15 Amazon gift voucher for participants. The invitation guided them to the experiment instrument, which was hosted in Google forms. We randomly selected each participant for either experimental group or the control group and forwarded them the corresponding Google form. 17 participants participated in the experimental group and 19 participants participated in the control group. Demographics of the participants participated in the study can be found in Table 7.

5.4 Study procedure

The participants who signed up for the experiment completed the experiment instrument remotely. For both the control group and the experimental group, the first page of Google forms provided information about the experiment and the purpose of conducting the experiment. The second page described the human research ethical aspects of participating in the experiment and sought the participants' consent. Once they agreed to participate in the study, the next section explained the task they had to perform. We explained that the experiment requires them to identify and report usability issues that a programmer has encountered by referring to the qualitative feedback that the programmer provided. This description was slightly different in the instruments provided for the two groups as the experimental group had to refer to the guidelines we developed for performing this task while the control group did not have this requirement. It also included a video that explained the programming task that the programmer completed using the Google authentication API prior to providing the feedback and each code change that programmers had to do by walking them through a sample solution.

Then, in the form provided to the experimental group, each section of the developed guidelines (i.e. a section related to one cognitive dimension at a time) were presented to participants. They were asked to go through the section of the guidelines. Once they did that, they were presented with the sample feedback related to the particular cognitive dimension and asked them to identify and list usability issues that the programmer had encountered based on their feedback. They had access to the guidelines document while completing this step. We advised them to indicate "no issues" if they feel that the programmer have not encountered any issues. They continued this process for sections of all 15 cognitive dimensions. Appendix B shows a sample section from the instrument used for the experimental group.

The form that was presented for the control group was similar to this, but they were not provided with the guidelines that we developed. As a guide, they were only provided with a simplified definition for a usability issue that said "A usability issue is an aspect of the API that negatively affects the efficiency, effectiveness and satisfaction of the programmers who are using the API". Then each section of the form showed a sample feedback (the same feedback was provided to the experimental group) related to the particular cognitive dimension and asked them to identify and list usability issues. A sample section from the instrument used for the control group is Available in Appendix C.

The participants of the experimental group were additionally asked to provide their overall feedback about the guidelines they followed. Finally, all participants answered several questions related to their demographics.

Once the data collection was complete, we compared the issues identified by each participant with the reference list we created and checked what portion of the issues was identified by each participant.

6. STUDY RESULTS

After the data collection step, the number of issues that are identified by participants and the number of incorrect issues that are reported by participants (false positives) were compared between the control group and the experimental group. The statistical power analysis [29] performed at the beginning of the study proved the appropriateness of the used participant sample and the reference issue set in order to determine the differences in the probability of identifying an issue and the validity of an identified issues. By conducting a Shapiro-Wilk test [71], the aforementioned two numerical data sets of both the experimental group and the control group are found to be normally distributed. Furthermore, the mean of variances are found to be equal among the two probability data sets as well as among the two validity data sets according to the F-test of equality of variances [72, 93].

We compared the probabilities of identifying each issue in the two groups using a paired t-test [44, 93]. Paired t-test was used because probabilities of identifying each issue was compared and the same set of usability issues were used in both the experimental group and the control group. This revealed that the probability of identifying an issue by participants in the experimental group is significantly higher than the probability of identifying an issue by participants in the control group. While the probability of identifying a particular issue was 0.44 for the experimental group, the probability of identifying an issue by participants in the control group was 0.28 ($p<0.05$).

We also compared the validity of issues identified by participants in the experimental group and the control group. Results of the unpaired t-test [93] revealed that validity of issues identified by the participants in the experimen-

Table 6: Issues in the reference list for each cognitive dimension

Cognitive Dimension	Number of Issues	Issues
Abstraction level	1	• Classes of the API are too low level
Learning style	2	• Difficult to learn the API by copying a sample code • No enough examples available on using the API
Working framework	1	• Has to work with too many classes simultaneously/has to keep track of while using the API
Work step unit	1	• Programmers has to write more code than they expect when using the API
Progressive evaluation	0	
Premature commitment	0	
Penetrability	4	• There is no general overview of what steps to follow when using the API • Difficult to find details about URL endpoints • Difficult to find details about scope values • Details about the usage of state parameters are not properly indicated.
API elaboration	0	
API viscosity	1	• Difficult to make changes to code that use the API
Consistency	1	• gapi.load and gapi.init looks similar but provide different functionalities
Role expressiveness	2	• Difficult to identify which classes and methods to use when using the API • Difficult to understand the role of certain parts of the API eg : Data Stores, HTTP transport
Domain correspondence	0	
Hard to misuse	1	• When gapi.load is missing, error messages does not help to identify the issue
End user protection	1	• Security of the end user will depend on how the programmer use the API
Testability	1	• API does not provide any guidance on what to test.

Table 7: Participant demographics summary

Demographic	Experimental Group	Control Group
Age		
Above 18 upto 24	7	3
Above 24 upto 32	7	3
Above 32 upto 40	1	6
Above 40	2	7
Gender		
Female	1	1
Male	16	15
Prefer not to say	0	3
Current Occupation		
Software Developer	12	12
Software Architect	0	4
Software Quality Assurance Engineer	0	0
Student (Computer Science/ IT)	4	2
Student (Other)	1	1
Participant has previous experience in developing/testing APIs		
Yes	16	16
No	0	3
Not Sure	1	0
Participant is a Native English Speaker		
Yes	5	11
No	12	8

tal group was significantly higher compared to the control group. While the validity was 0.61 for the experimental group, it was 0.40 for the control group ($p<0.05$).

We also measured if the participants who used our proposed guidelines (i.e. experimental group) have identified more issues of each cognitive dimension compared to the participants who did not used the guidelines. Results revealed that the experimental group had identified significantly more issues related to working framework, work-step unit, penetrability and end-user protection dimensions compared to the participants of the control group ($p<0.05$).

Even though the probability of identifying issues and validity of issues identified are significantly higher when using the proposed guidelines compared to when not using them, the probability of identifying an issue in the experimental group was not very good either. The average probability of a participant of the experimental group identifying a given issue was measured to be 0.44. From the cognitive dimensions that had at least one issue in the reference issue set, work-step unit (0.94), testability (0.64) and end-user protection (0.64) had the highest probability of getting identified by a participant. Issues of learning style (0.29), working framework (0.23) and role expressiveness (0.23) had the lowest rate for getting reported by a participant.

We asked participants of the experimental group about their opinion on how to improve the developed set of guidelines. Even though participants could not successfully identify all the issues by following the guidelines, they mentioned that the set of guidelines is intuitive and it was easy to extract information. Some participants provided suggestions to improve the guidelines as well.

Some participants thought that rules and descriptions provided for each issue type is generic and too verbose. They stated that it would be much easier to use the guidelines if those rules and descriptions are more specific. Most participants thought that examples provided for each issue type need to be improved, suggesting the inclusion of more examples and improving the quality of the examples.

7. DISCUSSION

Data analysis in usability studies, especially in API usability studies, is an interesting but a less discussed topic. Even though there are numerous studies that perform usability evaluations, those studies rarely give detailed insights about data analysis when qualitative data such as think aloud data or interview/open-ended survey responses are involved [35, 55, 56, 77, 78, 79, 80]. We observed and presented this in the systematic literature review we conducted, which investigated a subset of previous usability studies that used a CDF questionnaire for the usability evaluation. However, our view is that it is a very important component of a usability evaluation, especially when the evaluation is conducted to identify usability issues. Studies that involve quantitative data usually give proper descriptions about data analysis [1, 2, 69], but such studies give limited insights about usability issues of a system compared to studies that involve qualitative data.

Researchers might find it comfortable to analyse qualitative data to identify usability issues without proper guidance, because they might be skilled with related techniques such as grounded theory and thematic analysis. However, software developers may find it difficult as they are less familiar with related concepts compared to researchers. Furthermore, if they do not get the proper guidance, they might come to conclusions that are not based on the actual results they have, but based on their own perceptions of the system they are evaluating. To address this issue, in this research, we conducted a systematic literature review to review data analysis techniques that previous research have used to analyse CDF questionnaire responses. It revealed five methods that previous research have used for data analysis and four methods for identifying usability issues from CDF questionnaire responses.

Based on the findings of the systematic literature review, we developed a set of guidelines to guide software developers when they are analysing CDF questionnaire responses to identify usability issues of a security API. We evaluated the proposed method by conducting a between subject empirical investigation where two groups of participants tried to analyse feedback provided by programmers after using a security API.

Results of the experiment revealed that participants who used the proposed guidelines reported more usability issues with a higher validity compared to the participants who did not use them. This indicated that guidance similar to what was provided in the set of guidelines we developed can be useful for software developers when they are analysing qualitative feedback that usability evaluations provide. Especially, a significant difference of results we observed related to working framework, work-step unit, penetrability and end-user protection dimensions suggest that software developers do not see these issues as usability issues or are not aware of their impact if some sort of nudging was not done at the stage of issue identification.

This work developed two sets of guidelines, that are the researcher focused version and the developer focused version. The initial version we developed contained guidelines to identify both usability issues as well as positive usability aspects of the evaluated product. However, when conducting a usability evaluation in a software development environment, the main objective is to identify usability issues that exist in the product. Therefore, when converting the initial version into a developer focused version, we omitted the guidelines for identifying positive usability aspects of the evaluated security API. However, if researchers are using this set of guidelines as a thematic code to analyse CDF based qualitative data, they can use the researcher focused version of the proposed guidelines, so they can identify both positive and negative usability aspects.

The results we obtained in the experimental group of the experiment is not very promising, even though they are significantly better than the results of the control group. The observed probability of identifying an issue, which was 0.44, says that a developer who analyses CDF questionnaire with the help of the proposed guidelines will fail to identify a given issue more than half of the times. This indicates that further improvements are required for the proposed guidelines in order to to improve the effectiveness of using them.

There are a few possible reasons for these lower values. Even though the guidelines and the provided examples indicated the important information that needs to be stated when reporting a usability issue, participants consistently reported issues partially by only stating a few components of an issue. For example, some participants reported that “No proper error message to identify a misuse” as an issue under the ‘Hard to misuse’ dimension. However, we expected the reported issue to be “Error messages does not help to identify the issue when the **gapi.load() method is missing**”. The programmer who answered the security API version of CDF questionnaire has specifically mentioned that they encountered the issue due to lack of error messages when the *gapi.load()* method of the API was not used in their implementation. When reporting this issue, some participants ignored the context where the error messages were not present even though the guidelines emphasized to report the particular context of the issue if the programmer has mentioned it in their feedback. If the particular context of the issue was not reported when reporting an issue, the developer who would be fixing the issue would have to decide where the error messages are missing using their own judgement. Since the perception of the API developer might be different from the programmer’s who would use the API, it is possible for the API developer to overlook the particular context that the programmer faced. This type of partial reporting was a main reason for the lesser number of exact issues identified by participants. Furthermore, since we considered these partially reported issues as incorrect issues, they caused a reduced validity of results as well.

As stated in Section 4, the developed set of guidelines focuses on the version of CDF developed by Wijayarathna et al. [86] for security API. Guidelines focuses on the 15 dimensions of this CDF and covers usability issues of security APIs. Therefore, when applying this set of guidelines to analyse data collected through a different CDF questionnaire in a different context, this guidelines document should be adapted. However, the format introduced here can be used for such applications by introducing several changes. As some of the dimensions are common for most CDFs (e.g. progressive evaluation, role expressiveness), guidelines for those dimensions can be use by changing rules and examples to match the relevant context. For dimensions that are not covered in the developed set of guidelines, evaluators can develop their own guidelines by introducing new rules and examples following the presented approach. To identify rules, they can refer to the definitions of each dimension and then introduce examples to further clarify those rules.

8. LIMITATIONS

We note a few limitations of the studies reported in this paper that need to be considered when interpreting the results obtained in this research.

The reported issues not only depend on the developed guidelines or how participants analysed the programmer feedback, but also depend on the quality of the feedback provided by the programmer. If the programmer has not provided clear feedback about the API when answering the security API version of CDF questionnaire, it would make it difficult for the person who is analysing the feedback. The feedback we used in this experiment was obtained from an actual usability evaluation that used the security API version of CDF questionnaire to evaluate the usability of the Google authentication API [85]. Therefore, we assume that it represents the kind of feedback that would be obtained from the type of usability evaluation we are discussing here. Furthermore, when creating the reference usability list, two analysts agreed on the issues that were used in the reference issue set as communicated through the programmer feedback. Therefore, we believe that it is reasonable to expect the participants to identify usability issues of the reference list by referring to the programmer feedback.

Even though the guidelines and the programmer feedback were in English, not all participants participated in the experiment were native English speakers. Only five out of 17 participants of the experimental group were native English speakers. Interestingly probability of identifying an issue for these participants was 0.61 (validity=0.70), which compare higher to the probability obtained by all participants of the experimental group that used the proposed guidelines. The ideal way for testing the guidelines would have been to test it using native English speaking participants. However, we decided to use participants who are native English speakers as well as non native English speakers as we are targeting to introduce the set of guidelines for both groups. Most organizations use English as the primary language for their business purposes and most tutorial/help available for developers are in English. Unfortunately, results of the study recommend against this and a separate study can be designed to test how native English speaking professionals would benefit from the proposed guidelines.

The sample used in the experiment is predominantly male where 94% of the experimental group and 78% (16% preferred not to say) of the control group were male participants. However, this percentage of male participants represent the overall male/female ratio in the software development industry, which is also predominantly male [73]. In addition to that, the representation of each gender is approximately same in both groups and therefore, it is reasonable to assume that the used sample is suitable for the purpose of this study. Previous studies that investigated developer behaviour has also used samples with similar gender percentages [39, 62].

9. CONCLUSION AND FUTURE WORKS

Even though previous research has proposed a CDF based approach to evaluate the usability, there is no proper guidance on how the data analysis step of this approach should be performed. This usability evaluation method generates a set of qualitative responses that describe the usability of the system under evaluation. Analysing them can be a challenging task for software developers in order to identify usability issues. Therefore, in this work, we developed a set of guidelines to help software developers perform this step. We reviewed 70 papers that report usability studies that have used cognitive dimension questionnaires and reviewed data analysis techniques those studies have employed. We identified five techniques that those studies have used for qualitative data analysis.

From the information gathered by the systematic literature review, we developed a set of guidelines to follow when analysing CDF questionnaire responses that are collected when evaluating the usability of security APIs. We converted the developed guidelines into a software developer understandable language by asking four software developers to provide feedback after going through the guidelines. This resulted in a set of guidelines, which are understandable for software developers. Finally, we tested the proposed guidelines by conducting a between subject empirical investigation. In the study, two groups of participants analysed a response provided by programmers who used a security API to develop an application with and without the help of the developed set of guidelines.

The results of the experiment revealed that the participants who used the proposed guidelines identified significantly more usability issues with a higher validity compared to participants who did not. Hence the results revealed that using the proposed guidelines helped software developers to analyse CDF questionnaire responses more effectively compared to those who do not use the guidelines.

Software development community would be the main party that would benefit as they can use the proposed set of guidelines when performing CDF based usability evaluations to analyse CDF questionnaire responses that they collect in security API usability evaluations. It will enable them to identify usability issues exist in their security APIs and fix them to deliver a more usable security APIs. It will make the lives of programmers who use the security API much easier and comfortable, and will also lead them to develop more secure applications using the security API.

However, the amount of issues that identified after using the guidelines are still not satisfactory. Therefore, the guidelines need to be improved to enable people who use them to identify more usability issues and further research needs to be conducted to improve them. In addition to that, this work evaluated the proposed approach by comparing it to a scenario where programmers approach the task without any guidance. It would be interesting to see how the proposed guidelines perform compared to other possible approaches that can be followed to achieve the same task such as grounded theory. When conducting future research in this area, the proposed approach should be compared with other approaches that can be followed to analyse CDF questionnaire responses, specially with approaches such as grounded theory that was identified in the systematic literature review and not used in this research.

10. REFERENCES

[1] Y. Acar, M. Backes, S. Fahl, S. Garfinkel, D. Kim, M. L. Mazurek, and C. Stransky. Comparing the usability of cryptographic APIs. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 154–171. IEEE, 2017.

[2] Y. Acar, C. Stransky, D. Wermke, M. Mazurek, and S. Fahl. Security developer studies with GitHub users: Exploring a convenience sample. In *Symposium on Usable Privacy and Security (SOUPS)*, 2017.

[3] A. Adams and M. A. Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999.

[4] H. Beyer and K. Holtzblatt. *Contextual design: defining customer-centered systems*. Elsevier, 1997.

[5] M. H. Blackmon, P. G. Polson, M. Kitajima, and C. Lewis. Cognitive walkthrough for the web. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 463–470. ACM, 2002.

[6] A. F. Blackwell and T. R. Green. A cognitive dimensions questionnaire optimised for users. In *proceedings of the twelfth annual meeting of the psychology of programming interest group*, pages 137–152, 2000.

[7] A. F. Blackwell, T. R. Green, and D. J. Nunn. Cognitive dimensions and musical notation systems. In *Proceedings of International Computer Music Conference, Berlin*. Citeseer, 2000.

[8] A. Bobkowska. A methodology of visual modeling language evaluation. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 72–81. Springer, 2005.

[9] N. Bolloju. Conceptual modeling of systems integration requirements. *IEEE software*, 26(5), 2009.

[10] M. Boshernitsan. Program manipulation via interactive transformations. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 392–393. ACM, 2003.

[11] M. Boshernitsan, S. L. Graham, and M. A. Hearst. Aligning development tools with the way programmers think about code changes. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 567–576. ACM, 2007.

[12] R. E. Boyatzis. *Transforming qualitative information: Thematic analysis and code development*. sage, 1998.

[13] C. Cillien, A. Calitz, and J. Greyling. The application of the cognitive dimension framework for notations as an instrument for the usability analysis of an introductory programming tool. *Alternation*, 12(1b):543–576, 2005.

[14] S. Clarke. Evaluating a new programming language. In *13th Workshop of the psychology of programming interest group*, pages 275–289, 2001.

[15] S. Clarke. Using the cognitive dimensions - abstraction level. `http://blogs.msdn.com/b/stevencl/archive/2003/11/14/57065.aspx`, 2003. Accessed: 2018-10-12.

[16] S. Clarke. Using the cognitive dimensions - learning style. `http://blogs.msdn.com/b/stevencl/archive/2003/11/24/57079.aspx`, 2003. Accessed: 2018-10-12.

[17] S. Clarke. Using the cognitive dimensions - progressive evaluation. `http://blogs.msdn.com/b/stevencl/archive/2003/12/22/45143.aspx`, 2003. Accessed: 2018-10-12.

[18] S. Clarke. Using the cognitive dimensions - working framework. `http://blogs.msdn.com/b/stevencl/archive/2003/12/03/57112.aspx`, 2003. Accessed: 2018-10-12.

[19] S. Clarke. Using the cognitive dimensions -work step unit. `http://blogs.msdn.com/b/stevencl/archive/2003/12/22/45142.aspx`, 2003. Accessed: 2018-10-12.

[20] S. Clarke. Measuring API usability. *Dr. Dobb's Journal Windows*, pages S6–S9, 2004.

[21] S. Clarke. Using the cognitive dimensions - API elaboration. `http://blogs.msdn.com/b/stevencl/archive/2004/02/27/81317.aspx`, 2004. Accessed: 2018-10-12.

[22] S. Clarke. Using the cognitive dimensions - API viscosity. `http://blogs.msdn.com/b/stevencl/archive/2004/03/10/87652.aspx`, 2004. Accessed: 2018-10-12.

[23] S. Clarke. Using the cognitive dimensions - consistency. `http://blogs.msdn.com/b/stevencl/archive/2004/03/17/91626.aspx`, 2004. Accessed: 2018-10-12.

[24] S. Clarke. Using the cognitive dimensions - domain correspondence. `http://blogs.msdn.com/b/stevencl/archive/2004/05/17/133439.aspx`, 2004. Accessed: 2018-10-12.

[25] S. Clarke. Using the cognitive dimensions - penetrability. `http://blogs.msdn.com/b/stevencl/archive/2004/02/13/72646.aspx`, 2004. Accessed: 2018-10-12.

[26] S. Clarke. Using the cognitive dimensions - premature commitment. `http://blogs.msdn.com/b/stevencl/archive/2004/01/22/61859.aspx`, 2004. Accessed: 2018-10-12.

[27] S. Clarke. Using the cognitive dimensions - role expressiveness. `http://blogs.msdn.com/b/stevencl/archive/2004/04/23/119147.aspx`, 2004. Accessed: 2018-10-12.

[28] J. Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[29] J. Cohen. *Statistical power analysis for the behavioral sciences*. Academic Press, New York, 1969.

[30] M. A. Cohen, F. E. Ritter, and S. R. Haynes. Evaluating design: A formative evaluation of agent development environments used for teaching rule-based programming. In *proceedings of the Information Systems Education Conference*, pages 1542–7382, 2009.

[31] C. Datta. *Programming Behaviour of Personal Service Robots with Application to Healthcare*. PhD thesis, ResearchSpace@ Auckland, 2014.

[32] J. Diprose, B. MacDonald, J. Hosking, and B. Plimmer. Designing an API at an appropriate abstraction level for programming social robot applications. *Journal of Visual Languages & Computing*, 39:22–40, 2017.

[33] J. P. Diprose, B. Plimmer, B. A. MacDonald, and

J. G. Hosking. A human-centric API for programming socially interactive robots. In *Visual Languages and Human-Centric Computing (VL/HCC), 2014 IEEE Symposium on*, pages 121–128. IEEE, 2014.

[34] M. Duignan. *Computer mediated music production: A study of abstraction and activity*. PhD thesis, Victoria University of Wellington, 2008.

[35] B. Ellis, J. Stylos, and B. Myers. The factory pattern in API design: A usability evaluation. In *Proceedings of the 29th international conference on Software Engineering*, pages 302–312. IEEE Computer Society, 2007.

[36] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith. Why eve and mallory love android: An analysis of android ssl (in) security. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 50–61. ACM, 2012.

[37] S. Fahl, M. Harbach, H. Perl, M. Koetter, and M. Smith. Rethinking ssl development in an appified world. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 49–60. ACM, 2013.

[38] F. J. Fowler Jr and T. W. Mangione. *Standardized survey interviewing: Minimizing interviewer-related error*, volume 18. Sage, Thousand Oaks, CA, 1990.

[39] D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson. What happens when software developers are (un) happy. *Journal of Systems and Software*, 140:32–47, 2018.

[40] T. R. Green. Cognitive dimensions of notations. *People and computers V*, pages 443–460, 1989.

[41] T. R. G. Green and M. Petre. Usability analysis of visual programming environments: a âĂŸcognitive dimensionsâĂŹ framework. *Journal of Visual Languages & Computing*, 7(2):131–174, 1996.

[42] R. M. Groves, F. J. Fowler Jr, M. P. Couper, J. M. Lepkowski, E. Singer, and R. Tourangeau. *Survey methodology*, volume 561. John Wiley & Sons, Hoboken, NJ, 2011.

[43] J. C. Grundy, J. Hosking, K. N. Li, N. M. Ali, J. Huh, and R. L. Li. Generating domain-specific visual language tools from abstract visual specifications. *IEEE Transactions on Software Engineering*, 39(4):487–515, 2013.

[44] H. Hsu and P. A. Lachenbruch. Paired t test. *Wiley encyclopedia of clinical trials*, pages 1–3, 2007.

[45] ISO. 9126-3: Software engineering-product quality-part 3: Internal metrics. *International Organization for Standardization, Geneva, Switzerland*, 2003.

[46] ISO. 9126-4:-software engineering-product quality-part 4:-quality in use metrics. *International Organization for Standardization, Geneva, Switzerland. Switzerland*, 2004.

[47] G. Kadoda. A cognitive dimensions view of the differences between designers and users of theorem proving assistants. In *Proceedings of the Twelfth Annual Workshop of the Psychology of Programming Interest Group*, 2000.

[48] G. Kadoda, R. Stone, and D. Diaper. Desirable features of educational theorem provers? a cognitive dimensions viewpoint. In *Proceedings of Psychology of Programming Interest Group conference*, 1999.

[49] C.-M. Karat, R. Campbell, and T. Fiegel. Comparison of empirical testing and walkthrough methods in user interface evaluation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 397–404. ACM, 1992.

[50] B. Khazaei and E. Triffitt. Applying cognitive dimensions to evaluate and improve the usability of Z formalism. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 571–577. ACM, 2002.

[51] B. Kitchenham. Guidelines for performing systematic literature reviews in software engineering. Technical report, Keele University and University of Durham, 2007.

[52] Y. Kollet and T. J. Smedley. Message-flow programming in PdaGraph. In *Visual Languages and Human Centric Computing, 2004 IEEE Symposium on*, pages 229–232. IEEE, 2004.

[53] R. Korf. COBrA-A concept ontology browser for anatomy. Technical report, Informatics Report EDI-INF-IM030022, 2003.

[54] M. N. Marshall. Sampling for qualitative research. *Family practice*, 13(6):522–526, 1996.

[55] S. G. McLellan, A. W. Roesler, J. T. Tempest, and C. I. Spinuzzi. Building more usable APIs. *IEEE software*, 15(3):78–86, 1998.

[56] E. Mosqueira-Rey, D. Alonso-Rios, V. Moret-Bonillo, I. Fernandez-Varela, and D. Alvarez-Estevez. A systematic approach to API usability: Taxonomy-derived criteria and a case study. *Information and Software Technology*, 97:46 – 63, 2018.

[57] J. Nielsen. Finding usability problems through heuristic evaluation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 373–380. ACM, 1992.

[58] J. Nielsen. *Usability engineering*. Elsevier, 1994.

[59] J. Nielsen and R. L. Mack. *Usability Inspection Methods*. John Wiley & sons, Inc., 1994.

[60] J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249–256. ACM, 1990.

[61] H. Ossher, R. Bellamy, I. Simmonds, D. Amid, A. Anaby-Tavor, M. Callery, M. Desmond, J. de Vries, A. Fisher, and S. Krasikov. Flexible modeling tools for pre-requirements analysis: conceptual architecture and research challenges. *ACM Sigplan Notices*, 45(10):848–864, 2010.

[62] M. Perscheid, B. Siegmund, M. Taeumel, and R. Hirschfeld. Studying the advancement in debugging practice of professional software developers. *Software Quality Journal*, 25(1):83–110, 2017.

[63] H. Petrie and C. Power. What do users really care about?: a comparison of usability problems found by users and experts on highly interactive websites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2107–2116. ACM, 2012.

[64] M. Piccioni, C. A. Furia, and B. Meyer. An empirical study of API usability. In *Empirical Software Engineering and Measurement, 2013 ACM/IEEE international symposium on*, pages 5–14. IEEE, 2013.

[65] P. G. Polson, C. Lewis, J. Rieman, and C. Wharton. Cognitive walkthroughs: A method for theory-based evaluation of user interfaces. *International Journal of man-machine studies*, 36(5):741–773, 1992.

[66] R. Razali, C. Snook, M. Poppleton, and P. Garratt. Usability assessment of a UML-based formal modeling method using a cognitive dimensions framework. *Human Technology: An Interdisciplinary Journal on Humans in ICT Environments*, 2008.

[67] R. Razali, C. Snook, M. Poppleton, P. Garratt, and R. Walters. Usability assessment of a UML-based formal modelling method. In *19th Annual Psychology of Programming Workshop (PPIG'07)*, pages 56–71, 2007.

[68] E. M. Redmiles, Y. Acar, S. Fahl, and M. L. Mazurek. A summary of survey methodology best practices for security and privacy researchers. Technical report, University of Maryland, 2017.

[69] T. Scheller and E. Kühn. Automated measurement of API usability: The API concepts framework. *Information and Software Technology*, 61:145–162, 2015.

[70] A. Sears. Heuristic walkthroughs: Finding the problems without the noise. *International Journal of Human-Computer Interaction*, 9(3):213–234, 1997.

[71] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.

[72] G. W. Snedecor and W. G. Cochran. Statistical methods, eight edition. *Iowa state University press, Ames, Iowa*, 1989.

[73] StackOverflow. Developer survey results 2019. `https://insights.stackoverflow.com/survey/2019`, 2019. Accessed: 2019-07-17.

[74] A. Stead. User-configurable machine vision for mobiles. *Proceedings of the Psychology of Programming Interest Group*, 2011.

[75] A. Stead, A. F. Blackwell, and S. Aaron. Graphic score grammars for end-users. In *NIME*, 2012.

[76] A. Strauss and J. Corbin. *Basics of qualitative research: Techniques and procedures for developing grounded theory.* Sage publications, 1998.

[77] J. Stylos. *Making APIs more usable with improved API designs, documentation and tools.* PhD thesis, Carnegie Mellon University, 2009.

[78] J. Stylos and S. Clarke. Usability implications of requiring parameters in objects' constructors. In *Proceedings of the 29th international conference on Software Engineering*, pages 529–539. IEEE Computer Society, 2007.

[79] J. Stylos, B. Graf, D. K. Busse, C. Ziegler, R. Ehret, and J. Karstens. A case study of API redesign for improved usability. In *Visual Languages and Human-Centric Computing, 2008. VL/HCC 2008. IEEE Symposium on*, pages 189–192. IEEE, 2008.

[80] J. Stylos and B. A. Myers. The implications of method placement on API learnability. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 105–112. ACM, 2008.

[81] A. Teitelbaum. *Arts' Codes: a new methodology for the development of real-time embedded applications for control systems.* PhD thesis, RMIT University, 2006.

[82] M. Tukiainen. Evaluation of the cognitive dimensions questionnaire and some thoughts about the cognitive dimensions of spreadsheet calculation. In *Proceedings of the 13th Workshop of the Psychology of Programming Interest Group*, pages 291–301, 2001.

[83] M. Vinni. *Cognitive Dimensions in Software Development.* PhD thesis, Joensuu School of Computing, 2012.

[84] M. Weinstein. TAMS analyzer. `http://tamsys.sourceforge.net/`. Accessed: 2019-02-28.

[85] C. Wijayarathna and N. A. Arachchilage. An empirical usability analysis of the google authentication api. In *Proceedings of the 23rd International Conference on Evaluation and Assessment in Software Engineering 2019.* ACM, 2019.

[86] C. Wijayarathna, N. A. Arachchilage, and J. Slay. A generic cognitive dimensions questionnaire to evaluate the usability of security APIs. In *International Conference on Human Aspects of Information Security, Privacy, and Trust*, pages 160–173. Springer, 2017.

[87] C. Wijayarathna and N. A. G. Arachchilage. A methodology to evaluate the usability of security apis. In *9th IEEE International Conference on Information and Automation for Sustainability*, 2018.

[88] C. Wijayarathna and N. A. G. Arachchilage. Using cognitive dimensions to evaluate the usability of security apis: An empirical investigation. *Information and Software Technology*, 115:5 – 19, 2019.

[89] C. Wijayarathna and N. A. G. Arachchilage. Why johnny can't develop a secure application? a usability analysis of java secure socket extension API. *Computers & Security*, 80:54 – 73, 2019.

[90] C. A. Wingrave. *A Reflection Inspired and Language Derived Formalism for Overcoming 3D Interface Design Complexity.* PhD thesis, Virginia Polytechnic Institute and State University, 2005.

[91] C. A. Wingrave. *Concept-Oriented design in Chasm: Conversational domain language inspired 3D user interface design and development.* PhD thesis, Virginia Tech, 2008.

[92] C. Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, page 38. ACM, 2014.

[93] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering.* Springer Science & Business Media, 2012.

APPENDIX

A. PARTICIPANT RECRUITMENT STATISTICS OF THE EMPIRICAL EVALUATION

- Invitations sent for email addresses from GitHub : 2750
- Number of emails that bounced : 89
- Number of complaints received : 5
- Number of invitees participated within a week : 20
- Number of participants recruited via advertisements : 16

B. SAMPLE SECTION FROM THE EXPERIMENTAL GROUP INSTRUMENT (SECTION 1/15)

In this section of the study, we are trying to identify usability issues of a security API related to its abstraction level aspect. Following are the instructions from our guidebook for identifying usability issues of a security API related to the abstraction level from programmer feedback. Please read the following carefully.

Table from the guidelines document comes here.

Now you have went through a section of our guidebook, lets see if it helps you to identify usability issues from a feedback given by a programmer.

Following is the feedback that was provided by a programmer who used the Google authentication API to implement sign-in and sign-out functionalities into a simple web application. You can refer to changes they had to do at `https://drive.google.com/open?id=1SpfmMIPG1oEl1piK0Jku6Xxt0A0rjJUN` .

Q1 : Were you able to implement the core functionality of the task you completed using only one class from the API or more than one class?

A : More than one class was needed.

Q2 : Did you expect to use only one class or more than one class to implement the core functionality of the task you completed?

A : I expected to use more than one class.

Q3 : How would you describe your experiences with respect to the types of classes that you worked with while completing the task?

A : They were too low level.

Q4 : Explain.

A : I didn't expect that amount of complexity when trying to implement simple login, all these factories etc.

Use below space to list the usability issues related to the abstraction level of the Google Authentication API that the programmer has encountered. When reporting issues, try to be specific as possible and specify API components, documents, learning styles, etc. that are infected from the issue. (Use a new line when reporting a new issue. Mention 'no issues' if the programmer has not encountered any usability issue related to the API's abstraction level.)

C. SAMPLE SECTION FROM THE CONTROL GROUP INSTRUMENT (SECTION 1/15)

Following is a feedback that was provided by a programmer who used the Google authentication API to implement sign-in and sign-out functionalities into a simple web application. You can refer to changes they had to do at `https://drive.google.com/open?id=1SpfmMIPG1oEl1piK0Jku6Xxt0A0rjJUN` .

Q1 : Were you able to implement the core functionality of the task you completed using only one class from the API or more than one class?

A : More than one class was needed.

Q2 : Did you expect to use only one class or more than one class to implement the core functionality of the task you completed?

A : I expected to use more than one class.

Q3 : How would you describe your experiences with respect to the types of classes that you worked with while completing the task?

A : They were too low level.

Q4 : Explain.

A : I didn't expect that amount of complexity when trying to implement simple login, all these factories etc.

Use below space to list the usability issues of the Google Authentication API that the programmer has encountered. When reporting issues, try to be specific as possible and specify API components, documents, learning styles, etc. that are infected from the issue. (Use a new line when reporting a new issue. Mention 'no issues' if the feedback does not mention any usability issue.)

D. LIST OF PAPERS REVIEWED IN THE SYSTEMATIC LITERATURE REVIEW

Author/s	**Title**	**Year**	**Venue**
Starting Set			
Kadoda et al.	Desirable features of educational theorem provers? A Cognitive Dimensions viewpoint	1999	Psychology of Programming Interest Group conference
Blackwell and Green	A cognitive dimensions questionnaire optimised for users	2000	Psychology of Programming Interest Group conference
Papers Reviewed in the First Iteration			
Blackwell et al.	Cognitive dimensions and musical notation systems	2000	International Computer Music Conference
Kadoda	A cognitive dimensions view of the differences between designers and users of theorem proving assistants	2000	Psychology of Programming Interest Group conference
Clarke	Evaluating a new programming language	2001	Psychology of Programming Interest Group conference
Tukiainen	Evaluation of the cognitive dimensions questionnaire and some thoughts about the cognitive dimensions of spreadsheet calculation	2001	Psychology of Programming Interest Group conference
Khazaei and Triffitt	Applying cognitive dimensions to evaluate and improve the usability of Z formalism	2002	International Conference on Software Engineering and Knowledge Engineering
Blackwell and Green	Notational systems - the cognitive dimensions of notations framework.	2003	HCI Models, Theories, and Frameworks: Toward an Interdisciplinary Science.
Boshernitsan	Program manipulation via interactive transformations	2003	ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications
Korf	COBrA - A Concept Ontology Browser for Anatomy	2003	Masters Thesis - University of Edinburgh
Blackwell et al.	Champagne prototyping: A research technique for early evaluation of complex end-user programming systems	2004	IEEE Symposium on Visual Languages-Human Centric Computing
Kollet and Smedley	Message-flow programming in PdaGraph	2004	IEEE Symposium on Visual Languages-Human Centric Computing
Bobkowska	A methodology of visual modeling language evaluation	2005	International Conference on Current Trends in Theory and Practice of Computer Science
Cilliers et al.	The Application of the Cognitive Dimension Framework for notations as an instrument for the usability analysis of an introductory programming tool	2005	Alternation
Roast et al.	Enhancing contextual analysis to support the design of development tools	2005	People and Computers XVIII - Design for Life
Wingrave	A reflection inspired and language derived formalism for overcoming 3D interface design complexity	2005	Doctoral dissertation - Virginia Polytechnic Institute and State University
Boshernitsan and Graham	Interactive transformation of Java programs	2006	international conference on Software engineering
Dagit et al.	Using cognitive dimensions advice from the trenches	2006	Journal of Visual Languages & Computing
Duignan et al.	Activity theory for design from checklist to interview	2006	Human Work Interaction Design: Designing for Human Work
Green et al.	Cognitive dimensions: Achievements, new directions, and open questions	2006	Journal of Visual Languages & Computing
Teitelbaum	Arts' Codes: A new methodology for the development of real-time embedded applications for control systems	2006	Doctoral Dissertation - RMIT University
Boshernitsan et al. [11]	Aligning development tools with the way programmers think about code changes	2007	SIGCHI conference on Human factors in computing systems
Razali et al.	Usability assessment of a UML-based formal modelling method	2007	Psychology of Programming Interest Group conference
Duignan	Computer mediated music production: A study of abstraction and activity	2008	Masters Thesis - Victoria University of Wellington
Gupta	Houses in reach: A personal real estate monitoring and mining application	2008	Masters Thesis - University of Victoria
Martin and Hughes	Cognitive dimensions questionnaire applied to exploratory algorithm design	2008	conference on Innovation and technology in computer science education

Author/s	**Title**	**Year**	**Venue**
Razali et al.	Usability assessment of a UML-based formal modeling method using a cognitive dimensions framework	2008	Human Technology: An Interdisciplinary Journal on Humans in ICT Environments
Wingrave	Concept-Oriented design in Chasm: Conversational domain language inspired 3D user interface design and development	2008	PhD Thesis - Virginia Tech
Wingrave and Bowman	Tiered developer-centric representations for 3d interfaces: Concept-oriented design in chasm	2008	Virtual Reality Conference
Arcs and Razali	Cognitive dimensions and grounded theory in learning software modeling	2009	Procedia-Social and Behavioral Sciences
Bolloju	Conceptual modeling of systems integration requirements	2009	IEEE Software
Cohen et al.	Evaluating design: A formative evaluation of agent development environments used for teaching rule-based programming	2009	Information Systems Education Conference
Neumann et al.	End-user strategy programming	2009	Journal of Visual Languages & Computing
Wingrave et al.	A natural, tiered and executable UIDL for 3D user interfaces based on Concept-Oriented Design	2009	ACM Transactions on Computer-Human Interaction ACM Transactions on Computer-Human Interaction
Al Sarraj and De Troyer	Web mashup makers for casual users: A user experiment	2010	International Conference on Information Integration and Web-based Applications & Services.
Cohen et al.	Applying software engineering to agent development	2010	AI Magazine
Duignan et al.	Abstraction and activity in computer-mediated music production	2010	Computer Music Journal
Mohd Ali	Critic specification for domain-specific visual language tools	2010	Masters thesis - University of Auckland
Mohd Ali et al.	End-user oriented critic specification for domain-specific visual language tools	2010	IEEE/ACM international conference on Automated software engineering
Ossher et al.	Flexible modeling tools for pre-requirements analysis: conceptual architecture and research challenges	2010	ACM international conference on Object oriented programming systems languages and applications
Wingrave and LaViola	Reflecting on the design and implementation issues of virtual environments	2010	Presence, vol. 19, no. 2
Kamalrudin et al.	Improving requirements quality using essential use case interaction patterns	2011	International Conference Software engineering
Kutar et al.	The cognitive dimensions questionnaire: Adapting for non-expert users	2011	Psychology of Programming Interest Group conference
Stead	User configurable machine vision for mobiles	2011	Psychology of Programming Interest Group conference
Beckert and Grebbing	Evaluating the usability of interactive verification systems	2012	COMPARE
Cohen et al.	Dimensions of concern: A method to use cognitive dimensions to evaluate interfaces	2012	ACM Transactions on Computer-Human Interaction (TOCHI)
Nash	Supporting virtuosity and flow in computer music	2012	PhD Thesis - University of Cambridge
Nash and Blackwell	Liveness and flow in notation use	2012	New Interfaces for Musical Expression
Rizzolo	Learning based programming	2012	PhD Thesis - University of Illinois at Urbana-Champaign
Stead et al.	Graphic score grammars for end-users	2012	New Interfaces for Musical Expression
Trainer	Supporting trust in globally distributed software teams: The impact of visualized collaborative traces on perceived trustworthiness	2012	PhD Thesis - University of California
Vinni	Cognitive Dimensions in Software Development : Case of a Mobile Game Content Editor	2012	Masters Thesis - Joensuu School of Computing
Blackwell and Charampadis	Practice-led design and evaluation of a live visual constraint language	2013	Technical Report - University of Cambridge
Grundy et al.	Generating domain-specific visual language tools from abstract visual specifications	2013	IEEE Transactions on Software Engineering
Martin	Methods to support end user design of arrangement-level musical decision making	2013	PhD Thesis - The University of Sydney
Datta	7 programming behaviour of personal service robots with application to healthcare	2014	PhD Thesis - Researcg Space @ Auckland

Author/s	**Title**	**Year**	**Venue**
Diprose et al.	A human-centric API for programming socially interactive robots	2014	IEEE Symposium on Visual Languages and Human-Centric Computing
Liakopoulou and Mavrommati	UbiComp applications for assisting visually impaired people live an independent life: A participatory conceptualization design phase	2014	International Conference on Distributed, Ambient, and Pervasive Interactions
Nash	The cognitive dimensions of music notations	2015	International Conference on Technologies for Music Notation and Representation
Assal et al.	Cesar: Visual representation of source code vulnerabilities	2016	IEEE Symposium on Visualization for Cyber Security
Diprose et al.	Designing an API at an appropriate abstraction level for programming social robot applications	2017	Journal of Visual Languages & Computing
Lopez-Fernandez et al.	Designing and evaluating the usability of an API for real-time multimedia services in the Internet	2017	Multimedia Tools and Applications
Zevallos	A comparison of a Rule Definition Language (RDL) and the JAVA object oriented language for implementing a distributed system	Unknown	Unknown
Papers Reviewed in the Second Iteration			
Bobkawska	Cognitive dimensions questionnaire applied to visual modelling language evaluation - a case study	2003	Psychology of Programming Interest Group conference
Kutar et al.	Cognitive dimensions: An experience report	2000	Psychology of Programming Interest Group conference
Kutar et al.	A comparison of empirical study and cognitive dimensions analysis in the evaluation of UML diagrams	2002	Psychology of Programming Interest Group conference
Triffit and Khazaei	A study of usability of Z formalism based on cognitive dimensions	2002	Psychology of Programming Interest Group conference
Cohen	A theory-based environment for creating reusable cognitive models	2008	PhD Thesis - The Pennsylvania State University

Table 8: Papers Reviewed in the Systematic Literature Review