Test case design for context-aware applications

Santos, Ismayle de Sousa; Andrade, Rossana Maria de Castro; Rocha, Lincoln Souza; Matalonga, Santiago; de Oliveira, Káthia Marçal; Travassos, Guilherme Horta

Published in:
Information and Software Technology

DOI:
10.1016/j.infsof.2017.03.008

Published: 01/08/2017

Document Version
Peer reviewed version

Citation for published version (APA):
Santos, I. D. S., Andrade, R. M. D. C., Rocha, L. S., Matalonga, S., de Oliveira, K. M., & Travassos, G. H. (2017). Test case design for context-aware applications: Are we there yet? *Information and Software Technology*, *88*, 1-16. https://doi.org/10.1016/j.infsof.2017.03.008

Test Case Design for Context-Aware Applications: Are We There Yet?

Ismayle S. Santos[a,1,**], Rossana M. C. Andrade[a,2,*], Lincoln S. Rocha[a], Santiago Matalonga[b], Káthia M. Oliveira[c], Guilherme H. Travassos[d,3,*]

[a]*Federal University of Ceará, Ceará, Brazil*
[b]*Universidad ORT Uruguay, Montevideo, Uruguay*
[c]*University of Valenciennes, LAMIH, UMR CNRS 8201, Valenciennes, France*
[d]*Federal University of Rio de Janeiro, Rio de Janeiro, Brazil*

Abstract

Context: Current software systems have increasingly implemented context-aware adaptations to handle the diversity of conditions of their surrounding environment. Therefore, users are becoming used to a variety of context-aware software systems (CASS). This feature brings challenges to the software construction and testing because the context is unpredictable and may change at any time. Therefore, software engineers need to guarantee the dynamical context changing while testing CASS. Different test case design techniques (TCDT) have been proposed to support the testing of CASS. However, to the best of our knowledge, there is no analysis of these proposals on the advantages, limitations and their effective support to context variation during testing.
Objective: To gather empirical evidence on TCDT concerned with CASS by identifying, evaluating and synthesizing knowledge available in the literature.
Method: To undertake a secondary study (quasi-Systematic Literature Review) on TCDT for CASS regarding their assessed quality characteristics, used coverage criteria, test type, and test technique.
Results: From 833 primary studies published between 2004 and 2014, just 17 regard the design of test cases for CASS. Most of them focus on functional suitability. Furthermore, some of them take into account the changes in the context by providing specific test cases for each context configuration (static perspective) during the test execution. These 17 studies have revealed five challenges affecting the design of test cases and 20 challenges regarding the testing of CASS. Besides, seven TCDT are not empirically evaluated.
Conclusion: A few TCDT partially support the test of CASS. However, it has not been observed evidence on any TCDT supporting context-aware tests fully, i.e. that can adapt the expected output based on the context variation (dynamic perspective) during the test execution (runtime). It is an open issue deserving greater attention from researchers to increase testing coverage and ensure users confidence in CASS.

Keywords: Context Aware Application, Systematic Review, Software Testing

1. Introduction

In ubiquitous computing, the computers are merged with everyday objects that people use in their daily tasks. This is possible due to technological developments such as the continuously decreasing physical size, power consumption, production price, and ecological impact of computing devices, and due to their increasing processing power, storage capacity and communication bandwidth [1]. According to Dey and Abowd [2], context is any information that can be used to characterize the situation of an entity (person, place or object) that is considered relevant to the interaction between a user and an application, including the user and the applications themselves. So, aiming to help the user in daily activities, ubiquitous systems must be able to capture the information about the context and use it to guide their behavior and to assist users while they performing their tasks. As the context information impacts the system's output, testing the context-aware behavior requires an efficient test cases design to expose failures that can occur only in specific context situations.

The ISO/IEC/IEEE 29119-1 [3] defines test design technique (also known as test case design technique) as "activities, concepts, processes, and patterns used to construct a test model that is used to identify test conditions for a test item, derive corresponding test coverage items, and

[*]Corresponding author
[**]Principal corresponding author
Email addresses: ismaylesantos@great.ufc.br (Ismayle S. Santos), rossana@ufc.br (Rossana M. C. Andrade), lincoln@great.ufc.br (Lincoln S. Rocha), smatalonga@uni.ort.edu.uy (Santiago Matalonga), kathia.oliveira@univ-valenciennes.fr (Káthia M. Oliveira), ght@cos.ufrj.br (Guilherme H. Travassos)
[1]PhD Scholarship (MDCC/DC/UFC) sponsored by CAPES
[2]Researcher Scholarship - DT Level 2, sponsored by CNPq
[3]Researcher Scholarship - PQ Level 1D, sponsored by CNPq

subsequently derive or select test cases". Thus, once the test design technique defines what will be tested, the success of the testing activity is directly related to the choice of the adequate technique. In regards to test design for the context-aware systems, the challenge is related to continuous change of context information [4, 5] and can explode the permutation of available test space [6]. Different test techniques were proposed in the literature taking into account the context in the test case design, (See Section 4.4). Nevertheless, because the lack of an investigation about their state of the art, there is no answer to an important question in this scenario: "Are the current test case techniques adequate to test context-aware systems?"

To move towards the answer of this issue, in this paper we present the results of a quasi Systematic Literature Review (qSLR) [7], whose objective was to investigate the context influences in the design of test cases for context-aware applications. It is worth noting that a qSLR involves the same rigor and formalism that an SLR [8]. The difference is that a qSLR does not involve a meta-analysis [7], and this analysis is not adequate for our goal that was to characterize the state of the art of test case design techniques for context-aware applications.

Therefore, we investigate through a qSLR the following research questions: *(RQ1) What are the existing methods or techniques of test case design for context-aware application testing? (RQ2) Which quality characteristic are being evaluated by the test cases designed for context-aware applications? (RQ3) What are the coverage criteria used in the context-aware application testing? (RQ4) How does context influence the test case design in the context-aware application testing?*

It was possible to identify and characterize approaches from 17 papers for the test case design of context-aware applications. As the main result of the analysis of these papers, we observed a lack of test case design techniques supporting changes in the expected output according to the current context. We argue that context-aware testing oracles are necessary to deal with the dynamic nature of the context. Besides, we identified a set of 20 challenges related to context-aware application testing. Therefore, we believe that the results of this qSLR will benefit researchers and practitioners. Researchers will benefit because the results indicate gaps that need further investigation and the practitioners will benefit because the results are useful as a reference for testing context-aware applications.

The rest of the paper is organized as follows. In Section 2, we depict the basic concepts related to the test case design activity, as well as the related research project and our definition of context and context-awareness. In Section 3, we present the research protocol and the process used. In Section 4, we describe the results of this review. In Section 5, we discuss the techniques empirically validated, as well as the threats to validity and some opportunities for future research. In Section 6, we discuss related work. Finally, in Section 7, we conclude the paper.

2. Background

In this section, we discuss some basic concepts related to the testing activity and that are important to understanding the results presented in this paper. Furthermore, we present a brief description about the Context-Aware Testing for Ubiquitous Systems (CAcTUS) [4] Project, which is the research project linked to the qSLR presented in this paper, as well as the definitions of context and context-awareness used.

2.1. Test Case Design

According to Myers et al. [9], the testing "is the process of executing a program with the intent of finding errors", where an error is present if a program does not do what it is supposed to do or if it does what it is not supposed to do. It is worth noting that exhaustive testing is unlikely, and some rationales for this are: (i) the domain of possible inputs of a program is too large; (ii) it may not be feasible to simulate all possible system environment conditions; and (iii) the high cost of this activity, which could exceed 50 percent of the total cost of the software development [9]. Thus, the testing is performed with a finite set of test cases, suitably selected from the usually infinite executions domain, against the expected behavior" [10].

To design test cases with a higher probability of finding defects is important to use an adequate test design technique to the needs. In the context-aware applications domain, where the context information expands the possible usage scenarios, test case design techniques have a fundamental role to reduce the number of tests.

According to the ISO/IEC/IEEE 29119-4 [11], a test case design technique provides guidance on the derivation of *test conditions*, *test coverage items* and *test cases*.

- A *test condition* is a testable aspect of a test item, such as a function, transaction, feature, quality attribute or structural element identified as a basis for testing;
- *Test coverage items* are attributes of each test condition that can be covered during testing; and
- A *test case* is a set of preconditions, inputs and expected results, developed to determine whether or not the covered part of the test item has been implemented correctly.

The ISO/IEC 29119-4 [11] also describes a series of techniques that have wide acceptance in the software testing industry. It also classifies such techniques for three types of tests based on the source of information used to design the test cases: (i) Specification-based testing, where the source is the test basis (e.g. requirements); (ii) Structure-based testing, where the test item structure (e.g. source code) is used as information source; and

[4]http://lens.cos.ufrj.br/cactus/

(iii) Experience-based testing, where the tester's knowledge and experience are used as primary data source.

It is worth to highlight that characteristics of the application under test impact the test case designing activity. In context-aware software systems, the context information is a new kind of input affecting the application behavior [12] and, therefore, should be taken into account to design a test case.

2.2. The CAcTUS Project

To investigate the software quality in ubiquitous computing, a first research project called Maximum[5] was performed during 2012 to 2014 to propose software measures to assess the quality of ubiquitous systems focusing in the human-computer interaction [13, 14, 15]. From this project, we realized the need for investigating how the testing activity is impacted by the context awareness, as it is one of the main characteristics of ubiquitous systems [16, 17, 18].

Thus, Context-Aware Testing for Ubiquitous Systems (CAcTUS) is an ongoing research project that aims to understand and propose test strategies for ubiquitous systems. The CAcTUS (lens.cos.ufrj.br/cactus) involves researchers from three universities (Universidade Federal do Rio do Janeiro, Universidade Federal do Ceará both in Brazil and Université de Valenciennes et du Hainaut-Cambrésis in France). Since the interaction issues in ubiquitous systems goes beyond the human-computer relationship, encompassing the interaction among different devices and systems that we call **actor-computer interaction**. Therefore, its objective is to understand test strategies for the quality assessment of actor-computer interaction in ubiquitous systems. In this kind of interaction, the actor may be not only the user but also another computer (device) or software to better meet the ubiquitous software system non-functional requirements, such as invisibility [17].

With the purpose of enhancing the importance of the relationship between actors and computers by hinting that context can only be understood from the actor (human or system) viewpoint, we evolved Dey and Abowd's context definition in the CAcTUS research project. Our definition is that *"Context" is any piece of information that may be used to characterize the situation of an entity (logical and physical objects present in the systems environment) and its relations that are relevant for the actor-computer interaction.* Furthermore, in the CAcTUS project *"Context-Awareness" is a dynamic property representing a piece of information, which can evolutionarily affect the overall behavior of the software system in the interaction between the actor and the computer.*

To achieve CAcTUS' goal, it was defined as the activity to investigate the state-of-the-art through two quasi Systematic Literature Review - qSLR [7] on how context-awareness affects the testing activity.

[5]http://great.ufc.br/maximum/

Both qSLRs shared the aforementioned definitions of *context, context-awareness and actor-computer interaction.* Besides, they investigated the same population (context-awareness systems) and used the same classifications for the empirical study, test type, and test technique. These issues were defined in the first CAcTUS qSLR protocol [19] and are presented together with other information in Section 3 - Research Methodology and Protocol.

The first qSLR of the CAcTUS focused on the different activities of the testing process of a context-aware application, and their results were published in two previous work [6, 20]. The first paper [6] describes the initial efforts evaluating whether the observed techniques for testing context-aware software can be matched with the ISO/IEC/IEEE 29119 categories [11]. The second one [20] presents the challenges for testing context-aware applications that are related to hardware and network constraints, and to the context variation.

This paper, in turn, presents the results of the second performed qSLR that goes a step beyond in this investigation by focusing on the impact of the context in the test case design.

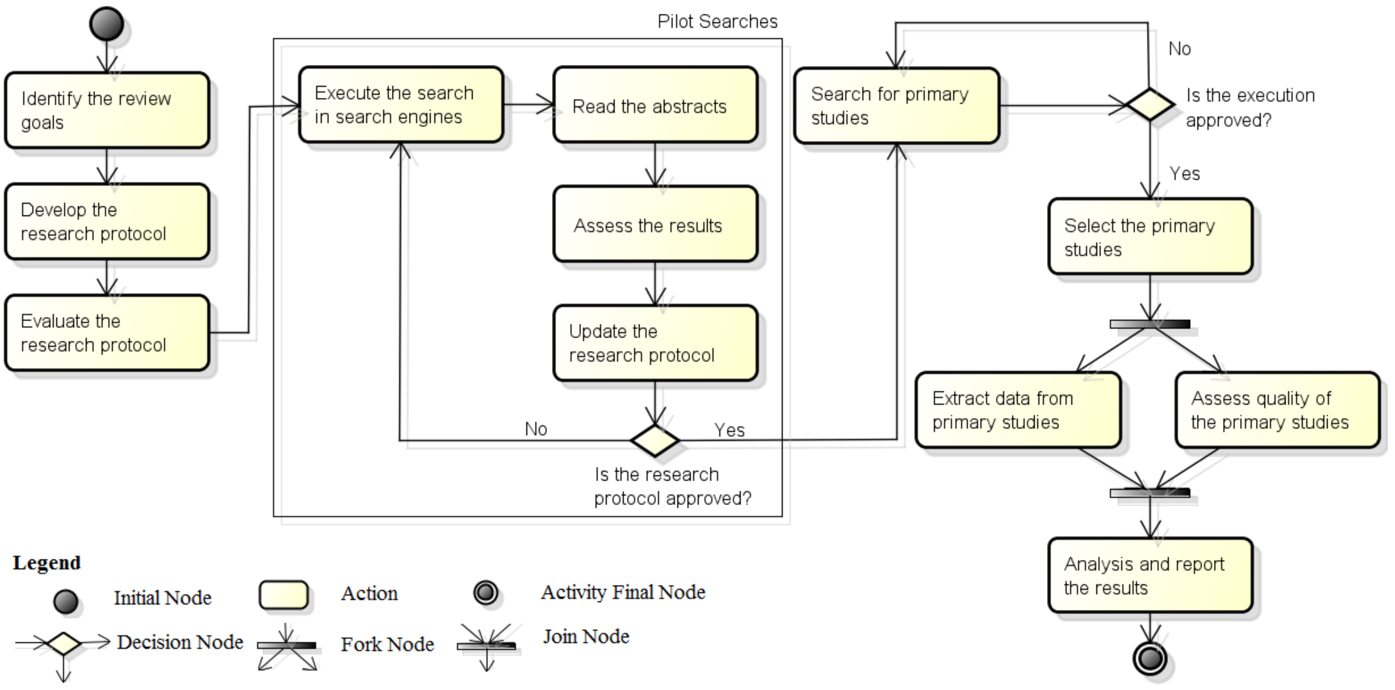
3. Research Methodology and Protocol

We conducted this review based on the guidelines by Biolchini et al. [21], and Kitchenham and Charters [22] for performing systematic literature reviews in Software Engineering. A Systematic Literature Review (SLR) is a form of secondary study that uses a clear methodology to identify, analyze and interpret all available evidence related to a specific research question in a way that is unbiased and (to a degree) repeatable [22]. Then, an SLR presents a fair evaluation of a research topic by using a rigorous and auditable methodology.

Although keeping the rigor of a systematic literature review, because of the objective of this study (mainly characterization) we did not apply any comparison among the studies. Therefore, we can classify it as quasi-Systematic Literature Review [7].

Figure 1 presents the steps performed in this qSLR. In the first step (*Identify the review goals*), we established the objective of this review that was to investigate test case design techniques for context-aware application testing. We also identified in this step that none of the previously published reviews were related to our objectives, as presented in Section 6 - Related Work.

The next step was *Develop the research protocol* whose goal is to guide the execution of a secondary study [22]. We highlight that our research protocol is an evolution of the first CAcTUS qSLR protocol [19] and, therefore, we used the same structure and population, for example. In the next subsections, the main parts of the research protocol used in this work are presented.



Pilot Searches
Identify the review goals
Develop the research protocol
Evaluate the research protocol
Execute the search in search engines
Read the abstracts
Assess the results
Update the research protocol
No
Yes
Is the research protocol approved?
Search for primary studies
No
Is the execution approved?
Yes
Select the primary studies
Extract data from primary studies
Assess quality of the primary studies
Analysis and report the results
Legend
Initial Node
Action
Activity Final Node
Decision Node
Fork Node
Join Node

Figure 1: Quasi-Systematic Literature Review steps.

After that, the task *Evaluate the research protocol* was performed by experienced researchers in secondary studies. We also conducted pilot searches aiming to improve the search strategies. Each pilot search as composed by the following tasks: *Execute the search in search engines*, *Read the abstract*, *Assess the results* and *Update the research protocol*. Thus, the pilot searched was an iterative process, performed 12 times, where we executed the search in a search engine (online database), read the abstract from the resulting papers and analyzed the results. Based on these analyses, we updated the research protocol by refining the search string or the selection criteria, for example. These pilot searches were conducted until we have tested all suggested keywords, the search string has returned the defined control papers[6] and the approval of the research protocol by all interested researchers.

Once the protocol was approved, we started its execution with the task *Search for primary studies*, where we applied the final search string in the selected online databases. If problems regarding the search engines were found, the searches were re-executed. Otherwise, we performed the task *Select the primary studies* using the inclusion and exclusion criteria. Then, from the final list of included papers, we simultaneously performed the steps *Extract data from primary studies* and *Assess quality of the primary studies*. In the first one, we extracted the data from the papers to answer our research questions. In the latter, we assessed quality of the papers, under the strict perspective of the research protocol and not regarding the general quality of the papers. The last step was *Analysis and report of the results* that are presented in Section 4 - Results.

[6]Papers selected to assess the quality of the search string.

3.1. Research Questions

Our goal was to investigate how context information influences the test case design and whether traditional approaches for test case design can be used in context-aware application testing. This research aims then to identify the different methods and techniques of test case design for context-aware applications. Thus, based on the concepts related to test design (*test case*, *test condition* and *test coverage item*) presented in Section 2, we defined the following research questions:

- **Research Question 1 (RQ1)**: *What are the existing methods or techniques of test case design for context-aware application testing?* This question is intended to identify the methods applied to design *test cases* for context-aware applications.

- **Research Question 2 (RQ2)**: *Which quality characteristics have been evaluated by the test cases designed for context-aware applications?* This question investigates which quality characteristics are addressed by the test design techniques found in our qSLR. A quality characteristic is a category of software quality attributes that bears on software quality [23]. Thus, with this question, we identified the focus of the *test conditions* used by the test techniques. To answer this question, we classified the identified approaches according to the quality characteristics defined in the ISO/ISEC 25010 [23].

- **Research Question 3 (RQ3)**: *What are the coverage criteria used in the context-aware application testing?* This question is intended to identify which testing coverage criteria are applied during the test design. The motivation for this question is the identification of the *test coverage items* used by the approaches. Therefore, this question allows us to verify whether these criteria are new (and specific to context-aware applications) or if they are the same criteria applied to the traditional applications [7] testing.

- **Research Question 4 (RQ4)**: *How does context influence the test case design in the context-aware application testing?* This question investigates how the context influences the test case design. The motivation for this question is the identification of what should be taken into account while deriving *test cases* for context-aware applications. The results of this question should highlight the difference between traditional application testing and context-aware application testing.

3.2. Sources, Search String and Control Papers

To organize and structure the search string, we used the PICO approach [24]. This approach separates the question into four parts: Population of interest, Intervention or exposure being evaluated, Comparison (if applicable) and Outcome. Because of the objective of this study (mainly characterization), we did not apply any comparison.

We tested several combinations of keywords until we had achieved a suitable search string. To assess the quality of the search string, we first defined three control papers, depicted in Table 1. The control paper **CP1** was already known by the researchers before this qSLR. The control papers **CP2** and **CP3** were defined based on the results of the pilot searches, before defining the final search string. However, after the quality assessment of them (see Section 4 - Results), the paper **CP2** did not remain as a control paper.

Table 2 presents the components of the search string, which had been applied to the Web of Science[8] and Scopus[9]. The use of these general indexing systems can be justified by their broad coverage [26] and stability. These search engines were used in others SLR [19, 27] and are recommended by the guidelines of Kitchenham and Brereton [8].

We did not use IEEE Xplorer[10], because it is indexed by Scopus [28] and it limits the command search to only 15 search terms [29], while our search string has 29 search terms. Furthermore, we discarded the ACM digital Library[11] since previous experience [20] of the researchers has shown that the ACM search algorithm does not favor the repeatability of the results.

Table 1: Initial set of control papers

Id	Title	Year	Reference
CP1	Automated Generation of Context-Aware Tests	2007	[12]
CP2	A model-based approach to test automation for context-aware mobile applications	2014	[5]
CP3	Generating Test Cases for Context-aware Applications Using Bigraphs	2014	[25]

Table 2: Search String

Population	Sensibility to the context
	"context aware" OR "event driven" OR "context driven" OR "context sensitivity" OR "context sensitive" OR "pervasive" OR "ubiquitous" OR "usability" OR "event based" OR "self adaptive" OR "self adapt"
Intervention	Test case design
	"test design" OR "test generation" OR "test case" OR "test suite" OR "test specification" OR "software testing specification" OR "system testing specification" OR "test data" OR "test procedure" OR "test input" OR "test set"
Comparison	-
Outcome	Methodology
	"approach" OR "strategy" OR "method" OR "methodology" OR "technique" OR "process" OR "algorithm" OR "tool"

3.3. Inclusion and Exclusion Criteria

The Inclusion Criteria (IC) define which primary studies are relevant for the research questions and must be included in the Systematic Review. On the other hand, the Exclusion Criteria (EC) define which studies must be excluded. These criteria are important to limit the scope of the review. Table 3 presents both the inclusion and exclusion criteria applied in this qSLR. They were defined based on the goal of this secondary study. We limit the searches for papers published after 2000 because the context definition more known by the researchers working with context aware applications is presented in a paper from Dey and Abowd [2] published in 2000.

[7]In this paper, "traditional application" means an application that is not context-aware.

[8]http://www.isiknowledge.com/

[9]http://www.scopus.com/

[10]http://ieeexplore.ieee.org/

[11]http://dl.acm.org/

Table 3: Inclusion and Exclusion criteria

Inclusion Criteria	Focus on the testing of context-aware software systems AND To talk about (test strategies OR test design OR test methods)
Exclusion Criteria	Not focus on the testing of context-aware software systems OR Being older than 2000 OR Not talk about (test strategies AND test design AND test methods)

3.4. Quality Assessment

We defined criteria to assess the quality of the selected papers by the interest of our research. Therefore, quality is only internally meaningfully and cannot be extrapolated to other studies. It aims to highlight those papers that may be more closely related to the investigation theme and to guide the interpretation of the findings. Thus, we did not exclude papers based on their quality score, but we considered it when discussing the empirically validated techniques in Section 5 - Discussion.

To guide the quality assessment, we defined a quality assessment form, where each quality criterion was evaluated through a question. One researcher performed this assessment and checked by another one. Besides that, in doubt cases, discussions were conducted with a third investigator.

As shown in Table 4, we used one criterion related to the systematic description of the test case design technique (Q1), one criterion assessing the information depicted in the study (Q2), two criteria related to test case generation (Q3 and Q4), one criterion associated with the proposal background theory or applicability (Q5), and six criteria related to the proposal evaluation (Q6, Q7, Q8, Q9, Q10 and Q11).

3.5. Study Selection and Data Extraction Process

The study selection process was performed in four steps by three researchers (two Ph.D. researchers and one Ph.D. student). First, the search string was used in the selected databases returning an initial set of papers. Next, duplicates were identified and removed. After that, we evaluated the papers according to the contents of their title and abstract using both the inclusion and exclusion criteria. Finally, the complete reading of the selected studies was performed, and they have been evaluated again according to the inclusion and exclusion criteria. On the final set of included papers were extracted the data to answer our research questions.

For each selected paper the information was extracted and managed using the JabRef reference tool[12] and Microsoft Access[13]. Each research question motivated some data extraction (see Table 5). We also collected information about the proposal evaluation (the last seven attributes in Table 5) and bibliographic information (e.g. year, source). Furthermore, to increase the results reliability, two researchers evaluated each study and any disagreement among them was resolved by discussing with a third researcher.

Table 4: Quality Assessment Form

Id	Question
Q1	Is there a systematic description of the test case design technique (e.g. using a formal language, a semi-formal language or procedures)? (1 pt)
Q2	Was it possible to extract all data of the data extraction form depicted in Table 5 ? (0.5 per data; max value: 9.0)
Q3	Is there any description about how the test cases have been derived? (1 pt)
Q4	Is there any description about restrictions or conditions about the applicability of the proposal? (1 pt)
Q5	Does the paper describe any adaptation/evolution of pre-existent approach? (1 pt)
Q6	Is there any description about measuring or evaluation of the test case design technique? (1 pt)
Q7	Does the article present different evaluation types of the proposal? (0.5 pt x number of evaluations types) Note: For instance of a Case Study research, complemented by another type of study (e.g. experiment) counts as 1 pt (0.5 + 0.5)
Q8	Does the article describe the application of the proposal in different settings (or applications)? (0.5 pt if less than 3 settings/applications; 1 pt if three or more settings/applications)
Q9	Is the goal of the evaluation cleared defined? (0.5 pt)
Q10	How many dependent variables are investigated? (0.5 per each dependent variable)
Q11	Are the hypothesis (null and alternative) explicitly described in the study? (1 pt)

4. Results

In the search process, we considered the studies published from January 2000 until December 2014. Figure 2 presents the study selection process. This qSLR started with 1087 primary studies returned from the following

[12]http://jabref.sourceforge.net/
[13]http://office.microsoft.com/en-001/access/

Table 5: Extraction form

RQ	Attribute	Description
RQ1	Test case design technique	Short description about the test technique
RQ1	Strategy type	Black-Box and/or White-box testing
RQ1	Test technique	Classification of the test design technique according to ISO/IEC/IEEE 29119 [11]
RQ1	Test type	Classification of the test type according to ISO/IEC/IEEE 29119 [11]
RQ1	Support tool	Support tool description
RQ2	Assessed quality characteristics	Description of the quality characteristics according to ISO/IEC 25010:2011 [23] assessed by the test case design technique
RQ3	Applied coverage criteria	Description of the applied test coverage criteria
RQ4	Challenges for the testing activity	Description of the challenges related to the context-aware application testing presented in the primary study
RQ4	Does the technique considers when the context changes occurs while testing the context-aware application (Yes/No)?	Discussion if the paper considers the context changes during the tests execution. Valid answers are YES or NO
-	Restrictions	Description of the restrictions or conditions about the applicability of the proposal
-	Context-aware test variables	Description of the context-aware variables that influences the tests
-	Type of experimental study	Types of empirical studies according to Kjeldskov and Graham [30]
-	Goal of the technique evaluation	Goal of the technique evaluation depicted in the primary study
-	Dependent variables	Variables investigated in the empirical evaluation
-	Assessed hypothesis	Hypothesis investigated in the empirical evaluation
-	Application type where the proposal has been applied	Application type according to Kotonya et al.s taxonomy [31]
-	Domain type where the proposal has been applied	Domain according to Kotonya et al.s taxonomy [31]
-	Software Systems category where the proposal has been applied	Software Category according to Pressman 2010 [32]

databases: Scopus (810) and Web of Science (277). After removing 254 duplicate studies and evaluating the title and abstract of the other papers, the result was a set of 49 potentially relevant studies. Finally, we evaluated the full paper resulting in 17 papers.

4.1. Overview of the included papers

Table 6 presents the included primary studies. As presented in this table, 23.53% (four papers) were published in 2014, and 47.06% (eight papers) were published in the last five years. In regards to the publication's source, two of the identified papers were published in journals and the others were published in international conferences. We did not identify a most prominent conference or workshop for context-aware application testing since most of the included papers were published in different sources. Only two of the included papers were published at the same conference, named "International Conference on Quality Software".

4.2. Classification of the primary sources

We also classified the papers with regards to the type of experimental study applied according to the classification proposed by Kjeldskov and Graham [30]. This classification was chosen because it is simple to implement and is based on the classification of computer-aided software engineering (CASE) research proposed by Wynekoop and Conger [46], which is an overall approach. Table 7 presents the results of this classification.

Ten of the included papers (58.82%) used a laboratory experiment and the other seven papers (41.18%) applied a basic research, where researchers develop new theories or study well-known problems to which neither specific solutions nor methods for accomplishing solutions are

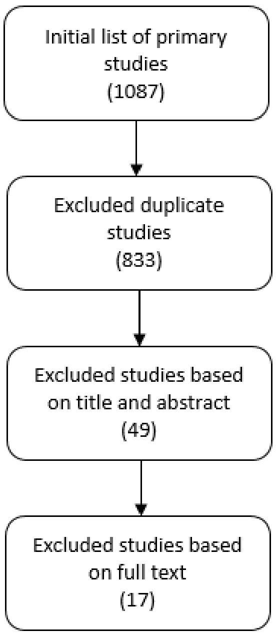

Initial list of primary studies
(1087)
Excluded duplicate studies
(833)
Excluded studies based on title and abstract
(49)
Excluded studies based on full text
(17)

Figure 2: Selection of the primary studies.

known [30]. In only two cases [5, 34] there is a disagreement between the authors' classification and our classification. In both cases, the authors classified the evaluation as "Case Study". However, according to Kjeldskov and Graham [30] a case study investigates a phenomenon within its real-life context, and this is not reported in those papers. Because of this, we classified the work of Wang et al. [34] as a laboratory experiment. In the case of the work of Griebe and Gruhn [5], they just discuss their proposal for an example, and then we classified it as basic research.

4.3. Quality assessment results of the included papers

Table 8 presents the quality assessment of the included papers and the rank of the papers according to the quality scores. In the cases of the same quality score (e.g., the papers from Chan et al. [43] and Amalfitano et al. [35]), the criterion to the rank was the year.

Considering the nonzero minimum score in all questions of the form depicted in Table 4, the minimum total score expected was 8.5. Based on this, only two papers did not reach the expected minimum score: the work of Tse et al. [45] and the work of Propp et al. [40]. However, we did not exclude these paper from this qSLR once we did not apply the quality assessment as an exclusion criterion (see Section 3 - Research Methodology and Protocol). A discussion considering only the papers with the best quality scores is presented in Section 5 - Discussion. In the following paragraphs, we discuss the quality assessment results.

In regards to the systematic description of the test design technique (Q1), only two papers [40, 41] do not present it. Furthermore, only two papers [33, 34] present all data from the extraction form depicted in Table 5 (Q2).

Table 6: Included papers ordered by year

Ref.	**Authors**	**Publication Source**	**Year**
[25]	Yu et al.	International Conference on Software Security and Reliability	2014
[5]	Griebe and Gruhn	ACM Symposium on Applied Computing	2014
[33]	Fredericks et al.	International Symposium on Software Engineering for Adaptive and Self-Managing Systems	2014
[34]	Wang et al.	ACM Transactions on Autonomous and Adaptive Systems	2014
[35]	Amalfitano et al.	International Conference on Software Testing, Verification and Validation Workshops	2013
[36]	Micskei et al.	International KES Conference on Agents and Multi-agent Systems, Technologies and Applications	2012
[4]	Püschel et al.	Modiquitous Workshop	2012
[37]	Wang et al.	International Conference on Quality Software	2010
[38]	Wang and Chan	International Conference on Automated Software Engineering	2009
[39]	Ye et al.	First Asia-Pacific Symposium on Internetware	2009
[40]	Propp et al.	Second Conference on Human-Centered Software Engineering	2008
[12]	Wang et al.	International Conference on Software Engineering	2007
[41]	Bo et al.	International Workshop on Automation of Software Test	2007
[42]	Lu et al.	International Symposium on Foundations of Software Engineering	2006
[43]	Chan et al.	International Journal of Software Engineering and Knowledge Engineering	2006
[44]	Chan et al.	International Conference on Quality Software	2005
[45]	Tse and Yau	International Computer Software and Applications Conference	2004

We identified that all the included papers present any description about how test cases can be derived (Q3). It

Table 7: Types of experimental study according to the classification proposed by Kjeldskov and Graham [30]

Ref.	Authors' Classification	According to Kjeldskov and Graham
[4]	Not classified	Basic Research
[5]	Case Study	Basic Research
[12]	Not classified	Laboratory Experiment
[25]	Experiment	Laboratory Experiment
[33]	Experiment	Laboratory Experiment
[34]	Case Study	Laboratory Experiment
[35]	Not classified	Laboratory Experiment
[36]	Not classified	Basic Research
[37]	Experiment	Laboratory Experiment
[38]	Experiment	Laboratory Experiment
[39]	Not classified	Basic Research
[40]	Not classified	Basic Research
[41]	Experiment	Laboratory Experiment
[42]	Experiment	Laboratory Experiment
[43]	Experiment	Laboratory Experiment
[44]	Not classified	Basic Research
[45]	Not classified	Basic Research

was expected because of the goal of this qSLR and the defined inclusion/exclusion criteria. We also identified that most of the papers describe some proposal limitation (Q4). Furthermore, we observed that only seven papers (41.18%) describe an adaptation or evolution of a pre-existing approach (Q5).

In regards to the proposal evaluation, six papers do not describe an assessment of the proposed test design technique (Q6). One paper [39] presents just a complexity evaluation. On the other hand, in regards to the applied evaluations types (Q7) the papers that present an evaluation use just one study type (e.g., a laboratory experiment - see Table 7) and only four studies [12, 34, 35, 41] perform it into more than three applications/settings (Q8). Ten of the papers (58.82%) describe the goal of the evaluation (Q9) and the measured dependent variables [47] (Q10). On the dependent variables, we highlight that most of the papers with empirical evaluation investigate only one or two variables. The study with more dependent variables was from Wang et al. [12], which assess six dependent variables in a laboratory experiment. Finally, only three papers (17.65%) describe the investigated hypothesis (Q11).

4.4. Description of the included papers

Before discussing the answer to each research question, we present below briefly each included paper focusing on the found test case design techniques.

Yu et al. [25] propose the use of Bigraphical Reactive Systems model to model the behavior of context-aware applications. Then, the bigraphical model-based testing uses the middleware and environment models as input to generate test cases to verify the interactions between the context-aware environment and the middleware together with domain services. To support the proposal, the authors also present two tools: i) BigM, which is a visual bigraphical modeling tool; and ii) BigSIM that is a bigraphical simulation tool. Since this approach uses as input a model of the states that the test item may occupy, we matched it with "State Transaction Testing" Test Design Technique according to the ISO/IEC/IEEE 29119-4. In regards to the test type, we classified their approach as "Interoperability Testing" according to the ISO/IEC/IEEE 29119-4, because of the focus of the model on the interactions between context-aware application and environment (e.g. sensors).

Griebe and Gruhn [5] propose a model-based approach to improving the context-aware mobile application testing. In their approach, first, the system models (i.e. UML Activity Diagrams) are enriched with context information. Then, these models are transformed into Petri Nets. From the Petri Net representation, a platform and technology-independent system testing model is generated that includes context information relevant for the test case execution. Tests are created to cover the all-transition-coverage criterion. Finally, platform and technology specific test cases are generated and can be executed using platform-specific automation technology. To execute the generated tests, the authors present a custom-built version of the calabash-android testing framework. We matched their approach as "Functional Testing" and "Scenario Testing" according to the ISO/IEC/IEEE 29119-4 because it uses UML Activity Diagrams modeling the system behavior and the scenarios of interactions between the test item and the user as input.

Fredericks et al. [33] introduce Veritas, an approach that uses evolutionary computation to adapt requirements-based test cases at runtime to handle different system and environmental conditions. Veritas monitors its environment for evidence of change and then automatically adapts individual test cases as necessary to ensure testing relevance while protecting against adapting test cases to pass under invalid conditions automatically. We matched their approach as "Functional Testing" according to the ISO/IEC/IEEE 29119-4 because its input is requirements-based test cases. We highlight that the goal of Veritas is to perform testing at runtime by selectively executing test cases and mutating test case parameters. We did not find a Test Design Technique according to the ISO/IEC/IEEE 29119-4 that could be matched with it. Furthermore, the paper does not describe details about the generation of the initial set of requirements-based test cases.

Wang et al. [34] define a metric named *context diversity* to capture the context changes in serial inputs and propose three strategies to study how context diversity may improve the effectiveness of the data-flow testing criteria. *Context diversity* measures the number of context changes inherent in a context stream. The authors also have implemented a support tool. We highlight that this study extends the studies [37] and [38]. We matched the testing approach presented in the paper as "Data Flow Testing"

Table 8: Quality Assessment

Ref.	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Q10**	**Q11**	**Total**	**Rank**
[4]	1.0	7.5	1.0	1.0	0	0	0	0	0	0	0	10.5	14º
[5]	1.0	7.5	1.0	1.0	0	0	0	0	0	0	0	10.5	13º
[12]	1.0	8.5	1.0	1.0	0	1.0	0.5	1.0	0.5	3.0	0	17.5	2º
[25]	1.0	8.5	1.0	1.0	0	1.0	0.5	0.5	0.5	1.5	0	15.5	4º
[33]	1.0	9.0	1.0	1.0	0	1.0	0.5	0.5	0.5	1.0	1.0	16.5	3º
[34]	1.0	9.0	1.0	1.0	1.0	1.0	0.5	1.0	0.5	2.5	1.0	19.5	1º
[35]	1.0	8.0	1.0	0	1.0	1.0	0.5	1.0	0.5	1.0	0	15.0	6º
[36]	1.0	7.5	1.0	1.0	1.0	0	0	0	0	0	0	11.5	11º
[37]	1.0	7.5	1.0	0	1.0	1.0	0.5	0.5	0.5	1.0	0	14.0	8º
[38]	1.0	7.5	1.0	0	0	1.0	0.5	0.5	0.5	1.5	0	13.5	9º
[39]	1.0	7.0	1.0	1.0	0	1.0	0	0	0	0	0	11.0	12º
[40]	0	6.0	1.0	0	0	0	0	0	0	0	0	7.0	17º
[41]	0	7.5	1.0	0	0	1.0	0.5	1.0	0.5	1.5	0	13.0	10º
[42]	1.0	8.5	1.0	0	1.0	1.0	0.5	0.5	0.5	0.5	1.0	15.5	5º
[43]	1.0	7.5	1.0	1.0	1.0	1.0	0.5	0.5	0.5	1.0	0	15.0	7º
[44]	1.0	6.5	1.0	1.0	1.0	0	0	0	0	0	0	10.5	15º
[45]	1.0	6.0	1.0	0	0	0	0	0	0	0	0	8.0	16º

since the authors apply context diversity to improve the effectiveness of data-flow testing criteria. According to the ISO/IEC/IEEE 29119-4 [11], in "Data Flow testing" a model of the test item that identifies control flow subpaths through the test item that link the definition and subsequent use(s) of individual variables shall be derived. Also, we classify the test type as "Functional Testing", because their approach could verify whether a function had been implemented according to its specified requirements.

Amalfitano et al. [35] present approaches based on the definition of reusable event patterns for the manual and automatic generation of test cases for mobile application testing. The test case design techniques presented by these authors which exploit the event patterns are: (i) Manual testing, where a tester manually uses event-patterns to define scenario-based test cases that include one or more instances of event-patterns; (ii) Mutation-based technique, where event-patterns are used to modify existing test cases, by applying mutation techniques that add event-pattern sequences inside already existing test cases; and (iii) Exploration-based technique, which establishes that event-patterns are used in automatic black-box testing processes based on dynamic analysis of the mobile application. Considering the characteristics of the approach of Amalfitano et al., we matched their approach with the "Error Guessing" Test Design Technique, which according to the ISO/IEC/IEEE 29119-4 [11] is when the aim is to derive a set of test cases covering likely errors, using the tester's knowledge and experience with previous test items. According to the ISO/IEC/IEEE 29119-4, the purpose of reliability testing is to evaluate the ability of the test item to perform its required functions, including evaluating the frequency which failures occur and when it is used under stated conditions for a specified period. Based on this definition, we matched the testing approach of Amalfitano et al. as "Reliability Testing".

Micskei et al. [36] present an approach that uses context modeling and graphical scenarios to capture the context and requirements of the system and automatically generates test data and test oracle. The testing is carried out in an interactive simulator environment and concentrated on the system-level behavior: detecting misbehavior on safety and robustness properties in diverse situations. As the requirements are expressed as scenarios, we matched this approach as "Scenario Testing" Test Design Technique according to the ISO/IEC/IEEE 29119-4. About the testing type, based on the test approach description we matched it as "Reliability Testing" according to the ISO/IEC/IEEE 29119-4.

Püschel et al. [4] present an approach applying Model-based Testing (MBT) and Dynamic Feature Petri Nets (DFPN) to define a test model from which an extensive test suite can be derived. A generator derives one test suite for each of the application's static configurations by combining the test model, context rules, and feature models. They also present an editor named Mobile Application Test Environment (MATE). This editor is based on Eclipse and provides a toolset for DFPN-based test models, the creation of context rules and the feature model. Once the test cases were created based on all possible traces (sequence of states), we matched the approach as "State Transition Testing" according to the ISO/IEC/IEEE 29119-4. About the testing type, we classified it as "Functional Testing", because the model defines the applications behavior.

Wang et al. [37] present an experiment on how much *context diversity*, proposed by Wang and Chan [38], for test cases, relates to fault-based mutants in context-aware software. Then, they investigate the trend on the number of mutants killed by individual test cases according to various *context diversity* values. Their approach was classified as "Functional Testing" and "Data Flow Testing" according to the ISO/IEC/IEEE 29119-4 due to the same reasons of the proposal of Wang et al. 2014 [34].

Wang and Chan [38] propose the metric named *context diversity* to measure how many changes occur in contextual values of individual test cases. They discuss how this notion can be incorporated into the test case generation process by pairing it with coverage-based test data selection criteria. They also propose to augment coverage-based testing techniques with the proposed metric. Their approach was classified as "Functional Testing" and "Data Flow Testing" according to the ISO/IEC/IEEE 29119-4 due to the same reasons of the proposal of Wang et al. 2014 [34].

Ye et al. [39] propose a testing approach to detect the possible behavior deviation of a context-aware application on different middleware. They present a formal model describing not only the behavior of context-aware applications but also the behavior of context-aware middleware under different implementation strategies. We matched this testing approach as "Error Guessing" technique, according to the ISO/IEC/IEEE 29119-4, because the tests are generated based on the implied scenarios, which are identified by deriving the application behavior on each middleware process with different strategies and deducing scenarios in which the different behaviors may occur. Also, we matched the approach as "Portability Testing", because of the focus on testing the replaceability of context-aware middleware.

Propp et al. [40] describe a method of model-based usability evaluation to evaluate interactive systems, with a particular focus on smart environments. Their objective is to provide usability evaluation methods for model-based smart environments, in all development stages. In their paper, a usability test case comprises a description of the test plan, the physical environment (e.g. needed device types) and involved user roles. The authors also present an engine that is used during design time to simulate a walkthrough on the task models, and during runtime, it acts as the logic within the smart environment providing assistance to the users. We matched their approach as "Usability Testing" as this is the focus of it, but we could not match it with one of the types of testing described in the ISO/IEC/IEEE 29119-4.

Wang et al. [12] propose an approach to improve the test suite of a context-aware application. This approach has the following steps: 1) It identifies key program points, represented by a control flow graph, where context information can effectively affect the application's behavior; 2) it generates potential variants for each existing test case exploring the execution of different context sequences based on the generated control flow graph; 3) it attempts to direct the application execution towards the generated context sequence dynamically. The outputs are Drivers (sequence of nodes that are be used to drive the test execution) and enhancement of an existing test suite for a context-aware application. The authors also present a supporting infrastructure. We matched this testing approach as "Functional Testing" because it allows the identification of whether a function had been implemented according to its specified requirements. Furthermore, we matched it as "Branch Testing" according to the ISO/IEC/IEEE 29119-4, because it traverses a control flow graph to generate a sequence of nodes that will be used to drive the test execution.

Bo et al. [41] introduce MobileTest, a tool supporting automated black box test for software on mobile devices. The tool adopts a sensitive-event based approach to simplify the design of test cases and enhance the test cases efficiency and reusability. This approach captures sensitive events (e.g. *inbox_full*, *incoming_call*, *call_end*, *inbox_empty*) from the target device in real-time and notify them to the test control layer. Based on the approach description, we matched it as "Compatibility testing" and "Performance-related testing." According to the ISO/IEC/IEEE 29119-4 [11], the goal of the "Performance-related testing" is to determine whether a test item performs as required when it is placed under various types and sizes of "load." The purpose of "Compatibility testing" is to determine whether the test item can function alongside other independent or dependent (but not necessarily communicating) products in a shared environment [11]. In regards to the test technique, we matched it as "Scenario Testing" because the tests are based on scenarios.

Lu et al. [42] define Context-aware Flow Graph (CaFG) to describe the control and data flow information in a context-aware middleware-centric application and also propose a family of context-aware data flow test adequacy measurement criteria. Thus, in this approach, the authors formalized the notion of context-aware data flow entities and new types of data flow associations to capture the context-aware dependencies. We matched the testing type of this approach as 'Functional Testing" since it can verify whether a function had been implemented according to its specified requirements. Besides, we matched this test case technique as "Data Flow Testing" according to the ISO/IEC/IEEE 29119-4 [11], because it derives a set of test cases that cover the paths between definitions and uses of context variables.

Chan et al. [43] propose an approach to integration testing of context-aware software. This paper proposes the use of source test cases to select follow-up test cases according to the metamorphic relations in question. To consider the changes in context that occur during the tests execution, they also propose the notion of checkpoints, at which the intermediate and final contexts can be used as test outcomes. Metamorphic relations can then be used to compare the test outcomes of multiple test cases to detect failures. We could not match this test design technique with the techniques described in the ISO/IEC/IEEE 29119-4, but about the testing type, we matched it as "Functional testing", because it could be used for the integration testing of the system functions.

Chan et al. [44] developed the notion of checkpoints to conduct integration testing, facilitating the checking of test results by the metamorphic testing approach. By treating checkpoints as the starting and ending points of a test case, they provide an environmental platform for the integration testing of the functions of a system. This setting offers an opportunity to test the functions in different parts of the application within the same environment. This approach is the same presented by Chan et al. [43] and, therefore, we classified it in the same way regarding the test technique and test type.

Finally, Tse et al. [45] propose to use isotropic properties of contexts as metamorphic relations for testing context-sensitive software. If a group of test cases and their corresponding outputs do not satisfy a specific metamorphic relation, then the program under test must contain a fault. Thus, we matched their approach with "Random Testing", where the aim is to derive a set of test cases covering the input parameters of a test item using values, which are chosen from this input domain at random [11]. Also, we classified the test type as "Functional Testing" because their approach can verify whether a function had been implemented according to its specified requirements.

4.5. Research Question 1 (RQ1): What are the existing methods or techniques of test case design for context-aware application testing?

Table 9 summarizes the found test design techniques. With our qSLR, we identified 12 different test case design techniques described in the 17 included papers. Five of these techniques are specification-based technique (Black-Box) and model-based. The models used to specify the context-aware application with testing purpose are: (i) Bigraphical Reaction System model from Yu et al. [25]; (ii) UML Activity Diagrams enriched with context information from Griebe and Gruhn [5]; (iii) extended UML 2 Sequence Diagrams from Mickei et al. [36]; (iv) Dynamic Feature Petris Nets Puschel et al. [4]; and (v) a formal model proposed by Ye et al. [39].

The other black-box techniques are: (i) test case design based on evolutionary computation from Fredericks et al. [33]; (ii) event-based test design from Amalfitano et al. [35]; and (iii) the proposal of using metamorphic relations in the test case design from Tse et al. [45].

From this group of black-box techniques, only two studies [35, 39] present an experience-based technique, in which the tester's knowledge and experience are used as the primary source of information during test case design [11]. Furthermore, most of the techniques in this group are related to "Functional Testing" using the testing technique "Scenario Testing," which utilizes a model of the sequences of interactions between the test item and other systems for the purpose of testing [11].

In the case of white-box approaches (structure-based techniques), we identified four test case design techniques:

Table 9: Classification of the testing approaches according to the ISO/IEC/IEEE 29119 [11]

Ref.	**Strategy**	**Test Type**	**Test Technique**
[4]	Black-Box	Functional Testing	State Transition Testing
[5]	Black-Box	Functional Testing	Scenario Testing
[12]	White-Box	Functional Testing	Branch Testing
[25]	Black-Box	Interoperability Testing	State Transaction Testing
[33]	Black-Box	Functional Testing	None
[34]	White-Box	Functional Testing	Data Flow Testing
[35]	Black-Box	Reliability Testing	Error Guessing
[36]	Black-Box	Reliability Testing	Scenario Testing
[37]	White-Box	Functional Testing	Data Flow Testing
[38]	White-Box	Functional Testing	Data Flow Testing
[39]	Black-Box	Portability Testing	Error Guessing
[40]	Black-Box	Usability Testing	None
[41]	Black-Box	Performance-related Testing and Compatibility testing	Scenario Testing
[42]	White-Box	Functional Testing	Data Flow Testing
[43]	White-Box	Functional Testing	None
[44]	White-Box	Functional Testing	None
[45]	Black-Box	Functional Testing	Random Testing

(i) the proposal of Wang et al. [34, 37, 38] of test design based on the measure context diversity; (ii) test design based on a control flow graph proposed by Wang et al. [12]; (iii) test case design technique with checkpoints and metamorphic relations presented by Chan et al. [43, 44]; and (iv) test case design technique based on Context-aware Flow Graph proposed by Lu et al.[42]. In this group, the test technique most used was "Data Flow Testing", where the aim is to derive a set of test cases covering the paths between definitions and uses of variables in a test item according to a chosen level of def-use coverage [11].

At last, only two of the included papers [40][41] do not answer this question since they did not describe systematically test case design technique. This could be observed in their quality assessment described in Table 8 (see **Q1** results). Also, as depicted in Table 9 some test case design techniques could not be classified according to the ISO/IEC/IEEE 29119-4 [11].

4.6. Research Question 2 (RQ2): Which quality characteristics have been evaluated by the test cases designed for context-aware applications?

Table 10 summarizes the quality characteristics, according to ISO/ISEC 25010 [23], evaluated by the test design techniques proposed in the included papers. This classification identifies the focus of the *test conditions* used to design test cases.

It is important to observe the classification presented in Table 10 is based on the researcher's interpretation, because none of the included papers describe explicitly the quality characteristics evaluated by the proposed test technique.

As presented in Table 10, most of the techniques are related to the characteristic *Functional Suitability* and the sub-characteristic *Functional Correctness.* It was expected since the assessment of these characteristics allow to verify whether a function had been implemented according to the specified requirements.

We also observed that the other quality characteristics are marginally addressed. The characteristics *Compatibility* and *Reliability*, for example, are treated by only two papers each. Furthermore, the characteristics *Portability*, *Performance efficiency* and *Usability* are addressed by only one test design technique each.

In regards to the test technique presenting tests cases to assess the *Usability* characteristic, we could not identify which of the usability sub-characteristics is the technique focus. It is because the paper from Propp et al. [40] does not present details enough about the created test cases.

We also highlight that some sub-characteristics are also marginally verified by the identified test techniques. The *Interoperability*, for example, is a sub-characteristic important for many context-aware software systems (e.g. smart environment), but was addressed by only one test case design technique [25].

Finally, we did not find any test case design technique related to the characteristics *Security* and *Maintainability*. Thus, it would be interesting to have further investigations to propose test techniques that assess these quality characteristics or/and other partially addressed by the current techniques.

Table 10: Quality characteristics evaluated by the test design techniques according to ISO/IEC 25010 [23]

Study	**Characteristic**	**Subcharacteristic**
[4]	Functional suitability	Functional correctness
[5]	Functional suitability	Functional correctness
[12]	Functional suitability	Functional correctness
[25]	Compatibility	Interoperability
[33]	Functional suitability	Functional correctness
[34]	Functional suitability	Functional correctness
[35]	Reliability	Recoverability, Fault tolerance
[36]	Reliability	Fault tolerance
[37]	Functional suitability	Functional correctness
[38]	Functional suitability	Functional correctness
[39]	Portability	Replaceability
[40]	Usability	None
[41]	Performance efficiency, Compatibility	Capacity, Co-existence
[42]	Functional suitability	Functional correctness
[43]	Functional suitability	Functional correctness
[44]	Functional suitability	Functional correctness
[45]	Functional suitability	Functional correctness

4.7. Research Question 3 (RQ3): What are the coverage criteria used in the context-aware application testing?

Table 11 presents the coverage criteria utilized in the included papers. We identify through this systematic review:

- Five papers [40, 41, 43, 44, 45] do not propose or use any testing coverage criteria. We highlight that it may be possible to integrate some coverage criteria with the test cases design techniques presented in these papers, but they do not describe any test coverage criteria;

- Five papers [5, 33, 34, 35, 37] use coverage criteria applied commonly to the traditional application testing. The code coverage and data-flow criteria, for example, are used by many techniques in the traditional application testing [48, 49];

- One paper [38] uses test adequacy criteria proposed in a previous paper [50] for testing context-aware applications. These criteria focus on the propagation of context variables in context-aware applications; and

- Six papers [4, 12, 25, 36, 39, 42] define coverage criteria according to the proposed testing approach, taking into account the context information. In this papers set, we highlight the work of Micskei et al. [36]

because it describes two general coverage criteria related to context information, which could be applied independently of the test technique.

Therefore, we could observe that traditional test coverage criteria are also applied in the context-aware application testing. It was expected since such criteria are often general for all kind of applications. However, we also identified new coverage criteria (e.g. Implied scenario coverage [39]) which consider the context information in the test coverage analysis.

Finally, the work of Wang et al. [34, 37, 38] proposes an additional criterion, named context diversity, which could be used to augment coverage-based testing techniques.

4.8. Research Question 4 (RQ4): How does context influence the test case design in context-aware application testing?

To understand how the context influences the test case design, we identified from the included papers: (i) the test design techniques addressing the context changes during the test execution; and (ii) the challenges related to the test case design for context-aware applications testing.

4.8.1. Changes in context and the test design

As context is dynamic, it is important to consider the changes in context that occur during the test execution. In this scenario, Chan et al. [44] classify the test cases into two categories: (i) test cases considering context changes (*Context-coupled test cases*); and (ii) test cases not considering context changes (*Context-decoupled test cases*). We then decided to use this classification because it classifies the test cases according to the way in which they deal with the context.

However, we realized that an additional category (Partial context coupled test cases) was needed to characterize those test cases that offered some limitations when considering context. In the following we describe the categories used and the classification of the identified test design techniques:

- *Context-decoupled test cases*: if the context remains static during the execution of the test cases or changes in the context do not activate any action [44]. In this category, we classified the following studies: [4], [25], [33], [34], [37], [38], [40], [41] and [45];

- *Partially context-coupled test cases*: test cases which allow the change of just some kinds of contexts or the combination of test context of different requirements during the tests execution. In this category, we classified the papers: [5], [12] and [36].

- *Context-coupled test cases*: in which sequences of context tuples, instead of isolated context tuples, are used. In other words, test cases allow the changes of context during the tests execution [44]. The following studies were classified in this category: [35], [39], [42], [43] and [44].

As the context continuously changes, the test design techniques that generate context-coupled test cases and partially context-coupled test cases are more interesting for the context-aware application testing. Therefore, more details about how the identified techniques deal with the context changes are described in the following paragraphs.

In the partially context-coupled test cases group, we identified three techniques [5, 12, 36]. The limitation of these approaches is that they limit the context changes. Griebe and Gruhn [5] assess their test approach creating test cases to verify a "call-a-cab" application in regards to the specification conforming reaction to the unexpected loss of network connection. Despite considering context change in all test scenarios, they do not address the changes of other context parameters during the tests execution.

Micskei et al. [36] handle test context by combining the initial context fragments of different requirements. Wang et al. [12] deal with the context changes during the tests by proposing a technique that generates tests to cover switches between context handlers. To do this, they identify key program points where context information can effectively affect the application's behavior and generates possible variants for each existing test case exploring the execution of different context sequences. Therefore, both the techniques of Micskei et al. [36] and Wang et al. [12] consider the context changes during the test execution, but these changes are only deemed to test the interaction among different functions of the application.

In the context-coupled test cases group we identified five papers [35, 39, 42, 43, 44]. Amalfitano et al. [35] propose the use of reusable event patterns for the manual and automatic generation of test cases for mobile application. In that way, their approach can consider the context changes during the tests execution because an event-pattern may define context changes or it may be included in other event sequences. For instance, the authors present the pattern MSVC that means "Magnetic Sensor value changed after any other event except the event itself". The main limitation of this approach is that their success is directly related to the tester experience since the event-pattern are specified by analyzing previous bug reports.

Ye et al. [39] investigate the impact of the middleware in the context-aware application behavior. They present different middleware implementation strategies (e.g., Allow Context Change vs. Keep Contexts during invocation of context handler), the effect of this (e.g. Missing contexts, Obsolete contexts, and Out-of-order contexts) and how to test the behavior deviation of a context-aware application running on different middlewares. Despite considering context changes in the tests execution, their focus is to check whether a context-aware application behaves consistently on two context-aware middleware before moving it from one to another.

Lu et al. [42] defined a family of testing criteria to measure the coverage of test sets of context-aware middleware-centric applications. Such criteria are based on a Context-

Table 11: Coverage criteria used by the test design techniques

Study	Coverage Criteria
[4]	They propose the test generation through a complete simulation of the combined Dynamic Feature Petri Nets (DFPN) consisting of the test model and the context branch to derive all possible traces, each of which corresponding to a specific test case
[5]	The authors use the all-transition-coverage criterion
[12]	The authors propose three coverage criteria (Context-Adequacy, Switch-To-Context-Adequacy, Switch-With-Preempted-Capp-Adequacy) to guide the test improvement to expose all types of contexts under execution and exposes the interactions between context handlers based on a control flow graph
[25]	They define a new coverage strategy similar to data-flow testing strategy, but defined on reaction rules regarding subbigraphs instead of variables in data flow approaches
[33]	Full coverage of a requirements specification
[34]	They use six data-flow testing criteria: All-Defs (AD), all-C-Uses (CU), all-P-Uses (PU), all-P-Uses/some-C-Uses (PUCU), all-C-uses/some-P-Uses (CUPU), All-Uses(AU). Frankl and Weyuker proposed these criteria [51]
[35]	The authors measure the resulting code coverage regarding lines of code (LOCs) and methods
[36]	Context related coverage metrics measure what part of the context model has been covered during testing (e.g., whether there are objects, which have not been present in any test runs) and what combinations of initial context fragments from different requirements were covered. Scenario related metrics measuring coverage on the scenarios, e.g., whether all scenarios have been triggered. Robustness related metrics measuring the thoroughness of the generation of extreme contexts by considering the coverage of violated constraints and potential boundary values from the context model
[37]	They applied data-flow testing criteria
[38]	They use three testing criteria All-Services (AS), All-Services-Uses (ASU), and All-2-Services-Uses (A2SU). Lu et al. proposed these criteria. [50]
[39]	They propose a new criterion (Implied scenario coverage) to generate and select test cases for checking the replaceability of middleware for a given context-aware application
[40]	None
[41]	None
[42]	The authors define three test coverage criteria (all-situations, all-def-situ-associations and all-pairwise-du-associations) for data flow testing of context-aware middleware-centric applications
[43]	None
[44]	None
[45]	None

aware Flow Graph (CaFG) that considers the update of a context variable, from environmental context acquisition, before its usage occurrence. The main limitation of this approach are: (i) the construction of the CaFG since the paper does not present a tool that automatically generates it; and (ii) the approach does not address the expected output definition. To assess the CaFG, the authors compared the execution results of faulty versions and the golden version.

Finally, Chan et al. [43, 44] developed the notion of checkpoints (circumstances where none of the situation expressions are relevant to the adaptive functions under test) to conduct integration testing, facilitating the checking of test outcomes by a metamorphic testing approach and allowing for changes of contexts during test case execution. The main limitation of this approach is that it depends on the definition of metamorphic relations and it can not be applied with an application that has not checkpoints.

4.8.2. Challenges to the test design for context-aware software testing

Aiming to identify the challenges affecting the test design, we first extracted from the included papers the challenges of testing context-aware applications. Next, we classified them into two categories according to the source of the challenge: (i) Middleware, which involves the test challenges related to the use of a context-aware middleware; and (ii) Context, which involves the test challenges resulting from the characteristics of a context information. Table 12 presents the results of this classification.

These challenges impact different activities of the testing process and can complement the challenges list presented in the previous CAcTUS paper [6]. We then an-

Table 12: Challenges to test context-aware applications

Category	Challenge	Description	Study
Middleware	Correctness of the application specification	There is no oracle to validate the correctness of the functions defined in a specification with the Situation-Aware Interface Denition Language (SAIDL)	[45]
	Combinatory explosion	There are many challenges in testing the prohibitive number of possible relationships between the application and the middleware	[45]
	Replaceability of context-aware middleware	The strategies in the implementation of middlewares affects the testing and they may not be specified explicitly in the middleware interface specification	[39]
	Hidden failures	Failure that occurs at a certain instant may be hidden again at the conclusion of a test case execution	[43][44][45]
	Identification of the termination of the test cases	As the middleware remains active and situations may continue to evolve, the termination of some test cases may not be easily identified	[43] [44]
	Component interaction	Complex interactions among the components (middleware, services, sensors, etc.) affect the testing activity	[12] [25]
	Unit Testing	Suppose the middleware was not included in the unit testing. In this case, testers should modify the application behavior	[45]
	Faulty propagation	The explicit interactions between application and middleware distribute the application logic over multiple tiers, and hence the faulty states of a program execution may be propagated among multiple architectural levels	[37]
	Feature Interaction	Applications sharing the same context-aware middleware may interact implicitly through their own features	[37]
Context	Continuously changes	The environment may change continuously (Contextual data as continuously changing)	[12][25] [42][45]
	State dependent	The current state of a device may depend on the state of other devices	[25]
	Complexity	The contextual information may be a complex data	[25][36]
	Combinatory explosion	The tests should consider various system's context (changing network availability, relocation, change in availability of sensor data)	[5][25] [35][36]
	Changes can occur at any time	Changes in the context parameters may occurs at any time and are usually unpredictable	[4][5] [12]
	Asynchronous	Contextual events must be handled asynchronously, and such handling must address the possibility of multiple interfering contextual changes	[12]
	Uncertain situations	Contextual data is uncertain. Uncertainties can include unexpected power drain and unexpected obstacles	[12] [33]
	Contextual data is temporal and spatial	Contextual data is temporal, spatial and these are some of the unique difficulties faced by testers of context-aware applications	[12]
	Imperfection of the context data	There are diverse kinds of noise in the captured contexts and significant errors in recovering actual situations based on given contexts	[12] [34] [38]
	Concurrency	Program-derived testing models must consider contextual data as well as the concurrency issues introduced by pervasive multi-threaded context handlers	[12]
	More control of the application	If it is necessary to feed continuously data to a context-aware application, then we must be able to exercise more control on the executing application	[12]

alyzed which test design activity they impact and identified the following challenges related to the test design for context-aware application testing:

- **Challenge 01: To define the test context**. In a traditional application a test condition is a testable aspect of a test item (e.g. a function). In a context-aware application, once the application behavior is affected by the context changes, it is important to define the context in which the test condition will be tested. Such context could be called *test context*. The challenge is how to specify the *test context* needed to the test case execution, considering the complexity and imperfection of the context data.

In this qSLR, we found 14 paper presenting a solution for this challenge. Yu et al. [25] address this issue modeling the application and the environment with a Bigraphical Reaction System. Other models used by the techniques identified to describe the context and the context-aware application behavior are: extended Activity Diagrams [5], Dynamic Feature Petri Nets [4], extended UML Sequence Diagrams [36], and state model [39]. Furthermore, Wang et al. [34, 37, 38] describe the *test context* through the "context diversity" measure together with the data-flow testing criteria. Amalfitano et al. [35] specify *test context* with event patterns. In the work of Wang et al. [12] the *test context* is defined by identifying context-aware program points where context changes may affect the application's behavior. Lu et al.[42], in turn, define the *test context* according to Context-aware Flow Graph. Finally, some proposals defines the *test context* based on metamorphic relations [43, 44, 45].

- **Challenge 02: To use the context information as a test coverage item**. In context-aware application testing, it is reasonable to think on using the context information itself as a coverage criterion. However, to define which context information should be considered during the tests is a challenging task because of the combinatorial explosion of this information.

  Addressing this challenge we found the context diversity metric [34, 37, 38]. Context diversity measures the number of context changes inherent in a context stream. For example, the context diversity for the sequences "meeting room, present report" and "home, watch football" is two, after summing up the context changes in the dimensions of location (where "meeting room" is different from "home") and activity (where "present report" is different from "watch football"). Based on this measure, Wang et al. [34] show how to select test cases with higher, lower, and more evenly distributed context diversity in constructing test suites that are adequate on the data flow testing criteria.

  The context related coverage criteria discussed by both Mickskei et al. [36] and Wang et al. [12] also address the **Challenge 02**. Mickskei et al. [36] describe as context related criteria: (i) what part of the context model has been covered during testing; and (ii) what combination of initial context fragments from different requirements were covered. Wang et al. [12] propose three criteria (Context-Adequacy, Switch-to-Context-Adequacy, Switch-With-Preempted-Capp-Adequacy) to guide the test improvement to expose all types of contexts under execution.

- **Challenge 03: To design test cases for dealing with uncertainty** Contextual data is uncertain. Examples of uncertainties are unexpected power drain and physical damages into the sensors. Therefore, it is interesting to specify test cases considering uncertainties.

  Fredericks et al. present one solution to deal with this challenge. [33] that use evolutionary computation to adapt test cases at runtime to handle different system and environment conditions. To do this, the test engineer should previously specify a range of values with which their test approach can adapt test case parameters. With this approach, it is possible to reduce the amount of test case failures due to the mismatch between test cases and operational context caused by uncertainty within the system and environment. Fredericks et al. show, for example, how their approach can deal with damage in a camera that reduces its ability to sense objects. However, this test design technique is not intended to add or remove test cases during execution dynamically. It just adapts pre-existing test cases.

- **Challenge 04: To specify when the expected output should be assessed**. In traditional applications, the output is assessed against the expected output after the application has processed the input data. In a context-aware application, as the context may change at any time, the challenge is to determine when the test driver should compare the application output with the expected output.

  Chan et al. [43, 44] proposed the concept of checkpoints as circumstances where the context-aware middleware will not trigger any adaptive function. Based on this idea, their approach generates test cases that start at a checkpoint and end at another. Therefore, this test design technique defines the end of a test case, addressing this challenge.

- **Challenge 05: To reflect the context-awareness in the test cases**. The context changes continuously and at any time. Thus, before the end of a test case, the context could be already changed and with it the application behavior too. In this scenario, the expected output of the test case could be outdated. Then, this challenge is about to design test cases allowing changes in the expected output according to the current context.

  Although we have found some test design techniques that allow for context changes during the test execution, none of them deal with changes in the test expected output. Thus, we did not find a test design technique with a context-aware oracle that assesses the application at any time, based on the current context.

5. Discussion

Through this qSLR, it was possible to identify 17 relevant papers to answer our research questions related to the test design for context-aware application testing. Based on the data extracted from these papers we then gathered some evidence about this topic.

In the following subsection, we discuss the experimentally validated techniques identified in this study. Next, we present some opportunities for future research and the threats to the validity of the results.

5.1. Experimentally validated techniques

Although we had identified 17 articles related to testing design, not all had reported systematic procedures for this activity. As depicted in Table 8 the papers of Propp et al. [40] and Bo et al. [41] have not scored in the question **Q1** of the quality assessment. This question was related to the systematic description of the test case design technique. In addition, more than one paper address the test design technique with checkpoints [43, 44] and the test design based on the *context diversity* measure [34, 37, 38]. Therefore, in this qSLR, we identified 12 different test techniques described systematically. From these, only seven test design techniques are presented with some empirical evidence through experiments or case studies. This could be observed during the quality assessment (see Table 8) and the classification of the experimental study (see Table 7).

It is worth noting that experimentally validated solutions present more confidence to practitioners. We then focus our discussion here on these seven test case design technique presented in the following: (i) test design based on Bigraphical Reaction System model from Yu et al. [25]; (ii) test design based on evolutionary computation from Fredericks et al. [33]; (iii) the proposal of Wang et al. [34, 37, 38] of test design based on the *context diversity* measure; (iv) event-based test design from Amalfitano et al. [35]; (v) test design based on a control flow graph proposed by Wang et al. [12]; the context-aware data flow testing proposed by Lu et al. [42]; and (vii) test design technique with checkpoints and metamorphic relations presented by Chan et al. [43, 44].

From these studies, only the oldest work from Chan et al. [44] did not present a laboratory experiment. It was classified as a basic research (see Table 7). It is the reason why this paper has not a good position in our quality rank. On the other hand, the another nine studies had achieved the highest scores in our quality assessment (see Table 8).

Considering only the ten papers above, we have found both white-box and black-box approaches, but we "missed" test techniques related to the portability testing, usability testing, and performance-related testing. The reason is that the test design techniques identified that deal with these test types (see Table 9) are not empirically assessed. As a consequence, the empirically validated techniques address only three (*Compatibility*, *Functional suitability* and *Reliability*) of the eight quality characteristics presented in the ISO 25010 [23].

In regards to the test design technique, we could not classify two of the seven test techniques according to the ISO/IEEE 29119-4 [11]. These techniques were: (i) test case design technique presented by Chan et al. [43, 44]; and (ii) test design technique proposed by Fredericks et al. [33]. It corroborates the evidence gathered in our previous work [6] that this international standard could not be broad enough to classify any form of software testing.

We also highlight that only the work of Chan et al. [43, 44] did not present a support tool. Yu et al. [25] present two tools (BigM and BigSIM) but did not reference them. Amalfitano et al. [35] uses the Android Ripper tool that was previously developed by them. Wang et al. [34, 37, 38] use an open-source tool to instrument the program under test and monitored the execution path of each test case. Fredericks et al. [33] presents the Veritas framework that implements their approach. Lu et al.[42] present a prototype tool that automates the test set generation based on their test criteria. Wang et al. [12] also use and reference two tools used to support their approach.

Also, four of the seven test design techniques consider the context changes during the test execution. The proposal of Wang et al. [12] generates partially context-coupled test cases to cover switches between context handlers. The test design techniques presented by Chan et al. [43, 44], Lu et al. [42] and Amalfitano et al. [35] generate context-coupled test cases. The first one depends on the definition of metamorphic relations. The second technique is based on Context-aware Flow Graphs. The third one depends on the definition of event pattern based on known bugs. Therefore, we could observe a lack of test case design techniques empirically assessed that consider context changes at the test execution.

Finally, we have found experimentally validated solutions to four of the five challenges related to testing case design for the context-aware application (see Section 4 - Results). Therefore, the challenge "To reflect the context-awareness in the test cases" has not been addressed. It is important to highlight that even considering all the 17 included papers, none of them address this challenge, making it an attractive opportunity for future research.

5.2. Opportunities for research

In accordance with the results of this review, we propose a few research topics for further research. The suggestions are:

- Controlled experiments comparing the results of applying traditional testing techniques and the techniques identified in this qSLR;

- Proposal of new testing approaches focusing on quality characteristics not take into account by the current approaches, but which are also relevant to the context-aware application. For example, no work found to deal with security issues or the efficiency in regards to the resource utilization;

- Proposal of new testing approaches in order to mitigate the identified challenges;

- Proposal of new testing approaches which generate context-coupled test cases. As presented in Section 4 - Results, we found some approaches that consider context changes during the tests execution, but they are not flexible to attend the different needs. Most of them require source-code [12, 43, 44]. The other are limited to predefined events [35], changes in the network connection [5] or behavior deviation due to the context-aware middleware used [39]; and

- Use of search-based approaches in context-aware application testing. In spite of the explosion of scenarios provided by the context information, only one study [33] from the included papers uses a search technique to improve the test approach.

5.3. Threats to validity

In this section we discuss the following threats to the validity of the results of this systematic review as well as how we mitigated them:

- The use of the part 4 of the ISO/IEC/IEEE 29119 [11] international standard that is in the draft status. This is a threat because the draft status means that the Test Design Techniques classification might not be stable yet. Other taxonomy could be used, which might affect the results. However, we believe that using ISO 29119 could be interesting because this represents the newest standard in the area. It was also used in the previous CAcTUS qSLR [6, 19];

- Selection Bias. This threat was mitigated by a rigorous research protocol, several tests with the search string, the use of control papers to assess the quality of the searches and discussion between the researchers at all the review steps;

- Inaccuracy of data extraction and bias on the synthesis of information. Both are threats since we could have misunderstood the intention of the authors whose work was studied. To mitigate these threats, we followed an extraction form defined in the protocol and had a discussion among the researchers about the data extracted. Furthermore, other classifications could be applied, but we believe that the data extracted was adequate to answer our research questions;

- Missing relevant studies for this systematic review. The main reasons for this threat are: i) the publisher sources of the study are not indexed by the databases used in this systematic review; ii) the search string did not hit the study, and; iii) the study is in a language other than English. To mitigate this threat, we used relevant electronic databases in Software Engineering, which are frequently used in many systematic reviews, and we conducted several trials until to construct the final version of the search string. Thus, we believe that this qSLR has a good coverage of state of the art, although, searches in other engines (e.g., ACM Digital Library) could identify papers not indexed by the used databases and, therefore, that were not analyzed in this review. Finally, the choice of the English language is justified to make this review study replicable and feasible.

6. Related Work

To the best of our knowledge, as a secondary study about testing in the scenario of context-aware applications there is only one previous work [6] from researchers of the CAcTUS project. In that paper, Matalonga et al. [6] discuss the initial results of our ongoing efforts to understand the testing of context-aware software, evaluating whether the observed techniques for testing context-aware software can be matched with the ISO/IEC/IEEE 29119 categories.

However, we found some non-systematic reviews [52, 53] and systematic reviews about software testing [27, 54, 55], as well as some surveys [56, 57, 58] and secondary studies about context-aware applications [59].

Bezerra et al. [52] identify the main challenges to perform usability testing in ubiquitous systems based on a non-systematic review of the literature studies and on their experience in testing this kind of systems. The authors also describe their ongoing work where they investigate usability testing to be designed considering context-awareness and to be supported by specific measurements to the ubiquitous computing scenario.

McMinn [53] reviews some common search algorithms, and some of the classic testing problems to which the search-based approach has been applied. Based on the analyses of the past work, the author discusses potential future research areas and open problems that remain in the Search-Based Software Testing.

Harman et al. [57] presents a survey about Search-Based Software Engineering (SBSE), identifying research trends and relationships between the techniques applied and the applications to which they have been implemented. In such survey, the authors highlight the overall trends in SBSE including Search-Based Testing. Furthermore, in other paper, Harman et al. [58] focus in the Search-Based Software Testing (SBST), presenting open problems and challenges of testing non-functional properties, multi-objective testing, and SBST for test strategy identification.

Porras et al. [27] report on a systematic mapping and review about automated testing approaches for mobile applications. The authors also investigate the major challenges in automated testing of mobile applications. According to the work of Porras et al. [27] the testing technique most used were the GUI-based models of the application. One conclusion is the need for further research to improve the creation of effective and efficient models for automated testing of mobile applications.

Qiu et al. [54] present a systematic mapping study about regression testing of Web Service. They present a qualitative analysis of the findings, including stakeholders, challenges, standards, techniques, and validations employed in the primary studies. They also identify gaps in current research work and reveals new insights for the future work.

Kanewala [55] present a systematic review about testing scientific software. This study aims to identify specific challenges, proposed solutions, and unsolved problems faced when testing scientific software. Also, they describe unsolved challenges and how software engineering researchers and practitioners can help to overcome them.

Perera et al. [56] examine a variety of popular and innovative Internet Of Things (IoT) solutions regarding context-aware technology perspectives. The survey of Perera et al. [56] is intended to serve as a guideline and a conceptual framework for context-aware product development and research in the IoT paradigm. These authors also discuss major lessons learned and opportunities for future research and development in the context-aware computing domain.

Champiri et al. [59] conducted a systematic review to identify the contextual information and methods used for making recommendations in digital libraries as well as the way researchers understood and utilized relevant contextual information. Based on the findings of this review, the authors highlight the need for more investigations on the concept of context from user viewpoint in scholarly domains.

The increasing presence of context-awareness in different types of software systems (such as recommendation systems, ubiquitous applications, and IoT applications) is evidence that this characteristic is important for the current and future software. Thus, the context-aware application testing is essential to guarantee software quality in many domains. In this scenario, a secondary study is interesting to provide an overview of the test design techniques applied in context-aware applications, to identify solutions, challenges and future directions in this topic.

We have found some systematic reviews about testing in other domains [27, 54, 55, 59] and Search Based Testing [53, 57, 58], but only one paper reporting the initial results of a secondary study about testing context-aware application [6]. In this latter, the focus was classifying the test techniques applied to verify whether the conventional testing techniques are useful to test context-aware software. Thus, our review and this previous one [6] are in the same domain (context-aware application testing). However, as they had different goals (and as a consequence, various search strings), the final set of papers included in these reviews are different.

Therefore, the contribution of the qSLR described in this paper is to present an overview of test case design for context-aware applications, highlighting the test case design techniques, the challenges for testing context-aware applications, the coverage criteria used, the quality characteristics evaluated by the identified techniques and gaps for future research.

7. Conclusion and Future Work

Context-aware applications use context information to guide their behavior. In this scenario, the context affects the testing activity since the application's behavior can change continuously as the context changes. We then investigated how context affects the testing of a context-aware application with a quasi Systematic Review answering the following questions: *(RQ1) What are the existing methods or techniques of test case design for context-aware application testing?; (RQ2) Which quality characteristics have been evaluated by the test cases designed for context-aware applications?; (RQ3) What are the coverage criteria used in the context-aware application testing?; and (RQ4) How does context influence the test case design in the context-aware application testing?*

Using this review, it was possible to identify 17 papers presenting different approaches to testing a context-aware application. Regarding the quality characteristics addressed, most of the approaches found are black-box and deal with the sub-characteristic "Functional correctness" of the quality characteristic "Functional suitability." We also could observe that traditional test coverage criteria are also applied in the context-aware application testing.

Although we found papers dealing with context changes during the test execution, further investigation is needed to assess the power and flexibility of these approaches in practice. Furthermore, we identified 20 challenges related to the context-aware application testing, which highlights the differences between the traditional application testing and the context-aware application testing. We also described five challenges related to the test case design for context-aware application and discussed the experimentally validated techniques. Finally, based on the results we presented some interesting topics for future research.

Finally, as a general result of this study we can say that much work remains to be done to make a test case design for context-aware applications a practical and effective reality. In this sense, we note that new research have been published after our study (e.g. the proposal of context simulator of Vieira et al. [60]) and we have launched ourselves into new ways of investigations. Indeed, we are currently working on new test criteria and a test case design technique focused in the generation of context-aware test cases to deal with the gaps identified. Besides that, we intend to investigate in an industrial case study the impact of the use of truly context-aware tests in the quality of the context-aware applications testing. Finally, other future work is a tertiary study on the results of the three systematic reviews conducted in the CAcTUS project to gather the acquired knowledge during this research project.

Acknowledgments

This work is a partial result of the "CAcTUS - Context-Awareness Testing for Ubiquitous Systems", project supported by CNPq (MCT/CNPq 14/2013 - Universal) under grant number 484380/2013-3.

References

[1] R. W. Horvth, Ivand Vroom, Ubiquitous computer aided design: A broken promise or a sleeping beauty?, Journal of Computer-Aided Design 59 (2015) 161–175.

[2] A. K. Dey, G. D. Abowd, Towards a better understanding of context and context awareness, in: Proceedings of the Workshop on the What, Who, Where, When and How of Context-Awareness, affiliated with the CHI 2000 Conference on Human Factors in Computer Systems, ACM, ACM Press, 2000.

[3] Software and Systems Engineering Software Testing Part 1: Concepts and Definitions, ISO/IEC/IEEE 29119-1:2013(E) (2013) 1–64doi:10.1109/IEEESTD.2013.6588537.

[4] G. Püschel, R. Seiger, T. Schlegel, Test modeling for context-aware ubiquitous applications with feature petri nets, in: Modiquitous Workshop, 2012.

[5] T. Griebe, V. Gruhn, A model-based approach to test automation for context-aware mobile applications, in: Proceedings of the 29th Annual ACM Symposium on Applied Computing, SAC '14, ACM, New York, NY, USA, 2014, pp. 420–427. doi:10.1145/2554850.2554942.
URL http://doi.acm.org/10.1145/2554850.2554942

[6] S. Matalonga, F. Rodrigues, G. H. Travassos, Matching context aware software testing design techniques to iso/iec/ieee 29119, in: T. Rout, R. V. OConnor, A. Dorling (Eds.), Software Process Improvement and Capability Determination, Vol. 526 of Communications in Computer and Information Science, Springer International Publishing, 2015, pp. 33–44.

[7] G. Travassos, P. dos Santos, P. Neto, J. Biolchini, An environment to support large scale experimentation in software engineering, in: Engineering of Complex Computer Systems, 2008. ICECCS 2008. 13th IEEE International Conference on, 2008, pp. 193–202. doi:10.1109/ICECCS.2008.30.

[8] B. Kitchenham, P. Brereton, A systematic review of systematic review process research in software engineering, Inf. Softw. Technol. 55 (12) (2013) 2049–2075.

[9] G. J. Myers, C. Sandler, T. Badgett, The Art of Software Testing, 3rd Edition, Wiley Publishing, 2011.

[10] I. C. Society, P. Bourque, R. E. Fairley, Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0, 3rd Edition, IEEE Computer Society Press, Los Alamitos, CA, USA, 2014.

[11] IEEE Draft International Standard for Software and Systems Engineering–Software Testing–Part 4: Test Techniques, ISO/IEC/IEEE P29119-4-FDIS April 2015 (2015) 1–147.

[12] Z. Wang, S. Elbaum, D. S. Rosenblum, Automated generation of context-aware tests, in: Proceedings of the 29th International Conference on Software Engineering, ICSE '07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 406–415. doi:10.1109/ICSE.2007.18.

[13] R. M. Santos, K. M. Oliveira, R. M. C. Andrade, I. S. Santos, E. R. Lima, Human Computer Interaction: 6th Latin American Conference, CLIHC 2013, Carrillo, Costa Rica, December 2-6, 2013, Proceedings, Springer International Publishing, Cham, 2013, Ch. A Quality Model for Human-Computer Interaction Evaluation in Ubiquitous Systems, pp. 63–70.

[14] R. M. Carvalho, R. M. C. Andrade, K. M. Oliveira, Distributed, Ambient, and Pervasive Interactions: Third International Conference, DAPI 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015, Proceedings, Springer International Publishing, Cham, 2015, Ch. Using the GQM Method to Evaluate Calmness in Ubiquitous Applications, pp. 13–24.

[15] R. M. Carvalho, R. M. C. Andrade, K. M. Oliveira, I. S. Santos, C. I. M. Bezerra, Quality characteristics and measures for human–computer interaction evaluation in ubiquitous systems, Software Quality Journal (2016) 1–53.

[16] L. S. Rocha, J. B. Ferreira Filho, F. F. P. Lima, M. E. F. Maia, W. Viana, M. F. Castro, R. M. C. Andrade, Ubiquitous software engineering: Achievements, challenges and beyond, in: Proceedings of the 2011 25th Brazilian Symposium on Software Engineering, SBES '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 132–137. doi:10.1109/SBES.2011.33.
URL http://dx.doi.org/10.1109/SBES.2011.33

[17] R. O. Spínola, G. H. Travassos, Towards a framework to characterize ubiquitous software projects, Inf. Softw. Technol. 54 (7) (2012) 759–785. doi:10.1016/j.infsof.2012.01.009.
URL http://dx.doi.org/10.1016/j.infsof.2012.01.009

[18] B. Neto, R. Andrade, M. Maia, A. Fonteles, W. Viana, A coordination framework for dynamic adaptation in ubiquitous systems based on distributed tuple space, in: Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International, 2013, pp. 1430–1435.

[19] F. Rodrigues, S. Matalonga, G. H. Travassos, Systematic literature review protocol: Investigating context aware software testing strategies, Tech. Rep. 11/2014, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil (2014).

[20] S. Matalonga, F. Rodrigues, G. H. Travassos, Challenges in testing context aware software systems, in: Proceedings of the 2015 Workshop on Systematic and Automated Software Testing, SAST '15, 2015.

[21] J. Biolchini, P. Mian, A. Natali, G. Travassos, Systematic review in software engineering, Tech. rep. (2005).
URL http://www.cos.ufrj.br/uploadfiles/es67905.pdf

[22] B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering, Tech. Rep. EBSE-2007-01, Department of Computer Science, University of Durham, Durham, UK (jul 2007).

[23] ISO/IEC 25010. International Standard ISO/IEC Systems and Software Engineering Requirements and Evaluation, ISO/IEC 25010:2011.

[24] M. Pai, M. McCulloch, J. D. Gorman, N. Pai, W. Enanoria, G. Kennedy, P. Tharyan, J. J. Colford, Systematic reviews and meta-analyses: An illustrated, step-by-step guide, National Medical Journal of India 17 (2).

[25] L. Yu, W. T. Tsai, Y. Jiang, J. Gao, Generating test cases for context-aware applications using bigraphs, in: Software Security and Reliability (SERE), 2014 Eighth International Conference on, 2014, pp. 137–146. doi:10.1109/SERE.2014.27.

[26] A. A. Chadegani, H. Salehi, M. M. Yunus, H. Farhadi, M. Fooladi, M. Farhadi, N. A. Ebrahim, A comparison between two main academic literature collections: Web of science and scopus databases, Asian Social Science 9 (5).

[27] A. Méndez-Porras, C. Quesada-Lpez, M. Jenkins, Automated testing of mobile applications: A systematic map and review, in: XVIII Ibero-American Conference on Software Engineering, URP,SPC,UCSP, UCSP, Lima-Peru, 2015, pp. 195–208.

[28] IEEE, Is IEEE Xplore digital library content indexed in SCOPUS? (2016).
URL https://supportcenter.ieee.org/app/answers/detail/a_id/510/~/is-ieee-xplore-digital-library-content-indexed-in-scopus%3F

[29] IEEE, IEEE command search (2015).
URL http://ieeexplore.ieee.org/Xplorehelp/#/searching-ieee-xplore/command-search

[30] J. Kjeldskov, C. Graham, A review of mobile hci research methods, in: L. Chittaro (Ed.), Human-Computer Interaction with Mobile Devices and Services, Vol. 2795 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2003, pp. 317–335.

[31] G. Kotonya, I. Sommerville, S. Hall, Towards a classification model for component-based software engineering research, in: Euromicro Conference, 2003. Proceedings. 29th, 2003, pp. 43–52. doi:10.1109/EURMIC.2003.1231566.

[32] R. Pressman, Software engineering: a practitioner's approach, McGraw-Hill Higher Education, New York, USA, 2010.

[33] E. M. Fredericks, B. DeVries, B. H. C. Cheng, Towards run-time adaptation of test cases for self-adaptive systems in the face of uncertainty, in: Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2014, ACM, New York, NY, USA, 2014, pp. 17–26. doi:10.1145/2593929.2593937.

[34] H. Wang, W. K. Chan, T. H. Tse, Improving the effectiveness of testing pervasive software via context diversity, ACM Trans. Auton. Adapt. Syst. 9 (2) (2014) 9:1–9:28. doi:10.1145/2620000.
URL http://doi.acm.org/10.1145/2620000

[35] D. Amalfitano, A. Fasolino, P. Tramontana, N. Amatucci, Considering context events in event-based testing of mobile applications, in: Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on, 2013, pp. 126–133. doi:10.1109/ICSTW.2013.22.

[36] Z. Micskei, Z. Szatmri, J. Olh, I. Majzik, A concept for testing robustness and safety of the context-aware behaviour of autonomous systems, in: G. Jezic, M. Kusek, N.-T. Nguyen, R. Howlett, L. Jain (Eds.), Agent and Multi-Agent Systems. Technologies and Applications, Vol. 7327 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 504–513. doi:10.1007/978-3-642-30947-2_55.

[37] H. Wang, K. Zhai, T. Tse, Correlating context-awareness and mutation analysis for pervasive computing systems, in: Quality Software (QSIC), 2010 10th International Conference on, 2010, pp. 151–160. doi:10.1109/QSIC.2010.57.

[38] H. Wang, W. Chan, Weaving context sensitivity into test suite construction, in: Automated Software Engineering, 2009. ASE '09. 24th IEEE/ACM International Conference on, 2009, pp. 610–614. doi:10.1109/ASE.2009.79.

[39] C. Ye, S. C. Cheung, J. Wei, H. Zhong, T. Huang, A study on the replaceability of context-aware middleware, in: Proceedings of the First Asia-Pacific Symposium on Internetware, Internetware '09, ACM, New York, NY, USA, 2009, pp. 4:1–4:10. doi:10.1145/1640206.1640210.

[40] S. Propp, G. Buchholz, P. Forbrig, Task model-based usability evaluation for smart environments, in: P. Forbrig, F. Paterno (Eds.), Engineering Interactive Systems, Vol. 5247 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 29–40.

[41] J. Bo, L. Xiang, G. Xiaopeng, Mobiletest: A tool supporting automatic black box test for software on smart mobile devices, in: Automation of Software Test , 2007. AST '07. Second International Workshop on, 2007, pp. 8–8. doi:10.1109/AST.2007.9.

[42] H. Lu, W. K. Chan, T. H. Tse, Testing context-aware middleware-centric programs: A data flow approach and an rfid-based experimentation, in: Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, SIGSOFT '06/FSE-14, ACM, New York, NY, USA, 2006, pp. 242–252. doi:10.1145/1181775.1181805.
URL http://doi.acm.org/10.1145/1181775.1181805

[43] W. K. Chan, T. Y. Chen, H. Lu, T. H. Tse, S. S. Yau, Integration testing of context-sensitive middleware-based applications: a metamorphic approach, International Journal of Software Engineering and Knowledge Engineering 16 (2006) 677–703.

[44] W. Chan, T. Chen, H. Lu, T. Tse, S. Yau, A metamorphic approach to integration testing of context-sensitive middleware-based applications, in: Quality Software, 2005. (QSIC 2005). Fifth International Conference on, 2005, pp. 241–249. doi:10.1109/QSIC.2005.3.

[45] T. Tse, S. Yau, Testing context-sensitive middleware-based software applications, in: Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International, 2004, pp. 458–466 vol.1. doi:10.1109/CMPSAC.2004.1342879.

[46] J. L. Wynekoop, S. Conger, A review of computer aided software engineering research methods, in: In Proceedings of the IFIP TC8 WG 8.2 Working Conference on The Information Systems Research Arena of The 90s, Copenhagen, Denmark, 1990.

[47] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, A. Wessln, Experimentation in Software Engineering, Springer Publishing Company, Incorporated, 2014.

[48] F. Elberzhager, A. Rosbach, J. Mnch, R. Eschbach, Reducing test effort: A systematic mapping study on existing approaches, Information and Software Technology 54 (10) (2012) 1092 – 1106.

[49] M. Shahid, S. Ibrahim, M. N. Mahrin, A study on test coverage in software testing, in: International Conference on Telecommunication Technology and Applications, 2011.

[50] H. Lu, W. Chan, T. Tse, Testing pervasive software in the presence of context inconsistency resolution services, in: Software Engineering, 2008. ICSE '08. ACM/IEEE 30th International Conference on, 2008, pp. 61–70. doi:10.1145/1368088.1368098.

[51] P. Frankl, E. Weyuker, An applicable family of data flow testing criteria, IEEE Trans. Softw. Eng. 14 (10) (1988) 1483–1498. doi:10.1109/32.6194.

[52] C. Bezerra, R. M. C. Andrade, R. M. Santos, M. Abed, K. M. de Oliveira, J. M. Monteiro, I. Santos, H. Ezzedine, Challenges for usability testing in ubiquitous systems, in: Proceedings of the 26th Conference on L'Interaction Homme-Machine, IHM '14, ACM, New York, NY, USA, 2014, pp. 183–188. doi:10.1145/2670444.2670468.
URL http://doi.acm.org/10.1145/2670444.2670468

[53] P. McMinn, Search-based software testing: Past, present and future, in: Proceedings of the 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, ICSTW '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 153–163. doi:10.1109/ICSTW.2011.100.

[54] D. Qiu, B. Li, S. Ji, H. Leung, Regression testing of web service: A systematic mapping study, ACM Comput. Surv. 47 (2) (2014) 21:1–21:46. doi:10.1145/2631685.
URL http://doi.acm.org/10.1145/2631685

[55] U. Kanewala, J. M. Bieman, Testing scientific software: A systematic literature review, Inf. Softw. Technol. 56 (10) (2014) 1219–1232. doi:10.1016/j.infsof.2014.05.006.
URL http://dx.doi.org/10.1016/j.infsof.2014.05.006

[56] C. Perera, C. Liu, S. Jayawardena, M. Chen, A survey on internet of things from industrial market perspective, Access, IEEE 2 (2014) 1660–1679. doi:10.1109/ACCESS.2015.2389854.

[57] M. Harman, S. A. Mansouri, Y. Zhang, Search-based software engineering: Trends, techniques and applications, ACM Comput. Surv. 45 (1) (2012) 11:1–11:61. doi:10.1145/2379776.2379787.

[58] M. Harman, Y. Jia, Y. Zhang, Achievements, open problems and challenges for search based software testing, in: 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST), 2015, pp. 1–12. doi:10.1109/ICST.2015.7102580.

[59] Z. D. Champiri, S. R. Shahamiri, S. S. B. Salim, A systematic review of scholar context-aware recommender systems, Expert Syst. Appl. 42 (3) (2015) 1743–1758. doi:10.1016/j.eswa.2014.09.017.
URL http://dx.doi.org/10.1016/j.eswa.2014.09.017

[60] V. Vieira, K. Holl, M. Hassel, A context simulator as testing support for mobile apps, in: Proceedings of the 30th Annual ACM Symposium on Applied Computing, SAC '15, ACM, New York, NY, USA, 2015, pp. 535–541.