TECHNICAL REPORT

RT-ES-760

A Rapid Review on Testing of Context-Aware Contemporary Software Systems

Domenico Amalfitano[1]
(domenico.amalfitano@unina.it)

Santiago Matalonga[2]
(smatalonga@gmail.com)

Andrea Doreste[3]
(doreste@cos.ufrj.br)

Anna Rita Fasolino[1]
(fasolino@unina.it)

Guilherme H. Travassos[3]
(ght@cos.ufrj.br)

1 2 3

Systems Engineering and Computer Science Department

August, 2019

A Rapid Review on Testing of Context-Aware Contemporary Software Systems

1. INTRODUCTION

1.1. Authors

Domenico Amalfitano - UniNa - domenico.amalfitano@unina.it

Santiago Matalonga - UoWE - smatalonga@gmail.com

Andrea Doreste - COPPE/UFRJ - doreste@cos.ufrj.br

Anna Rita Fasolino - UniNa - fasolino@unina.it

Guilherme H. Travassos - COPPE/UFRJ - ght@cos.ufrj.br

1.2. Context

In the investigation regarding Contemporary Software Systems (CSS) - such as the Internet of Things, Industry 4.0, and Smart Cities – it has been observed that these modern software systems offer challenges for their construction since they are challenging our traditional methods for software development. Usually, they rely on different technologies and devices that can interact-capture-exchange information, act, and make decisions. It leads to concerns that rather than just developing software, we need to engineer software systems embracing multidisciplinarity, integrating different areas.

In this context, previous investigations identified a lack of software testing technologies to support the evaluation of context-awareness, one of the main features of CSS. The observations made so far revealed that, despite applying conventional testing technologies to such software systems, it seems not to be possible to test the behavior of the system considering the variation of the context (at runtime). Besides, the design of test cases uses to miss context variables, which reduces the testing coverage and do not contribute to reducing the likelihood of failures due to context-related defects.

However, all of these technical deficiencies and challenges have not avoided the industry on building CSS and, regardless of the coverage, testing their essential features.

The software industry, considering the current situation and growing demand for contemporary software systems, had been struggling to test such systems taking into account the unavailability of a body of knowledge indicating the software technologies (processes, methods, tools, techniques, practices) supporting the full testing of CSS.

Therefore, this Rapid Review (RR) aims to analyze the available software technologies supporting the practitioners to test the variation of context in CSS, that have been reported applications in industrial settings. The organization of this RR research protocol is based on the work of Victor V. Ribeiro [12] and Rebeca Motta in the ESE group at COPPE/UFRJ.

2. DEFINITIONS

The definitions regarding the basic concepts of the software testing process come from (ISO/IEC/IEEE 29119 Organizational test process). Other specific terms are defined as follow:

- Context [4]: "any information that can be used to characterize the situation of entities that are considered relevant to the interaction between users and applications."
- Context-awareness [5]: "is a dynamic property representing a piece of information, which can be evolutionary affect the overall behavior of the software system in the interaction between the actor and the computer."
- Contemporary software systems [6]. In our perspective, CSS represent the modern systems of systems, context-aware, ubiquitous systems, Industry 4.0, Internet of Things, smart cities, intelligent environments, and Cyber-physical Systems.

3. RESEARCH QUESTIONS

3.1. A Proposal for Goal and RQs

- Goal: To characterize how software engineers are testing Contemporary Software Systems (CSS) and which software technologies (processes, methods, techniques, practices, and tools) they use to *deal with the context* when testing such systems.
    - RQ1: Are software engineers dealing with the context when testing CSS in the industry?
        - Rationale: It is important to observe whether testing is performed for CSS when dealing with the context and its variation. In this scenario, we are interested in characterizing whether they are considering the context in testing, to understand whether the context is instantiated and varied during the execution of the test cases and whether the system under test interacts at runtime with the running context. Otherwise, the other questions can not be answered. In this case, this research protocol should evolve into a multivocal study, taking into account other sources of information, such as grey literature, blogs, and industrial posts to look for some empirical evidence on the testing of CSS in the industry.
        - RQ1.1: Which are the software technologies supporting the **planning** of test activities dealing with the context of CSS?
            - Rationale: To understand how the context and its variations are being considered when planning to test.
        - RQ1.2: Which are the software technologies supporting the **management** of test activities dealing with the context for CSS?
            - Rationale: To understand how the context and its variations are being considered when managing testing activities.
        - RQ1.3: Which are the software technologies supporting the **execution** of test activities dealing with the context of CSS?
            - Rationale: To understand the support provided for the execution of CSS testing. We want to understand what support is given, to software practitioners, for supporting

Context and its variation during the execution of CSS testing.

4. SEARCH STRATEGY

We used Scopus as the leading search engine and ACM Digital Library as secondary.

We applied our search engine and selection procedures in SCOPUS and repeated the exercise in ACM Digital Library. The following articles, taken from the results, were used as the control papers (though they do not present industrial applications):

1. Wang, H., Chan, W.K., Tse, T.H.: Improving the Effectiveness of Testing Pervasive Software via Context Diversity. ACM Trans. Auton. Adapt. Syst. 9, 9:1--9:28 (2014).
2. Tse, T.H., Yau, S.S., Chan, W.K.K., Lu, H., Chen, T.Y.Y.: Testing context-sensitive middleware-based software applications. In: Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC). Pp. 458–466. IEEE (2004).

4.1. Scopus Search string

The Scopus[1] search engine was first selected, and the following search string support this RR, built using PICOC with five levels of filtering:

Population - contemporary software systems

Synonyms: ("Ambient Intelligence" OR "Assisted Living" OR "Multiagent Systems" OR "Systems of Systems" OR "Internet of Things" OR "Cyber Physical Systems" OR "Autonomous Systems" OR "Autonomic Computing" OR "Multi-Agent Systems" OR "Pervasive Computing" OR "Mobile Computing" OR "Distributed Systems" OR "Cooperative Robotics" OR "Adaptive Systems" OR "Industry 4.0" OR "Fourth Industrial Revolution" OR "Web of Things" OR "Internet of Everything" OR "Contemporary Software Systems" OR "Smart Manufacturing" OR Digitalization OR Digitization OR "Digital Transformation" OR "Smart Cit*" OR "Smart Building" OR "Smart Health" OR "Smart Environment" OR "Digital Transformation")

Intervention - Software Testing

Synonyms: ("Test* Management" OR "Test* Planning" OR "Test* Monitoring" OR "Test* Control" OR "Test* Completion" OR "Test* Design" OR "Test* Type" OR "Test* Implementation" OR "Test* Environment" OR "Test* Execution" OR "Test* Reporting" OR "software test*" OR "software validation" OR "software verification")

Comparison – no

Outcome - Software Testing Technologies

[1] https://www.scopus.com

Synonyms - (“Technique” OR “Technolog*” OR “Method” OR “Activity” OR “Tool” OR “Process” OR “Practice” OR “Mechanism” OR “Instrument” OR “Task” OR “Service” OR “Strategy”)

Context – (“Variation” OR “Context” OR “Context Awareness” OR “Context Variation”)

Limited to articles from 2002 to 2019

Limited to Computer Science and Engineering

AND (LIMIT-TO (SUBJAREA, "COMP") OR LIMIT-TO (SUBJAREA, "ENGI")) AND PUBYEAR AFT 2001）

```
TITLE-ABS-KEY(("ambient intelligence" OR "assisted living" OR "multiagent systems" OR "systems of systems" OR "internet of things" OR "Cyber Physical Systems" OR "autonomous systems" OR "autonomic computing" OR "Multi-Agent Systems" OR "Pervasive Computing" OR "Mobile Computing" OR "Distributed Systems" OR "Cooperative Robotics" OR "adaptive systems" OR "Industry 4" OR "fourth industrial revolution" OR "web of things" OR "Internet of Everything" OR "contemporary software systems" OR "smart manufacturing" OR digitalization OR digitization OR "digital transformation" OR "smart cit*" OR "smart building" OR "smart health" OR "smart environment" OR "digitalization" OR "digital transformation") AND ("Test* management" Or "test* planning" or "Test* Monitoring" Or "Test* control" or "test* completion" or "Test* design" or "test* type" or "test* implementation" or "test* environment" or "test* execution" or "test* reporting" OR "software test*" OR "software validation" OR "software verification" ) AND ("technique" OR "technolog*" OR "method" OR "activity" OR "tool" OR "process" OR "practice" OR "mechanism" OR "instrument" OR "task" OR "service") AND ("mutation" OR "variation" OR "metamorphic")) AND PUBYEAR > 2002 AND ( LIMIT-TO ( SUBJAREA,"COMP" ) OR LIMIT-TO ( SUBJAREA,"ENGI" ) )
```

4.2. ACM Digital Library search string

We adapted the search string to fit the restrictions of the ACM Digital Library. The resulting search string was:

```
+("ambient intelligence" "assisted living" "multiagent systems" "systems of systems" "internet of things" "Cyber Physical Systems" "autonomous systems" "autonomic computing" "Multi-
```

```
Agent Systems" "Pervasive Computing" "Mobile Computing"
"Distributed Systems" "Cooperative Robotics" "adaptive systems"
"Industry 4" "fourth industrial revolution" "web of things"
"Internet of Everything" "contemporary software systems" "smart
manufacturing" digitalization digitization "digital
transformation" "smart city" "smart building" "smart health"
"smart environment" digitalization "digital transformation")
+("Testing management" "testing planning" "Testing Monitoring"
"Testing control" "testing completion" "Testing design"
"testing type" "testing implementation" "testing environment"
"testing execution" "testing reporting" "software testing"
"software validation" "software verification") +("technique"
"technology" "method" "activity" "tool" "process" "practice"
"mechanism" "instrument" "task" "service") +("mutation"
"variation" "metamorphic")
```

4.3. Snowballing

Snowballing was applied with the sources selected for data extraction from the automatic search strategies. We applied forward and backward snowballing using Scopus as the tools for identifying further references.

5. SELECTION PROCEDURE

One researcher - different from the one who defined the search string - performs the following selection procedure:

1. Run the search string in Scopus;
    a. Apply the inclusion criteria based on the paper Title;
    b. Apply the inclusion criteria based on the paper Abstract;
    c. Apply the inclusion criteria based on the paper Full Text, and;
2. Run the ACM search string in ACM
    a. Apply the inclusion criteria based on the paper Title;
    b. Apply the inclusion criteria based on the paper Abstract;
    c. Apply the inclusion criteria based on the paper Full Text, and;

3. After finishing the selection from the search engines, use the included papers set to:
    a. Execute snowballing backward (one level) and forward:
        i. Apply the inclusion criteria based on the paper Title;
        ii. Apply the inclusion criteria based on the paper Abstract;
        iii. Apply the inclusion criteria based on the paper Full Text.

The JabRef Tool[2] must be used to manage and support the selection procedure.

6. INCLUSION CRITERIA

1. The paper must be in the context of **software engineering**; and
2. The paper must be in the context of **contemporary software systems**; and

[2] http://www.jabref.org/

3. The paper must report a **primary**; and
4. The **primary** study was conducted within an **industrial context** or **setting,** and
5. The paper must report an **evidence-based study** grounded in empirical methods (e.g., interviews, surveys, case studies, formal experiment, and others); and
6. The paper must provide data to **answering** at least one of the RR **research questions**, and
7. The paper must be written in the **English language**.

7. EXTRACTION PROCEDURE

For each candidate source, the extraction procedure is performed by one researcher, using the form presented in Section **7.1.** A Second researcher performed a proof-read of the data extraction, to verify the quality and integrity of the extraction.

7.1. Extraction Form

<paper_id>:<paper_reference>	
Abstract	Abstract
Description	A brief description of the study objectives and personal understanding
Study type	Extracted from the source as described by it.
Application Domain	The industrial domain where the study took place
Type of Modern software system	Type of Modern Software System
RQ1: How are software engineers dealing with the context when testing CSS in the industry?	- Description of context considerations. - Descriptions of the outlook of context (Frame of thought / Positivism / Context as a Problem/constraint or as an element of the environment?
RQ1.1: Which are the software technologies supporting the **planning** of test activities dealing with the context of CSS?	- Extraction of the planning process and its relationship to the context. References to Test Design Techniques / Test Types Techniques in terms of planning
RQ1.2: Which are the software technologies supporting the **management** of test activities dealing with the context for CSS?	- Extraction of the management process and its relationship to context. (References to the environment (either intended operational environment and/or the Test Environment)
RQ1.3: Which are the software technologies supporting the **execution** of test activities dealing with the context of CSS?	Extraction of the execution process and its relationship to context. (References to Test Design Techniques / Test Types Techniques during the execution).

8. SYNTHESIS PROCEDURE

In this RR, the extraction form provides a synthesized way to represent extracted data. Thus, we do not perform any synthesis procedure.

However, the synthesis is usually performed through a narrative summary or a Thematic Analysis when the number of selected papers is not high [2].

9. REPORT

An Evidence Briefing [3] reports the findings to ease the communication with practitioners.

10. EXECUTION RESULT

- Scopus Execution date: 2018/12/04
  - Jab ref file with results from Scopus
- ACM Execution Date: 2019/02/08
  - Jab Ref File with Results from ACM papers
- Included after applying inclusion criteria:
  - From Scopus:
    - Jab Ref. Final Set from Scopus
  - From ACM:
    - No paper selected from ACM
  - From snowballing
    - Backward snowballing
    - Forward snowballing
- Included for Data Extraction (and used for snowballing seeds):
  - Data Extraction files
- Included on Snowballing:
  - Data Extraction files

The final set of selected papers:

- **From Automatic Search Engines**

A1. Shin, S. Y. et al. (2018) 'Test case prioritization for acceptance testing of cyber physical systems: a multi-objective search-based approach,' in Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis - ISSTA 2018. New York, New York, USA: ACM Press, pp. 49–60. doi: 10.1145/3213846.3213852.

A2. Shin, S. Y. et al. (2018) 'HITECS: A UML Profile and Analysis Framework for Hardware-in-the-Loop Testing of Cyber-Physical Systems,' in Proceedings of the 21st ACM/IEEE International Conference on Model-Driven Engineering Languages and Systems - MODELS '18. New York, New York, USA: ACM Press, pp. 357–367. doi: 10.1145/3239372.3239382.

A3. Lahami, M., Krichen, M. and Jmaiel, M. (2016) 'Safe and efficient runtime testing framework applied in dynamic and distributed systems,' Science of Computer Programming. doi: 10.1016/j.scico.2016.02.002.

A4. Fröhlich, J. et al. (2016) 'Testing safety properties of cyber-physical systems with non-intrusive fault injection – An industrial case study,' in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). doi: 10.1007/978-3-319-45480-1_9.

- **From Snowballing**

S1. Ma, T., Ali, S. and Yue, T. (2018) 'Modeling foundations for executable model-based testing of self-healing cyber-physical systems,' Software & Systems Modeling. doi: 10.1007/s10270-018-00703-y.

S2. Arrieta, A. et al. (2017) 'Automatic generation of test system instances for configurable cyber-physical systems,' Software Quality Journal, 25(3), pp. 1041–1083. doi: 10.1007/s11219-016-9341-7.

S3. Fredericks, E. M., DeVries, B., and Cheng, B. H. C. (2014) '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. New York, New York, USA: ACM Press, pp. 17–26. doi: 10.1145/2593929.2593937**.**

S4. Sama, M. et al. (2010) 'Context-Aware Adaptive Applications: Fault Patterns and Their Automated Identification,' IEEE Transactions on Software Engineering, 36(5), pp. 644–661. doi: 10.1109/TSE.2010.35.

11. RAPID REVIEW REPORT

This section presents our responses to the research questions. Through the rapid review process, we have observed how the terms (context and context-awareness) are still used with several interpretations in the literature. For instance, Ali and Yue [8], use the term "context" to describe the Test environment (a source of confusion we had already reported in [8]). Another source of confusion comes from the interpretation of "industrial application." The pre-conception, when starting the Rapid Review, was to identify empirical studies taking place in the industry. Our running assumption is that context-aware software systems are being built and deployed. However, our research shows that there is still little evidence in the technical literature of serious-academic studies that take place in industrial/commercial-led settings.

The primary sources that were discarded at the Data Extraction stage were discarded on account of these two mentioned motives. Nonetheless, we believe that this also conveys a path to research. For instance, two of them [9, 10] mention the concept that is is possible to duplicate the deployment environment and use live input to the "production" system into the "duplicate" system. It should enable the tester to capture real-time input, but, we have found no evidence of this strategy being applied to a context-aware software system.

11.1. RQ1: Are software engineers dealing with the context when testing CSS in the industry?

Although the findings do hold firmly to provide a clear and confirmatory answer, it is possible to observe some initiatives towards taking the context into account when considering the context when performing the testing of contemporary software systems, such as mentioned and someway exemplified in the primary sources of information [A1-A4; S1-S4]. The following research questions are going to be discussed and, someway, answered considering this scenario.

11.1.1. RQ1.1: Which are the software technologies supporting the planning of test activities dealing with the context of CSS?

- **Domain-specific platform**. An approach observed in the primary sources of information is the development of a domain-specific platform for monitoring and testing context-aware software systems.
    - A2 presents a platform for testing satellites software systems, that enables the specification and execution of test cases considering the changes in the context.
    - A4 presents a syntax for describing text cases incorporating uncertainty (contextual changes) in self-driving cars.
    - S2 presents a platform that can modify configurations of the SUT. In this case, the SUT is a commercial drone family of products.

- **Context Change Runtime Software System Monitor and Test Case Selector.** In the approaches observed in the primary sources of information, the software system is deployed to the operational environment with a monitoring platform that observes the behavior of the system when exposed to different changes in the environment and selects a suitable set of Test Cases to run when context changes are detected. (A1, A3, S3).

  Upon detecting contextual changes, the next problem is to design a suitable Test suite to which the SUT must be executed:

    - A1 uses search-based techniques to select a suitable test suite.
    - A3 injects faults in “non-critical subsystems” and sandbox the execution of this Test Case/Suite.
    - S3 applies an evolutionary algorithm to adapt the initial set of test suites to the newly identified values for the context.

- **Contextual Architecture Model**. The approach in S4, is different from the other selected sources, in that it assumes that the SUT architecture must have a middleware taking care of detecting and adapting the contextual changes. As a result of this assumption, the authors developed a Finite State Machine that enables the detection of faults in the SUT adaptation logic and its behavior.

11.1.2. RQ1.2: Which are the software technologies supporting the management of test activities dealing with the context for CSS?

From the primary sources, only S3 incorporates both tool development and process technologies, thereby recommending to the practitioners which activities to perform at different stages of the CSS testing process.

11.1.3. RQ1.3: Which are the software technologies supporting the execution of test activities dealing with the context of CSS?

Some of the technologies used to design the Test Cases also support their execution, as the cases described in A2, S1, S2, and S3.

In the case of A2, a technological infrastructure was designed to support methodological innovation. As described, the technological infrastructure exploits Test Suites, designed by the software tester into the platform, to generate simulated test suites.

In the case of S1, the environment was built specifically to monitor the execution of Test Cases. In this case, the environment injects faults in the sensor data and observes how the SUT behaves to this simulated changing in the context.

In S2, the test designer identifies the "variability points" of the system within the platform. The test designer also defines the set of configurations of the SUT that is going to take part in the scope of the test. The platform then takes care of designing the valid set of test suites given those both set of inputs (variability points, SUT configurations).

In S3, the platform monitors the operating environment (context) for checking evidence of change, and it determines which test cases are relevant to the current conditions of execution.

11.2. Research directions abstracted from not include primary sources.

From those papers that we extracted the data, but we did not include since they did not present an industrial application [X1-X?], an emerging pattern is an intent of monitoring the behavior of the system at runtime. This approach - which has not been consistently named - "*in-vivo*" testing [9, 10] or runtime testing [11] - relies on continually monitoring the SUT sensors in the operational environment (like in S3). We argue that this approach needs to be further explored as there is little empirical evidence (just mentioned in S3), but it might seem promising for supporting the testing of CSS in the industry whether some issues regarding the duplication of environments and costs reduction can be mitigated.

11.3. Confronting pre-conceptions

On a final note, it is notable that we were not able to find evidence of data-intensive approaches for testing the context in CSS. For instance, large companies usually trace the use of their systems and have plenty of contextual data available regarding their software systems, such as self-driving cars, robots, jet airplanes, among others. However, none of these software systems or initiatives regarding their testing are reported in the included primary sources. On the one hand, it is likely such information represents some strategic asset that software companies would not share with their peers. On the other hand, the news we know by the media can also indicate a lack of adequate software technologies to support the testing of context-aware CSS, as the failures currently reported by these software systems (cars, robots, airplanes, among others).

12. CONCLUSIONS

As far as we could observe, there is shallow evidence on the testing of context-awareness in CSS. Most of the identified approaches do not support the full observation of the changes in the context when testing CSS. Besides, the observed approaches require extensive investment in technological and human resources for their deployment in the industry. Therefore, little evolution has been observed in the technical literature since the results published in [8], despite the growing demand for context-awareness CSS in the industry.

Research and working agenda should be discussed between software researchers and practitioners aiming to minimize the risks of lacking adequate software testing technologies for context-awareness CSS. Such an agenda should include topics regarding how model-driven testing approaches could be adopted for context-awareness CSS, as well as how the introduction of automated testing approaches may improve the cost-effectiveness of V&V processes for such software systems.

13. REFERENCES

[1] C. Tricco et al. A scoping review of rapid review methods. BMC Medicine, 2015.

[2] B. Cartaxo et al.: The Role of Rapid Reviews in Supporting Decision -Making in Software Engineering Practice. EASE 2018.

[3] B. Cartaxo et al. Evidence briefings: Towards a medium to transfer knowledge from systematic reviews to practitioners. ESEM, 2016.

[4] A. K. Dey and G. D. Abowd, “Towards a better understanding of context and context-awareness,” in Proceedings of the CHI 2000 Workshop on The What, Who, Where, When and How of Context-Awareness, 2000, vol. 4, pp. 1–6.

[5] I. de S. Santos, R. M. de C. Andrade, L. S. Rocha, S. Matalonga, K. M. de Oliveira, and G. H. Travassos, “Test case design for context-aware applications: Are we there yet?,” Information and Software Technology, vol. 88, pp. 1–16, Aug. 2017.

[6] R. C. Motta, K. M. de Oliveira, G. H. Travassos. On Challenges in Engineering IoT Software Systems. SBES 2018. September 17-21, 20. São Carlos, Brazil. ACM ISBN 978-1-4503-6503-1/18/09

[7] S. Ali and T. Yue, "U-Test: Evolving, Modelling and Testing Realistic Uncertain

Behaviours of Cyber-Physical Systems," *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*, Graz, 2015, pp. 1-2. doi: 10.1109/ICST.2015.7102637.

[8] Matalonga, S., Rodrigues, F. and Travassos, G. H. (2017) 'Characterizing testing methods for context-aware software systems: Results from a quasi-systematic literature review,' *Journal of Systems and Software*, 131, pp. 1–21. doi: 10.1016/j.jss.2017.05.048.

[9] C. Murphy, G. Kaiser, I. Vo, and M. Chu, "Quality Assurance of Software Applications Using the In Vivo Testing Approach," *2009 International Conference on Software Testing Verification and Validation*, Denver, CO, 2009, pp. 111-120. doi: 10.1109/ICST.2009.18

[10] M. Chu, C. Murphy, and G. Kaiser, "Distributed In Vivo Testing of Software Applications," 2008 1st International Conference on Software Testing, Verification, and Validation, Lillehammer, 2008, pp. 509-512. doi: 10.1109/ICST.2008.13A.

[11] Kane, T. Fuhrman and P. Koopman, "Monitor Based Oracles for Cyber-Physical System Testing: Practical Experience Report," 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Atlanta, GA, 2014, pp. 148-155. doi: 10.1109/DSN.2014.28

[12] Ribeiro, V. V., Cruzes, D. S., and Travassos, G. H. (2019). "Study on Practices, Moderator Factors, and Decision-Making of Security and Performance Verification". Submitted to Software Quality Journal.