Protocol for Systematic Literature Review

Current Version: 1.0
Current Date: 2022/05/14

Investigators: Ricardo Caldas[1], Patrizio Pelliccione[2], Genaina Rodrigues[2]

1 Motivation

As robotic systems crawl from industry to service applications, it is essential for ensuring safe behavior against unexpected inputs [1]. Verification, validation, and testing activities are commonly conducted during the development cycle to both reveal faults before deployment and study failures reported during execution. Yet, assurance techniques undertaken in the development cycle are limited to simulated environments, opening a gap between design-time guarantees and runtime stimuli. Failures stemming from runtime cannot be fully prevented, posing a challenge to roboticists designing for highly complex and dynamic environments.

Techniques fostering the collection of runtime data and treatment of unexpected scenarios play an essential role in enabling autonomous behavior despite the intricacies of the environment. For instance, the Robotic Operating System (ROS) is a middleware that naturally emerged from this need. ROS enables seamless dynamic interactions between independent computing nodes in a peer-to-peer design. Such middleware facilitates the development of modular robotic software with guarantees of compliance with ever-changing environments. Moreover, building confidence in ROS applications is hard, for the interactions between computing nodes are enacted (and often re-enacted) only at production.

Field-based testing is known for coping with the complexity, unpredictability, evolvability and size of modern software [2]. In addition, other sixty (60) tools for runtime verification present a wide plethora of opportunities for assuring dynamic behavior in ROS-based applications [3]. However, such techniques and works are not tailored to ROS-based applications, questioning whether and why there is no complete solution to ensure ROS programs [4].

Therefore, we are keen to build upon these works and provide an up-to-date review of the literature on the characteristics and usage of field-testing and runtime monitoring tailored for ROS-based applications. Such a review will provide researchers with the state of the art overview and has the potential to identify open challenges. To that end, we will perform a systematic literature review study. This document describes the protocol for this study.

2 Research Method- Systematic Literature Review

We will conduct a systematic literature review study on the published knowledge in the runtime verification and validation of ros-based applications. Our systematic literature review consists of a process with four steps as shown in Figure 1: Research Questions Definition, Conduct Search, Screening of Papers, and Data Extraction and Synthesis.

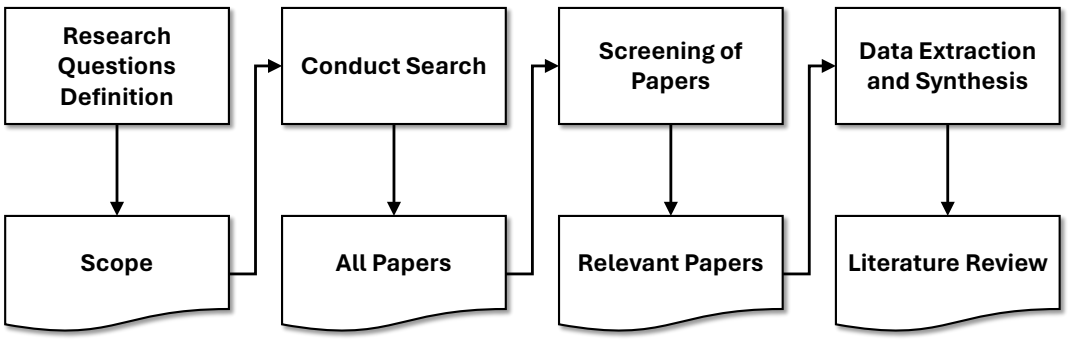

Figure 1: Four steps of the literature review.

In the remainder of this document we elaborate on each of these steps and explain how the study is organized accordingly. With this document in hands, the reader should be able to execute or replicate the literature review.

3 Research Questions Definition

The research questions drive the literature review and all the following steps are based on them. The process of eliciting and specifying the research questions involves all researchers that participate in the literature review. We define the overall goal of the literature review using the Goal-Question-Metric (GQM) approach [5], see Table 1.

Table 1: Goal-based research questions.

Goal	
Purpose	Characterize
Issue	runtime monitoring (RM) and field-testing (FT) for
Object	building confidence on ROS-based applications
Viewpoint	from researchers perspective.
RQ1	Whether and how runtime monitoring or field-testing been used to build confidence in ROS-based applications?
RQ2	What quality criteria are considered when using the aforementioned techniques?
RQ3	What are the assumptions about the system for applying RM or FT?
RQ4	Why are runtime monitoring or field-testing used for building confidence about ROS-based applications?
RQ5	Which other techniques are used to build confidence on ROS-based applications at runtime?

With RQ1 we aim to create an overview of runtime monitoring and field-testing for gaining confidence about ROS-based applications. Using general knowledge on runtime verification [3] and field-testing [2], we will extract particular information about how the technique is used, possibly highlighting gaps from the state-of-the-art and the state-of-practice within ROS domain.

With RQ2 we deepen into the quality criteria that are targeted by runtime monitoring and field-testing. Different quality models may be more or less suitable to different variants of the aforementioned techniques. This analysis may help to understand how different variants of RM and FT may benefit ROS-developers targeting specific quality criteria.

With RQ3 we aim to get an understanding of what kind of systems are enabled by each technique. Also, we aim to extract whether the runtime monitoring or field-testing are domain dependent.

With RQ4 we are keen to characterize the scientific/practical context, motivatives, and rationale that lead researchers into proposing and using runtime monitoring and field-testing.

Finally, with RQ5 we aim to look at runtime monitoring and field-testing among other techniques that can be used for gaining confidence on ROS-based applications at runtime. Our interests are to analyse what is particular to the techniques in question (RM and FT) and what is generalizeable. This question helps in delimiting the boundaries of runtime monitoring and testing in ROS-based applications.

4 Conduct Search

We use automatic search-engines for collecting an initial set of papers, i.e., IEEEXplore, ACM Digital Library, and Scopus.

Search String = (ros OR ”robotic operating system”) AND (”runtime verification” OR ”runtime assurance” OR ”online assurance” OR ”online verification” OR ”runtime monitoring” OR ”runtime testing” OR ”online testing” OR ”on-line testing” OR ”field-based testing” OR ”field testing” OR ”in-vivo testing”)

The set of all publications are returned applying the search string in the aforementioned search engines is 75 papers.

5 Screening Papers

Screening papers consists of passing the complete set of potentially useful papers through a filter. The outcome is the set of relevant papers. Thus, in this section we define the filter, by means of inclusion and exclusion criteria, and then the procedure to be followed when discarding papers.

Table 2: Inclusion Criterion (IC) and Exclusion Criterion (EC)

ID	Description	Reasoning
IC_1	ROS-based application. Including ROS1 and ROS2.	Robotic Operating System (ROS) is a must.
IC_2	Explicit description (or reference to peer-reviwed venue) of the verification, validation, or testing technique.	Papers that do not explicitely describe the employed technique may lead to ambiguous interpretation.
EC_1	Tutorial, artifact, short paper (less than 5pgs), keynote, secondary studies, roadmaps, duplicated study[1]	Such papers do not provide enough contextual information.
EC_2	Verification, validation, or testing techniques that do not make use of the ROS Ecosystem[2]	Papers targeting V&V of non-ROS applications should be excluded.
EC_3	Verification, validation, or testing techniques that do not address solely hardware properties	Papers targeting V&V of hardware should be excluded.

The individual work of screening consists of checking if the paper is in compliance with the criteria presented in Table 2. This is done by parsing the paper in search for ROS or "robotic operating system", which is usually spotted in the abstract and introduction. Then, looking for verification, validation, and testing techniques names (e.g., runtime monitoring, testing, theorem proving, model checking). Once a technique is found, the surrounding text is read to confirm if there is an explicit description of how it is used–or reference. Finally, scanning whether the used technique uses the ROS ecosystem.

The papers containing all the inclusion criteria and none of the exclusion criteria are deemed relevant for data extraction.

6 Data Extraction and Synthesis

Finally, the data is extracted from the set of relevant papers and synthesized in a spreadsheet. However, this process demands a scheme containing the precise information that is useful for answering the research questions. Laying on knowledge from papers that inspired this work [3, 2, 6] we elicit and specify the data items to be extracted. Then, we provide clear-cut definitions and examples for each data item. See Table 3.

(F1-F3) Metadata. Data about the papers collected. For documentation.

(F4) Runtime Monitoring (RM) or Field-Testing (FT)? Whether the authors from the paper use RM or FT, or none.

(F5) Adherence to Taxonomy In case it is classified as Runtime Monitoring, this item asks for a deeper analysis of the technique in terms of Specifications, Traces, Interference, Reaction, Monitor, Deployment [3]. In case it is classified as Testing, this item asks for a deeper analysis of the technique in terms of In-House/Field-based, Environment (Development/Production), Moment (Ex-vivo, Offline or Online) [2].

(F6) Process View When explicitely described, collect the step-by-step in list format to reproduce the technique.

(F7) Quality type Defines what quality criteria is analysed. E.g. Safety, Reliability, Performance, Security.

(F8) Quality definition. Defines the quality criteria. Usually presented in natural language description but can also be found as mathematical model.

(F9) Use Case Typically used by researchers for demonstrating the effectiveness and efficacy of the approach. It can also be the subject of design in cased of a design paper.

(F10) System Category. Service or Autonomous Systems. Sub-catergories of both domains are also helpful. E.g., Autonomous Underwater Vehicle (AUG), Autonomous Driving Vehicle (ADV).

Table 3: Data items to collect.

ID	Field	Use
F1	Author(s)	Documentation
F2	Publication Year	Documentation
F3	Title	Documentation
F4	Runtime Monitoring or Field-Testing?	RQ1
F5	Taxonomy	RQ1
F6	Process View	RQ1
F7	Quality type	RQ2
F8	Quality definition	RQ2
F9	Use Case	RQ3
F10	System Category	RQ3
F11	Application Domain	RQ3
F12	Link to Repository	RQ3
F13	Context	RQ4
F14	Motivation	RQ4
F15	Rationale	RQ4
F16	Technique name	RQ5
F17	Technique type	RQ5
F18	Dynamicity	RQ5
F19	Offline/Online	RQ5

(F11) Application Domain. Use case domain. E.g. Robotics, Self-Driving Vehicles, Internet of Things, Healthcare, Transportation, Mobile Communication, Finance, Business Analytics & eCommerce, Decision Support Systems, Forestry Farming & Urban Informatics, Manufacturing and Process Control, Logistics and Maintenance, Public Safety.

(F12) Link to Repository When the use case or proposed tool, technique, or method is implemented and open source, provide a link to the repository.

(F13) Context A description of the context surrounding the proposed/used technique.

(F14) Motivation A description of the motivation used by the authors to apply this technique.

(F15) Rationale A description of the rationale used by the authors to contrast the technique to others.

(F16) Technique name The name used by the authors to define the proposed or used technique.

(F17) Technique type In which of the common categories do the technique best fits. E.g., Runtime Verification (aka Runtime Monitoring) [3], Testing [2], Model Checking [7], Theorem Proving, Program Analysis [8], or Simulation.

(F18) Dynamicity Verification, validation and testing techniques might need to execute the system under scrutiny. Techniques that a require running version of the system (or sub-system) are categorized as dynamic, otherwise, they are categorized as static.

(F19) Offline/Online Techniques using the production version of the system under scrutinity during execution are said to be online. Otherwise, they are offline [2, 3].

Data extraction and synthesis is done by manual parsing, identification, classification among the data items defined in the classification scheme. Ultimately we fill a spreadsheet that are available online [3]. By the end of this step, the researchers shall be able to answer all the research questions and provide insights delimiting the baseline for further research.

7 Quality Assurance

To ensure the quality of the research, we follow the checklist:

- Protocol validation.

[3] https://ros-rvft.github.io/replication_package_extras/literature_review.xlsx

- Validation measures in conducted processes.
- All the data is publicly available.

The protocol validation consists of internal validation by peer-reviewing with knowledgeable researchers with different expertise. By internal validation we mean the researchers that helped to conceive the idea of the ongoing study.

As validation measures we include internal validation for the inclusion/exclusion of the papers, the extracted data, and the reported results.

All the data and text will be made available in the format of a replication package.

References

[1] C. Hutchison, M. Zizyte, P. E. Lanigan, D. Guttendorf, M. Wagner, C. Le Goues, and P. Koopman, "Robustness testing of autonomy software," in *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*. IEEE, 2018, pp. 276–285.

[2] A. Bertolino, P. Braione, G. D. Angelis, L. Gazzola, F. Kifetew, L. Mariani, M. Orrù, M. Pezze, R. Pietrantuono, S. Russo *et al.*, "A survey of field-based testing techniques," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–39, 2021.

[3] Y. Falcone, S. Krstić, G. Reger, and D. Traytel, "A taxonomy for classifying runtime verification tools," *International Journal on Software Tools for Technology Transfer*, vol. 23, no. 2, pp. 255–284, 2021.

[4] R. Halder, J. Proença, N. Macedo, and A. Santos, "Formal verification of ros-based robotic applications using timed-automata," in *2017 IEEE/ACM 5th International FME Workshop on Formal Methods in Software Engineering (FormaliSE)*. IEEE, 2017, pp. 44–50.

[5] V. R. B. G. Caldiera and H. D. Rombach, "The goal question metric approach," *Encyclopedia of software engineering*, pp. 528–532, 1994.

[6] I. Malavolta, G. A. Lewis, B. Schmerl, P. Lago, and D. Garlan, "Mining guidelines for architecting robotics software," *Journal of Systems and Software*, vol. 178, p. 110969, 2021.

[7] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[8] F. Nielson, H. R. Nielson, and C. Hankin, *Principles of program analysis*. Springer Science & Business Media, 2004.