Verification of ROS-based Systems: Guidelines for Developers and QA teams

We have synthesized 8 guidelines that materialize the state-of-the-art in verification through dynamic analysis, i.e. runtime verification and field-based testing, targeting ROS-based systems design and assurance argumentation.

With this questionnaire, we intend to collect your impression, as a researcher, on whether each synthesized guideline is clearly expressed and useful. Moreover, we intend to understand whether our proposed set is complete or there are missing concerns.

Involved researchers:
Ricardo Caldas (contact: ricardo.caldas@chalmers.se),
Patrizio Pelliccione,
Genaína Rodrigues

* Indicates required question

Research Interests and Experience

1. Current Research Interest (comma-separated keywords) *

2. Expertise in the Robot Operating System (ROS) *

Mark only one oval.

	1	2	3	4	5	
Inex	◯	◯	◯	◯	◯	Expert

3. Expertise in Runtime Verification *

Mark only one oval.

	1	2	3	4	5	
Inex	◯	◯	◯	◯	◯	Expert

4. Expertise in Field-based Testing *

Mark only one oval.

	1	2	3	4	5	
Inex	◯	◯	◯	◯	◯	Expert

How ROS-based systems should be designed to enable and facilitate runtime verification and field-based testing?

This section introduces four (4) guidelines targeting developers of ROS-based systems.

Guidelines:
(D1). ROS Nodes should be Responsible for One Functional Unit.
(D2). Ensure Global Time Monotonicity of Events/States.
(D3). Specify Properties of Interest
(D4). Understand and Set overhead boundaries

For each guideline, we provide a title, description, strengths, weaknesses and related work.

We invite you to give your opinion on whether you agree, and whether the guideline is clear and useful.

(D1). ROS nodes should be Responsible for One Functional Unit.

Short Description: The design team should implement ROS nodes following the single responsibility principle.

Long Description: ROS fosters a rich toolkit for designing modular robotics software. Developers face the design decision of how to implement a given set of requirement using ROS fundamental concepts (e.g., nodes, topics, services, packages). To promote verifiability, the system's computational units should be designed such that they can be safely tested with warrantied no side effects on the running system. Therefore, the design team should implement ROS nodes following the single responsibility principle. Meaning that each node should be responsible for an indivisible and complete system function. In such a way that it is enough to inspect the inputs and outputs of a single ROS node to analyse a minimal behavior of the system.

Strengths: Fine-grained observation of system requirements; Clear interfaces for runtime inspection; Allows mocking and substituting behavior.

Weaknesses: Feedback loops with the environment; Dependent on requirements specification.

Related Work: In accordance, Hartswell, C. et al., 2019, define components as building blocks that may be hierarchically composed to fulfill the system's functional requirements. From a different stance, Malavolta et. al., 2020, suggests robotics practitioners to design for isolation of feature at package-level.

5. Do you agree with guideline D1 ? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Totally

6. Explain

7. Is guideline D1 clearly expressed? *

Mark only only one oval.

	1	2	3	4	5	
Sen	◯	◯	◯	◯	◯	Clear

8. Explain

9. Is guideline D1 useful? *

Mark only one oval.

	1	2	3	4	5	
Usel	◯	◯	◯	◯	◯	Useful

10. Explain

(D2). Ensure Global Time Monotonicity of Events/States.

Short Description: The development team should ensure that the collection of events/states is globally ordered, in monotonic time.

Long Description: Requirements considering the physical time ask for gaining confidence in face of specifications that are monotonic in time. Which is the case of ROS-based applications. ROS carries out the scheduling and execution of ROS nodes and services, which is transparent to the user (i.e., ROS application developer). Nodes may operate with varying frequencies depending on the need. ROS elements may run distributed in several computers. Hardening the runtime assurance of the robotic behavior. The development team should ensure events/states in monotonic time. Collecting timestamps with precision asks for inline instrumenting ROS nodes. This instrumentation is available during design-time. The robotic developers should enhance the code with globally ordered timestamps for consistent assurance of the robotics behavior.

Strengths: Consistent traces; Avoids incorrect verdicts due to time drift in distributed clocks;

Weaknesses: Computational overhead due to synchronization

Related Work: In accordance, Malavolta et. al., 2020, suggests collecting timestamps from multiple sources to cope with heterogeneous hardware devices

11. Do you agree with guideline D2? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Totally

12. Explain

13. Is guideline D2 clearly expressed? *

Mark only one oval.

	1	2	3	4	5	
Sens	◯	◯	◯	◯	◯	Clear

14. Explain

15. Is guideline D2 useful? *

Mark only one oval.

	1	2	3	4	5	
Usel	◯	◯	◯	◯	◯	Useful

16. Explain

(D3). Specify Properties of Interest

Short Description: The development team should specify properties of interest during design.

Long Description: Gaining confidence on ROS-based systems asks for contrasting the system specification, i.e., properties, against the software behavior. Precise specifications either come from the design process or are created during quality assurance. Manually generating property specifications is error-prone, requiring knowledge of the requirements. The best personnel to specify property specifications are the development team, since they implement the code after the system requirements. The development team should specify properties of interest during design. Acquainted with the system requirements and how they are implemented entails property specification responsibility to the designers. They should define properties in precise notation which will support he quality assurance team to have confidence in their properties.

Strengths: Accuracy, avoids incorrect verdicts due to imprecise property specification;

Weaknesses: May introduce bias in the verification; Maintenance effort to propagating changes to property specifications.

Related Work: -

17. Do you agree with guideline D3 ? *

Mark only one oval.

	1	2	3	4	5	
Not	○	○	○	○	○	Totally

18. Explain

19. Is guideline D3 clearly expressed? *

Mark only one oval.

	1	2	3	4	5	
Sen:	◯	◯	◯	◯	◯	Clear

20. Explain

21. Is guideline D3 useful? *

Mark only one oval.

	1	2	3	4	5	
Usel	◯	◯	◯	◯	◯	Useful

22. Explain

(D4). Understand and Set Overhead Boundaries

Short Description: The development team should set boundaries to acceptable verification overhead.

Long Description: Runtime assurance comes at a computational price, also known as overhead. According to real-time constraints and quality of service expected from the robotics system, the accepted overhead might vary. The variation imposes needs on stricter or more flexible techniques for gaining confidence on the robotic system. Flexibility usually juggles with precision of the assurance arguments and computational resources usage. Such tradeoff is not always elicited by the stakeholders and/or requirements of the system.
The development team should set boundaries to acceptable verification overhead. In opposition to leaving the quality assurance team to blindly seek for minimizing the caused overhead with limited knowledge of the acceptable conditions. The development team is aware of the implementation and requirements including the technical knowledge on performance constraints. Placing the development team in the right position for defining and documenting the acceptable boundaries without incurring sensible depreciation in the quality of service delivery.

Strengths: Provides useful information to the QA team;

Weaknesses: -

Related Work: In accordance, Bertolino A. et al., 2021, reinforces the need to run experiments that measure the verification overhead when isolating components for testing purposes.

23. Do you agree with guideline D4? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Totally

24. Explain

25. Is guideline D4 clearly expressed? *

Mark only one oval.

	1	2	3	4	5	
Sens	◯	◯	◯	◯	◯	Clear

26. Explain

27. Is guideline D4 useful? *

Mark only one oval.

	1	2	3	4	5	
Usel	○	○	○	○	○	Useful

28. Explain

How to perform runtime verification and testing in the field for ROS-based applications?

This section introduces four (4) guidelines targeting quality assurance teams of ROS-based systems.

Guidelines:
(QA1). Check with your Models before going to the Field
(QA2). Use Specification Patterns for Checking domain-specific Behavior
(QA3). Use Standard Libraries for Recording Observations
(QA4). The Doors to Introspection: topics, services, parameters.

For each guideline, we provide a title, description, strengths, weaknesses and related work.

We invite you to give your opinion on whether you agree, and whether the guideline is clear and useful.

(QA1). Check with your Models before going to the Field

Short Description: The quality assurance team should consider using models of the environment and mocking third-party components before testing in the field.

Long Description: ROS applications typically interact with the physical environment and other third-party modules. Gaining confidence on ROS-based applications asks for execution in the field, which is often a costly process due to isolation from side effects, security preservation, controllability, and observability. The quality assurance team should consider using models of the environment and mocking third-party components before testing in the field. Activities such as calibrating sensors and finding parameter values may not need to wait for execution in the field. Lewis T. et. al, for instance, propose the simulation of turbulent plume dispersion scenarios, for setting up parameters for testing the physical robot in the field. A field test with similar setup in the field demands a signification amount of time and resources.

Strengths: Reduced cost. Reuse of parameters.

Weaknesses: Depending on the fidelity of the model, imprecise assurance arguments may cause rework in the field.

Related Work: Malavolta I. et al., 2020, look at the problem from the interfaces between subject under test and environment. They suggest using standardized messaging interfaces for nodes interacting with simulators and hardware devices. Another promising approach, according to the authors, is building on low-fidelity environmental simulations to reproduce bugs found in the field. Guerrero E., et al, 2021, uses StoneFish and the Turbot AUV dynamics for simulation before taking the system to field, including, also, a realistic image in the simulator as ground-truth.

29. Do you agree with guideline QA1 ? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Totally

30. Explain

31. Is guideline QA1 clearly expressed? *

Mark only one oval.

	1	2	3	4	5	
Sens	◯	◯	◯	◯	◯	Clear

32. Explain

33. Is guideline QA1 useful? *

Mark only one oval.

	1	2	3	4	5	
Usel	◯	◯	◯	◯	◯	Useful

34. Explain

____

____

____

____

____

(QA2). Use specification patterns for Checking domain-specific Behavior.

Short Description: The quality assurance team should use specification patterns to precisely describe robotic behavior in ROS.

Long Description: Checking whether a ROS-based application complies with a set of requirements asks for specifying the expected behavior. Specifying robotics behavior with precise formalism, however, is not trivial. Property specification patterns ease the verification of autonomous behavior by providing template specifications to recurring design decisions. The quality assurance team should use specification patterns to precisely describe robotic behavior in ROS. Such specification patterns can be implicit or explicit to the verification. Implicit properties free the QA team from the burden of specification of recurring behavior in ROS. For example, ROS-Immunity internally encodes fuzzing and honeypot for synthesizing fingerprints of attackers of vulnerable ROS applications. Explicit properties empower the QA team enabling the verification unique behavior. To ease the specification process, we recommend using specification patterns tailored to the ROS domain.

Strengths: Reuse; Ease in specifying expected behavior.

Weaknesses: -

Related Work: -

35. Do you agree with guideline QA2? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Totally

36. Explain

37. Is guideline QA2 clearly expressed? *

Mark only one oval.

	1	2	3	4	5	
Sen:	◯	◯	◯	◯	◯	Clear

38. Explain

39. Is guideline QA2 useful? *

Mark only one oval.

	1	2	3	4	5	
Usel	◯	◯	◯	◯	◯	Useful

40. Explain

(QA3). Use Standard Libraries for Recording Observations.

Short Description: The quality assurance team should use standard ROS libraries for recording observations.

Long Description: Gaining confidence on ROS-based applications asks for reproducing, reusing, and replicating scenarios. On the one hand, members of the QA team can reuse known corner cases to find other cases. On the other hand, the design team may closely understand the faults by simulating the execution trace in controlled environment. Both the QA team and the development team benefit from recorded execution traces, and how the observations are recorded matters. The quality assurance team should use standard ROS libraries for recording observations. The standard library for recording observations in ROS is rosbag (http://wiki.ros.org/rosbag). The rosbag package records exchanged messages in so-called bag files and provides means for analyzing and processing such bags, visualizing exchanged messages and reproducing the messages exchange.

Strengths: Reduced need of maintenance;

Weaknesses: -

Related Work: As an example, Beul M. et al., 2017, extensively used bags for recording the state of their unmanned aerial vehicle during field testing keeping rosbag activated even in-between experiments to avoid setting up overhead. The authors kept logging data between trials for performance purposes. When all the trials were done, the authors simply filtered and analyzed the data using in-house tools.

41. Do you agree with guideline QA3? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Totally

42. Explain

43. Is guideline QA3 clearly expressed? *

Mark only one oval.

	1	2	3	4	5	
Sens	◯	◯	◯	◯	◯	Clear

44. Explain

45. Is guideline QA3 useful? *

Mark only one oval.

	1	2	3	4	5	
Usel	◯	◯	◯	◯	◯	Useful

46. Explain

(QA4). The Doors to Introspection: topics, services, parameters.

Short Description: The quality assurance team should use topics, services or parameters to gather observations for assuring ROS-based applications.

Long Description: Gaining confidence on applications based on ROS asks for attesting the behavior of a composite of ROS nodes. Such nodes encapsulate the data and methods for operating robots. There are three types of interfaces between ROS nodes: topics, services, and parameters. Topics are publish-subscribe channels. Services lay on request-response communication. Parameters follow a shared database communication paradigm, similarly to blackboards.The quality assurance team should use topics, services or parameters to gather observations for assuring ROS-based applications.
- How to use topics? Typically, tools offering observation of ROS nodes employ topic-based monitoring. In general, they deploy an extra ROS node subscribing to the topic under observation.
- How to use services? Using services require a node requesting specific data to the node under observation.
- How to use parameters? The parameter-based approach asks for an observer fetching information from the parameter server.

Strengths: Independence from source code (not always available, observable, controllable);

Weaknesses: Whenever the nodes are not implemented as self-contained functional units of software, there might be the need of either inspecting the code for hidden behavior.

Related Work: ROSMonitoring, R. Ferrando, 2020 , for example, employs topic-based monitoring.

47. Do you agree with guideline QA4? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Totally

48. Explain

49. Is guideline QA4 clearly expressed? *

Mark only one oval.

	1	2	3	4	5	
Sen	◯	◯	◯	◯	◯	Clear

50. Explain

51. Is guideline QA4 useful? *

Mark only one oval.

	1	2	3	4	5	
Usel	◯	◯	◯	◯	◯	Useful

52. Explain

Are we missing a Guideline or Recommendation?

In sum, we list the guidelines:

How ROS-based systems should be designed to enable and facilitate runtime verification and field-based testing?

(D1). ROS Nodes should be Responsible for One Functional Unit.
(D2). Ensure Global Time Monotonicity of Events/States.
(D3). Specify Properties of Interest
(D4). Understand and Set overhead boundaries

How to perform runtime verification and testing in the field for ROS-based applications?

(QA1). Check with your Models before going to the Field
(QA2). Use Specification Patterns for Checking domain-specific Behavior
(QA3). Use Standard Libraries for Recording Observations
(QA4). The Doors to Introspection: topics, services, parameters.

53. Given your experience with Runtime Verification, Field-based Testing, and Robotics (ROS).
Do we miss Guidelines or Recommendations? *

Mark only one oval.

- Yes
- No

54. If yes, what else should we recommend to developers or QA teams?

Thanks for contributing with research on Robotics Software Engineering!

If you would like, we will update you with our key insights for later releases of this project.

To this end, **please provide your email contact**.*

*Not required. You can Submit the form without this information.

55. Name

56. Email