Architecting ROS-based Systems - Survey

Dear Participant,

Thank you for considering to participate in this study!

We are a group of researchers from the Vrije Universiteit Amsterdam (The Netherlands), the Carnegie Mellon Software Engineering Institute (USA) and the Institute for Software Research at Carnegie Mellon University (USA). We are currently conducting a study about guidelines for architecting ROS-based systems.

Within this study:

- you will be asked to answer 9 brief questions about your experience developing ROS-based systems
- your answers will only be stored and used in an anonymized form
- to complete the survey it will take you 10-15 minutes

The target respondents for this questionnaire are contributors involved in open-source ROS-based projects.

If you would like to receive the final research report, we will email it to you when all the statistical information is analyzed and conclusions are drawn.

Let's go to the first question!
*Required

Your Experience with ROS

1. How many years of experience do you have developing ROS-based systems? *

_____

2. How many ROS packages have you contributed to in your career? *

Mark only one oval.

- ◯ 1
- ◯ Between 2 and 5
- ◯ Between 6 and 10
- ◯ More than 10

3. What is your primary motivation for working with ROS? *

Mark only one oval.

- ◯ I develop ROS-based systems for a company
- ◯ I develop ROS-based systems as a hobby
- ◯ I develop ROS-based systems for academic purposes
- ◯ Other: _____

Guidelines for Architecting ROS-based Systems

4. In your last ROS-based project, how useful would each of the following 39 guidelines for architecting ROS-based systems have been? *

Mark only one oval per row.

	Absolutely useful	Useful	NOT useful	Absolutely NOT useful	Don't know
G1 - Expose a single ROS node with interfaces for third-party users for the most common use cases					
G2 - Design the system to be as hardware-independent as possible					
G3 - Decouple ROS nodes from variations in the execution environment					
G4 - Identify variation points of the system in advance, and design the system so that it can be extended by third-party users without modifying its core nodes					
G5 - Take into account that the data exchanged between nodes of the system may not be fully compatible (semantically), incorrect, out-of-order, or redundant					
G6 - By design, limit unnecessary computationally-heavy operations by carefully analyzing the execution scenarios of the system					
G7 - Use a dedicated node to store and represent globally-relevant data (e.g., the physical environment where the system operates) and use it as the single source of truth for all the other nodes in the system					
G8 - If context-specific (hardware) configuration is needed, then load it at bringup time					
G9 - Decouple nodes with responsibilities that naturally work at different rates and use different rates for different purposes					
G10 - For real-time requirements, collect timestamps from as many sources as possible (i.e., do not rely on ROS-based timestamps only)					
G11 - Publish empty messages when triggering atomic actions					
G12 - Nodes that potentially produce/consume large amounts of messages should be configurable in terms of their publish/subscribe rates					
G13 - Selectively limit the data exchanged between nodes to provide only the information that is strictly necessary for completing tasks					
G14 - Use different communication channels and different hardware depending on the criticality and real-time needs of the nodes					
G15 - Provide at least one globally-reachable node capable of receiving run-stop messages and stopping/resetting the whole system					
G16 - Nodes interacting with simulation or physical platforms should implement identical ROS messaging interfaces					
G17 - Use standard ROS message formats as much as possible, possibly supporting also their legacy versions					
G18 - Each ROS package should be responsible for one and only one specific feature of the system or robot capability and provide a well-defined interface for it					
G19 - If the system is remotely distributed,					

constantly observe the status of the communication channels, hosts, and machines on the network					
G20 - Nodes with high-frequency operations should be configurable so that they can operate according to available computational resources					
G21 - State estimation nodes should support an arbitrary number and different types of sensors					
G22 - State estimation nodes should be resilient with respect to the amount and frequency of the data received by the sensors					
G23 - Include data- and node-health information in messages containing critical data (e.g., strength of GPS signal)					
G24 - Use services when starting up robots, instead of publishing to topics					
G25 - Provide dedicated nodes for doing introspection and querying the lower levels of the system					
G26 - Systems interacting with other non-ROS systems should provide two types of interfaces: a ROS-independent interface for non-ROS systems and a ROS-based interface for ROS tools (e.g., Rviz, Qt)					
G27 - Avoid persisting raw data (e.g., a full resolution video) if only part of it will be used					
G28 - If different types of data are always sent/received together and must be synchronized, then package them into a single message					
G29 - Each single node should also be runnable in isolation					
G30 - The behavior of each node should follow a well-defined lifecyle, which should be queryable and updatable at run-time					
G31 - If a node is stateful and its behavior strongly depends on the time and order of arrival of messages, specify the protocol of the messages expected by the node					
G32 - When possible, ROS nodes should be stateless and their behaviour should not depend on previous operations or received messages					
G33 - Transform data only when it is used					
G34 - Assign meaningful names to components (e.g., nodes, topics, messages, services) and group them by adopting standard prefixes/suffixes					
G35 - When possible, core algorithms, libraries, and other generic software components should be ROS-agnostic					
G36 - Pay special attention to race conditions when persisting data received from other ROS nodes within the system					
G37 - Use a dedicated node for persisting and querying long-term data and short-term data (e.g., in the order of seconds)					
G38 - ROS nodes should be agnostic of the underlying communication mechanisms (e.g., network protocols, deployment topology, etc.)					
G39 - Group nodes and interfaces into cohesive sets, each of them with its own					

responsibilities and well-defined dependencies

5. From your experience, are there any guidelines in addition to those in the previous question that ROS-based developers should know?

Quality Requirements for ROS-based Systems

6. What are the top-3 quality requirements you considered when working on your last ROS-based system? *

Examples of quality requirements include (in no specific order): performance, compatibility, usability, reliability, security, maintainability, portability, energy efficiency, safety, and any other quality requirement you think is important in the context of ROS-based systems.

Architecture documentation in your repository

7. It appears that there is no software architecture documentation in the repository you contributed to. Was it recorded somewhere else, or is there some other reason? *

Submit your answers

Thank you! Please, do not forget to click on the Submit button at the bottom of this page!

8. Do you have any final comments or suggestions?

9. Your e-mail address

Optional, we will use it only once for sending the results of our study

This research is carried out jointly by the Vrije Universiteit Amsterdam (The Netherlands), the Carnegie Mellon Software Engineering Institute (USA), and the Carnegie Mellon University School of Computer Science (USA).

This is the list of the investigators involved in this study, feel free to contact us for any question, comment, or discussion.
- Ivano Malavolta (i.malavolta@vu.nl), Vrije Universiteit Amsterdam
- Grace Lewis, Carnegie Mellon Software Engineering Institute
- Bradley Schmerl, Carnegie Mellon University School of Computer Science
- Patricia Lago, Vrije Universiteit Amsterdam
- David Garlan, Carnegie Mellon University School of Computer Science