MARVer: Robotik Sistemlerin Emniyetini Doğrulama Aracı
MARVer: A Tool for Verification of Robotic System's Safety

Zekeriyya Demirci[1], Metin Ozkan[2], M. Talha Sahin[2], H. Can Ergun[2], Ahmet Yazici[2]

[1]Yazılım Mühendisliği Bölümü
Osmangazi Üniversitesi, Eskişehir
zekeriyya.demirci@ogu.edu.tr

[2]Bilgisayar Mühendisliği Bölümü
Osmangazi Üniversitesi, Eskişehir
meozkan@ogu.edu.tr

Özetçe

Robotik sistem geliştiricileri genellikle geliştirilen sistemlerin emniyet doğrulamasını deneyimlerine dayalı olarak test yöntemleri ile gerçekleştirmektedir. Ancak robotik sistemler üretim süreçlerini daha esnek hale getirme ihtiyacından dolayı daha otonom ve karmaşık hale geldi. Bu durumda robotik sistemlerin emniyet doğrulamasını daha güvenilir, insan faktörüne en az bağımlı ve daha az iş yüküne sahip hale getirmek için resmi ve otomatik doğrulama araçlarına ihtiyaç duyulmaktadır. Bu motivasyonla bu çalışmada robotik sistemler için emniyet doğrulaması için geliştirilmiş bir yazılım aracı sunmaktadır. Model Destekli Çalışma Anı Doğrulama Aracı (MARVer), model doğrulama ve çalışma anı doğrulama yöntemlerini bir arada kullanarak doğrulama sürecini daha resmi, öngörülebilir ve tekrarlanabilir hale getirmenin yanı sıra doğrulama kapsamını artırır. (ROKOS) kalite kontrolü için otomatik robotik denetim hücresi üzerinde bir deney yapıldı.

Abstract

Robotic system developers generally perform the safety verification of the developed systems with testing methods based on their experience. However, robotic systems have become more autonomous and complex, with the need to make production processes more flexible. In this case, there is a need for formal and automated verification tools to make the safety verification of the robotic systems more reliable, least dependent on the human factor, and less workload. With this motivation, this paper presents a tool developed for safety verification for robotic systems. The Model Aided Runtime Verification Tool (MARVer) employs model-checking and runtime verification methods combinatory making the verification process more formal, predictable, and repeatable manner as well as increasing the scope of verification. An experiment was conducted on an automated robotic inspection cell for the quality control of automotive body-in-white (ROKOS).

1. Introduction

The increasingly widespread use of robotic systems causes us to encounter these systems more often at different points, such as medical services, unmanned vehicles, and especially industrial production [1], [2]. In particular, these systems, which are equipped with sensors and have autonomous features, have the ability to perceive their environment, make decision and implement it. Having all these capabilities makes robotic systems more complex than ever before. Furthermore, robotic systems share their workspace with mobile entities such as humans or other autonomous systems. Therefore, an error that may occur for any reason can cause irreversible and even fatal accidents. For this reason, very intensive testing processes take place in transforming safety-critical systems into products. For a system to be highly safe, all possible scenarios must be considered in the verification and validation (V&V) processes. However, traditional test methods are not efficient in the verification processes of complex systems such as robotics. Instead, V&V methods such as formal verification should be implemented, which ensures that properties will be provided as long as the system is properly modeled.

Formal verification is a technique in which the system is modeled mathematically and checks whether the given properties are met [3, 4]. [5] offers state of the art in formal verification for autonomous robotic systems by analyzing the challenges. They categorize formal verification under three main titles: model checking, theorem proving, and runtime verification. Model-checking is a widely used formal verification method [6]. Using formalism methods such as timed automata or Petri nets, systems with distinct characteristics such as concurrent, probabilistic, or timed systems can be modeled by verifying whether it meets the desired properties utilizing the model-checking method [5]. The desired properties are expressed as logical formulas such as temporal logic and verified on the model by considering all possible system traces [6]. On the other hand, runtime verification is a lightweight dynamic analysis method that observes whether the systems meet the given specifications [7, 8]. While runtime verification verifies systems at runtime, model checking verifies systems before deployment [1].

The software development life cycle (SDLC) consists of many steps where different V&V methods are applied. Static verification methods such as model checking are applied at the early stage of SDLC, while dynamic verification methods such as runtime verification (RV) are applied at runtime when the system is operational. Performing more than one verification method together on a system can increase the scope of verification as well as add a more formal structure to the process. Using different verification methods in combination, especially in developing quite complex systems such as robotics, can make the system more reliable against possible errors.

This study aims to perform a comprehensive verification process for robotic systems that bring multiple V&V methods together. For this purpose, a Model Aided Runtime Verification Tool (MARVer) is proposed for ROS-compatible robotic systems that utilize model checking and runtime verification methods. MARVer is basically constructed on the ROSMonitoring [9], an open-source runtime verification tool for ROS-compliant systems, and the UPPAAL [10], a well-known model checking tool for real-time systems. MARVer can process an UPPAAL model file, convert queries to a suitable temporal logic (TL) format that the verifier can manage, and perform a runtime verification on the system. MARVer allows the entire verification process to be easily managed through an interface. Thus, it acts as a bridge between users and the complex verification process while making it easier to manage. Moreover, applying a transformation prevents the rewriting of the properties to be verified for each method. In addition, the safety of robotic systems is not approved just by inspecting the robotic system's software. For instance, a collision is the most critical safety issue in robotic systems. Generally, the software of the robotic system does not provide data about the distances between the robot and obstacles. MARVer includes add-ons to get data like distances not provided by the system software by supporting additional data sources. Thus, it reduces the need for domain-specific knowledge to build and execute the verification process.

The rest of the paper is organized as follows. First, the architecture of the MARVer tool is presented in Section 2. Then, the application is conducted, and the results are given in Section 3. Finally, the conclusion of the study is given in Section 4.

2. Verification Tools for Robotic Systems

Although various approaches have been published for verifying and validating robotic systems, most have focused on a specific system. Instead, there is a need for more comprehensive methods and tools that can be used for most robotic systems. While there is an intention to combine different methods and tools to achieve more automated, flexible, and repeatable approaches, formal solutions with a wide range of uses are needed. With the combination of two verification techniques, model checking with UPPAAL and CoFI model-based testing with ConData, [11] present ConTEA, a software tool for automatically connecting UPPAAL to ConData, and discuss their contribution to industrial software development. [12] offer a model-based approach including modeling, verification, and automatic code generation for ROS systems. UPPAAL, a model checker, is used as a verifier and generates C++ code from timed automata. They carry out a case study to verify the process of grasping and moving a cup by a robot. In order to achieve an automated model-based testing process for multi-robot systems, [13] propose a testing tool, namely TestIt. It is designed to work with many tools used in the test development process. They present a use case using the UPPAAL family tool. To advance the safety and security of ROS-compliment robotics systems, [14] offer ROSRV, a lightweight runtime verification framework. It is based on ideas that track the system execution by capturing messages, modifying them if necessary, and performing user-defined actions. [15] offer an approach to testing real-time systems, integrating model-based testing with ROS-based systems. They model the system in UPPAAL and develop an adapter and interface between DTRON and ROS. They demonstrate the feasibility of the studied approach by applying a case study navigating an autonomous platform simulation in a limited space to reach the target point.

3. MARVer Architecture

MARVer is a model-aided runtime verification tool developed to verify ROS-compatible systems, offering an extended verification process utilizing ROSMonitoring [9] and UPPAAL [10] tools. Essentially it imports an UPPAAL model file, extracts the queries used for verifying the model-by-model checking, and utilizes them in the runtime verification process. MARVer automatically converts the specifications written in UPPAAL query language into suitable format that ROSMonitoring accepts. Thus, it saves users the hassle of rewriting queries and does not require expert knowledge. Thanks to the flexible structure of MARVer, users can easily integrate their systems into MARVer through ROS, along with the services they have written.

MARVer provides a graphical user interface (GUI) with which you can manage the entire verification process. This interface makes it possible to view the UPPAAL model of the system, prepare the necessary configuration file for the verification process, specify the properties to be verified, and monitor the runtime verification process. The overall architecture of MARVer is shown in Figure 1. The components indicated by the demo arrow were used in the experiment.

3.1. Toolchain

MARVer utilizes many tools as a toolchain: ROS, ROSMonitoring, Gazebo, SRVT, and UPPAAL. ROS is an open-source, meta-robot operating system that provides structured communication on top of operating systems[16]. Communication is mainly based on the publisher/subscriber structure, and messages are announced through topics.

ROSMonitoring [9] is an open-source runtime verification tool for ROS-compatible systems. It subscribes to topics published over ROS and verifies the system by checking incoming messages for violations against formally specified properties. As a result, it broadcasts a message over ROS indicating whether the violation has occurred. UPPAAL is a model checker in which the system is modeled as a network of timed automata, and properties are expressed as a timed computation tree logic (TCTL) [17]. It is capable of modeling, simulating, and verifying systems. It uses queries written in UPPAAL query language to verify systems. Gazebo [18] brings a fresh approach to simulation with a complete toolbox of development libraries and cloud services to make 3D realistic simulation easy. Iterate fast on new physical designs in realistic environments with high-fidelity sensor streams.

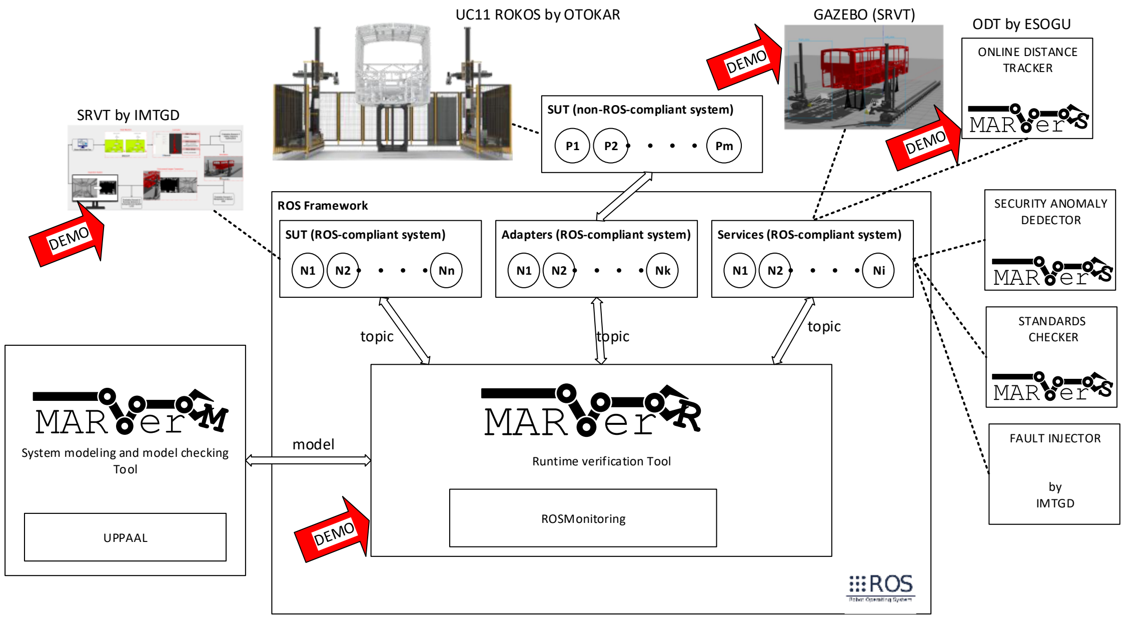


Figure 1 MARVer architecture

SRVT [19] is the simulation-based robot verification testing tool (SRVT). SRVT transfers ROKOS to the GAZEBO simulation environment for the V&V of the system. Simulation environment using Gazebo, trajectory planning using Moveit [20], and task management using ROS Smach [21] package are built in a single ROS package.

3.2. Architecture

MARVer is mainly composed of three components: MARVer-M, MARVer-R, and MARVer-S. The belief explanations of these components are given below.

3.2.1. MARVer-M

MARVER-M supports the modeling phase for the verification of the system. This component has already been under development, and the main focus is to facilitate the modeling of the systems. Model checker tools generally save the model specifications into a structured file. Similarly, UPPAAL produces an XML file including all entities in the model as well as queries used to be verified. MARVer-M is where the UPPAAL model is previewed, and database actions (import or export) are realized. It can similarly display an UPPAAL model from a local repository or database on the interface. Thus, it is aimed for users to get a more straightforward idea about the system. On the other hand, MARVer-M allows the designed models to be stored in the database.

3.2.2. MARVer-R

MARVer-R is utilized to manage the whole runtime verification process. It facilitates the configuration, instrumentation, specification, monitoring, and verification. There are three steps to implement verification:

(i) Configuration

ROSMonitoring automatically generates monitors through a configuration file shown in Table 1. MARVer facilitates the construction of the configuration file by using the model description of the system. In fact, a configuration file is a YAML file that contains detailed arrangements for the ROS-compatible system including the number of monitors to be generated. Besides, for each monitor the following information is required: the topics to be intercepted and their details such as the name of the topic, the message type and the type of actions, node-related information such as the name of the node and the path of the launch file to be remapped.

Table 1 Configuration file example

```
nodes: # list of nodes to monitor
  - node:
      name:
      package:
      path:
monitors: # list of monitors to generate
  - monitor:
      id:
      log:
      silent:
      warning:
      oracle:
        port:
        url:
        action
      topics: # list of topics monitor will intercept
        - name:
        type:
        action:
        publishers:
          -  (node name)
```

Monitors establish the communication via ROS with the system and send the content of ROS messages to the verifier. MARVer-R enables to specify of whole configurations via the interface. It's also possible to save a config file or import an existing config file. Once a config file is imported, the content can be updated.

(ii) Property specifications

Properties are sentences formally written in temporal logic to indicate the system's expected behavior. MARVer verifies the systems against whether the properties are violated or not. These properties must be determined in Reelay format [22] before starting the verification process. To do this, MARVer offers two ways. First, properties can be manually specified via the interface. It requires expertise in the process of expressing the properties in Reelay format. In a second way, the properties already expressed in the modeling are used. MARVer is capable of converting UPPAAL queries to Reelay format that the verifier can understand. To do this, MARVer asks for a verified UPPAAL model file including the queries. After the property is specified, it can be saved into a JSON file for later use.

(iii) Runtime verification

Before the runtime verification process a configuration file, a property file, and your ROS workspace path have to be specified. Then MARVer generates monitors and relocates them into the ROS workspace. Once everything is ready, the runtime verification process can be started. MARVer automatically runs the monitor and starts to track the system over ROS topics. The monitor gets the message and transmits the content to the verifier. The verifier checks for any violations and emits a verdict. MARVer provides an interface to observe the verification results at runtime. Moreover, it logs the results into a log file for further analysis.

3.2.3. MARVer-S

The safety of robotic systems is not verified just by inspecting the system's software. The most critical safety issue in robotic systems is a collision. Generally, the software of the robotic system does not provide data about the distances between the robot and obstacles. MARVer includes add-ons to get data like distances not provided by the system software. MARVer-S represents the add-ons that provide the data needed to verify the robotic systems. MARVer is an open-source, extensible tool to add your services to MARVer-S. In this study, an online distance tracker (ODT) included in MARVer-S is employed to verify the robotic systems.

4. Applications

An experiment was conducted to verify ROKOS, an automated robotic inspection cell for quality control of automotive body-in-white. The experiments are implemented by using the Simulation-Based Robot Verification Tool (SRVT) [19]. This tool simulates a robot and a bus skeleton in the Gazebo simulation environment as shown in Figure 2. It is constructed on ROS architecture and communicates with other system components over topics by publishing messages including the current task ID.

Experimental verification was realized to verify whether a given safety distance threshold is exceeded while the robot performs its task. The safety distance threshold refers to the

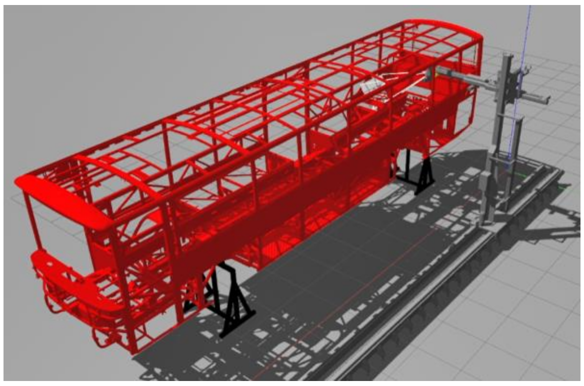


Figure 2 Gazebo simulation environment of SRVT

the distance that must be maintained between the links of the robot and the bus skeleton. There are six distances between the links of the robot and the bus skeleton, but the violation is checked by looking at the minimum value among them. In this experiment, the value of the safety distance threshold is used as 0.2 meters. In order to get distance values during the robot is operating, an additional service is employed: Online Distance Tracker (ODT). It is responsible for publishing a message over ROS, including the minimum distance between the robot and the bus skeleton.

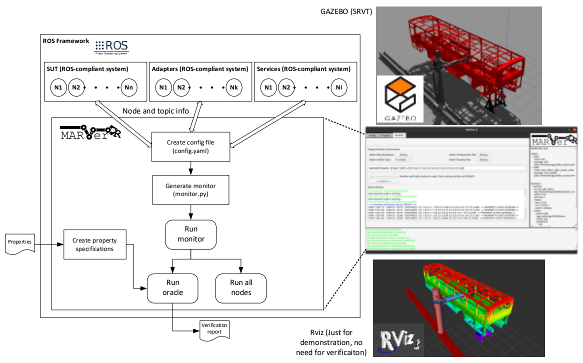

GAZEBO (SRVT)
ROS Framework
ROS
SUT (ROS-compliant system)
Adapters (ROS-compliant system)
Services (ROS-compliant system)
N1
N2
Nn
Nk
Ni
Node and topic info
MARVer
Create config file (config.yaml)
Generate monitor (monitor.py)
Run monitor
Properties
Create property specifications
Run oracle
Run all nodes
Verification report
GAZEBO
RViz
Rviz (Just for demonstration, no need for verificaiton)

Figure 3 Demonstration For Experiment

Figure 3 shows some demonstration of tool during the experimentthe experiment's overall architecture, including the additional services. MARVer subscribes to topics published by SRVT and ODT, tracks the system execution at runtime, and emits a verdict on whether the system violates the given properties. For this purpose, the configuration file was filled with the information about SRVT and ODT nodes which publishes task ID and distance values, respectively. Then, the properties already verified by the UPPAAL model checker on the system model were used.

The property R1 is expressed in the UPPAAL query language as given below. MARVer-R converts it into Reelay format. While the ODT generates the distance values, the threshold value is also 0.2 meters.

R1:

Does an error occur while the robot is performing the task with ID 2?

UPPAAL query language:

E<> distance<threshold && taskID == 2

Reelay format:

{distance < threshold and taskID == 2}

Once the experiment is started, MARVer generates the monitor to intercept the topics specified in the configuration file. Then the monitor begins to track the system through topics and send messages to the verifier. The properties are verified for each message individually and the verifier produces a result.

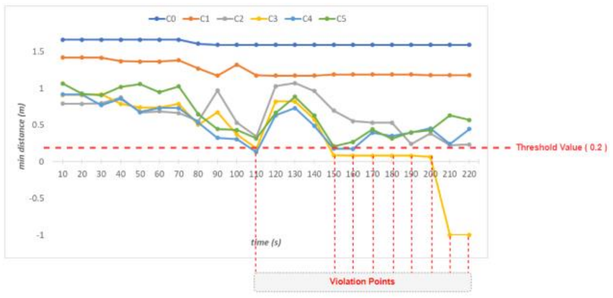
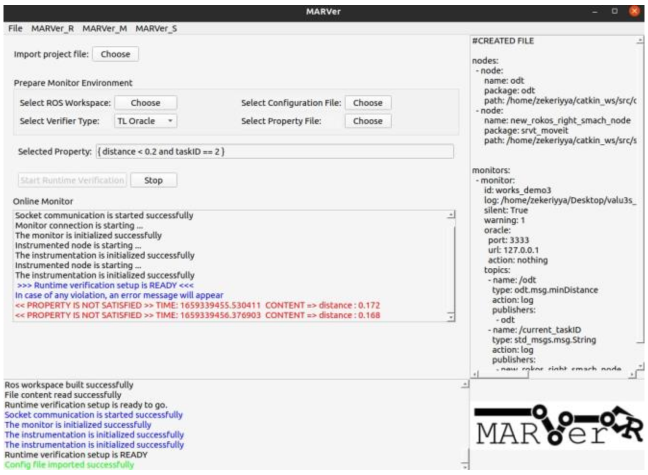


Figure 4 Obtained distance values

The values published by the ODT for six minimum distances of the robot's links during the experiment are visualized in Figure 4. MARVer checks for violation by looking at the smallest of these values at that moment. Values below the threshold value cause a violation. If a violation occurs, MARVer displays a warning with red color on the user screen.



```
Import project file: Choose

Prepare Monitor Environment
Select ROS Workspace: Choose          Select Configuration File: Choose
Select Verifier Type: TL Oracle       Select Property File: Choose

Selected Property: { distance < 0.2 and taskID == 2 }

Online Monitor
Socket communication is started successfully
Monitor connection is starting ...
The monitor is initialized successfully
Instrumented node is starting ...
The instrumentation is initialized successfully
Instrumented node is starting ...
The instrumentation is initialized successfully
 >>> Runtime verification setup is READY <<<
In case of any violation, an error message will appear
<< PROPERTY IS NOT SATISFIED >> TIME: 1659339455.530411  CONTENT => distance : 0.172
<< PROPERTY IS NOT SATISFIED >> TIME: 1659339456.376903  CONTENT => distance : 0.168

Ros workspace built successfully
File content read successfully
Runtime verification setup is ready to go.
Socket communication is started successfully
The monitor is initialized successfully
The instrumentation is initialized successfully
The instrumentation is initialized successfully
Runtime verification setup is READY
Config file imported successfully
```

```
#CREATED FILE

nodes:
 - node:
    name: odt
    package: odt
    path: /home/zekeriyya/catkin_ws/src/c
 - node:
    name: new_rokos_right_smach_node
    package: srvt_moveit
    path: /home/zekeriyya/catkin_ws/src/s


monitors:
 - monitor:
    id: works_demo3
    log: /home/zekeriyya/Desktop/valu3s_
    silent: True
    warning: 1
    oracle:
     port: 3333
     url: 127.0.0.1
     action: nothing
    topics:
     - name: /odt
      type: odt.msg.minDistance
      action: log
      publishers:
       - odt
     - name: /current_taskID
      type: std_msgs.msg.String
      action: log
      publishers:
```

Figure 5 Runtime verification screen of MARVer-R

Users are able to see the ROS-time that the violation occurred as well as the content of the message that cause the violation via MARVer interface as shown in Figure 5. MARVer saves each message into a log file until the verification process is completed. If necessary, more detailed information can be obtained by looking at the log file where all the results are recorded by matching ROS-time.

5. Conclusion

The model-aided runtime verification tool (MARVer) for ROS-compatible systems was proposed to eliminate the safety concerns that arise especially in systems such as robotics, which are becoming increasingly complex. MARVer offers a verification process using two formal verification methods: model checking, and runtime verification. In this way, besides a more comprehensive verification process, it is aimed to make the process a formal structure. Thanks to its flexible structure, MARVer can be easily integrated into ROS-compatible systems. In addition, MARVer, which has an interface for users, makes the verification process easier, especially compared to existing tools. An experiment was conducted using the SRVT tool to verify ROKOS, an approach to checking the quality of systems using robots. In the experiment, it was verified whether the robots exceeded the minimum distance during their movement and the results were evaluated.

Acknowledgement

This work was supported by theVALU3S project that has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union’s Horizon 2020 research and innovation program and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey.

The views expressed in this work are the sole responsibility of the authors and do not necessarily reflect the views or position of the European Commission. The authors, the VALU3S Consortium, and the ECSEL JU are not responsible for the use which might be made of the information contained in here.

This work is supported by the Scientific and Technical Research Council of Turkey (TUBITAK), Contract No 120N800, project title: “Verification and Validation of Automated Systems' Safety and Security ".

References

[1] C. Hu, W. Dong, Y. Yang, H. Shi, and G. Zhou, “Runtime Verification on Hierarchical Properties of ROS-Based Robot Swarms,” IEEE Transactions on Reliability, vol. 69, no. 2, pp. 674–689, 2020, doi: 10.1109/TR.2019.2923681.

[2] R. Wang et al., “From Offline Towards Real-Time Verification for Robot Systems,” IEEE Transactions on Industrial Informatics, vol. 14, no. 4, pp. 1712–1721, 2018, doi: 10.1109/TII.2017.2788901.

[3] E. Gjondrekaj et al., Towards a Formal Verification Methodology for Collective Robotic Systems. 2012. doi: 10.1007/978-3-642-34281-3_7.

[4] F. Ingrand, “Recent Trends in Formal Validation and Verification of Autonomous Robots Software,” in 2019 Third IEEE International Conference on Robotic Computing (IRC), 2019, pp. 321–328. doi: 10.1109/IRC.2019.00059.

[5] M. Luckcuck, M. Farrell, L. Dennis, C. Dixon, and M. Fisher, “Formal Specification and Verification of Autonomous Robotic Systems: A Survey,” ACM Computing Surveys, vol. 52, pp. 1–41, 2019, doi: 10.1145/3342355.

[6] A. A. Tolga Ovatman Davut Polat, Ali Osman Ünver, "An overview of model checking practices on verification of PLC software," Software & Systems Modeling, vol. 15, pp. pages937–960, 2016.

[7] S. Pinisetty, T. Jéron, S. Tripakis, Y. Falcone, H. Marchand, and V. Preoteasa, "Predictive runtime verification of timed properties," Journal of Systems and Software, vol. 132, pp. 353–365, 2017, doi: https://doi.org/10.1016/j.jss.2017.06.060.

[8] C. Sánchez et al., "A survey of challenges for runtime verification from advanced application domains (beyond software)," vol. 54, no. 3, pp. 279–335, 2019.

[9] A. Ferrando, R. C. Cardoso, M. Fisher, D. Ancona, L. Franceschini, and V. Mascardi, "ROSMonitoring: A Runtime Verification Framework for ROS," in Towards Autonomous Robotic Systems, 2020, pp. 387–399.

[10] G. Behrmann, A. David, and K. G. Larsen, "A Tutorial on Uppaal," in Formal Methods for the Design of Real-Time Systems, 2004, pp. 200–236. doi: 10.1007/978-3-540-30080-9_7.

[11] E. Villani, R. P. Pontes, G. K. Coracini, and A. M. Ambrósio, "Integrating model checking and model based testing for industrial software development," Computers in Industry, vol. 104, pp. 88–102, 2019, doi: https://doi.org/10.1016/j.compind.2018.08.003.

[12] R. Wang et al., "A Formal Model-Based Design Method for Robotic Systems," IEEE Systems Journal, vol. PP, pp. 1–12, 2018, doi: 10.1109/JSYST.2018.2867285.

[13] G. Kanter and J. Vain, "Model-based testing of autonomous robots using TestIt," Journal of Reliable Intelligent Environments, vol. 6, no. 1, pp. 15–30, 2020, doi: 10.1007/s40860-019-00095-w.

[14] C. E. Jeff Huang Yi Zhang, Brandon Moore, Qingzhou Luo, Aravind Sundaresan, Grigore Rosu, "ROSRV: Runtime Verification for Robots," International Conference on Runtime Verification, vol. 8734. Springer, pp. 247–254, 2014.

[15] Artur Gummel, "Model-Based Testing With TestIt: The Robot Operating System Case Study," TALLINN UNIVERSITY OF TECHNOLOGY, Tallinn, 2018.

[16] M. Quigley et al., "ROS: an open-source Robot Operating System," in ICRA workshop on open source software, 2009, vol. 3, no. 3.2, p. 5.

[17] M. Iftikhar and D. Weyns, "A Case Study on Formal Verification of Self-Adaptive Behaviors in a Decentralized System," Electronic Proceedings in Theoretical Computer Science, vol. 91, 2012, doi: 10.4204/EPTCS.91.4.

[18] "Gazebo," Jul. 18, 2022. https://gazebosim.org/ (accessed Jul. 18, 2022).

[19] E. A. Yayan U., "Endüstriyel Robot Hareket Planlama Algoritmaları Performans Karşılaştırması," Journal of Science, Technology and Engineering Research, pp. 31–45, 2021.

[20] "MoveIt," Jul. 18, 2022. https://moveit.ros.org/ (accessed Jul. 18, 2022).

[21] "ROS Smach," Jul. 18, 2022. http://wiki.ros.org/smach (accessed Jul. 18, 2022).

[22] D. Ulus, "Online Monitoring of Metric Temporal Logic using Sequential Networks," Jan. 2019, Accessed: Jul. 01, 2022. [Online]. Available: http://arxiv.org/abs/1901.00175