



CHALMERS
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

SmartTestudo: Adaptive Monitoring for Cyber-Physical Systems

A proof of concept applied in an industrial environment simulation with TurtleBot3

Master's Thesis in Software Engineering and Management

Andrea Riccardi

MASTER'S THESIS 2025

SmartTestudo: Adaptive Monitoring for Cyber-Physical Systems

A proof of concept applied in an industrial environment simulation
with TurtleBot3

Andrea Riccardi



UNIVERSITY OF
GOTHENBURG



CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

SmartTestudo: Adaptive Monitoring for Cyber-Physical Systems
A proof of concept applied in an industrial environment simulation with TurtleBot3
Andrea Riccardi

© Andrea Riccardi, 2025.

Supervisor: Rebekka Wohlrab, Department of Computer Science and Engineering
Examiner: Chih-Hong Cheng, Department of Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in L^AT_EX
Gothenburg, Sweden 2025

Andrea Riccardi
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

With the advent of Industry 4.0, Cyber-Physical Systems (CPS) have started to become more adopted in different kind of industries, increasing productivity and efficiency. This rapid adoption has led various companies to develop their in-house solutions to stay competitive and try to dominate the market in selling robotic systems. This division in different ways of building CPS led to standardization problems and made making new innovations and discoveries harder. To solve this issue of a lack of a framework and solid bases, two researchers developed the Robot Operating System (ROS) to have a common baseline to develop CPSs. With new advancement in hardware technologies ROS had it's bottlenecks, so a new version called ROS2 was developed. This brought new possibilities to improve CPSs and overall productivity.

As CPS became more adopted, new problems started to arise, mainly regarding safety, maintenance and monitoring. Each CPS produced big amounts of data that isn't always useful and risks to clutter the system and robot performance.

Solutions to this issue have been proposed in existing research, and the objective of this research is to analyze the existing literature to identify potential limitations and issues with the current state of the art, and propose an innovative way of monitoring CPS with a technique called adaptive monitoring.

To achieve this objective, a design science approach has been devised, with three iteration, one regarding the analysis and the second two regarding the development and verification of the artifact. After each iterations results are presented and discussed, with the artifact improved.

The proposed artifact consists in an adaptive monitoring system which inner workings are based on the MAPE-K loop, where the knowledge is specified by the user with the use of a Domain Specific Language (DSL) and the values of the frequency of each topic is optimized and adjusted with the adoption of the Quality Performance Model (QUPER).

The findings of this research highlight the limitations faced by existing literature and how the current artifact approach comes in aid.

Keywords: Adaptive monitoring, CPS, TurtleBot3, ROS, ROS2, DSL, MAPE-K, QUPER

Acknowledgements

This thesis has truly been a rollercoaster of emotions, with people joining and leaving the ride along the way. I am incredibly grateful to all those who have been part of this journey whether for a moment or for the long haul [67].

First and foremost, I would like to thank my family for their unwavering support throughout this entire experience abroad. I can't find the words to express how thankful and grateful I am.

To my close friends, a big thank you for always being there, for the constant support, for all the laughs, for all the Italian dinners and for the amazing moments shared throughout this journey.

A special thanks to my company, Freeloop, and my great colleagues, for allowing me to continue gaining valuable experience while working alongside such an amazing team. Your flexibility, understanding, and encouragement have meant a lot. I'm also especially grateful that some of you came to visit and took part in running the half-marathon.

To all the new friends made during this experience.

Last but certainly not least, I would like to express my sincere gratitude to my supervisor, Rebekka Wohlrab. Even while navigating a significant and special chapter in her life, she was always available, supportive, and incredibly dedicated. Her guidance and encouragement were invaluable throughout this journey. I wholeheartedly wish her all the best in this new chapter of life.

Andrea Riccardi, Gothenburg, May 2025

Contents

List of Figures	xiv
List of Tables	xv
List of Acronyms	xvii
1 Introduction	1
1.1 History of Cyber-Physical Systems	1
1.2 Monitoring Systems	2
1.3 Problem Description	2
1.4 Purpose of the Study	3
1.5 Research Questions	4
1.6 Motivating Example	5
1.7 Significance of the Study	6
1.7.1 Implications for Research	6
1.7.2 Implications for Practice	6
1.8 AI Use Statement	6
1.9 Thesis Outline	7
2 Background	9
2.1 Cyber-Physical Systems	9
2.2 Adaptive Monitoring	9
2.3 ROS and ROS2	10
2.4 TurtleBot3	11
2.5 TurtleBot3 Topics	11
2.6 Gazebo	13
2.7 RViz	13
2.8 Rqt	14
2.9 Rosbag2	15
2.10 DSL	16
2.11 Mathematical Optimization	17
3 Related Work	19
3.1 ROS/ROS2 in Industry	19
3.2 Adaptive Monitoring in Software Engineering	20
3.3 Challenges in Monitoring CPS	21
3.4 Monitoring Techniques Analysis	21

3.5	MAPE-K	23
3.6	QUPER Model	25
3.7	Evaluating the cost-benefit tradeoff of monitoring	26
4	Research Methodology	27
4.1	Research Design	27
4.1.1	Design Science	27
4.2	Iteration Plan	28
4.3	Iteration 1	29
4.3.1	Data Collection	29
4.3.2	Goals and Data Analysis	29
4.4	Iteration 2	29
4.4.1	Data Collection	29
4.4.2	Goals and Data Analysis	30
4.5	Iteration 3	30
4.5.1	Data Collection	30
4.5.2	Goals and Data Analysis	30
5	Problem Identification	31
5.1	Process	31
5.2	Data Collection	31
5.3	Data Analysis	33
5.3.1	Papers analysis	33
5.4	Common Issues Identified	36
5.5	Artifact's Requirements	38
5.5.1	Stakeholders	38
5.5.2	Functional Requirements	39
5.5.3	Non-Functional Requirements	40
5.5.4	Prioritization	41
5.5.5	Requirements considerations	42
6	SmartTestudo	43
6.1	Adaptive Monitoring Strategy	44
6.2	ADIRF Ruler	46
6.2.1	TextX ADIRF Model	47
6.2.2	ADIRF Rule Tool GUI	49
6.2.3	ADIRF Rules use case examples	51
6.2.4	OBI Bandwidth Optimizer	53
6.2.5	Consideration from mathematical models	57
6.3	SmartTestudo	57
6.3.1	Setup and Launch of SmartTestudo Package	59
6.4	Rosbag visualizer	60
6.5	Artifact Code Repositories	61
7	Findings	63
7.1	Data collection from simulator	63
7.1.1	SmartTestudo configuration	63

7.1.2	Warehouse environment	65
7.1.3	Data collection	65
7.2	Simulations Findings	66
7.2.1	Data analysis from simulations	66
7.2.1.1	Topics with Dynamic Frequency Adaptation	67
7.2.1.2	Topics with Static Frequency	69
7.2.1.3	Frequency analysis table	70
7.2.1.4	Bandwidth analysis table	70
7.2.2	Consideration of the Artifact from the Data	71
8	Discussion	73
8.1	Research questions	73
8.1.1	RQ1: What are the current limitations in monitoring in ROS2 based CPS?	73
8.1.2	RQ2: What potential solutions could mitigate the limitations of current monitoring approaches in ROS2 based CPS?	75
8.1.3	RQ3: To what extent can the current limitations of monitoring approaches in ROS2 based CPS be solved by the potential solutions?	78
8.2	Limitations	81
8.2.1	Threats to internal validity	81
8.2.2	Threats to external validity	81
8.2.3	Threats to construct validity	82
8.2.4	Delimitations	82
9	Conclusion	85
9.1	Takeaways	85
9.2	Future Work	85
	Bibliography	87
A	Appendix	I
A.1	ReqGenie requirement elicitation	I
A.2	ADIRF Ruler Tool Screenshots	I
A.3	Findings Graphs	III

List of Figures

2.1	TurtleBot3 platforms. Figure reused from [74].	11
2.2	Gazebo with the industrial warehouse scenario.	13
2.3	RViz with the industrial warehouse scenario.	14
2.4	Rqt topic viewed and camera view.	15
2.5	Graph made with rqt_graph.	15
2.6	Rosbag2 info on a bag from the industrial warehouse scenario.	16
2.7	Example of DSL schema using TextX. [20]	17
2.8	Example of DSL rules using TextX. [20]	17
3.1	MAPE-K Architecture diagram. Image adapted from [3]	24
3.2	The QUPER benefit view. Modelled from [84].	26
4.1	Iteration plan overview.	28
6.1	Roman legion in testudo formation.	43
6.2	Overall artifact architecture.	44
6.3	Artifact architecture - ADIRF Ruler	47
6.4	DSL ADIRF Model tree.	47
6.5	Default rule block	50
6.6	Cnditional rule block	50
6.7	ADIRF Ruler - GUI Overview	51
6.8	SmartTestudo Architecture	57
6.9	Rosbag Vizualizer Architecture	60
7.1	Warehouse environment with security zones highlighted.	65
7.2	TurtleBot3 data collection path.	66
7.3	/cmd_vel Topic Frequency Analysis.	68
7.4	/cmd_vel Bandwidth Frequency Analysis.	68
7.5	/odom Topic Frequency Analysis.	68
7.6	/odom Topic Bandwidth Analysis.	69
7.7	/camera/image_raw Topic Frequency Analysis.	69
7.8	/camera/image_raw Topic Bandwidth Analysis.	70
7.9	/camera/camera_info Topic Frequency Analysis.	70
7.10	/camera/camera_info Topic Bandwidth Analysis.	70
A.1	ADIRF Tool Home screen.	I
A.2	ADIRF Tool Topic parameter setup for QUPER model OBI optimizer.	II

A.3	Output from ADIRF Tool.	III
A.4	/scan Topic Frequency Analysis.	III
A.5	/scan Topic Bandwidth Analysis.	IV
A.6	/camera/image_raw/compressed Topic Frequency Analysis.	IV
A.7	/camera/image_raw/compressed Topic Bandwidth Analysis.	IV
A.8	/imu Topic Frequency Analysis.	IV
A.9	/imu Topic Bandwidth Analysis.	IV
A.10	/joint_state Topic Frequency Analysis.	V
A.11	/joint_state Topic Bandwidth Analysis.	V
A.12	/tf Topic Frequency Analysis.	V
A.13	/tf Topic Bandwidth Analysis.	V

List of Tables

5.1	List of papers analyzed.	32
6.1	Formalized Quality Levels Based on the QUPER Model	57
6.2	TurtleBot3 Topic Frequencies	58
6.3	TurtleBot3 Topic Bandwidths, and Message Sizes	59
7.1	Frequency Comparison Between Default and SmartTestudo	70
7.2	Bandwidth Comparison Between Default and SmartTestudo	71

List of Acronyms

- AWS** Amazon Web Services. 36
- CNC** Computer Numerical Control. 36
- CPS** Cyber-Physical Systems. v, ix, 1–7, 9–11, 13, 19, 21–23, 28, 29, 73–77, 79, 81, 83, 85
- DDS** Data Distribution Service. 34, 35
- DSL** Domain Specific Language. v, xiii, 16, 17, 22, 45–47, 53, 57, 58, 76, 77, 79–81, 85
- FPGA** Field-Programmable Gate Array. 22
- FRET** Formal Requirements Elicitation Tool. 22, 33, 37, 77, 78
- GUI** Graphical User Interface. 14, 45
- IoT** Internet of Things. 1, 9, 35–37
- LiDAR** Light Detection And Ranging. 11, 12, 52
- MAPE-K** Monitor-Analyze-Plan-Execute over a shared Knowledge. v, 22–25, 44–46, 57, 77–79, 81, 85
- OGMA** Operational Goal-Based Mission Analysis. 22, 33, 37, 77, 78
- PID Controller** Proportional–integral–derivative controller. 23
- QUPER** Quality Performance Model. v, xiii, 25, 26, 45, 51, 55, 64, 67, 68, 79, 80, 85
- ROS** Robot Operating System. v, 1, 3, 7, 9–11, 13–15, 19, 35, 73, 76, 86
- ROS2** Robot Operating System 2. v, xi, 2–4, 6, 7, 9–11, 14–16, 19–23, 27–31, 33–40, 44, 45, 58, 59, 73–83, 85
- RViz** ROS Visualizer. xiii, 3, 13, 14, 29, 30
- SDLC** Software Development Life Cycle. 38
- SLAM** Simultaneous Localization and Mapping. 11, 12, 14, 23
- SuM** System under Monitoring. 21, 30, 34, 46, 57, 60
- UAV** Unmanned Aerial Vehicle. 6
- WoT** Web of Things. 9

1

Introduction

This chapter introduces the main research topic of this thesis, covering the initial background of the scope, the description of the problem, the purpose of the study, the research questions, the motivating example, and the significance of the study. It provides an overview of the aim, scope, and end goals of the project.

1.1 History of Cyber-Physical Systems

Since the first industrial revolution, the breakthrough in technology with the steam engine brought fundamental changes in how people worked, where they lived, what potential economic surplus was available, and how many people could be supported around the world. These changes inevitably had ramifications reaching into almost every aspect of human experience into the habits of thought and the relations between men and women as well as into systems of production and exchange [81]. These new improvements and changes in the way of thinking regarding work, technology, and efficiency, led to the development and research in the field. The results of these improvements were clear, thanks to outside events that pushed the need to increase the productivity and efficiency of factories, this led the way to the second, third, and fourth industrial evolutions [44]. The Fourth Industrial Revolution has significantly advanced beyond the first, building upon the Third by expanding the integration of robots into industrial processes. It leverages the Internet of Things (IoT) and its supporting technologies as the foundation for Cyber-Physical Systems (CPS), with smart machines driving the optimization of production chains [44]. A Cyber-Physical Systems (CPS) is an orchestration of computers and physical systems in which embedded computers monitor and control physical processes, usually with feedback loops, where physical processes affect computations and vice versa [42]. CPS can be utilized in any industry, including engineering, manufacturing, transportation, instrumentation, water management systems, trains, physical security (access control and monitoring), asset management and distributed robotics, and even health care [35]. Another pushing factor of this increase in the adoption of robots in the different fields has been thanks to the development of a standardized Robot Operating System called ROS [68]. This operating system helped developers reduce time in the development of new features by not having to rewrite the basic functions of the robot every time. ROS brought a lot of improvements in the development of robotic systems and highlighted some issues that had to be fixed, such

as bottlenecks and multi-robot operations. To aid and fix these issues a new and improved ROS released, ROS2 which fixed the major issues found in the previous version and is still the de-facto standard in robotic systems development [48].

1.2 Monitoring Systems

CPS are systems that have both physical and digital components, and to manage actual activities and gather data from sensors, the embedded software interacts with the network. As a result, a feedback loop is created that helps to improve behavior at the boundary between the physical and virtual worlds [35]. Data constitutes a pivotal element in CPS, as it enables monitoring robot's behavior and performance. A crucial part in any CPS is a good monitoring system, that allows to understand what is happening and remove any uncertainty, such missing information, unreliable resources, stochastic phenomena and inherently vague concepts [66].

The key functionalities of a monitoring system can be categorized into three main areas:

Fault detection, which signals that something is wrong within the monitored system;

Fault isolation, which classifies the specific issue occurring;

Fault identification, which assesses the severity of the fault [26].

Monitoring systems are crucial to maintaining safety and efficiency of operations in the robot, however this monitoring can sometimes be expensive and intrusive, thus, the design of a monitoring system (i.e., the software system that implements monitoring capabilities) usually involves tradeoffs between the impact caused by the action of monitoring and its expected quality of results, such as data accuracy, freshness and coverage [94]. To deal with these issues of maintaining a high level of monitoring while taking into account the different constraints, researchers and engineers developed a way to make current monitoring systems, *adaptive*.

1.3 Problem Description

The primary issue this research seeks to address is the challenge of managing the real-time constraints and efficiently processing large volumes of data in monitoring systems for Cyber-Physical Systems (CPS). As mentioned in the previous section, CPS interact with the environment through the use of a full suite of sensors [4]. These sensors allow the CPS to reduce the uncertainties and understand what is the state of the environment surrounding it. Removing the uncertainties is fundamental to guaranteeing that the mission objectives are met and that the safety of the system and of the environment is maintained. However in multiple CPS applications the number of sensors can be high and monitoring each one to grant system safety may prove impractical, time-consuming and could put a strain in the communication or processing power of the CPS.

To mitigate this risk, various solutions can be proposed, one of which is criteria-based sensor management. This approach helps in selecting and utilizing a finite set

of sensors alongside available computational resources to optimize the utility of the information about the system at any given time [4]. While the core of this research revolves around the use of an adaptive monitoring system, to perform this action of evaluating a strategy to balance the trade-off between reducing monitoring power and safety of the mission.

A feature present in ROS2 is the high-frequency monitoring messages that are necessary to ensure system safety and functionality. As mentioned above, a drawback of this monitoring is that it can create a significant strain on network bandwidth capabilities and computational overhead, putting stress on the infrastructure. The challenges of efficient monitoring become even more critical in industrial environments, due to the risk of hazard to the personnel and strict resource constraints of the robot's computational power. This problem of reducing monitoring has been studied in ROS2 unmanned drones [34] and research on the topic of adaptive monitoring systems has brought good results, demonstrating their potential to reduce unnecessary data transmission and computational loads without compromising system reliability. While adaptive monitoring strategies have been explored in other ROS applications [89], there remains a lack of research addressing how such strategies can be systematically designed, optimized, and validated for CPS.

This research addresses the potential to design and implement an adaptive monitoring system for ROS2 based CPS systems. Such a system would aim to dynamically adjust the frequency or volume of monitoring data based on its criticality to mission objectives whether scientific or operational. By reducing unnecessary data transmission during less critical phases, the system could optimize bandwidth usage without compromising mission safety or effectiveness. To develop this adaptive monitoring system, more thorough research is needed to understand the different necessities and requirements of the different mission objectives so that the different trade-offs in monitoring certain aspects can be evaluated [91].

1.4 Purpose of the Study

The purpose of this study is to understand the current state of adaptive monitoring in ROS2 based CPS, evaluate the gaps, understand the issues in it and then develop and evaluate a strategy for implementing adaptive monitoring in a ROS2 based CPS system. The end goal will be to produce an artifact with an adaptive monitoring strategy and evaluate it against the current monitoring. The platform that this research will be applied to is a TurtleBot3 [2] device simulated in the Gazebo simulator [39] in an industrial scenario to evaluate the results. The RViz tool [36] will also be used to visualize the robot model. This research aims to explore the trade-offs [91] between monitoring the safety, functionality, and scientific operations of the CPS during its mission activities. By addressing these trade-offs, the study aims to optimize data management and reduce unnecessary transmission without compromising the main objectives of the CPS such as safety, longevity, and mission objectives.

This research will be a foundation analysis to manage the growing challenge of big data monitoring in CPS. The findings are anticipated to benefit future development of adaptive monitoring systems, and expanding research on the existing ROS2 based

CPS systems. Additionally, the results may contribute to broader fields such as autonomous systems, resource-constrained environments, and big data monitoring optimization in robotics.

1.5 Research Questions

To better understand the existing challenges and limitations of monitoring systems in a ROS2 based Cyber-Physical Systems (CPS), and to lay the foundations for this research, the following research questions have been formulated.

RQ1: What are the current limitations in monitoring in ROS2 based CPS?

To address this overarching question, we analyze the monitoring landscape from multiple perspectives:

- **RQ1.1:** What are the technical limitations of current ROS2 topic monitoring tools?
- **RQ1.2:** What usability and integration issues do developers encounter when deploying these monitoring tools?

RQ2: What potential solutions could mitigate the limitations of current monitoring approaches in ROS2-based CPS?

Based on the limitations uncovered in RQ1, we aim to explore adaptable monitoring mechanisms:

- **RQ2.1:** What rule-based strategies can adapt monitoring behavior based on environmental context?
- **RQ2.2:** Are there alternative or complementary solutions that can further enhance monitoring adaptability?
- **RQ2.3:** How feasible is the implementation of these strategies in real-world ROS2 environments?

RQ3: To what extent can the current limitations of monitoring approaches in ROS2 based CPS be solved by the proposed solutions?

To evaluate the effectiveness of the solutions explored in RQ2:

- **RQ3.1:** How much improvement in bandwidth efficiency and topic frequency control can be observed?
- **RQ3.2:** What are the architectural strengths and limitations of applying the proposed adaptive monitoring approach to ROS2 based CPS?

These research questions guide the structure and methodology of this work. Experiments using TurtleBot3 in the Gazebo simulation environment serve as a reference testbed for exploring and evaluating the proposed solutions.

1.6 Motivating Example

The main example used to validate this research is through the use of a TurtleBot3 rover inside a industrial warehouse scenario simulation. In this scenario the TurtleBot is tasked to fetch some items and return them to the main area. During this test, the TurtleBot will navigate through different safety zones, with humans involved, in these areas different kind of monitoring frequencies are necessary to maintain safety while in other the requirements can be more relaxed. These requirements allow to use a more laid down approach regarding monitoring and allow to develop some adaptive monitoring strategies to make the monitoring system of the TurtleBot, adaptive and more efficient regarding the transmission of data through it's antenna. The main motivating factors for this research regarding the need of Adaptive Monitoring Systems in CPS are, *Hardware/Bandwidth Constraints*, *Environment Safety* and *System Resilience and Fault Tolerance*.

Hardware/bandwidth constraints: Since in warehouses or other environments a high number of CPS are being used and tasked with simple delivery goals, robotics manufacturer build them to be cost effective to achieve their requirements. So often the main boards are made with enough computing power to do these operations. Since monitoring overhead communication can take up some computing power, this limited boards can be seen as a limiting factor in monitoring and lifespan of a rover. A solution to alleviate the load of a boards and improve it's lifespan and safety is through the use of adaptive monitoring that alleviates the overheard of monitoring by adapting it and reducing the bandwidth transmitted, reducing the strain on the motherboard.

Environment safety: Since the environments in which CPS operate can be filled with hazards, the CPS must be able to identify the dangers and adapt to the context of the situation. By adapting the frequency of the sensors when inside dangerous environments, the system can monitor with an increased fine grained performance to allow the user to maintain control and safety in the warehouse. This change in monitoring can also alert the user instantly, making the system more effective in displaying what is happening in the environment.

System Resilience and Fault Tolerance: These are two critical aspects that a CPS must have while operating in potentially hazardous environments such as warehouses. In such contexts, issues like sensor failures, communication losses, or unexpected obstacles may occur. An adaptive monitoring system enhances resilience by reallocating resources and adjusting monitoring intensity based on the current state of the system. For instance, if a sensor begins to malfunction or if abnormal behavior is detected, the system can increase monitoring granularity or redirect sensor usage to maintain situational awareness and ensure continued safe operation. This adaptability allows the workers to reduce the downtime and enhance the system capacity to function despite partial faults.

1.7 Significance of the Study

This research aims to refine, revise, and extend existing knowledge on adaptive monitoring systems within the context of ROS2 based CPS. This study is positioned to make meaningful contributions to both the theoretical understanding and practical implementation of adaptive monitoring systems, fostering advancements in the fields of robotics, and data management.

1.7.1 Implications for Research

This study contributes to the broader field of robotics by providing a systematic approach to developing adaptive monitoring systems that can be applied to various ROS2 based robotic platforms beyond TurtleBot3. Future studies could expand upon this framework to investigate machine learning or AI-driven decision-making for adaptive monitoring. Additionally, the research offers valuable insights into the capabilities and limitations of ROS2 based CPS systems, encouraging further enhancements or adaptations of this open-source platform.

1.7.2 Implications for Practice

The developed adaptive monitoring system enhances mission efficiency by enabling practitioners in the robotics industry to optimize monitoring and processor usage, ensuring that critical data is prioritized while non-essential data is deferred or discarded. The framework also improves operational safety and longevity by focusing on safety-critical monitoring, which helps extend the operational lifespan of ROS2 based CPS. Additionally, the methodology provides a cost-effective alternative to testing on physical hardware by offering simulation-based validation of new systems before deployment. Beyond industrial applications, the principles of adaptive monitoring can be adapted to other applications, such as unmanned aerial vehicles (UAVs), autonomous cars, and industrial robots, to address similar challenges of data prioritization in resource-constrained settings.

1.8 AI Use Statement

During the development of this thesis, OpenAI's ChatGPT was used to assist with coding-related tasks and troubleshooting configuration errors regarding the development of the artifact in the ROS2 TurtleBot system. The use of ChatGPT helped reducing the time spent fixing configuration errors in the Gazebo simulator and in the initial ROS2 configuration phase. It also helped boost the productivity related to programming tasks in the development of the strategy. Another tool developed by the University of Gothenburg called ReqGenie [25] has been used during the elicitation phase of the requirement to build the artifact.

1.9 Thesis Outline

Section 2 will present all relevant background theory of the topic. It will include notions regarding Cyber-Physical Systems (CPS), adaptive monitoring, ROS and ROS2, TurtleBot3 and the main tools used in which the artifact for this research will be implemented and evaluated on.

Section 3 will present the related work on which the thesis builds its foundations on. It will be regarding the existing challenges and gaps in adaptive monitoring in software engineering and in CPS. It will also touch some points regarding the internal workings of an adaptive monitoring system, understanding the cost-benefit and trade-offs of such system into practice.

Section 4 will present the research methodology used for this thesis, with a focus on the theory aspect behind it and how it's applied in this research.

Section 5 will present the a literature review of the issues in ROS2 monitoring, it will cluster the results and lay the requirements for the artifact.

Section 6 will present the artifact, how it's been developed and what strategies have been adopted to implement it.

Section 7 will present all the findings identified from the three iterations of this study. It will display the data found from the simulations and leave it for discussion in the section after.

Section 8 will present the main discussion points for each research question. It will discuss the results from the previous section and evaluate the research objectives and limitations of the topic.

Section 9 will present the main conclusion of the thesis which will include a recap with the main takeaways from this research.

2

Background

This chapter introduces the background theory required to understand the topic of this thesis. It discusses key concepts including Cyber-Physical Systems (CPS) with its definitions and applications, adaptive monitoring with its theoretical foundations, ROS and ROS2, TurtleBot3, and the tools used to perform the simulations for this research.

2.1 Cyber-Physical Systems

A Cyber-Physical Systems (CPS) integrates a physical system with a cyber system, combining sensing, computation, communication, and control to enable real-time interaction between the digital and physical worlds [45]. The main components of a CPS are: *Sensors* to allow it to read the data from the surrounding environment, *Actuators* to interact with the environment, *Computing and control center* to process and analyze the data with feedback loops and *Communication network* to interact and communicate with a control center. Cyber-Physical System (CPS) have provided an outstanding foundation to build advanced industrial systems and applications by integrating innovative functionalities through Internet of Things (IoT) and Web of Things (WoT) to enable connection of the operations of the physical reality with computing and communication infrastructures. A wide range of industrial CPS-based applications have been developed and deployed in Industry 4.0 [46]. CPS allows us to envision, construct, develop, refine, and perpetuate smart systems in domains that lead to the improvement of companies, communities, and individuals [35].

2.2 Adaptive Monitoring

As mentioned in Section 2.1 a key component in any CPS is sensors. A big suite of sensors allows the CPS to understand its current state in the environment and analyze what is happening around it. As described in Section 1.2, monitoring system have their drawbacks and weaknesses if not properly implemented. A possible solution to this problem is the use of monitoring strategy called *Adaptive monitoring*. Adaptive monitoring can be seen as a strategy or as a framework of rules used to monitor a system or an environment efficiently. The main idea behind adaptive

monitoring is about finding a set of rules to monitor a phenomenon that take into consideration the different constraints of the system under monitoring or the observer. The main constraints in CPS can be regarding resource management (time spent monitoring, hardware constraints, trade-offs, etc.). The topic of adaptive monitoring is also present in other domains such as biology, natural science, marine wildlife monitoring, wildfire monitoring, software engineering, robotics, and other fields [94].

Regarding the actual definition of Adaptive Monitoring, it varies between fields of study but stays more or less the same regarding the end goal objective.

In the field of Software Engineering, “*The ability an online monitoring function has to decide and to enforce, without disruption, the adjustment of its behavior for maintaining its effectiveness, in respect of the variations of both functional requirements and operational constraints, and possibly for improving its efficiency according to self-optimization objectives*” [55].

In the field of ecology, “*Adaptive monitoring is considered to be an iterative process that requires experience and knowledge of an ecosystem before implementing a monitoring programme, assessing results and interacting with users*” [69].

2.3 ROS and ROS2

As mentioned in Section 1.1 with the advent of robotic systems, and improvements in technology such as metallurgy, electronics and software capabilities, robotic systems started to become a crucial part in industrial settings. From painting, to welding, to moving boxes, this revolutions brought great improvements in productivity and worker safety. As new robotic systems were being developed, a common issue was still present, each robotic system had it’s own software architecture, so whenever a new model was being developed, it was necessary to start from scratch, losing valuable time re-developing older features, rather than creating new. An initial solution has been developed and it’s Robot Operating System (ROS). ROS was designed to meet a specific set of challenges encountered when developing large-scale service robots as part of the STAIR project at Stanford University and the Personal Robots Program at Willow Garage, but the resulting architecture is far more general than the service-robot and mobile-manipulation domains [68].

With the improvements in technology robotic systems started to become more powerful with new functionalities and so the ROS framework wasn’t enough [33, 86]. A new open-source ROS framework was developed called ROS2 [93]. ROS2 has been upgraded to improve the performance bottlenecks and architectural design limitations found in ROS [93] and is now being used as the de-facto standard in industrial robots, and custom in-house forks are developed to improve the existing functionality and add new ad-hoc additions [24].

2.4 TurtleBot3

TurtleBot3 [73] is a small, affordable, highly customizable, programmable, ROS based mobile robot for use in education [2], research, hobby, and product prototyping. The goal of TurtleBot3 is to dramatically reduce the size of the platform and lower the price without having to sacrifice its functionality and quality, while at the same time offering expandability. The use of TurtleBot3 helped researchers to simulate different experiments and evaluate results.

There are three different versions of TurtleBot3: *Burger*, *Waffle* and *Waffle Pi*.

The *Burger* is the simplest version equipped with just a 360° LiDAR sensor and limited sensors. On the other hand *Waffle* constitutes a more advanced version, having both a 360° LiDAR, a camera and other improved sensors. Finally *Waffle Pi* is quite similar to *Waffle* but using a Raspberry Pi as a processor [74].

One of the key features of TurtleBot3 is its ability to map unknown environments using LiDAR in conjunction with SLAM [13]. The two platforms *Burger* and *Waffle* can be seen in Figure 2.1 taken from the documentation website [74].

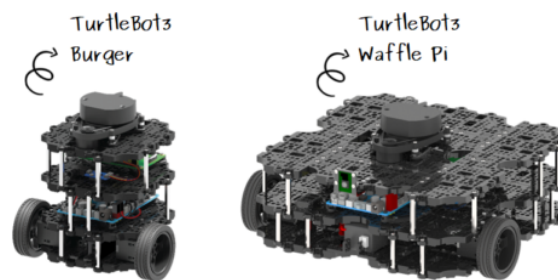


Figure 2.1: TurtleBot3 platforms. Figure reused from [74].

2.5 TurtleBot3 Topics

In ROS2 a topic is a named communication channel used for publishing and subscribing to messages between different nodes in a robotic system. These topics enable asynchronous, many-to-many communication, meaning one or more nodes can publish data to a topic, and multiple other nodes can receive that data in real time. Each topic is associated with a message type that defines the structure of the data being transmitted for example, velocity commands, sensor readings, or camera images. Topics are a key part of ROS2 *publish-subscribe* architecture, making them ideal for handling continuous streams of data such as those from sensors, control commands, or localization updates. TurtleBot3 being a ROS2 based CPS adopts the same topics communication mechanism which help communicates the different states of the rover without being tightly coupled, promoting modularity, scalability, and easier debugging.

For this research the following topics have been used and analyzed:

The `/cmd_vel` topic uses the `geometry_msgs/Twist` message type and is responsible for conveying motion commands to the robot, including both linear and angular

2. Background

velocities. It is typically published by the navigation stack or teleoperation nodes and directly influences the movement of the robot.

The `/odom` topic publishes `nav_msgs/Odometry` messages, providing estimates of the robot's position and velocity over time. This information is critical for tracking the robot's motion, supporting localization algorithms, and enabling autonomous navigation.

The `/camera/image_raw/compressed` topic, with message type `sensor_msgs/CompressedImage`, delivers compressed camera data to reduce bandwidth usage while still enabling basic image processing tasks such as object detection or obstacle recognition.

In contrast, the `/camera/image_raw` topic uses the `sensor_msgs/Image` message type to stream uncompressed camera frames. This topic is particularly important for high-resolution vision tasks like SLAM or visual servoing, though it consumes significantly more bandwidth.

The `/imu` topic publishes `sensor_msgs/Imu` messages, which include data on orientation, angular velocity, and linear acceleration. This is essential for estimating the robot's pose and motion dynamics, especially when fused with odometry or GPS information.

The `/joint_states` topic uses `sensor_msgs/JointState` messages to report the positions, velocities, and efforts of the robot's joints. This data is key for understanding the robot's mechanical state and for executing accurate joint-space control.

The `/scan` topic provides `sensor_msgs/LaserScan` messages, delivering 2D range measurements from a laser scanner or LiDAR. This is a foundational sensor input for obstacle detection, mapping, and reactive navigation.

The `/tf` topic publishes `tf2_msgs/TFMessage` messages containing dynamic coordinate transforms between frames (e.g., from base to camera or sensor frames). These transformations are updated over time and are essential for spatial awareness in a ROS-based system.

Finally, the `/camera/camera_info` topic delivers `sensor_msgs/CameraInfo` messages, which include calibration parameters like focal length and distortion coefficients. These parameters are necessary for accurate image processing, projection, and camera-based navigation.

These topics can be seen in Figure 2.4 and how they interact with each other in Figure 2.5.

2.6 Gazebo

A key component in any empirical research is where the data is sourced from. In experiments regarding expensive scenarios where replicability is a key requirement, simulators come in aid. In the case of TurtleBot3 or in any ROS based CPS, Gazebo simulator [59] is the de-facto standard. Gazebo is a robot simulation toolbox and is commonly used to develop tests for software developed in any ROS version. It provides a framework, extending ROS, which unifies aspects such as physics-based simulation and visualization using an XML-based notation to define objects, and a plugin architecture [63]. Gazebo is designed with the goal to accurately reproduce the dynamic environments a robot may encounter. All simulated objects have mass, velocity, friction, and numerous other attributes that allow them to behave realistically when pushed, pulled, knocked over, or carried. These actions can be used as integral parts of an experiment, such as construction or foraging [39]. It also offers a rich environment to quickly develop and test multi-robot systems in new and interesting ways. It is an effective, scalable, and simple tool that has also potential for opening the field of robotics research to a wider community [39]. Additionally, it contains high quality graphics and convenient graphical and programmatic interfaces [19]. Another powerful feature of Gazebo is the ability to create your own world simulations directly inside the tool rather than just importing an existing one. In Figure 2.2 an example of an industrial warehouse is presented, it shows the flexibility of the tool, not only in being a real time simulator but also a world editor.

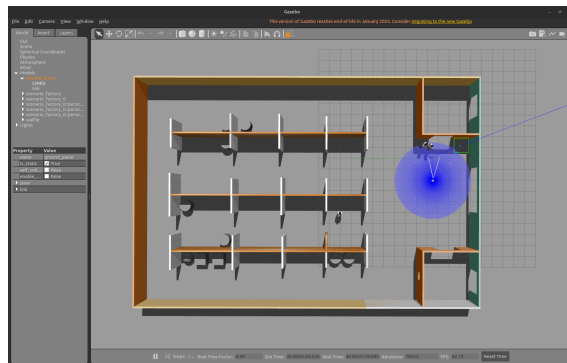


Figure 2.2: Gazebo with the industrial warehouse scenario.

2.7 RViz

In computational science and computer graphics, there is a strong requirement to represent and visualize information in the real domain [36]. RViz [76] [75] is a three-dimensional viewer whose main objective is to display sensor data and ROS state information. When using RViz, it is possible to view the current virtual robot model configuration. It is also feasible to display live representations of sensor values on ROS topics, including camera data, infrared distance measurements, sonar data, laser sensor measurements, among many other applications [19]. The real strength of RViz is the ability to visualize the data read from the TurtleBot3 whether in a

2. Background

simulation or deployed in a real environment. Figure 2.3 shows a local map created with the SLAM module of TurtleBot, this allows to create a map of the environment to let the TurtleBot understand where it is and what operations to perform to reach an objective.

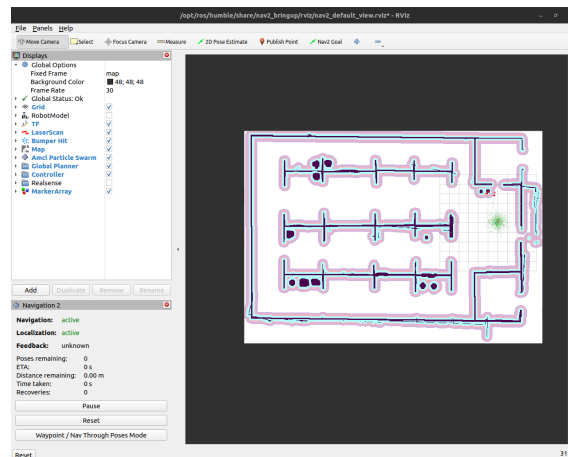


Figure 2.3: RViz with the industrial warehouse scenario.

2.8 Rqt

Rqt [16] is a visualization tool for ROS and ROS2. It provides a Graphical User Interface (GUI) to visualize and interact with the topics that are monitored by the system. Rqt is a tool that allows developers to simplify their workflow by reducing the need for manual terminal commands, and simplifies the debugging of the system. Regarding visualization the main features allow the developer to see the various topics, and image view of what the system sees. Figure 2.4 shows the topic visualized on the left, displaying information regarding the frequency of the messages and the bandwidth. By opening the single topic more information can be displayed such as the content of the message. The figure on the right displays the image view read from the camera/image_raw topic, allowing developers to have a quick glance at what the robot is seeing.

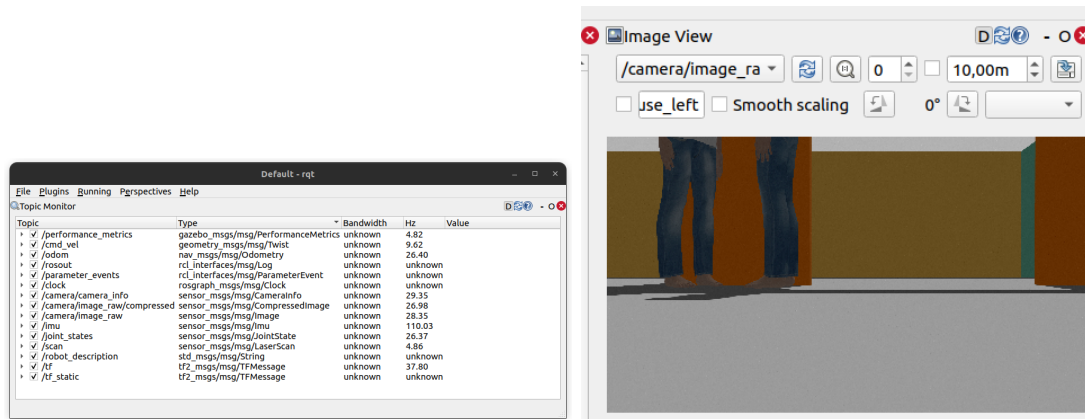


Figure 2.4: Rqt topic viewed and camera view.

Another great tool is `rqt_graph` [17] which allows the visualization of how the nodes communicate, making it easier to understand the flow of topics and services in the system. Figure 2.5 shows the generated nodes graph of the current system, highlighting the key relationship between subscriber and publisher of the topic in nodes.

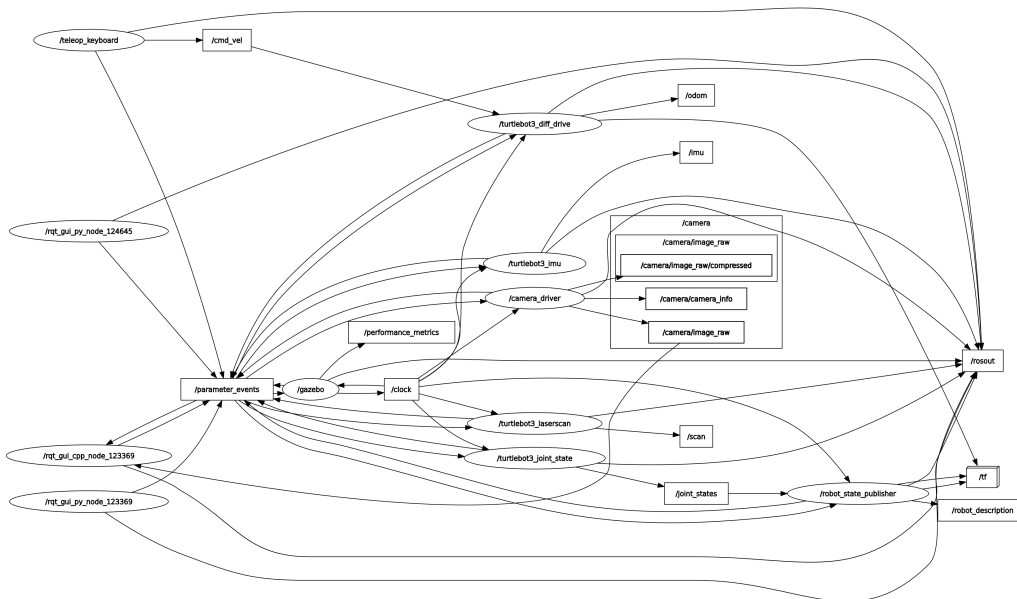


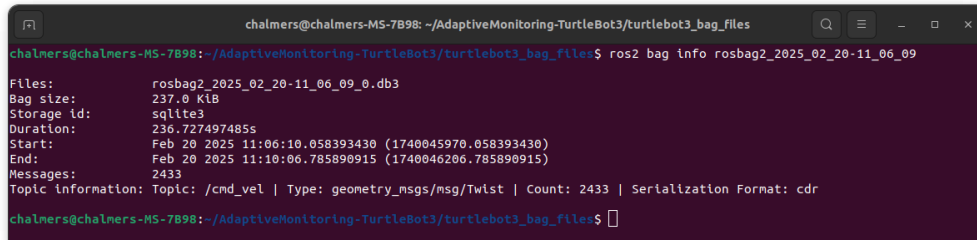
Figure 2.5: Graph made with `rqt_graph`.

2.9 Rosbag2

Rosbag2 is a tool in ROS2 used for recording and replaying messages that are exchanged between nodes in a system. It is the successor to rosbag from ROS but with improvements such as support for different storage formats, better performance, and

2. Background

built-in compression. The main features of rosbag2 are the ability to record all the topics during a run, and then being able to play them back to re-play the scenario the exact way it happened. Its an essential tool for debugging and doing data analysis in ROS2 applications. It allows the replay of different simulations recorded and so it allows to make comparisons between data gathered and scenarios. Figure 2.6 displays the output of the command `ros2 bag info <name_ofbag>`.



```
chalmers@chalmers-MS-7B98: ~/AdaptiveMonitoring-TurtleBot3/turtlebot3_bag_files
chalmers@chalmers-MS-7B98: ~/AdaptiveMonitoring-TurtleBot3/turtlebot3_bag_files$ ros2 bag info rosbag2_2025_02_20-11_06_09
Files:          rosbag2_2025_02_20-11_06_09_0.db3
Bag size:       237.0 KiB
Storage id:     sqllite3
Duration:       236.727497485s
Start:          Feb 20 2025 11:06:10.058393430 (1740045970.058393430)
End:            Feb 20 2025 11:10:06.785890915 (1740046206.785890915)
Messages:      2433
Topic information: Topic: /cmd_vel | Type: geometry_msgs/msg/Twist | Count: 2433 | Serialization Format: cdr
chalmers@chalmers-MS-7B98: ~/AdaptiveMonitoring-TurtleBot3/turtlebot3_bag_files$
```

Figure 2.6: Rosbag2 info on a bag from the industrial warehouse scenario.

2.10 DSL

“A Domain Specific Language (DSL) is a language designed to provide a notation tailored toward an application domain, and is based only on the relevant concepts and features of that domain“ [41].

A Domain specific language is similar to a programming language but it has a higher abstraction level that encapsulates the need for the domain specific problem that aims to aid in solving. DSL are built by domain expert to express the problem and the need in a clear way. The objective of the experts is to make the DSL easy to use for non-programmers, who will use it to define objectives in the program and define instructions or rules.

The main benefits of using a DSL are regarding improvements in: overhead performance in generation of code, separation of concerns from implementation and declaration, increased maintainability and interoperability.

The main drawbacks of adopting a DSL are mainly regarding the upfront cost of developing such artifact. The development requires a consult of a domain expert that will guide the developer into building such language.

To understand if a DSL is needed in the artifact that is being built, a guide that summarizes and analyses the different development phases has been developed [53]. This paper analyses DSL’s and developed a step by step guide to make the developer understand if the need for a DSL is justified, by cost and requirement, since in not all cases a DSL is needed, due to the different drawbacks explained above.

In Figure 2.7 an example of a schema of a DSL made with the Python library TextX [20] is seen. This is the dictionary that the parser will use to interpret the example rules seen in Figure 2.8.

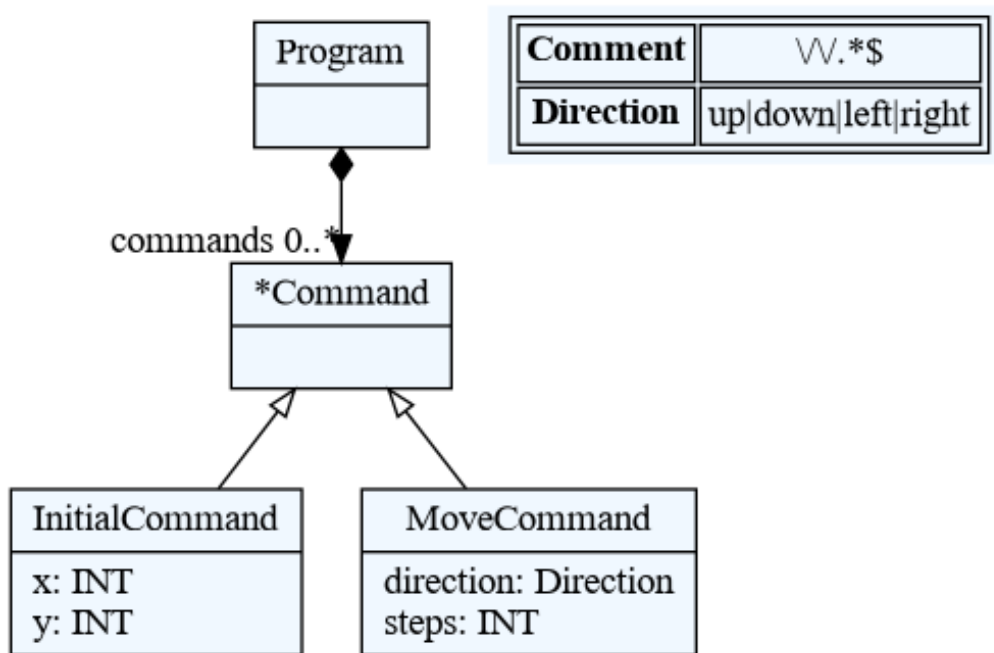


Figure 2.7: Example of DSL schema using TextX. [20]

```

begin
  initial 3, 1
  up 4
  left 9
  down
  right 1
end
  
```

Figure 2.8: Example of DSL rules using TextX. [20]

2.11 Mathematical Optimization

Mathematical optimization is a branch of applied mathematics focused on finding the best possible solution from a set of feasible alternatives. It involves defining an objective function that needs to be maximized or minimized such as cost, profit, or efficiency subject to a set of constraints that represent the limits or requirements of the problem domain. Optimization problems are widely used in fields such as operations research, engineering, finance, logistics, and data science. They come in various forms, including linear programming (LP), integer programming (IP), and nonlinear programming (NLP), depending on the nature of the objective function

and constraints. In practical applications, optimization problems are often solved using specialized software tools. PuLP [85] is a Python library that provides a simple interface for defining and solving linear programming problems. It allows users to model optimization problems in a natural and readable way and connects with powerful solvers like CBC [15], GLPK [28], or CPLEX [31] to compute solutions efficiently.

“In more general terms Mathematical Optimization may be described as the science of determining the best solutions to mathematically defined problems, which may be models of physical reality or of manufacturing and management systems.”[79]

“Optimization is everywhere from routine business transactions to important decisions of any sort, from engineering design to industrial manufacturing, and from choosing a career path to planning our holidays. In all these activities, there are always some things (objectives) we are trying to optimize and these objectives could be cost, profit, performance, quality, enjoyment, customer-rating and others. The formal approach to these optimization problems forms the major part of the mathematical optimization or mathematical programming.”[92]

3

Related Work

This chapter introduces the related work, focusing on the current state of the art and existing challenges in the field of adaptive monitoring. Key aspects include adaptive monitoring in software engineering, challenges in monitoring practices, evaluation of trade-offs between cost and benefits, and analysis of ROS2-based CPS monitoring. The chapter broadens the perspective on the topic by highlighting current gaps and unresolved problems in both industry and research.

3.1 ROS/ROS2 in Industry

One of the main advantages of having a ready to go architecture such as ROS2, is the ability to perform all the research and development needed without wasting time in building your in-house solution first. This "research led" approach allowed practitioners to develop their own proof of concepts, focusing on the concept rather than all the architecture behind. Mentioned in the previous sections, ROS brought an increase in the development of CPS technology, this increase of usage can be seen by the high number of company in all the different fields of industry that are adopting ROS2 as either product or as a development tool for their solution. The data for all the companies can be seen in this GitHub repository [51]. ROS2 is also used in other fields such as research or in school, this map created by [54] shows the different institutions, research centers and companies which actively use ROS2. Another way to show this increase in usage is by analyzing the history of the ROS2 metrics on the website forums and documentation section which can be found online [60].

An interesting research regarding an empirical study on the ROS ecosystem [40], performed an analysis on all the ROS packages and contributors. The main conclusions drawn from this research is that even though the ROS ecosystem is constantly growing with new adopting it, the release of new packages is almost linear, meaning that most of the users use the existing packages and don't provide some new features to the ROS ecosystem. This can be seen as expected, as mentioned in the title "It Takes a Village to Build a Robot"[40], the results show that to perform such innovative task of introducing a new improvement, a big team of developers is needed. An example of such industry application and proof of concept can be seen from this research by Bonci et al. [8] in this research the authors perform an analysis of the current state of ROS2 applications and perform their own proof of concept to

improve the middleware of ROS2 for industrial applications.

3.2 Adaptive Monitoring in Software Engineering

Adaptive monitoring is a broad topic with applications across various fields such as biology, robotics, computer science, and any other domain that requires a dynamic form of monitoring. This dynamic approach arises from the need to enhance resource management and consumption in existing monitoring strategies. The resources involved can differ depending on the application and may include human effort, hardware constraints, computational power, and more. As discussed in Section 2.2, the primary goal of adaptive monitoring is to optimize current monitoring processes by implementing strategies that improve resource consumption while ensuring that constraints, particularly in safety-critical environments, are not violated.

Significant research has been conducted on adaptive monitoring in various fields. In the context of Software Engineering, a systematic mapping study by Zavala et al. (2019) [94] provides an overview of the current state of the art. The study analyzed 110 research papers and classified them into 81 different approaches. Its main objectives were to identify key concepts of adaptive monitoring, examine research trends in the area, analyze how adaptive monitoring is conducted and evaluated, and identify common strategy patterns. The study's findings highlight several key characteristics of adaptive monitoring artifacts.

In terms of contributions to the field, the analyzed artifacts are mainly categorized as algorithm-based and architecture-based solutions. Most solutions are problem-specific, with only a few being generic. The predominant objective of adaptive monitoring is to address trade-offs, such as balancing monitoring efforts with overhead costs or reducing bandwidth usage and monitoring frequency. The primary elements that undergo adaptation include sampling points, sampling rates, and monitored metrics. Adaptation is usually triggered by system faults, requirement violations, state changes, or context-based factors. Decision-making is predominantly policy-based, and execution is mostly automated, with only a few solutions incorporating semi-automatic or manual interventions.

Regarding the types of adaptation, structural adaptation is the most common, while parameter-based adaptations are less frequent. The test environments used in research vary, with 59% of studies relying on simulations, while others use real-world industry cases. Evaluation is typically carried out in sensor networks and service-based systems.

The issue of evaluating trade-offs in adaptive monitoring systems is further explored by Wohlrab et al. (2022) [91], who discuss the challenge of analyzing and understanding different quality attributes. This topic, particularly the cost-benefit trade-offs of monitoring systems, will be elaborated in Section 3.7.

3.3 Challenges in Monitoring CPS

As discussed in Sections 1.2 and 1.3, one of the primary challenges in monitoring systems within Cyber-Physical Systems (CPS) is managing the number of sensors and the volume of data processed by the system's processor. To effectively monitor a System under Monitoring (SuM) system, the main system must be capable of reporting data without compromising its normal functionality. The primary concern identified by Stadler et al. (2023) [80] relates to the safety of operators working in environments where CPS are deployed. The paper outlines various risks associated with these scenarios.

One significant risk factor is the occurrence of industrial accidents involving robots, where operators may be injured or crushed by robotic systems. Another major risk arises from non-routine operations, such as maintenance, repair, and inspection, which often lead to accidents when systems deviate from their standard operating models. Additionally, the complexity of collaborative CPS presents a challenge, as multiple robots working together require real-time monitoring to ensure safety and coordination.

These challenges are not only tied to the monitoring system itself but also to the broader strategy and policies governing the use of CPS in safety-critical environments. The author identifies five key challenges in monitoring CPS. First, adaptive monitoring is essential, as the system must dynamically adjust monitoring parameters based on context, such as increasing monitoring intensity during maintenance activities. Second, precautionary mode transitions should be implemented, allowing CPS to automatically switch to safer operational modes when risks are detected. Third, the co-evolution of monitoring must be considered, ensuring the system can adapt to changes like the integration of new sensors or additional CPS units. Fourth, effective aggregation of monitoring data is crucial for detecting system-wide patterns, requiring the analysis of data from multiple CPS. Finally, data persistence for post-incident analysis is necessary to facilitate learning from past incidents and improving future safety measures.

Another relevant aspect of CPS in industrial settings concerns ethical considerations, as discussed by Enzing et al. (2023) [22]. These ethical issues include transparency and accountability, worker rights and well-being, and potential biases in monitoring practices. While these concerns are not the focus of this research, they remain important factors to consider when designing monitoring systems for CPS in environments involving human operators.

3.4 Monitoring Techniques Analysis

The main objective of monitoring is to give the various information to the user and remove any uncertainties of what is happening to the System under Monitoring (SuM). To perform such monitoring operation, ROS2 has its own monitoring system present under the hood, it works by monitoring the various topic by pub-

lishing them, and anyone can listen to their messages. This is a simple but effective approach in the everyday use, and it fits the requirement of being general but also allows for some customization to accommodate for other people requirements.

ROS2 being open source and customizable, as discussed in Sections 3.1, different researchers have developed their own in house solution and strategy to make monitoring more efficient. The list below analyzes different papers in which the authors address the topic of monitoring and propose different approaches.

Cheng et al. [12] proposed a proof of concept for a self adaptation operations in ROS2 applied to a CPS EvoRally autonomous rally car. In this approach they evaluated the need to apply a MAPE-K loop to maintain the monitoring requirements would be satisfied during runtime.

Perez et al. [65] discuss the importance of runtime verification monitoring for safety critical environments. To maintain the requirements satisfied two main tool are used: FRET and OGMA. These two tools are used to generate the monitoring nodes and publish the violations.

River et al. [70] they bring to the table a ROS-FM, a high-performance inline network-monitoring framework with a security policy enforcement tool and distributed data visualization. To further validate their research the authors compare the overhead of this framework against the generic ROS monitoring tools. In this approach they used a technical solution with the integration of a DSL to allow the user to configure the monitoring system.

Mayoral et al. [52] propose the topic of adaptive computing in the field of robotics. Adaptive computing allows the system to change its computational behavior at runtime using reconfigurable hardware like FPGAs. This solution allows to create a software-defined hardware model, where hardware behavior is controlled like software through ROS 2.

Ichnowski et al. [32] explain how ROS2 limited computing resources both on board and on the cloud can be overwhelmed by the amount of data collected. To solve this issue they developed FogROS2 an adaptive cloud-fog offloading platform for ROS2 based CPS.

Lim et al. [58] addressed the safety monitoring for collaborative robots. They developed a solution to limit hazards in industry by designing a redundant fault detection system composed of a simulated cobot controlled via ROS2 nodes, a gazebo simulation mimics the expected behavior, if the expected result from the simulation is different from reality a fault is detected and stopped.

NVIDIA Jetson is pushing the development of AI solutions for ROS2[72]. NVIDIA Jetson is a product from NVIDIA which consists is a single-board computer designed specifically for AI applications. It is much more performant than a Raspberry Pi, this feature allows it to develop such AI oriented solutions. However such solutions at the moment are mainly oriented to technologies such as image recognition, SLAM improvements and computer vision. While aspects such as monitoring are not yet being analyzed by AI solutions.

Luckcuck et al. [47]’s research is a survey of the use of formal methods in the specification and verification of autonomous robotic systems. The authors analyze how formal approaches can complement or surpass traditional techniques like testing and simulation, especially for ensuring safety and correctness. This research’s result also look at techniques such as runtime monitoring and prove crucial for the verification of such systems.

3.5 MAPE-K

“The MAPE-K (Monitor-Analyze-Plan-Execute over a shared Knowledge) feedback loop is the most influential reference control model for autonomic and self-adaptive systems.”[3]

“The MAPE-K feedback loop has been established as the primary reference model for self-adaptive and autonomous systems in domains such as autonomous driving, robotics, and Cyber-Physical Systems.”[14]

CPSs during their existence face challenges in a changing environment to which they have to react accordingly. From the basics of the Proportional–integral–derivative controller (PID Controller), another feedback loop has been created Monitor-Analyze-Plan-Execute over a shared Knowledge (MAPE-K).

Self-adaptive systems are designed to autonomously adjust their behavior in response to changes in their environment, system state, or operational requirements. A widely adopted approach for enabling self-adaptation is the MAPE-K feedback loop, which is a fundamental architectural pattern in autonomic computing.

The MAPE-K loop consists of the following stages that can be seen in Figure 3.1.

Monitor: This phase is responsible for collecting relevant data from the system and its operating environment. Sensors, logs, and performance metrics are commonly used to gather information.

Analyze: The collected data is examined to detect trends, patterns, or anomalies. Advanced techniques such as statistical analysis, machine learning, and rule-based

reasoning can be used to assess the system's state and predict future behavior.

Plan: Based on the analysis, a decision-making process determines the necessary adaptations. This phase involves defining adaptation strategies, optimization techniques, and selecting the best course of action to ensure the system maintains its objectives such as performance, security, energy efficiency.

Execute: The planned adaptation actions are applied to the system through actuators, configuration changes, or policy updates. This ensures that the necessary modifications are implemented to maintain or improve system performance.

Knowledge (K): The entire process is supported by a shared knowledge base, which stores historical data, system models, policies, and learned behaviors. This component enhances decision-making by providing contextual information and enabling continuous learning.

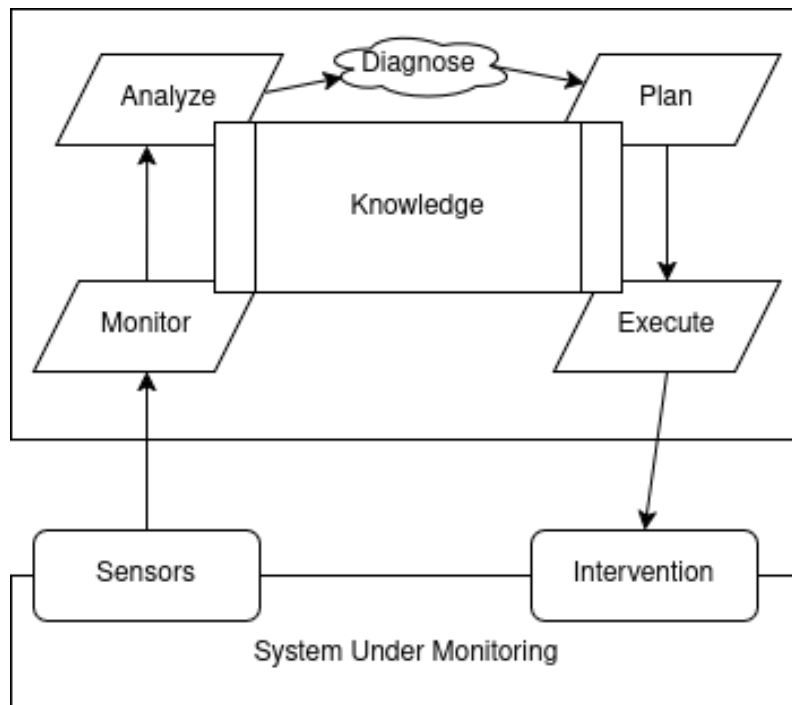


Figure 3.1: MAPE-K Architecture diagram. Image adapted from [3]

The MAPE-K loop facilitates the design of self-adaptive systems across various domains, including cloud computing, cybersecurity, IoT, and robotics. By incorporating this loop, systems can proactively respond to environmental changes, workload fluctuations, security threats, and hardware failures, ensuring optimal operation without human intervention. This architecture plays an important role in the development of any adaptive system. Within the research body there have been different applications, such as employing MAPE-K in reflective middlewares [90] and Human-Machine Teaming [14]. For this research and development of the artifact the core architecture will expand on the existing battle hardened MAPE-K solutions with some knowledge in form of requirements specified by the user.

3.6 QUPER Model

The purpose of the model is to provide concepts for qualitative reasoning of quality levels in the decision-making of setting actual targets of quality requirements for coming releases of the product [6].

The Quality Performance Model (QUPER) [84] is a decision-support framework designed to facilitate high-level release planning for quality requirements in software development. It provides a structured approach for evaluating the trade-offs between quality attributes, development costs, and market expectations to assist stakeholders in making informed decisions.

The key components of the QUPER model are listed below [6].

Thresholds and Breakpoints: The model identifies critical points at which improvements in a quality attribute lead to significant increases in user satisfaction or competitive advantage. These points help prioritize quality enhancements based on their potential impact.

Cost-Utility Trade-offs: QUPER supports the evaluation of the relationship between quality improvements and associated costs, enabling decision-makers to determine whether an increase in quality is justified based on its added value to the system.

Market and Competitive Influence: The model takes into account market expectations and competitive benchmarks to assess how various quality attributes contribute to a product's positioning in the marketplace.

The QUPER model is built upon two key concepts: breakpoints and barriers. Breakpoints define the non-linear relationship between quality and benefit, while barriers capture the non-linear relationship between quality and cost. These concepts form the foundation of QUPER's three perspectives: the Benefit View, the Cost View and the Roadmap View. [84]

The Benefit View displayed in Figure 3.2 introduces three distinct breakpoints:

Utility Breakpoint: This marks the transition from useless to useful quality. A product with quality below this threshold is not recognized for its value and is unlikely to be accepted in the market.

Differentiation Breakpoint: This represents the shift from useful to competitive quality, where the product gains a strong competitive position in the market.

Saturation Breakpoint: This indicates the transition from competitive to excessive quality, where further improvements provide no significant practical benefit within the given usage context.

QUPER has been widely applied in domains [82, 83] where quality characteristics are a key differentiator, such as mobile devices, embedded systems, and software products. It enhances decision-making by providing a structured methodology to balance quality aspirations with practical constraints. The QUPER model can prove useful in combination with the MAPE-K in the development of the adaptive monitoring system artifact, by combining the feedback loop with the knowledge and quality goals defined with the use of the QUPER model, allowing the user to fine

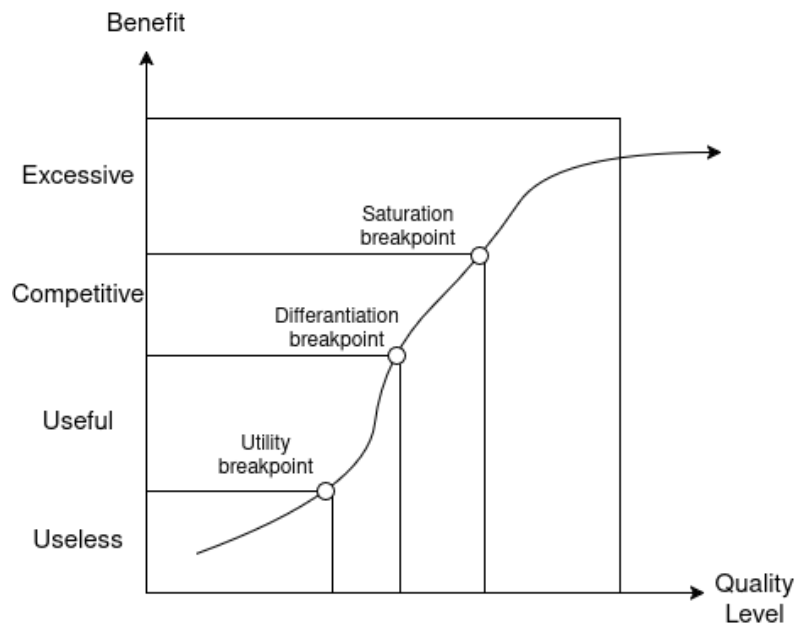


Figure 3.2: The QUPER benefit view. Modelled from [84].

tune the system with the correct parameters.

3.7 Evaluating the cost-benefit tradeoff of monitoring

As explained in Section 3.2 a bit influential factor in making an adaptive monitoring strategy is regarding evaluating the different trade-offs that a system will be impacted by. These trade-offs are necessary to make the system adapt based on the context that it finds itself in. An example of this trade-off analysis is from this paper [91] where the authors performed an analysis of trade-offs in automated planning for adaptive systems. The main challenge in the paper was defining an utility function that took into account parameters that the stakeholders might not fully understand. The different quality attributes that a robotic system might have are speed vs energy efficiency, privacy vs safety and safety vs cost. The goal by the paper was to define an utility function that evaluated these trade-offs to make the robot take the optimal path. In this paper the main solution and defining factor was a machine learning based solution that would analyze the performance and make the decision.

4

Research Methodology

This chapter introduces the methodology used to conduct this research. It explores the theoretical foundations of the chosen approach and explains how the methodology is applied within the context of this study. It also outlines the processes for data collection and analysis.

4.1 Research Design

A design science research approach [21] was followed to investigate how adaptive monitoring systems can benefit the performance and efficiency of ROS2 systems and to be able to answer the research questions. It was considered to be best suited for this project due to the iterative nature of the process and the need to deliver an artifact after each iteration. Design science research focuses on the design process and gaining knowledge about designing the artifact as well as the artifact itself [21]. The artifact in this research was developed over the second and third iteration, with the first iteration being more a theoretical analysis of the state of the art to elicit the requirements for the following iterations. At the end of each iteration, the artifact was evaluated and validated, which served as the basis for the implementations in the next iteration.

4.1.1 Design Science

There exists multiple definitions of design science:

- *Design science research aims to study, research, and investigate the artificial and its behavior from an academic and organizational standpoint [5].*
- *Design science research is a rigorous process of designing artifacts to solve problems, to evaluate what was designed, or what is working and to communicate the results [10].*
- *Design science is the epistemological basis for the study of what is artificial. Design science research is a method that establishes and operationalizes research when the desired goal is an artifact or a recommendation. In addition, research based on design science can be performed in an academic environment*

and in an organizational context [87].

All these definitions have the same meaning, Design science is a research methodology used when a problem requires empirical data to be gathered and processed, and the development of an artifact through iterations to solve or analyze a problem in research.

The use of design science in software engineering problems has started to become more popular due to the good fit in the development of an artifact such as software systems, frameworks and guidelines. New research has been made in the field of design science, and Knauss proposed seven guidelines for applying design science research for Master's Thesis work [38].

Here are the seven guidelines:

- **G1:** Define the artifact early.
- **G2:** Work in iterations. Contribute to each research question in each iteration.
- **G3:** Define research questions with respect to the regulative cycle, i.e., one related to the problem, one related to potential solutions and their construction, and one related to evaluation.
- **G4:** Have regular meetings.
- **G5:** Shift emphasis between cycles. Work on each research question in each cycle, but put more emphasis on the problem in Cycle one, on the solution in Cycle two, and on the evaluation in Cycle three.
- **G6:** Have a dedicated section to describe the artifact.
- **G7:** Write the thesis document as you go.

4.2 Iteration Plan

For this research regarding adaptive monitoring systems, following Knauss's guidelines [38], the research has been divided in three iteration in which the end goal will be to develop an artifact to improve the adaptive monitoring systems in ROS2 based Cyber-Physical Systems (CPS).

The Iteration plan for this research is shown in the picture below.

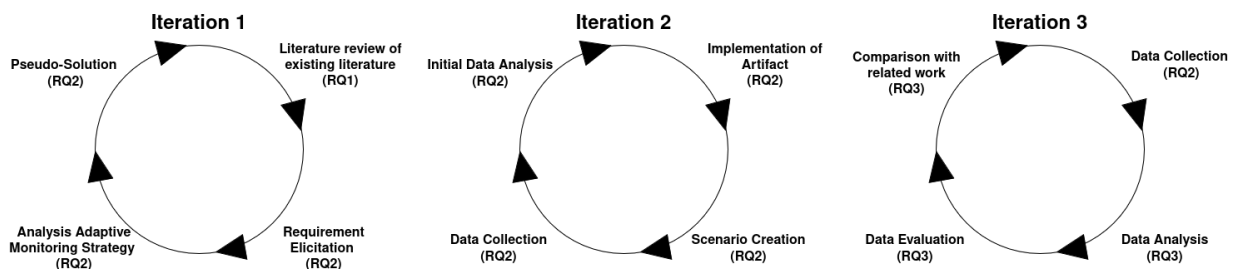


Figure 4.1: Iteration plan overview.

4.3 Iteration 1

Following **G2** and **G3** from the guidelines mentioned above, the focus in Iteration 1 revolves around the understanding and discussion of the problem [38]. In this first iteration the main analysis of the problem will be through the use of a literature review of the monitoring problem and the end goal will be to have a clear understanding of the issues by looking at the state of the art and approach a discussion for **RQ1**, the second goal will be to set the foundations by eliciting the initial requirements for the development of the artifact to answer **RQ2**.

4.3.1 Data Collection

To understand the problem and bring a solution for **RQ1** and start the requirement elicitation phase for **RQ2**, the main data collection will be through literature review and analysis. Through the use of existing research in the topic of monitoring in ROS2 based CPS, data will be collected from different sources. In this iteration a brainstorming session with an expert in the domain has been planned to discuss about possible strategies to adopt for the artifact. Other data sources of data will be through some consult of existing forums and other papers of information will be analyzed.

4.3.2 Goals and Data Analysis

The main goal of Iteration 1 is to answer **RQ1** and give some initial ideas for **RQ2**. The main result produced by this iteration would be an analysis of the issues found in ROS2 systems in different field and a list of functional and non-functional requirements that will be used for Iteration 2. The main data analysis would be through the analysis of results in different papers and the data will be grouped into different categories. After having analyzed the data gathered, there will be a requirement elicitation phase with the use of the tool ReqGenie [25]. This tool will help to define the requirements based on the issues found in research, and will help guide through the development phase in iteration 2.

4.4 Iteration 2

Following **G2** and **G5** and building on the requirements elicited from Iteration 1, the main end goal of this second iteration is to develop an artifact to solve **RQ2** and to lay some initial boards for **RQ3**. This iteration will prove critical, putting emphasis on the development of a solution to the problem of monitoring in ROS2 based CPS.

4.4.1 Data Collection

To evaluate the results from the implementation of the artifact, empirical data must be collected. In this iteration the main empirical data will be gathered through the use of the TurtleBot3 ROS2 based robot in the Gazebo simulator and RViz.

The data will be gathered through the use of a warehouse scenario in which the TurtleBot3 robot will be put through. The data will be collected using rosbag2 [18], and the configuration of the scenario and the artifact will be explained in Section 7.1.

4.4.2 Goals and Data Analysis

The main goal for Iteration 2 is to answer to **RQ2** and to start evaluating the artifact for **RQ3**. This iteration's goal is to bring a working artifact to the table that is able to produce some meaningful result regarding the monitoring issue. This artifact will be put to the test in a warehouse scenario explained in Section 7.1, and the main data analysis strategy will be through comparison with baseline configuration against the one with the artifact. The main topics will be monitored and comparison graphs will be plotted regarding frequency and bandwidth. The end goal will be to clearly show the improvements or the drawbacks of the system.

4.5 Iteration 3

The last iteration will focus on **G5** and **G6**. This final iteration will focus on improving the artifact regarding **RQ2** and making evaluations and analysis showing results to discuss and solve **RQ3**. The main challenge in this iteration will be to show the findings in a clear way and understanding the comparison of the existing artifacts against the one developed.

4.5.1 Data Collection

The data collection will be the same as Iteration 2. It will be gathered through the use of the TurtleBot3 ROS2 based robot in the Gazebo simulator and RViz. The data will be gathered through the use the warehouse scenario mentioned in which the TurtleBot3 robot will be put and evaluated. The data will focus on monitoring parameters to gather useful information regarding the SuM.

4.5.2 Goals and Data Analysis

The main goal of this iteration is to solve **RQ2** and **RQ3**, by analyzing the data gathered, with the use of comparative analysis and different plots to highlight the pros and cons, strengths and weaknesses of the artifact. The end goal will be to deliver a working artifact that can fulfill what asked in the research questions.

5

Problem Identification

This chapter focuses on identifying the problems within the current state of monitoring in ROS2 systems. It explores the topic through a literature review and clusters the various problems identified. This approach highlights common challenges in monitoring and lays the foundation for extracting requirements for the development of the artifact.

5.1 Process

The goal of this section is to identify the problems and gaps in monitoring in ROS2, based on a literature review. This review mapping aims to discover some common issues present in analyzed papers and cluster them, with the end goal to identify requirements to specify for the artifact. The findings from this section will help answer **RQ1**, by highlighting the current limitations in monitoring, via a theoretical analysis.

5.2 Data Collection

To perform the data collection for this iteration, a literature review of the existing literature regarding the topic of monitoring applied on ROS2 has been performed. This way of collecting data allows to understand the existing challenges and future work that is still missing in this topic.

The way data has been collected is through the analysis of 15 papers mentioned in the table below. These papers all adopt ROS2 in their research.

ID	Resource Reference
R1	Abdulrahman Al-Batati, Anis Koubaa, and Mohamed Abdelkader. Ros 2 in a nutshell: A survey. Preprints, October 2024
R2	Elias E. Hartmark and Tage Andersen. Runtime verification of autonomous robotic systems in ros 2. Master’s thesis, Norwegian University of Life Sciences, Norway, 2024. 30 ECTS, Applied Robotics
R3	Ivan Perez, Anastasia Mavridou, Tom Pressburger, Alexander Will, and Patrick J. Martin. Monitoring ros2: From requirements to autonomous robots. In Proceedings of FMAS2022 ASYDE2022, volume 371, pages 208–216, 2022
R4	Gijs M.C. van den Hoven. Introducing a performance observation framework to ros2. Master’s thesis, Mathematics and Computer Science, March 2024.
R5	Christophe Bédard, Ingo Lütkebohle, and Michel Dagenais. ros2_tracing: Multi-purpose low-overhead framework for real-time tracing of ros 2. IEEE Robotics and Automation Letters, 7(3):6511–6518, July 2022.
R6	Andrea Bonci, Francesco Gaudeni, Maria Cristina Giannini, and Sauro Longhi. Robot operating system 2 (ros2)-based frameworks for increasing robot autonomy: A survey. Applied Sciences, 13(23):12796, November 2023.
R7	Jonas Peck, Johannes Schlatow, and Rolf Ernst. Online latency monitoring of time-sensitive event chains in ros2. Technical Report 202101271521-0, Institute of Computer and Network Engineering, TU Braunschweig, January 2021.
R8	Yuya Maruyama, Shinpei Kato, and Takuya Azumi. Exploring the performance of ros2. In 2016 International Conference on Embedded Software (EMSOFT), pages 1–10, 2016
R9	M.A. Roach, J. Pennney, and B.H. Jared. Exploring a supervisory control system using ros2 and iot sensors. Technical report, University of Texas at Austin, 2023.
R10	Maryam Ghaffari Saadat, Angelo Ferrando, Louise A. Dennis, and Michael Fisher. Rosmonitoring 2.0: Extending ros runtime verification to services and ordered topics. Electronic Proceedings in Theoretical Computer Science (EPTCS), 411:38–55, November 2024
R11	Jongkil Kim, Jonathon M. Smereka, Calvin Cheung, Surya Nepal, and Marthie Grobler. Security and performance considerations in ros 2: A balancing act.
R12	Yixiao Li, Yutaka Matsubara, Hiroaki Takada, Satoru Funahashi, and Hiroki Kawashima. Monitor and analyze rare ros2 performance issues with a unified tracing framework. In Proceedings of the 2024 The 6th World Symposium on Software Engineering (WSSE), pages 95–104. ACM, 2024. December 2024
R13	Musab Talha Cakin. Ros 2 data flow scheduling: Design and implementation of a cross-process real-time executor. Master’s thesis, University of Applied Sciences Technikum Wien, Wien, Austria, September 2023.
R14	Tobias Betz, Maximilian Schmeller, Harun Teper, and Johannes Betz. How fast is my software? latency evaluation for a ros 2 autonomous driving software. 2023 IEEE Intelligent Vehicles Symposium (IV), pages 1–7, Chengdu,China,June 2023. IEEE June 2023
R15	Deep Patel, Chayan Maiti, and Sreekumar Muthuswamy. Real-time performance monitoring of a cnc milling machine using ros 2 and aws iot towards industry 4.0. In IEEE EUROCON 2023 - 20th International Conference on Smart Technologies, pages 776–781, 2023.

Table 5.1: List of papers analyzed.

5.3 Data Analysis

To perform the data analysis for this iteration the gathered papers have been analyzed and searched to understand and highlight the open challenges and issue that are present in the ROS2 systems. In the section below a highlight of all the papers and the issues they faced with ROS2 is presented.

5.3.1 Papers analysis

R1: The paper “ROS 2 in a Nutshell: A Survey” [1] highlights several existing gaps in ROS 2’s real-time performance and scalability. Despite significant advancements, current solutions still face challenges in ensuring scalable, flexible, and holistic real-time performance, particularly in highly distributed systems where network delays and jitter introduce unpredictability. Additionally, while various priority-driven schedulers have been proposed, integrating them into existing ROS 2 frameworks often introduces additional overhead. There is also a growing need for tools that enable fine-grained runtime monitoring and online latency management, ensuring that real-time constraints are met under dynamic conditions.

R2: The Master thesis “Runtime Verification of Autonomous Robotic Systems in ROS 2” [29] is a research with the use of existing tools for runtime verification of requirements in ROS2, the tools used are FRET [56] and OGMA [57]. For this research the main limitation that has been found is regarding what is being monitored by the monitoring system. In this case, the monitoring system only monitors the topics produced by ROS2, but in more complex monitoring systems, other parameters are also monitored, such as external commands. By monitoring different parameters the user can understand more about the current situation, but to develop such monitoring system, more architectural decisions have to be made. This limitation has an impact in the development of a monitoring system, depending on how fine grained the monitoring requirements need to be.

R3: The paper “Monitoring ROS2: from Requirements to Autonomous Robots” [65] is a research on the topic of runtime verification and monitoring applied to ROS2. The paper highlights how writing good monitors can be challenging and as mentioned in paper R2, they use FRET and OGMA to make it more consistent and less error prone. The main limitations and challenges faced in this research were regarding the difficulties to specify what properties to monitor, being programmed in low level languages like C++. Another issue was regarding the need to make repetitive and error prone boiler plate code to allow the data flow to the monitors. The two tools find a solutions to this errors.

R4: The master thesis “Introducing a Performance Observation Framework to ROS2” [88] is a research regarding the topic of analysis of performance in ROS2. The thesis makes use of experiments, monitoring the latency and jittering of the

topics during different scenarios. The main discussion points regarding the issues found from this research are regarding the fact that measuring the performance of ROS2 application is not trivial and not standard methods exist for this evaluation. The author focus on specifying that developing a real-time or safety critical ROS2 robot is challenging and that a lot of research on the requirements regarding the criticality level must be addressed before hand.

R5: The paper “ros2_tracing: Multipurpose Low-Overhead Framework for Real-Time Tracing of ROS 2” [9] is a research on the development of a new tool called ros2_tracing which allows the developer to use tracing which is an established approach for performance analysis, popular for operating system-level performance analysis and for distributed systems, to analyze ROS2. The paper highlights some drawbacks in monitoring ROS2 with tools such as rosbag2 and similar middle-ware tools, the drawbacks being a significant resource cost in both CPU and memory usage and big effort into understanding the system. This tool can provide good diagnostics of the system by showing the stacktrace and being lightweight on the SuM.

R6: The paper “Robot Operating System 2 (ROS2)-Based Frameworks for Increasing Robot Autonomy: A Survey” [8] is another survey research regarding the development of frameworks to increase the autonomy of ROS2 based robots in complex tasks. This research bring an introspective into ROS2, highlighting some challenges in developing a framework, one of which is the double edged sword of being open-source, meaning that is constantly evolving, therefore lacking the reliability of commercial software developers, by not having customer service, nor being user friendly, and consequently not very appealing to companies. Another challenge and limitation being discussed in the paper is regarding ensuring real-time performance, because it’s granted by the DDS in the communication level, but they have to be managed by the developer on system level.

R7: The paper “Online latency monitoring of time-sensitive event chains in ROS2” [64] is a research regarding monitoring of time-sensitive events for middleware-centric architectures with end-to-end weakly-hard real-time constraints applied on ROS2. This research demonstrated via a proof of concept applied on ROS2 a new approach to solve the issue of monitoring end-to-end latencies, which cannot be sufficiently monitored with existing methods. It mentions how this new method leverages on the same processor shared memory with little overhead.

R8: The paper “Exploring the Performance of ROS2 ” [50] is a research regarding a performance analysis of the middle-ware DDS of ROS2. This research presents a lot of empirical data tables regarding various parameters such as cpu usage, memory usage, latency and jittering, it’s useful to understand the inner workings of the middle-ware to understand which processes impact the performance the most. As mentioned in the conclusions this research is valuable for many people because the

research doesn't only apply to ROS2 based systems but to any system with a DDS middle-ware.

R9: The paper “Exploring a Supervisory Control System Using ROS2 and IoT Sensors” [71] is a research with goal to demonstrate the use of ROS2 and IoT sensors to monitor a process. The main findings are from empirical results and demonstrate that the collection and recording of pyrometer sensor data at 4Hz is possible using ROS2 and allows for highly modular and scalable systems of increasing complexity. The discussion also goes into detail regarding the future research of making such monitoring possible at different frequencies based on the context of the scenarios.

R10: The paper “ROSMonitoring 2.0: Extending ROS Runtime Verification to Services and Ordered Topics”[78] is a research with objective to implement an extension of the existing ROSMonitoring framework. This framework is designed to enable runtime verification of robotic applications in ROS. This research improves the existing tool by facilitating the verification of services and accommodated ordered topics rather than solely unordered ones. This paper highlights the architecture of the system and presents the weaknesses of the framework, improving and expanding it.

R11: The paper “Security and performance considerations in ROS 2: A balancing act” [37] is a research regarding the security and performance of ROS2. The researchers go into depth into the architecture of ROS2 and evaluate the various security risks and vectors posed by such architecture. The main conclusion drawn from this research is that, ROS2 being still under rapid development and with new versions being updated and published every few months, monitoring the various implementations to check for violations is a must do, to avoid any vulnerabilities that might get introduced. Developing the tools and procedures that enable developers to deploy during and after the development process will reduce the risk of security vulnerabilities of resulting CPS systems.

R12: The paper “Monitor and analyze rare ros2 performance issues with a unified tracing framework” [43] is a research with a proof of concept regarding the implementation of an open source tracing framework. The tool similar to the previous papers, aims to efficiently monitor the trace events, and create unified trace file of each issue detected for detailed analysis. This framework has been tested and demonstrates that the CPU usage can be reduced and the system trace file can give very useful information regarding the system. Future work is still being investigated such as the development of a visualization tool for the trace files.

R13: The master thesis “ROS 2 data flow scheduling: Design and implementation of a cross-process real-time executor” [11] is a research regarding the improvement

of data flow scheduling regarding the topic of self driving car. This research raised some points regarding ROS2 executors, which face challenges in meeting real-time demands due to lacking precise timing guarantees for callback execution, their non-deterministic behavior in callback order and timing can result in unpredictable system responses, they also lack fine-grained control over callback execution sequence and timing, which is essential for synchronization. Other problems such as inefficient resource utilization, scheduling overheads, and communication complexities introduce variability in performance.

R14: The paper “How fast is my software? latency evaluation for a ROS 2 autonomous driving software” [7] is a research regarding the understanding of latencies in the software stack for autonomous vehicles. In the paper they present an evaluation workflow to inspect software and the occurring latencies for ROS 2 applications. The research highlights some of the bottlenecks in ROS2 applications.

R15: The paper “Real-time performance monitoring of a CNC milling machine using ROS 2 and AWS IoT towards industry 4.0” [62] is a research regarding a proof of concept of monitoring a CNC machine with AWS and IoT sensors. The end goal of the research was to create a general solution to improve the existing IoT solutions for industry 4.0.

5.4 Common Issues Identified

From the data gathered and analyzed in the section above, the papers have all in common different issues with ROS2. Seven common issues have been identified in the papers and are explained below.

IF1 *Real-time performance challenges:* R1, R4, R6, R13 and R14 highlight issues with real-time performance in ROS2. The unpredictability of network delays and jitter affects the real-time constraints, making it hard to ensure a deterministic execution and result. R13 faces the same issue with executors having to struggle with precise timing guarantees, leading to non deterministic behavior in call back execution. R4 and R14 highlight this issue by explaining that a lack of standardized performance evaluation methods makes it hard to measure and optimize latency and jittering.

IF2 *High overhead in monitoring tools:* R5 and R12 present the issue of existing monitoring tools like rosbag2 which introduce a significant resource cost in terms of CPU and memory usage, making them not efficient for real-time and long term analysis. R5 and R12 also present a new way of monitoring tools, tracing. This frameworks aim to reduce the overhead resource cost, but there is still future work to be done to improve this field of low latency and low resource consumption moni-

toring.

IF3 *Difficulty in Specifying Monitoring Properties:* R3 presents an issue regarding the need for writing requirements for monitoring, the authors highlight the difficulty in writing efficient monitoring scripts and defining what properties to monitor. This difficulty is also from a code point of view, with low level programming language as C++ requiring repetitive boilerplate code. R3 tries to patch this issue by using tools such as FRET and OGMA, which help to reduce the complexity needed to write monitors for ROS2 based systems. R2 also discuss the topic of FRET and OGMA which help automate the aspects of writing monitoring, but highlight the main limitation of these tools which is that they only perform topic based monitoring, which doesn't take into account external commands to achieve a more fine grained monitoring.

IF4 *Lack of Standardized Monitoring Methods:* R4, R12 and R14 all use different monitoring techniques leading to inconsistencies in results due to factors. R4 and R14 highlight how there are no universally accepted best practices for monitoring ROS2 applications, which makes it difficult to compare results across different research papers. R5,R10 and R12 connect to this issue by presenting `ros2_tracing` and `ROSMonitoring` which address parts of the problem but lack a standardized monitoring framework.

IF5 *Security Risks in Monitoring:* R11 highlights key points regarding the security risks in ROS2, by showing how the rapid development and evolution of ROS2 is a double-edged sword, by improving the system but also risking to introduce new vulnerabilities. The system must adapt to detect potential security risks, and at the moment there are no clear security focused monitoring frameworks, but mostly mainly focus on performance monitoring rather than security auditing.

IF6 *Challenges in Industrial and IoT Applications:* R9 and R15 both have a proof of concept with IoT applied in ROS2, this introduces new challenges regarding monitoring and high-frequency data collection scenarios, one mentioned is scalability.

IF7 *Open-Source Evolution and Maintainability Issues:* R6 gives highlight to the point of ROS2 being open-source, being the nature of the system constantly evolving it can introduce some challenges into the adoption of the system for customers. An example of this challenge is the need to keep up to date the versions if some vulnerability gets discovered, and unlike commercial software, ROS2 lacks a dedicated customer support, making debugging, monitoring and maintaining the system harder for new developers.

5.5 Artifact's Requirements

To develop an artifact in the field of software engineering a common approach is following the process of Software Development Life Cycle (SDLC) [77]. An important phase in developing an artifact is the requirement elicitation phase, this phase is critical to put some stakes to the artifact [27]. A crucial step before eliciting the requirements is understanding the underlying architecture and system, the data collected and analyzed in the previous section proves fundamental for this scope, it highlighted the weaknesses and gaps of ROS2.

To approach the first development of the artifact a series of Functional and Non-Functional requirements has been created.

ReqGenie

To perform requirement elicitation, the tool ReqGenie [25] has been used and its results are listed below.

The tool can be found at this link <https://chat.openai.com/g/g-xfNpHp5ju-reqgenie>.

The repository at this https://github.com/tavantzish/ReqGenie_Appendix/tree/main.

The conversation with ReqGenie can be found in the Appendix Section A.1.

5.5.1 Stakeholders

The first step in requirement elicitation is the identification of the Stakeholders. This step allows it to understand the main protagonists of this artifact in development, and better process the information to satisfy their requests.

Primary Stakeholders

- **Robotics System Developers / Engineers:** Design and implement the adaptive monitoring framework within ROS2 environments. Responsible for ensuring the system performs within defined resource constraints.
- **Researchers in Robotics and CPS:** Use the system as a framework for further study or to validate adaptive monitoring strategies in various applications.
- **Simulation and Testing Teams:** Validate the monitoring strategies using virtual environments before deploying them on physical hardware.
- **Industrial Automation Engineers:** Apply the system to real-world robots in factory or warehouse environments to improve operational efficiency and safety.

Secondary Stakeholders

- **End Users / Field Operators:** Benefit from improved reliability and reduced risk during robot operation.
- **ROS2 Open-Source Community:** Gains insights and potential contributions to improve the ROS2 platform and ecosystem.
- **Business Decision-Makers / Project Managers:** Use research outcomes to inform investment or adoption of adaptive monitoring in future robotics initiatives.
- **Safety and Compliance Auditors:** Monitor how adaptive monitoring aligns with safety-critical operations and standards.

5.5.2 Functional Requirements

To ensure the effectiveness of the Adaptive Monitoring System for ROS2 TurtleBot3, the following functional requirements have been defined. These requirements focus on real-time performance monitoring, security, scalability, and usability, ensuring the system meets the needs of ROS2 based applications. Since the artifact is divided into two artifacts the functional requirements are divided as well.

Artifact

- **FR1:** *The system shall dynamically adjust the monitoring frequencies of ROS2 topics based on a user-defined rule file.*
- **FR2:** *The system shall implement a MAPE-K feedback loop on top of an existing ROS2 monitoring system to continuously evaluate system state and apply monitoring adaptations.*
- **FR3:** *The system shall parse and apply monitoring constraints defined by the user*
- **FR4:** *The system shall provide fallback/default behaviors in the absence of a valid rule file.*
- **FR5:** *The system shall allow users to define monitoring rules using a simple language.*
- **FR6:** *The system shall use the QUPER model to let users specify quality thresholds for each monitored topic (e.g., ideal, good-enough, unacceptable).*
- **FR7:** *The system shall apply optimization logic (mathematical formulas) to resolve trade-offs between monitoring frequency and system performance.*
- **FR8:** *The system shall output a syntactically correct rule file.*

- **FR9:** *The system shall provide validation or linting features to check rule consistency and completeness before output.*

5.5.3 Non-Functional Requirements

In addition to the functional requirements, the Adaptive Monitoring System for ROS2 TurtleBot3 must also meet several non-functional requirements to ensure efficiency, security, and reliability. These requirements follow the ISO/IEC 25010 quality model ensuring that the system remains robust and adaptable in various deployment environments. The key non-functional requirements are outlined below:

Compatibility

- **NFR1:** *The system shall be compatible with ROS2 Humble and later distributions.*
- **NFR2:** *The system shall support deployment across multiple TurtleBot3 robots without requiring modification to their core ROS2 stack.*

Functional Suitability

- **NFR3:** *The system shall support topic-based monitoring in ROS2.*
- **NFR4:** *The system shall correctly apply user-defined rules to all topics defined in the rule file with no more than 1% error margin.*

Usability

- **NFR5:** *The system shall provide a graphical or CLI-based interface for rule creation that requires no knowledge of C++.*
- **NFR6:** *The system shall include clear documentation and examples for defining QUPER thresholds and writing rules.*

Maintainability

- **NFR7:** *The system shall allow for modular rule file updates without requiring a system restart.*

Portability

- **NFR8:** *The system shall run on both x86_64 and ARM64 architectures under Ubuntu 22.04.*
- **NFR9:** *The system shall support deployment in both physical TurtleBot3 hardware and Gazebo/ROS2 simulation environments.*

Safety

- **NFR10:** *The system shall prioritize monitoring of high priority topics and never throttle them below a user-defined minimum frequency.*

5.5.4 Prioritization

To perform the prioritization of the requirements the 100\$ method [30] has been used. This technique allows to give weights and priorities to the different functional and non functional requirements, with different stakeholders in mind. The end result is divided into three types of requirements: *High priority* which are the ones necessary to deliver, *Medium priority* which are the ones that are nice to have time permitting and *Low priority* which are also nice to have but can also be seen as future work.

High Priority

- **FR1, FR2:** Adaptive monitoring via user-defined rules and a MAPE-K feedback loop
- **FR3, FR4:** Rule parsing and robust fallback/default behavior
- **FR5, FR6:** User-friendly rule language + QUPER model integration
- **NFR3, NFR4:** Accurate topic-based monitoring and rule enforcement
- **NFR9:** Simulation and hardware deployment support (Gazebo + TurtleBot3)
- **FR9:** Rule validation/linting for correctness

Medium Priority

- **FR7:** Trade-off optimization using formulas
- **FR10:** Rule import/export support for reusability
- **NFR5, NFR6:** CLI/GUI for rule authoring + documentation and usability aids
- **NFR7:** Modular updates to rule files without restart
- **NFR1, NFR2:** ROS2 Humble+ compatibility + multi-robot deployment readiness

Low Priority

- **FR8:** Automatic rule generation/output (syntactic correctness)

- **NFR8:** Cross-architecture portability (x86_64 + ARM64)
- **NFR10:** Prioritization of critical topics and enforcement of minimum monitoring frequency

5.5.5 Requirements considerations

These requirements have been generated using the ReqGenie tool [25]. The tool demonstrated is capabilities by expanding existing requirements and giving insightful suggestions. To evaluate the requirements an initial phase of requirements elicitation without the tool has been performed, and then these initial requirements have been fed in the tool to evaluate. The results are impressive and the work of these tool is very valuable.

Regarding the ways to determine if a requirement has achieved its goal or has violated it, this artifact will adopt the research of Berntsson ‘‘How you choose the right level of quality for your non-functional requirements – the QUPER model’’ [82] regarding the QUPER model. This research outlines some guidelines on analyzing the benefit view with the cost view, highlighting that after a certain level, the improvement in benefit is minimal while cost grows.

6

SmartTestudo

This chapter explains the artifact, focusing on the theoretical concepts behind its implementation, the system architecture, and the development process across iterations.



Figure 6.1: Roman legion in testudo formation.

Source: <https://www.pngegg.com/en/png-ccdzs>

This quote from Zavala is part of the foundation for this research, it envisions the core of the meaning of adaptive monitoring in a simple way, by putting the definition into simple terms.

“Adaptive monitoring is a method used in a variety of domains for responding to changing conditions“ [94].

This is what the goal of the research aim to achieve, an artifact that adapts to the changing conditions applied to the TurtleBot3 simulation environments.

From the systematic mapping Zavala identified a common threat that affects every adaptive monitoring system that she analyzed, *Generalization*.

This threat is to be taken lightly though. As noted in her paper, a generic solution is not always the most effective. The goal of his research was to present the current state of the art, providing developers with a clear understanding of existing

adaptive monitoring strategies to build upon. She emphasized that ad-hoc solutions can sometimes outperform generic ones and that fine-tuning can lead to significant improvements.

6.1 Adaptive Monitoring Strategy

The main challenge in creating and developing an adaptive monitoring system is defining the underlying strategy to make the system react to the external factors and adapt. A common framework model identified in the systematic mapping from Zavala [94] is the MAPE-K model. This feedback loop is the most influential reference control model for autonomic and self-adaptive systems [3]. Developers adopt this MAPE-K model as a ease of use to specify the requirements and use it as an abstract state machine. One challenge identified in developing self-adaptive system is “Self-adaptive systems are generally difficult to specify, validate, and verify due to their high complexity and dynamic nature“ [3]. This architecture solves this challenge by dividing the different phases, there is a separation between the adaptation logic and the functional logic, allowing a easier management of the building blocks of this abstract state machine.

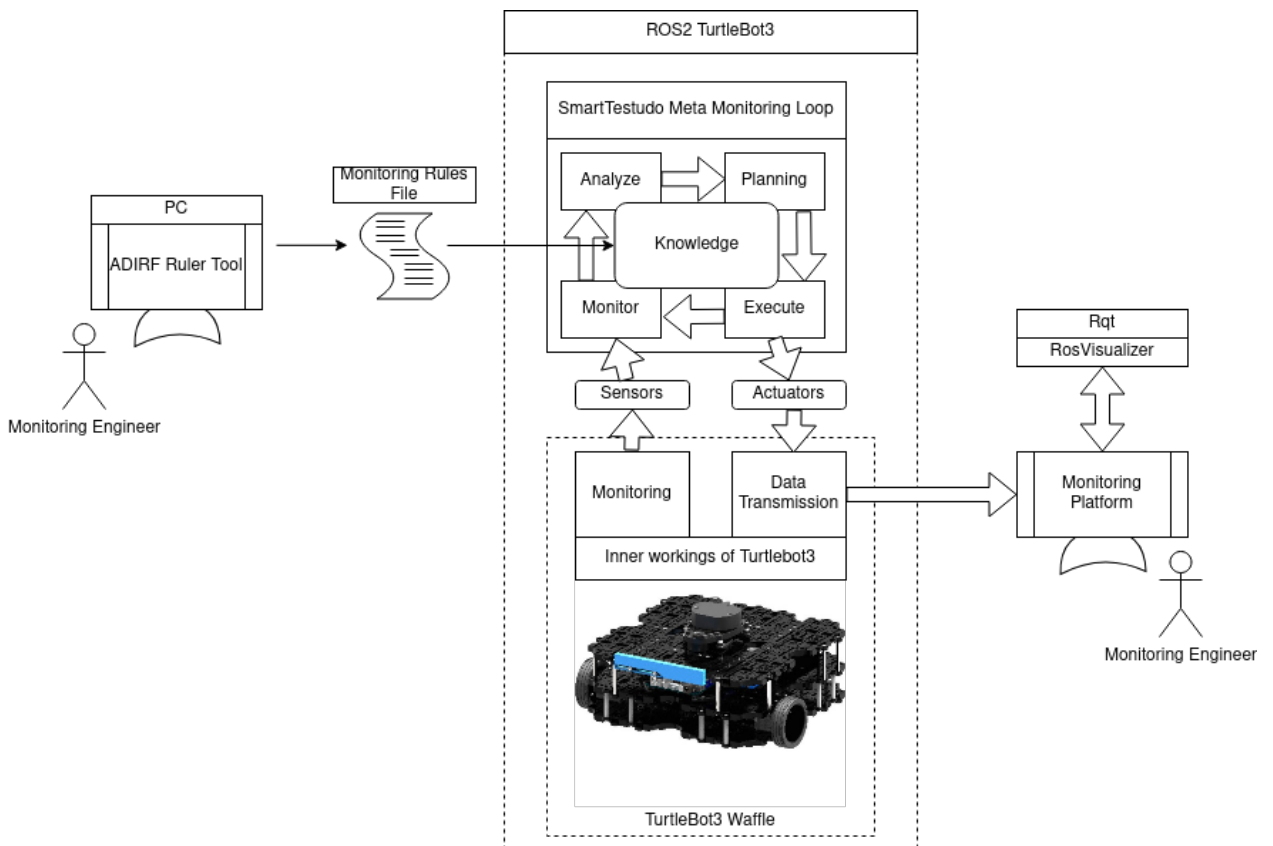


Figure 6.2: Overall artifact architecture.

Figure 6.2 shows the artifact main architecture. It consists of a meta monitoring MAPE-K loop that builds on top of the existing monitoring loop of ROS2 Turtle-

Bot3.

The meta monitoring tool SmartTestudo will work, by analyzing data from Sensors, which in this case are the existing monitoring topics, with their frequency and bandwidth. After that it will process the knowledge from the Monitoring rules specified with the use of the external tool called ADIRF Ruler, in which the user will specify the constraints in which the frequencies will be adapted. With the knowledge acquired, the system will adapt and alter the frequencies that will then get sent to the monitoring platform controlled by the engineer via tools such as Rqt mentioned in Section 2.8 or the evaluation tool RosVisualizer.

The way the artifact will work and adapt will be regarding context analysis in the different scenarios and in combination with parameters specified by the user following the QUPER model [83], this will make the algorithm evaluate and plan with the knowledge specified by the user and by modifying the frequency of monitoring based on the context it will improve the bandwidth of the TurtleBot3, reducing strain on the system.

The three main artifacts developed in this thesis are the following:

SmartTestudo: An adaptive monitoring tool that is built on top of the existing ROS2 monitoring system. It will be a meta monitoring system adopting the MAPE-K loop. The knowledge of the system will be specified through the use of DSL rules. These rules will be used as knowledge to analyze, plan and execute. In the end the actuators will modify the frequencies of the topics based on the evaluation of such rules. This is the main artifact and is shown in the figure above.

ADIRF Ruler: A GUI tool designed to facilitate the definition and configuration of monitoring rules through the usage of a defined DSL. Inside this tool a bandwidth optimizer called “OBI Bandwidth optimizer“ is present. This optimizer allows the user to input the maximum bandwidth of the system and outputs the frequency necessary to achieve that. This artifact serves as the user-facing component, abstracting away technical complexities and enabling domain experts to interact with the system in an intuitive manner. The user will have to specify his desired QUPER level for each topic to be monitored and this will allow the system to use the resources in an efficient way. This artifact is shown in the figure above, being used by the monitoring engineer, which with his knowledge of the system he will specify the rules for the thing.

Rosbag Visualizer: An external supporting tool dedicated to the visualization and inspection of recorded system data. It provides a clear and interactive representation of system behaviors over time, aiding both in debugging and in post-analysis of the monitoring process. This tool will be a group of different command line python scripts that will allow the decoding of the various rosbag2 and make analysis. This artifact is present is used in the testing phase of the monitoring tool, to plot graphs and understand the behavior.

The initial phase of development focuses on establishing a robust architecture that adheres to the principles of clean architecture and separation of concerns. To achieve this, the system is divided into three well-defined artifacts, each with a distinct responsibility within the overall workflow. This modular design ensures that components remain independently testable, extensible, and maintainable over time.

This architecture aligns with key software engineering principles such as the single responsibility principle, loose coupling and high cohesion, open-closed principle and layered architecture. This architecture not only enhances maintainability and scalability but also lays a solid foundation for future expansion, such as integration with other adaptive systems or extension to different domains.

6.2 ADIRF Ruler

A key building block in the artifact is the “ADIRF Ruler“. Following the theory noted in the Section 6, where the MAPE-K loop is explained, this tool provides the user the ability to give the necessary knowledge to the SmartTestudo adaptive monitoring system by specifying rules through the usage of a DSL. Another key feature of this tool is the ability to model the Topic frequency and bandwidth usage by using a mathematical optimizer that adapts based on how much bandwidth is available to use.

In Figure 6.3 the architecture of the system is shown. The artifact is divided into frontend and backend.

The frontend is responsible of allowing the user to create the rules by specifying parameters and how the system should adapt under which conditions. It also gives a tool to calculate the frequency of each topic based on the available bandwidth of the System under Monitoring (SuM). The technology used for the frontend is React Native.

The backend is responsible for the validation of the rules and the calculation of the optimized values. It performs this action by using the python TextX library[20] in which a DSL has been defined. To perform the optimization, the python library Pulp [85] is being used. The main framework to handle the backend is Flaks. Figure 6.3 shows on the left the Monitoring Engineer using the ADIRF Ruler tool, and on the right the architecture of the tool is zoomed and displayed.

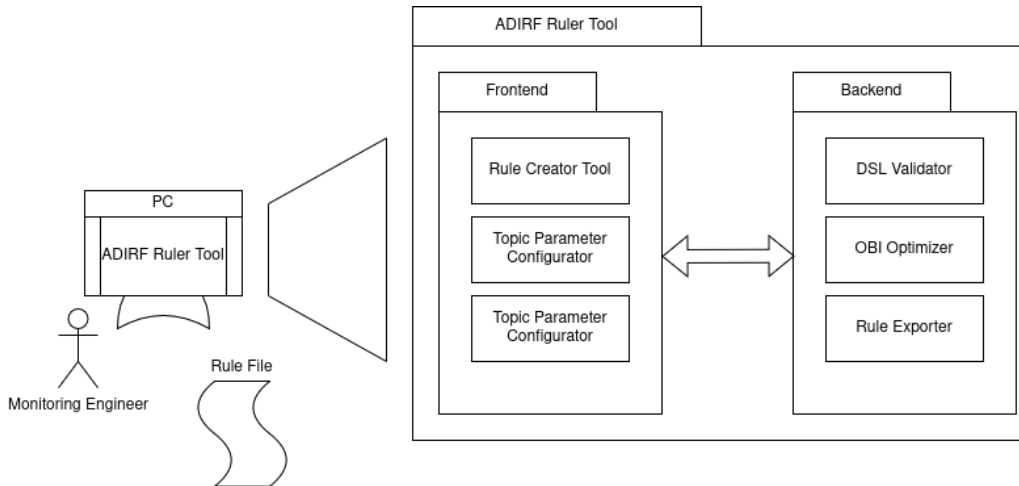


Figure 6.3: Artifact architecture - ADIRF Ruler

6.2.1 TextX ADIRF Model

Mentioned in the previous section, behind the validation and elicitation of the adaptive monitoring system rules, there is a model, a DSL. As mentioned in Section 2.10 the use of DSL has it's pros and cons, but in this case after an initial cost-estimate analysis following the Mernik [53] guide, the use of a DSL proves crucial for the specification of the different rules that the model will adapt to. After a period of analyzing the different use cases and different possible rules, a DSL model has been built and can be seen in the figure 6.4 below.

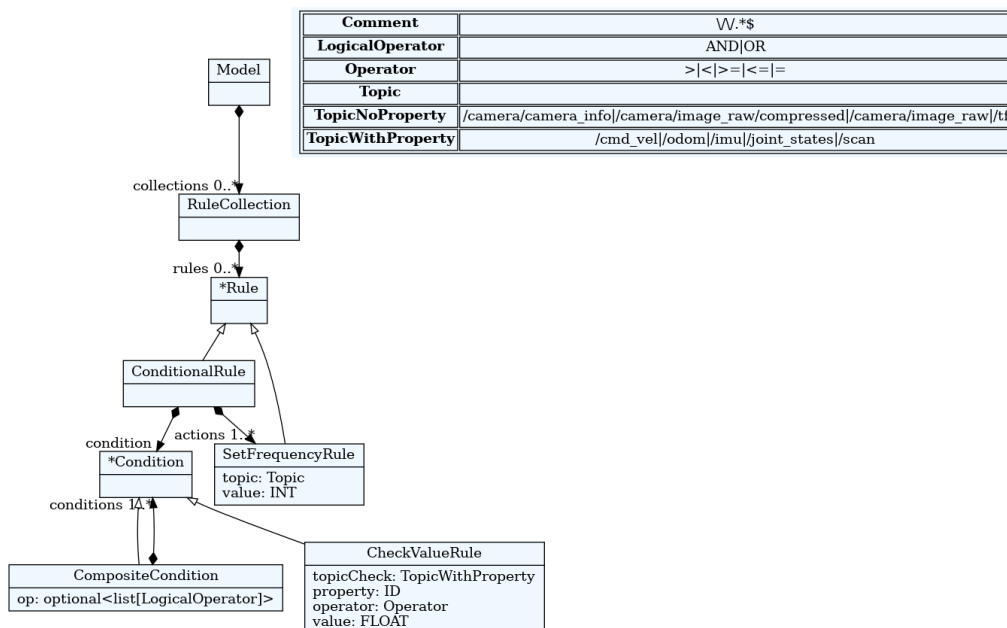


Figure 6.4: DSL ADIRF Model tree.

```
begin
  // Global optimized default values
  SetFrequency /odom 10
  SetFrequency /cmd_vel 7
  SetFrequency /scan 3
  SetFrequency /camera/camera_info 3
  SetFrequency /camera/image_raw 3
  If(
    CheckValue /odom pose_y < 5.4
    AND CheckValue /odom pose_y > 1
    AND CheckValue /odom pose_x > -13
    AND CheckValue /odom pose_x < 6
  ) Then
    SetFrequency /odom 4
    SetFrequency /cmd_vel 4

  If(
    CheckValue /odom pose_y < 25.5
    AND CheckValue /odom pose_y > 6
    AND CheckValue /odom pose_x > -13
  ) Then
    SetFrequency /odom 10
    SetFrequency /cmd_vel 10
end
```

Listing 1: DSL ADIRF Rules example.

Figure 6.4 shows the building blocks that define how the rules should be written and Listing 1 show an example of written rules. The main format of the file as show above is divided into building blocks.

Analyzing the way the syntax is formed:

Model is the whole rule document, it allows to have multiple rule collection under it. It can be seen as the main project file that works as a container that holds the different RuleCollection.

Rule Collection is the item that contains the different Rules for this particular rule set. The body of the collection starts with *begin*, then its filled with Rules that can be Conditional Rules or SetFrequency Rules, it then closes the body with a *end* statement. It can be seen as the container of the rules for each subset of adaptive actions that the system will need to perform. This allows the user to specify different block of rules to help de-clutter the syntax and reduce errors.

Rule is the object that contains the specification on what to perform and how, a

Rule can be a Conditional Rule or a SetFrequency Rule. Multiple rules are present under a rule collection and the goal is to provide knowledge to the adaptive monitoring system.

SetFrequency Rule is a type of rule which role is to assign the frequency of the topic. The way its syntax is simple “SetFrequency /topicName INT“. This operation directly assigns the value to the adaptive monitoring system without any condition. It’s mainly used on topic which frequency can be reduced directly, and on topics that have been optimized with the OBI Bandwidth Optimizer.

Conditional Rule is a type of rule which role is to have a condition to evaluate inside and then set a frequency. This is the main building block that allows the user to specify it’s conditions with logical operators.

Condition is a component that can be either a Composite Condition or a Check Value Rule. It’s role is to be inside a conditional rule and be used to perform an action.

Composite Condition is a type of Condition which allows the evaluation of multiple Conditions with the use of logical operators. This evaluation allows to create more complex logic inside the rules.

Check Value Rule is a type of Condition which allows the user to check the value of a property of a topic and perform a comparison with the logical operators. It’s the lowest building block and it’s key to perform adaptive monitoring, since it allows to specify different scenarios and condition checks.

6.2.2 ADIRF Rule Tool GUI

The objective of this tool is to reduce the skill level required to program a monitoring tool. It does so by proposing a simple and intuitive GUI which allows the Monitoring Engineer to create its own set of rules, which the monitoring system will adapt itself to.

The GUI is structured in a simple and accessible way, the user is welcomed by the home screen in Figure A.1 and has three main buttons.

The **Create New Project** button allows the user to create a rule project, clicking this button will open the screen as seen in Figure 6.7.

The **Configure Topic Parameters** button, opens the screen seen in Figure A.2, which allow the user to configure the topics different required frequencies to setup both the QUPER levels and the bandwidth limits.

The **Documentation** button allows the user to read the documentation and see how the tool works.

After creating a new project the screen presented in Figure 6.7 is show. This view allows the user to start customizing his rule file, by adding different blocks. Such blocks are explained in the DSL model above, and are the Add global default block and the conditional block.

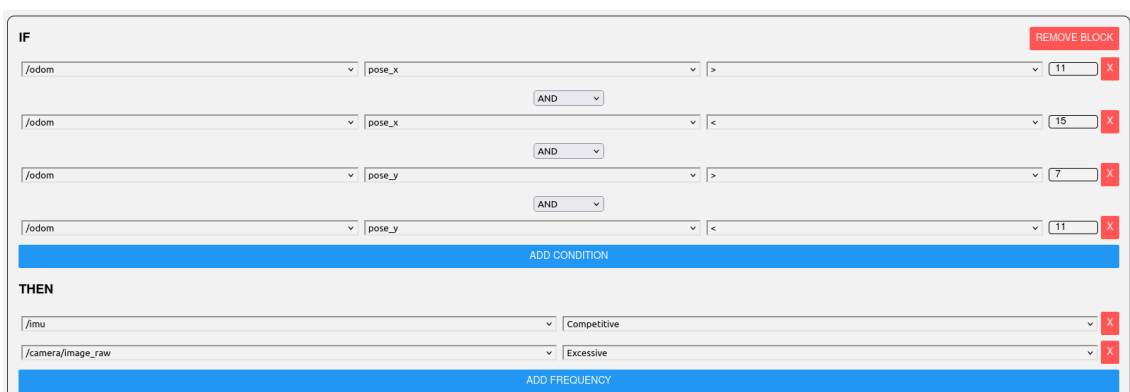
The **Global default rule** allows the user to set the global default QUPER OBI optimized value of the frequency of a topic. The user adds a block, selects the topic and then the monitoring level that he wants to have for that topic. The monitoring level of each topic is specified in the topic configuration screen and the levels are mentioned in Table 6.1. If the user doesn't specify the value of a topic, it will be by default put with the least amount of resources.



The screenshot shows a rectangular window titled "Set Frequency" with a close button (red 'X') in the top right corner. Inside the window, there are two dropdown menus. The first dropdown menu has the text "/cmd_vet" selected. The second dropdown menu has the text "Desirable" selected.

Figure 6.5: Default rule block

The **Conditional rule block** allows the user to create some conditional logic based on the values from other topics and then alter the frequency of other topics following the QUPER model defined. This block is the most useful in defining a logic, which can be more or less complex depending on the amount of logical operators added.



The screenshot shows a complex interface for creating a conditional rule block. It is titled "IF" and has a "REMOVE BLOCK" button in the top right corner. The interface is divided into two main sections: "IF" and "THEN".

IF Section: This section contains four rows of conditions. Each row consists of a topic dropdown menu, a variable dropdown menu, a comparison operator dropdown menu, and a value input field with a red 'X' button to its right. The conditions are:

- Row 1: "/jodom" > "pose_x" > ">" > "11"
- Row 2: "/jodom" > "pose_x" > "<" > "15"
- Row 3: "/jodom" > "pose_y" > ">" > "7"
- Row 4: "/jodom" > "pose_y" > "<" > "11"

Between the first and second rows, and between the third and fourth rows, there is an "AND" dropdown menu. Below the fourth row is a blue button labeled "ADD CONDITION".

THEN Section: This section contains two rows of actions. Each row consists of a topic dropdown menu, a frequency dropdown menu, and a red 'X' button to its right. The actions are:

- Row 1: "/imu" > "Competitive"
- Row 2: "/camera/image_raw" > "Excessive"

Below the second row is a blue button labeled "ADD FREQUENCY".

Figure 6.6: Conditional rule block

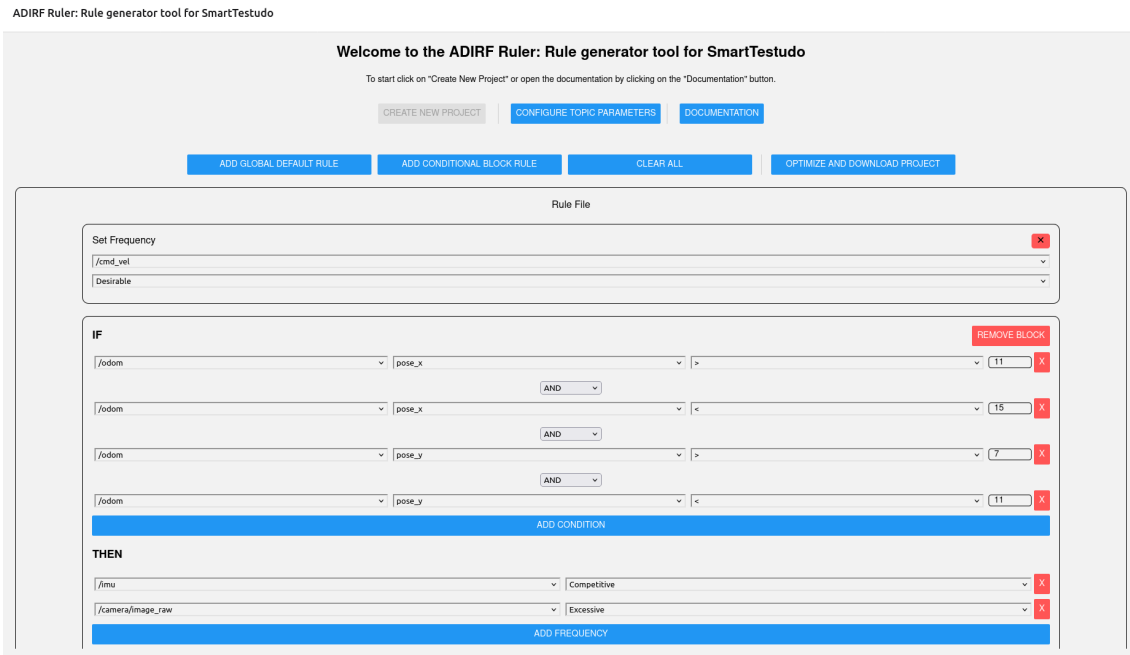


Figure 6.7: ADIRF Ruler - GUI Overview

After having created the rule project, the User can export and download the project, by clicking the **Optimize and Download** button, which will run the OBI Optimizer, and replace the QUPER model specified values with the optimized frequency values.

Considerations on the ADIRF tool

This tool is a proof of concept of how a simple and accessible GUI can make creating custom rules simpler. The building mechanic is based off the Scratch [49] programming language, which allows the user to program by connecting different building blocks and perform actions. Scratch has proven really successful in making the learning process easier [23] that's why a similar approach has been chosen. Being devised which such block based architecture can presents some limitations in writing more complex rules, further investigation on the topic in future iteration is needed to fill this gap. As a proof of concept some aspects regarding accessibility and UI/UX principles have been partially ignored, the plan is to build on the knowledge gathered from this artifact and create a new improved version, addressing all the issues found.

6.2.3 ADIRF Rules use case examples

With the model explained above, the user can specify different custom rules with different conditions, based on the requirements and scenarios. Some examples are:

Different Context Monitoring Sectors: The use of the Check Value Condition

on the odometry topic allows the user to create different monitoring zones, in which the robot will reduce or increase the monitoring frequency.

```
begin
  If(
    CheckValue /odom pose_y < 5.4
    AND CheckValue /odom pose_y > 1
    AND CheckValue /odom pose_x > -13
    AND CheckValue /odom pose_x < 6
  ) Then
    SetFrequency /odom 4
    SetFrequency /cmd_vel 4
end
```

Listing 2: Topic values with zone constraints.

Topic values: With the Check value condition the user can do as he wants to change the topic frequency based on topic conditions. A simple example being the increase of distance with the LiDAR sensor, can cause a increase in monitoring in the `cm_vel` topic. This allows a lot of different conditions to exist.

```
begin
  If(
    CheckValue /odom pose_y > 1
  ) Then
    SetFrequency /odom 2
end
```

Listing 3: Topic values with check value condition.

Topic values with multiple conditions: Similar to the sectors and the topic values, this technique allows to have multiple condition monitoring, allowing the user a more complex use case scenario.

```
begin
  If(
    CheckValue /cmd_vel vel > 2
    AND CheckValue /scan distance > 3
  ) Then
    SetFrequency /odom 2
end
```

Listing 4: Topic values with multiple conditions.

Battery-Aware Monitoring: Adapt frequency based on battery level to conserve power when it's low, and increase it when there's more capacity.

```
begin
  If(
    CheckValue /battery_state level < 20.0
  )Then
    SetFrequency /camera/image_raw 2
    SetFrequency /scan 3
  end
```

Listing 5: Battery aware monitoring.

Emergency Situations / Safety Triggers: Increase frequency dramatically in potentially dangerous or emergency situations.

```
begin
  If(
    CheckValue /scan distance < 0.5
    OR CheckValue /odom speed > 2.5
  )Then
    SetFrequency /cmd_vel 100
    SetFrequency /scan 50
  end
```

Listing 6: Emergency situations.

With these building blocks provided by the DSL different combinations of possible conditions can be made. These blocks allows the Monitoring Engineer to customize the system as much as needed in any situation.

6.2.4 OBI Bandwidth Optimizer

The second module in the ADIRF Ruler is the OBI Bandwidth optimizer. This tool allows the user to optimize the topics frequency based on an optimization strategy taking into account the total bandwidth that the system can handle. By using mathematical optimization explained in Section 2.11 and providing the different parameters necessary to run the tool, it will optimize the frequency values and allow the user to select them to apply in the rules.

This module is the key part of this research, it optimizes the frequencies guaranteeing that the constraints are not violated. It does so by using the QUPER model level specified by the user and works around that. For the evaluation of the tool, two other mathematical optimization models were devised, but in the end only the QUPER model remained. The other models were based on bandwidth budget, and on priority.

Bandwidth budget mathematical optimization

The Bandwidth budget mathematical model, takes into consideration each topic without any priority and optimizes their frequency based on the total bandwidth available specified by the user. The user specifies the topic frequency max - min, topic message sizes and total bandwidth available and the system calculates the optimized parameters. Below the mathematical problem is explained.

We are given a set of ROS topics with the following parameters:

- s_i : message size of topic i in KB
- f_i : frequency of topic i in Hz (decision variable)
- f_i^{\min}, f_i^{\max} : minimum and maximum allowable frequency for topic i

Let T be the set of all topics. The total bandwidth consumed by a topic i is $s_i \cdot f_i$ (in KB/s). We are given a total bandwidth budget B (in KB/s).

Objective: Maximize the total frequency across all topics:

$$\max_{f_i} \sum_{i \in T} f_i$$

Subject to:

$$\begin{aligned} \sum_{i \in T} s_i \cdot f_i &\leq B && \text{(Bandwidth constraint)} \\ f_i^{\min} &\leq f_i \leq f_i^{\max} && \forall i \in T \quad \text{(Frequency bounds)} \end{aligned}$$

The data that we know:

- $B = 15000$ KB/s (bandwidth budget)
- Each topic has a known s_i , f_i^{\min} , and f_i^{\max}

The result gives an optimal allocation of frequencies f_i that maximizes the total frequency while staying within the bandwidth limit.

Topic priority mathematical optimization

The Topic priority mathematical model uses the same approach as seen above but adds the possibility for the user to add a priority to which topic should be optimized last and which should be more optimizable. This addition allows the user to have a more fine grained control over the optimization process. The user specifies same as

before, the topic frequency max - min, topic message sizes, total bandwidth budget and then a priority list of the topics, after this the system calculates the optimized parameters. Below the mathematical problem is shown.

We are given a set of ROS topics with the following parameters:

- s_i : message size of topic i in KB
- f_i : frequency of topic i in Hz (decision variable)
- f_i^{\min}, f_i^{\max} : minimum and maximum allowable frequency for topic i
- w_i : priority weight of topic i

Let T be the set of all topics. The bandwidth consumed by topic i is $s_i \cdot f_i$ (in KB/s). We are given a total bandwidth budget B .

Objective: Maximize the *weighted* total frequency across all topics:

$$\max_{f_i} \sum_{i \in T} w_i \cdot f_i$$

Subject to:

$$\begin{aligned} \sum_{i \in T} s_i \cdot f_i &\leq B && \text{(Bandwidth constraint)} \\ f_i^{\min} &\leq f_i \leq f_i^{\max} && \forall i \in T \quad \text{(Frequency bounds)} \end{aligned}$$

Notes:

- $B = 15000$ KB/s (bandwidth budget)
- The weights w_i adjust the importance of maximizing frequency for each topic.
 - $w_i > 1$: more important
 - $w_i = 1$: neutral
 - $w_i < 1$: less important

This formulation allows prioritizing some topics over others in the frequency allocation while staying within bandwidth constraints.

QUPER Model mathematical optimization

The QUPER Model based mathematical model uses the same approach as the Bandwidth budget, but includes the QUPER model into the frequency ranges. The user

inputs the different frequency ranges of each topic, following Table 6.1 and selects how the optimizer should behave, selecting the different modes, after that the system calculates the optimal frequencies for the quality level selected. Below the mathematical problem is shown.

Let T be the set of topics to optimize. For each topic $i \in T$, we are given:

- s_i : message size (KB)
- $\text{tiers}_i = [\tau_i^0, \tau_i^1, \dots, \tau_i^5]$: a vector of 6 tier thresholds (in Hz)
- d_i : desired tier (an integer from 0 to 5)

Let f_i be the frequency variable (in Hz) to optimize for each topic i .

Tier-Based Frequency Bounds:

Each tier d_i defines a frequency interval:

$$f_i \in \begin{cases} [0, \tau_i^0] & \text{if } d_i = 0 \quad (\text{Unacceptable}) \\ [\tau_i^{d_i-1}, \tau_i^{d_i}] & \text{if } 1 \leq d_i \leq 4 \\ [\tau_i^5, \tau_i^5 + \epsilon] & \text{if } d_i = 5 \quad (\text{Excessive}) \end{cases}$$

where ϵ is a small constant buffer for tier 5 (e.g., $\epsilon = 10$).

Bandwidth Constraint:

Let B be the total bandwidth budget (KB/s), and let $F \subset T$ be the subset of topics with fixed frequencies \hat{f}_j .

Then the constraint is:

$$\sum_{i \in T \setminus F} s_i \cdot f_i + \sum_{j \in F} s_j \cdot \hat{f}_j \leq B$$

Objective Function:

The goal is to maximize the total frequency of topics being optimized:

$$\max_{f_i} \sum_{i \in T \setminus F} f_i$$

Summary:

This optimization is solved multiple times:

- Once globally with no fixed topics ($F = \emptyset$)

- Then for each conditional block, optimizing a subset of overridden topics, while treating others as fixed at global values

Quality Level	QUPER Interpretation
Unacceptable	Below Utility Breakpoint
Marginal	Approaching Utility
Acceptable	At Utility Breakpoint
Desirable	Between Utility and Differentiation
Competitive	At Differentiation Breakpoint
Excessive	Beyond Saturation Breakpoint

Table 6.1: Formalized Quality Levels Based on the QUPER Model

6.2.5 Consideration from mathematical models

During the development of the ADIRF artifact, the three mathematical models explained above have been tested and analyzed internally. However after consideration, to conduct the research only the QUPER model has been used and implemented in the final ADIRF Ruler tool.

6.3 SmartTestudo

The second artifact for this thesis is the SmartTestudo adaptive monitoring system. This tool is responsible for parsing the rules produced by the ADIRF Ruler and apply them on the SuM. Explained in Section 3.5 and Section 6 this artifact acts as a MAPE-K loop, by having the knowledge passed from the DSL rules specified and built by the user and optimized by the OBI optimizer tool.

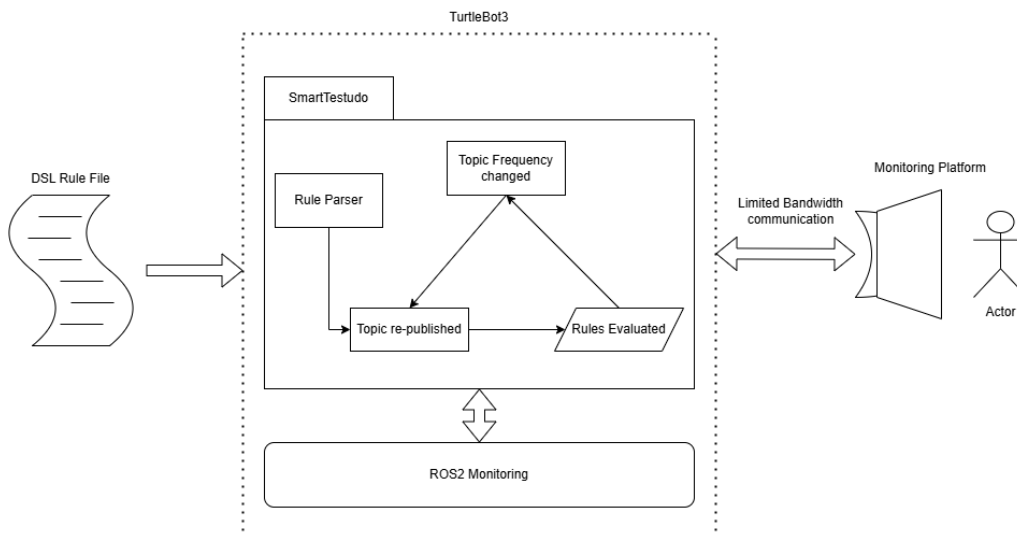


Figure 6.8: SmartTestudo Architecture

The main inner working of SmartTestudo are simple:

Rule parser: This module is responsible for parsing the rules validating them through the use of the same DSL created in ADIRF Ruler. This block saves the rules for later use.

Topic republished: SmartTestudo is built on top of the existing ROS2 monitoring system, not directly modifying it, so the initial step that the system performs is the republishing of the existing ROS2 topics. It can be seen as an adaptive system applied on what data will be sent to the external monitoring platform, so a meta-monitoring system. Each topic is republished through the use of a thread.

Rules evaluated: A common thread handles the evaluation of the rules.

Topic frequency changed: After the evaluation of the rules the conditions are applied to the topic's frequencies and change happens.

Initial Topic Analysis

Table 6.2 and Table 6.3 list the TurtleBot3 topics that are being used as a baseline for this thesis research. They have been initially analyzed using the commands found below.

Frequency:

```
ros2 topic hz /cmd_vel
```

```
average rate: 9.893
```

```
min: 0.095s max: 0.110s std dev: 0.00112s window: 10000
```

Topic Name	Default Frequency
/cmd_vel	10 Hz
/odom	25.5 Hz
/camera/camera_info	29 Hz
camera/image_raw/compressed	28 Hz
camera/image_raw	29 Hz
/imu	195 Hz
/joint_state	5 Hz
/scan	5 Hz
/tf	30 Hz

Table 6.2: TurtleBot3 Topic Frequencies

Bandwidth:

```
ros2 topic bw /camera/image_raw
```

Subscribed to [/camera/image_raw]

2.36 MB/s from 2 messages

Message size mean: 6.22 MB min: 6.22 MB max: 6.22 MB

Topic Name	Summary (Freq, Bandwidth, Msg Size)
/cmd_vel	518 B/s, 52 B (mean/min/max)
/odom	0.66 KB/s, 0.72 KB
/camera/camera_info	11.00 KB/s, 0.38 KB
camera/image_raw/compressed	11.64 MB/s, 0.42 MB
camera/image_raw	23.17 MB/s, 6.22 MB
/imu	195 Hz, 64.75 KB/s, 0.32 KB
/joint_state	3.48 KB/s, 0.12 KB
/scan	4.68 KB/s, 2.94 KB
/tf	11.20 KB/s, 0.17 KB (0.11–0.21 KB)

Table 6.3: TurtleBot3 Topic Bandwidths, and Message Sizes

6.3.1 Setup and Launch of SmartTestudo Package

The SmartTestudo is a custom ROS2 component developed within the turtlebot3_ws workspace. Below is a step-by-step guide on how to build and run the package.

Building the Package

After cloning the repository, navigate to the root of the workspace and build the package using colcon:

```
cd ~/AdaptiveMonitoring-TurtleBot3/turtlebot3_ws
colcon build --packages-select smarttestudo
```

This command compiles only the SmartTestudo package, reducing build time and isolating it from other workspace packages.

Sourcing the Workspace

After building, source the workspace to overlay the newly built packages onto the environment:

```
source install/setup.bash
```

Running the Node

To launch the main executable node from the package, use:

```
ros2 run smarttestudo smart_testudo
```

This command starts the SmartTestudo node from the SmartTestudo package. Ensure that any required dependencies or launch configurations are already set up in your environment.

6.4 Rosbag visualizer

The third artifact is a support tool called Rosbag Vizualiser. This tool allows the user to plot graphs and analyze the rosbag2 files generated during the simulation scenarios. This tool is a support command line tool for the SmartTestudo but can be used to decode any rosbag2 file independently of the SuM. Figure 6.9 gives a good overview on the architecture with the different functions that the tool should perform.

This tool is a support script suite that will allow the user to produce three different outputs based on what he chooses.

Topic graphs: This will output the frequency graph of each topic in the system, allowing the user to the changes in monitoring during the execution.

Bandwidth graphs: This will output bandwidth graph and will allow the user to see the usage of each topic during the execution of the system.

Comparison graphs: This will output a comparison graph of selected topics between two rosbag2 fed as input. This gives the user the ability to compare data on the same graph.

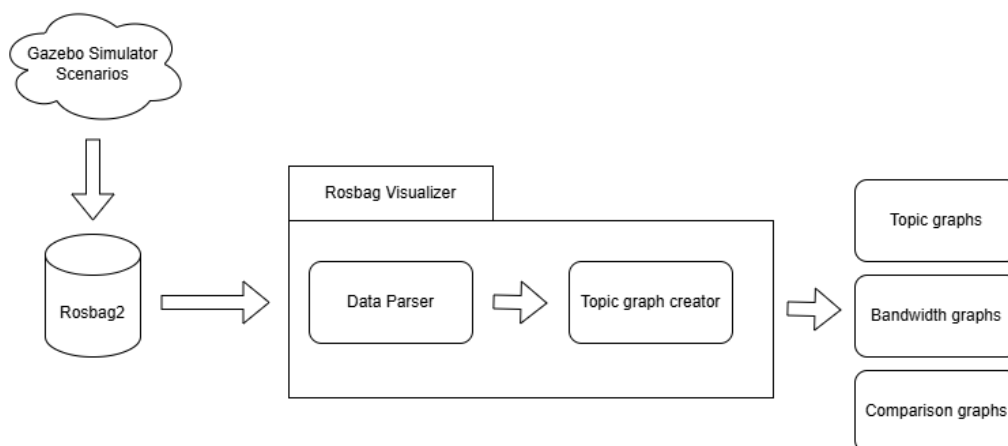


Figure 6.9: Rosbag Vizualizer Architecture

The input file produced by the Gazebo Simulator scenario with the tool rosbag2 is discussed in Section 2.9.

6.5 Artifact Code Repositories

The different tools can be found in their own GitHub repository.

- **ADIRF Ruler:**
<https://github.com/AndreaBForce/ADIRFRuler>
- **SmartTestudo:**
<https://github.com/AndreaBForce/SmartTestudo-AdaptiveMonitoring>
- **RosBag Visualizer:**
<https://github.com/AndreaBForce/SmartTestudoSupportToolVisualizer>

7

Findings

This chapter presents and displays the findings from the different iterations.

7.1 Data collection from simulator

To perform a qualitative analysis of the artifact, the SmartTestudo adaptive monitoring system has been tested in warehouse environment in the Gazebo simulator, as mentioned in the methodology Section 4.

7.1.1 SmartTestudo configuration

To setup the SmartTestudo as mentioned in the sections above, a configuration file with the different behavior rules has to be passed into the system. To generate this rules file the tool ADIRF Ruler has to be used. The tool allows the user to optimize each topic monitoring frequency/bandwidth values based on performance requirements needed.

For the validation of this artifact the current rule file has been devised.

```
begin
  // Global optimized default values
  SetFrequency /odom 10
  SetFrequency /cmd_vel 7
  SetFrequency /scan 3
  SetFrequency /camera/camera_info 3
  SetFrequency /camera/image_raw 3
  SetFrequency /camera/image_raw/compressed 3
  SetFrequency /imu 10
  SetFrequency /joint_states 3
  SetFrequency /tf 10

  // Security zone 2 - Warehouse Corridor
  If(
    CheckValue /odom pose_y < 5.4
    AND CheckValue /odom pose_y > 1
    AND CheckValue /odom pose_x > -13
    AND CheckValue /odom pose_x < 6
  ) Then
    SetFrequency /odom 4
    SetFrequency /cmd_vel 4
    SetFrequency /scan 4

  // Security zone 3 - Scaffolding Corridor
  If(
    CheckValue /odom pose_y < 25.5
    AND CheckValue /odom pose_y > 6
    AND CheckValue /odom pose_x > -13
    AND CheckValue /odom pose_x < 6
  ) Then
    SetFrequency /odom 10
    SetFrequency /cmd_vel 10
    SetFrequency /scan 10
    SetFrequency /camera/image_raw/compressed 10
end
```

Listing 7: Rule files used for artifact evaluation.

The rule file above allows the testing of the SmartTestudo artifact. The current file is divided into two sections, using the features explained in the ADIRF Section. The first part regards the global optimized topic values, after having specified the QUPER model requirements for each topic, the tool optimizes the values and outputs the current global defaults. These frequency values stay operative until a new condition is broken.

The second half regards the context based monitoring part, the user can specify different zones where the monitoring system should behave differently. With these initial configuration the bandwidth data can be collected.

7.1.2 Warehouse environment

A critical part in any simulation based research is to get the collected data as close to reality as possible. Gazebo comes in aid with it's simulation physics model that renders a TurtleBot3 in all it's glory. To make the simulation real, a world with an industrial warehouse has been created. The world can be seen in Figure 2.2. The Figure 7.1 below shows the warehouse environment with highlighted the different security zones to visually aid where are the different context changes.

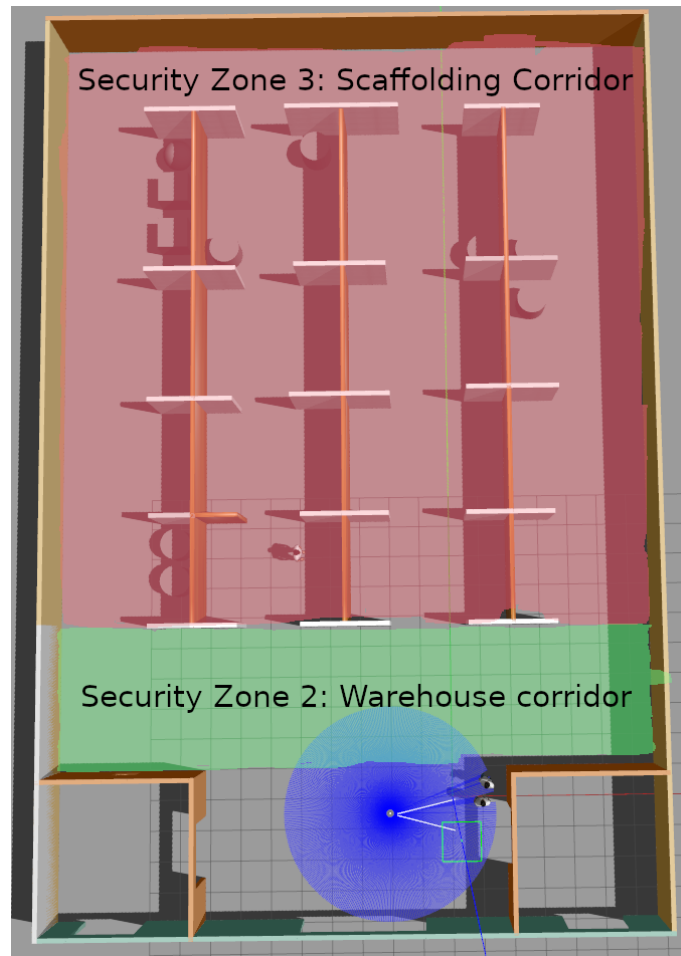


Figure 7.1: Warehouse environment with security zones highlighted.

7.1.3 Data collection

The most critical challenge in collecting data to compare is to have the same conditions in the simulation environment, to reduce bias in the collected data. The approach used to grant this requirement is through the use of Rosbag2 tool explained in Section 2.9.

The initial step was the recording of a rosbag with the common path recorded, after running the environment and started the recording a path was created and save in the bag. After that two different configurations data have been collected. For each simulation, the rosbag was replayed and the movements were applied to the

TurtleBot3, in parallel a new rosbag was being recorded and data collected. Below are the two simulations environments.

No Adaptive monitoring: This configuration was used to get the default data of the TurtleBot3, to have it as a baseline to compare.

SmartTestudo with rules specified above: This configuration allowed the testing of the adaptive monitoring system to see how it would react and what improvements it would bring.

After having collected the data from the simulations, the rosbags are parsed using the scripts present in the RosVisualizer tool, and graphs are plotted. The graphs can be seen in the Findings section.

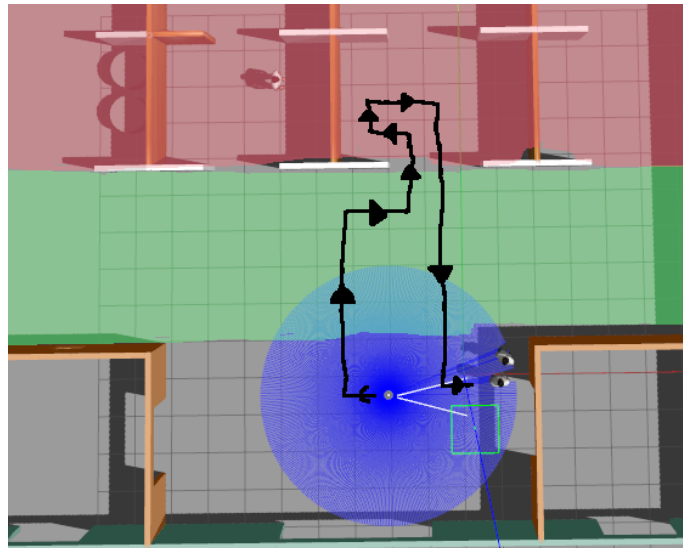


Figure 7.2: TurtleBot3 data collection path.

7.2 Simulations Findings

From the data gathered from the TurtleBot3 warehouse scenario simulation in Gazebo explained in Section 4, the results are displayed below and show clear the benefits of adopting an adaptive monitoring systems as a solution.

7.2.1 Data analysis from simulations

To evaluate the effectiveness of the SmartTestudo adaptive monitoring system, we analyzed simulation data using the **RosVisualizer** tool, which generated graphs of message frequency and bandwidth for each monitored topic. Each topic has its own graph, and its behavior is discussed in the sections below. For context, all monitored topics are introduced in Section 2.5.

Evaluation Goal: The goal of this analysis is to determine how the SmartTestudo system adapts topic frequencies based on environmental context, and whether these adaptations lead to meaningful differences compared to the default (non-adaptive) monitoring behavior.

Evaluation Approach: To support this goal, we compare the performance of:

- The default monitoring configuration (no adaptation).
- The adaptive SmartTestudo configuration (with rule-based frequency changes).

For each topic, we analyze both frequency and bandwidth over time, focusing on how SmartTestudo’s rule-based system affects communication behavior.

Structure of Analysis: To make the comparison clearer, we group topics based on how their frequency was adapted:

- **Dynamic Frequency Adaptation:** Topics whose frequency changes dynamically during runtime, based on TurtleBot3’s position or contextual triggers defined in the rule file explained above.
- **Static Frequency Adaptation:** Topics whose frequency was adapted once at the start or changed globally, without further dynamic adjustments during runtime.

This structure helps highlight the benefits and limitations of SmartTestudo’s adaptation strategy in different scenarios.

7.2.1.1 Topics with Dynamic Frequency Adaptation

This group includes topics whose frequency changes depending on the rule file mainly of TurtleBot3’s location within the simulated environment. These changes are explicitly defined in the SmartTestudo rule file and reflect a responsive adaptation mechanism for monitoring sensitive areas.

Topics in this group:

- `/cmd_vel`
- `/odom`
- `/scan`
- `/camera/image_raw/compressed`

Behavior Summary: As the TurtleBot3 moves in the warehouse scenario passing through different security zones, the monitoring frequency adapts and changes. In the default zone, each topic operates with the global QUPER optimized frequency values (7Hz for `/cmd_vel`, 10Hz for `/odom` etc.). When the robot enters the Warehouse Corridor (green zone), the frequencies drop to reduce monitoring overhead.

7. Findings

In the Scaffolding Corridor (red zone), the frequencies increase to provide higher-resolution data in a higher-risk area.

This behavior can be seen best for Frequency in Figures 7.3, 7.5, A.4 and A.6, while for Bandwidth analysis in Figures 7.4, 7.6, A.5 and A.7. where the global QUPER optimized frequency values are being applied in the beginning, and then as the TurtleBot3 changes areas the change of frequency can be seen with an increase or reduction.

Observation: In `/cmd_vel`, the frequency starts at 7Hz. As the TurtleBot enters the green zone, it drops to 4Hz, and once in the red zone, it increases to 10Hz, as shown in Figures 7.3 and 7.4. This dynamic behavior mirrors the rule conditions precisely and validates the adaptive policy mechanism implemented via SmartTestudo. This same pattern can be seen in the other topics, the graphs represent similar features but with different values specified in the rule file.

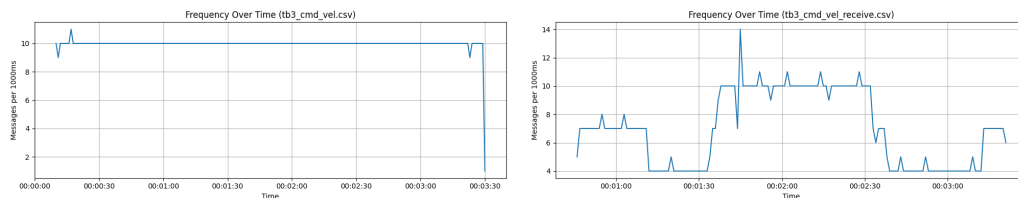


Figure 7.3: `/cmd_vel` Topic Frequency Analysis.

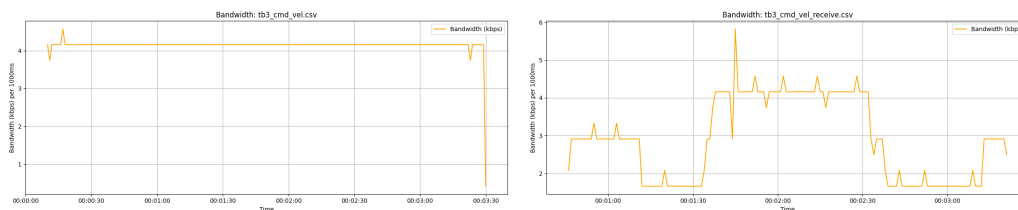


Figure 7.4: `/cmd_vel` Bandwidth Frequency Analysis.

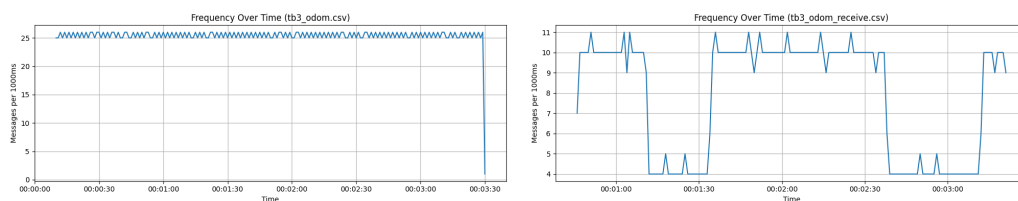


Figure 7.5: `/odom` Topic Frequency Analysis.

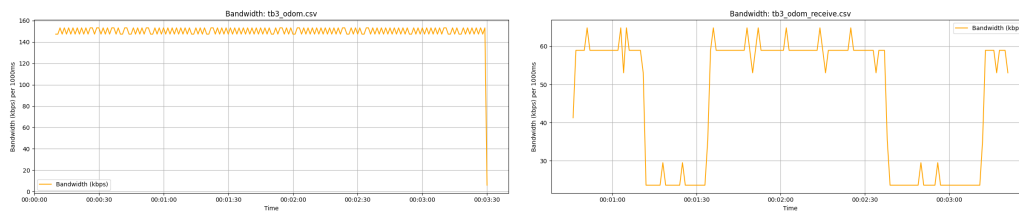


Figure 7.6: /odom Topic Bandwidth Analysis.

7.2.1.2 Topics with Static Frequency

These topics are only defined once in the rule file with a global default and remain constant throughout the simulation. No conditional adaptation is applied.

Topics in this group:

- /imu
- /joint_states
- /tf
- /camera/image_raw
- /camera/camera_info

Behavior Summary: The frequency and bandwidth of these topics remain steady across all scenarios. This is because the SmartTestudo rule file does not specify dynamic adjustments based on the robot's position for these topics. This pattern can be seen in the graphs below, for both Frequency in Figure 7.7, 7.9, A.8, A.10 and A.12 and in Bandwidth Figure 7.8, 7.10, A.9, A.11 and A.13

Observation: The /imu topic operates at 10Hz throughout the simulation with consistent bandwidth usage (as shown in Figures A.8). This indicates that it is not considered sensitive to location-based context changes or that its data granularity is already deemed sufficient.

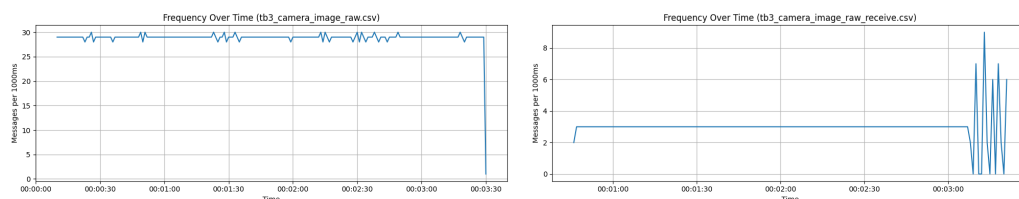


Figure 7.7: /camera/image_raw Topic Frequency Analysis.

7. Findings

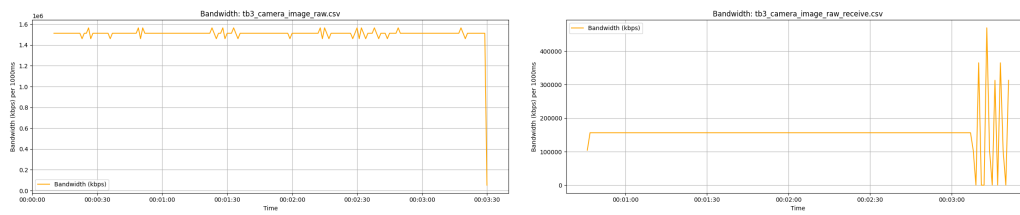


Figure 7.8: /camera/image_raw Topic Bandwidth Analysis.

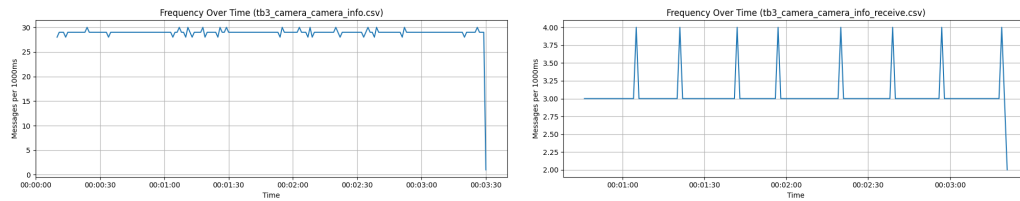


Figure 7.9: /camera/camera_info Topic Frequency Analysis.

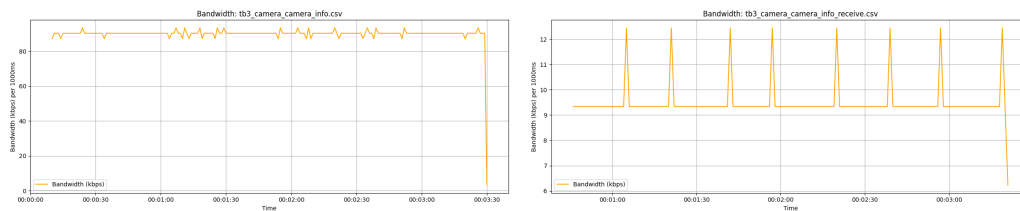


Figure 7.10: /camera/camera_info Topic Bandwidth Analysis.

7.2.1.3 Frequency analysis table

Table 7.1 shows the average frequency for each topic during the simulation scenario. The data displays what's already seen in the different graphs from the section above but gives a number regarding the percentage change in the frequency.

Topic	Default (Hz)	SmartTestudo (Hz)	Change (%)
tb3_camera_camera_info_receive	28.85	3.04	-89.45
tb3_camera_image_raw_compressed_receive	28.87	5.47	-81.06
tb3_camera_image_raw_receive	28.85	2.99	-89.64
tb3_cmd_vel_receive	9.95	7.00	-29.65
tb3_imu_receive	189.02	10.02	-94.70
tb3_joint_states_receive	4.98	3.04	-38.86
tb3_odom_receive	25.37	7.83	-69.15
tb3_scan_receive	4.98	5.87	+18.02
tb3_tf_receive	29.86	10.03	-66.43

Table 7.1: Frequency Comparison Between Default and SmartTestudo

7.2.1.4 Bandwidth analysis table

Table 7.2 highlights the improvements in bandwidth usage. It shows how the adaptive monitoring works and improve the transmitted kbps data by adapting it to the context of what the user needs.

Topic	Default (kbps)	SmartTestudo (kbps)	Change (%)
tb3_camera_camera_info_receive	89.78	9.48	-89.45
tb3_camera_image_raw_compressed_receive	101699.77	19264.72	-81.06
tb3_camera_image_raw_receive	1505089.72	155862.47	-89.64
tb3_cmd_vel_receive	4.14	2.91	-29.65
tb3_imu_receive	494.48	26.21	-94.70
tb3_joint_states_receive	4.86	2.97	-38.86
tb3_odom_receive	149.60	46.15	-69.15
tb3_scan_receive	119.80	141.39	+18.02
tb3_tf_receive	41.57	13.96	-66.43

Table 7.2: Bandwidth Comparison Between Default and SmartTestudo

7.2.2 Consideration of the Artifact from the Data

The data suggests that the adaptive monitoring system SmartTestudo performs effectively compared to the default static configuration. The two tables above (Table 7.1 and Table 7.2) highlight measurable improvements in data transmission efficiency, with significant reductions in bandwidth usage while maintaining adequate frequencies for safety-critical topics.

Across all monitored topics, SmartTestudo achieves an average bandwidth reduction of approximately 64.9%, with some topics such as `/camera/image_raw_receive` and `/imu_receive` seeing reductions as high as 94.7%. Even the minimum observed reduction among downscaled topics (`/cmd_vel_receive`) still reached 29.7%. Notably, `/scan_receive` was dynamically increased by 18.0% in certain zones, reflecting context-aware adaptation rather than uniform downscaling.

This demonstrates that SmartTestudo not only reduces communication overhead, particularly in bandwidth-intensive topics like `/camera/image_raw/compressed`, but does so without compromising critical information flow in safety-relevant scenarios.

8

Discussion

This chapter discusses the research questions based on the findings from the previous section. It dives deep into the meaning of the research by providing a clear explanation of each issue.

8.1 Research questions

8.1.1 RQ1: What are the current limitations in monitoring in ROS2 based CPS?

RQ1's goal is to understand and highlight the existing limitations and gaps in the current state of the art of ROS2 based CPS. Following the design science methodology [38], this question's objective is to analyze the problem in a theoretical way and to lead the way by eliciting requirements for RQ2. From the data gathered and analyzed from the literature review in Section 5, there is a clear number of issues still open in the field of monitoring in ROS2 based CPS. The limitations span across multiple aspects of ROS2 highlighting different weaknesses in this domain. The issues can be summarized by clusters: *real-time performance*, *monitoring overhead*, *specification complexity*, *standardization gaps*, *security concerns*, *scalability challenges*, and *maintainability issues*. These issues reflect deeper structural trends in robotics development. Historically, companies have relied on proprietary, in-house solutions for system monitoring and diagnostics. This development model led to duplicated effort and inconsistent methodologies, slowing innovation. The introduction of ROS and later ROS2 marked a shift towards a unified, open-source robotics middleware, enabling shared infrastructure and faster progress. Yet, the youth of ROS2 presents a double-edged sword: while its modularity and extensibility have fostered rapid growth, its evolving architecture and continuous updates introduce challenges in long-term stability and tool compatibility. Different ROS2 distributions modeled after Linux distributions help manage this evolution by providing stable environments, yet they also contribute to fragmentation and require careful version alignment. Nevertheless, the community-driven nature of ROS2, supported by active forums such as [61], suggests a promising outlook. Many of the current limitations are actively being addressed through community feedback and distribution upgrades. For instance, the transition from ROS to ROS2 [33] itself demonstrates the ecosystem's ability to learn from past shortcomings and deliver

substantial architectural improvements.

From this initial discussion a more in detail analysis can be made in the following sub-research questions below.

RQ1.1: *What are the technical and architectural limitations of existing ROS2 monitoring tools?*

This question dives into more details regarding the technical limitations of existing monitoring in ROS2 based CPS. To answer this, a thorough analysis with a literature review of existing literature on monitoring approaches has been conducted and data has been analyzed in Section 5. The literature review revealed several open issues that persist across the field. These limitations have been grouped into the following categories.

IF1 - Real-time performance: This issue regards the difficulty in ensuring consistent, deterministic behavior under high system load.

IF2 - Monitoring overhead: The tools themselves can introduce computational or network load, affecting system behavior.

IF3 - Specification complexity: Defining what, when, and how to monitor in ROS2 systems requires specialized knowledge.

IF4 - Standardization gaps: Lack of unified practices or tools leads to fragmented solutions across different implementations.

IF5 - Security concerns: Monitoring tools can unintentionally expose system data or increase the attack surface.

IF6 - Scalability challenges: Difficulty in deploying consistent monitoring strategies across distributed or large-scale systems.

IF7 - Maintainability issues: Compatibility issues between ROS2 distributions and monitoring tools can hinder long-term support.

A more in depth analysis and explanation of each issue found can be seen in the Problem Identification Section 5. From the data gathered an analysis on the technical limitation has been performed, connecting each issue and reasoning, here are the results of it.

These limitations collectively illustrate the multifaceted technical debt in monitoring for ROS2 based CPS. They reveal not only immaturity in tooling but also fragmentation in methodology, making integration and consistent deployment across different domains difficult. Among the limitations, real-time performance (IF1), monitoring overhead (IF2), and lack of standardization (IF4) emerge as the most pressing. These three factors create a compounded effect: high-latency tools that are difficult to compare or integrate across systems introduce unpredictable system behavior, especially in time-critical applications such as robotics and autonomous navigation. Additionally, challenges like specification complexity (IF3) and maintainability (IF7) suggest that even technically sound monitoring solutions face adoption barriers, especially among non-expert developers. The absence of security-aware monitoring (IF5) is also concerning, as CPS systems are increasingly deployed in networked, safety-critical environments.

Overall, the technical and architectural limitations point to a lack of holistic design thinking in current ROS2 monitoring approaches. They are often reactive (built

to address narrow problems) rather than proactive (designed with extensibility, security, and scalability in mind). These findings directly inform the next step of this research (RQ2), which focuses on evaluating whether adaptive, rule-based, and low-overhead monitoring systems like SmartTestudo can effectively address these limitations.

RQ1.2: *What practical challenges do developers face in applying monitoring in ROS2 environments?*

Building from RQ1.1, this question is regarding the main challenges faced by developers in applying such monitoring systems in ROS2 based CPS. In practice, developers working with ROS2 face a range of challenges when attempting to integrate monitoring into their systems. These challenges are not only technical but also stem from the complexity of ROS2’s ecosystem and its evolving nature.

One major issue is the steep learning curve involved in setting up effective monitoring. Developers must often write low-level code or configure multiple tools manually, with limited support for abstraction or automation. This is compounded by the specification complexity noted in the literature (IF3), where defining what and how to monitor requires significant domain knowledge and tool-specific expertise. A study regarding the adoption in industry of ROS2 mentioned in Section 3.1 highlights this issue of a high entry level barrier, and that to improve the ecosystem a big team of developers is needed[40].

Another barrier is the overhead introduced by available monitoring tools, such as rosbag2 and tracing frameworks (IF2). These tools may not be optimized for lightweight or real-time applications, making them impractical for resource-constrained environments like embedded or mobile robots.

Furthermore, due to the lack of standardization across monitoring approaches (IF4), developers are frequently forced to rely on fragmented or incompatible tools, which adds further integration effort and increases the chance of runtime inconsistencies or silent failures.

Maintainability is another key concern. With ROS2 distributions evolving rapidly, tools can become obsolete or require frequent updates to remain compatible (IF7). Unlike commercial platforms, ROS2 lacks centralized support, so developers must often rely on community forums and trial-and-error debugging, which slows down development and increases frustration. These practical challenges significantly impact the adoption of robust monitoring in real-world CPS projects. As a result, many developers either implement minimal monitoring or defer it until late stages, when problems are harder to detect and fix. This underlines the importance of creating flexible, low-overhead, and developer-friendly solutions goals which directly motivate the SmartTestudo artifact proposed in this research.

8.1.2 RQ2: What potential solutions could mitigate the limitations of current monitoring approaches in ROS2 based CPS?

RQ2 revolves around what’s found and discussed in RQ1 and builds on the existing related work. Research already existing on the topic of monitoring and how men-

tioned in the Problem identification chapter solution have been developed. These solutions span from, upgrading the hardware components, updating the software to change the behavior of monitoring, define a common standard and allow the user to customize their need of monitoring. These can fall into different monitoring solutions, simpler approaches which alter the frequency, or more complex solutions that take into consideration the context. Study with AI solution has also been done and has proven beneficial but with it's own drawbacks. The only clear conclusion from these question is that no approach is yet a defined standard on how to improve monitoring, so the issue identified in RQ1 still stands even after investigation. That's a common reality in the field, where with new discoveries it's difficult to make it prevail and become a de-facto standard. The goal that has been set in RQ2 was to identify the existing approaches and leverage on their pros and cons to develop an artifact that could bring results and improve the monitoring limitations.

The artifact presented in this research builds on already existing approaches, taking the benefits and improving on them. The artifact as mentioned in the sections above takes into account different strategies, from implementing a DSL, optimizing frequency values based on bandwidth requirements and user needs and making it context dependent as well. Consulting with an expert in the field of adaptive monitoring and robotics and reading the ROS forum, the way of adopting an adaptive monitoring system as a solution seemed like the best approach to tackle the challenges faced discovered in RQ1.

After this initial analysis the following sub-questions's goal is to dive into more detail and analyze the data from the related work section.

RQ2.1: What rule-based strategies can adapt monitoring behavior based on environmental context?

In contrast to static monitoring approaches commonly used in ROS2, rule-based adaptive strategies provide a dynamic mechanism to tailor monitoring behavior according to environmental context or runtime conditions. These strategies allow the monitoring system to selectively reduce or intensify data collection based on pre-defined conditions, thereby improving performance, resource efficiency, and system responsiveness.

SmartTestudo is designed following this principle. It enables the user to define environment-dependent monitoring rules through customizable configuration files. These rule files specify under which system states or contextual parameters certain monitoring actions should be triggered, adjusted, or paused entirely. This allows for a more context-aware monitoring experience compared to traditional approaches that apply uniform monitoring across all states, regardless of system load or criticality.

While SmartTestudo introduces a flexible rule-driven approach, several related works point to the increasing interest in adaptive or context-aware monitoring in ROS2-based CPS systems. For example, Cheng et al. [12] apply a MAPE-K feedback loop to enable runtime adaptation in a ROS2-based autonomous vehicle, illustrating a method of maintaining monitoring guarantees during execution. Similarly, Ichnowski et al.'s FogROS2 [32] provides offloading capabilities between cloud and

fog layers depending on computational load, an architectural-level adaptation driven by runtime conditions. Other works, such as River et al. [70], take a DSL approach to user-configurable monitoring, allowing custom specification of monitoring policies, though they focus more on network performance and security enforcement rather than full environmental adaptability. Unlike SmartTestudo, which emphasizes fine-grained behavioral tuning through rule files, many of the existing solutions implement broader adaptive mechanisms for example hardware reconfiguration (Mayoral et al. [52]) or simulation-based safety verification (Lim et al. [43]). These are valuable but often lack the lightweight, user-editable flexibility that SmartTestudo offers. Therefore, SmartTestudo fills a gap in the current landscape by offering a low-overhead, user-configurable, and environment-reactive monitoring framework. Its strategy aligns with the principles seen in other adaptive systems but brings them into a more accessible and customizable format for developers needing runtime adaptability without high complexity.

RQ2.2: *Are there alternative or complementary solutions that can further enhance monitoring adaptability?*

In addition to rule-based strategies like SmartTestudo, several alternative and complementary approaches have emerged that support greater adaptability in monitoring ROS2 based CPS systems.

One major complementary direction is the use of **model-driven engineering (MDE)** and formal specification tools such as FRET and OGMA [65]. These tools allow developers to specify monitoring properties at a high level and automatically generate monitoring code. While they do not inherently adapt at runtime, their modular design allows them to integrate with adaptive frameworks that apply context-driven logic to selectively activate these monitors.

Another promising solution is the use of **MAPE-K feedback loops**, as presented by Cheng et al. [12], which integrate monitoring into a self-adaptive control architecture. In such setups, the monitor is part of a feedback cycle that detects deviations, analyzes context, and adjusts behavior in real time, thus complementing rule-based strategies with autonomous decision-making.

Furthermore, **edge-cloud coordination frameworks** such as FogROS2 [32] offer architectural adaptability by dynamically offloading monitoring tasks to the cloud or fog layers depending on the current system load and latency requirements. While not rule-based per se, these systems enhance scalability and flexibility and can be used alongside rule-driven tools to offload computation-heavy monitoring dynamically.

In summary, formal specification tools, MAPE-K loops, and cloud-edge orchestration frameworks represent complementary approaches to rule-based strategies. Their integration either through hybrid architectures or layered deployment has the potential to significantly enhance the adaptability, responsiveness, and resilience of monitoring in ROS2 environments.

RQ2.3: *How feasible is the implementation of these strategies in real-world, real-time ROS2 environments?*

The feasibility of deploying adaptive monitoring strategies in real-world, real-time ROS2 systems largely depends on the computational overhead, timing determinism,

and complexity of integration.

Rule-based systems like SmartTestudo demonstrate high feasibility due to their lightweight design and modular integration with existing ROS2 nodes. Because they operate by reading environmental context and adjusting monitoring granularity or frequency through configuration files, they impose minimal computational overhead, making them suitable even for resource-constrained embedded systems such as TurtleBot3.

More complex adaptive strategies, such as MAPE-K feedback loops or cloud-offloading systems like FogROS2, introduce higher feasibility barriers. These solutions often require robust infrastructure support and add latency due to additional decision-making or data transfer steps. While highly effective in large-scale or industrial systems, they may be less suitable for small-scale robots or real-time safety-critical tasks.

Formal tools such as FRET and OGMA are partially feasible in real-time applications: their monitors are efficient, but the difficulty of writing correct specifications and integrating them into existing systems can pose development bottlenecks.

Moreover, ROS2's real-time performance remains a known limitation [8]. Adaptive strategies must be carefully profiled and tested to ensure that they do not violate deterministic timing constraints—especially in safety-critical applications such as autonomous vehicles or industrial robotics. While lightweight rule-based strategies offer high feasibility for real-time deployments, more advanced adaptive mechanisms require careful integration and may be better suited for systems with robust infrastructure or soft real-time requirements.

8.1.3 RQ3: To what extent can the current limitations of monitoring approaches in ROS2 based CPS be solved by the potential solutions?

RQ3's objective is to identify the effects of the artifact by using empirical data from simulations. After having run the simulation scenario and having applied the artifact with the rules file as explained in the section above, data has been gathered and analyzed in the findings. The graphs shown and analyzed in the Findings section, highlight the improvements on the TurtleBot3 itself. They show a clear reduction of bandwidth usage by the monitoring system and allow it to adapt to the current need, reducing the strain on the system. Despite an initial cost, the need for adaptive monitoring and optimizing the usage of data from a rover is worth on the long run. In this research, such tests on long iteration can't support the statement from before, but data from other research highlighted that reducing the strain on hardware components increases the lifespan of them reducing the maintenance and replacement costs.

RQ3.1: *How much improvement in bandwidth efficiency and topic frequency control can be observed?*

The results from the warehouse simulation scenario indicate that SmartTestudo delivers substantial improvements in bandwidth efficiency and topic frequency control. On average, the system achieved a bandwidth reduction of approximately 65% across all monitored topics. In some cases, such as the `/imu` and `/camera/image_raw`

topics, the reduction exceeded 90%, demonstrating that SmartTestudo effectively suppresses high-frequency data streams when they are not contextually necessary. These are particularly bandwidth-intensive topics, and their downsampling significantly reduces overall communication overhead. Surprisingly, not all topics were downsampled. For instance, the `/scan` topic saw a frequency increase of 18%, a result of SmartTestudo’s context-aware mechanism that raises the monitoring precision in high-risk or dynamic zones. This selective amplification shows that the system does not apply a blanket reduction strategy but instead makes nuanced, rule-based adjustments based on environmental context. Control-related topics such as `/cmd_vel` experienced more modest reductions (30%), which aligns with expectations. These topics require more consistent update rates to maintain responsiveness and safety, especially in real-time control scenarios.

These findings support the hypothesis that rule-based adaptive monitoring in ROS 2 can significantly reduce data overhead without compromising system responsiveness or situational awareness. Compared to the general-purpose monitoring approaches built into ROS 2 where all topics are broadcast continuously at fixed rates—SmartTestudo introduces a more granular and context-sensitive alternative. This aligns with trends seen in recent literature. For example, Cheng et al. [12] proposed a MAPE-K loop for runtime adaptation in autonomous vehicles, confirming the relevance of dynamic monitoring strategies. Similarly, Ichnowski et al. [32] highlighted the problem of cloud and onboard resources being overwhelmed by high-frequency data streams—SmartTestudo directly addresses this by reducing such load at the source. Unlike static solutions or even more complex platforms like FogROS2, SmartTestudo offers a lightweight and rule-driven mechanism that can be integrated directly into the ROS 2 stack, without relying on offloading or extensive infrastructure. This positions it as a practical middle ground between flexibility and performance, especially in resource-constrained environments.

RQ3.2: *What are the architectural strengths and limitations of applying the proposed adaptive monitoring approach to ROS2 based CPS?*

The particular architecture and approach of this artifact has been explained clearly in the artifact section. SmartTestudo brings to the table a solution which combines MAPE-K approach in close knit with rule customization with QUPER and mathematical optimization, all specified with a custom DSL. This composition results in a powerful yet modular adaptive monitoring system. However, the architectural design introduces both strengths and limitations.

The main strengths identified from the data gathered in the findings and the literature analyzed to similar approaches are the following:

Modularity and Maintainability: The MAPE-K loop ensures clear separation of responsibilities: sensing the system state, analyzing conditions, planning rule applications, and executing adjustments. This modularity allows each phase to evolve independently, which is crucial for iterative development and testing in CPS. Similar approach has been used by Cheng et al. [12] in their EvoRally rover.

Rule-Based Customization with DSL for Usability: SmartTestudo’s DSL enables in-

tuitive and expressive definition of rules governing topic behavior based on spatial and contextual triggers. This makes it accessible to non-expert users or system integrators who may not wish to modify code. The rule files are human-readable and decouple logic from implementation. This approach of using a DSL can be seen in Section 2.10 and in the related work where similar use has been applied by River et al. [70].

QUPER based Quality Trade-off Modeling: The QUPER model explained in Section 3.6 applied in SmartTestudo supports structured reasoning about trade-offs between quality attributes like resource usage and performance. This abstraction helps guide users toward better balance between efficiency and system responsiveness. Using the GUI ADIRF Ruler tool, the user can specify the level of resources and quality level that he wants to apply to different topics, allowing the optimizer to take this weight into account.

Mathematical Optimization for Precision Tuning: The planning phase using the ADIRF Ruler tool, allows the user to specify different weights and QUPER model weights, this leverages optimization techniques to select the best parameter settings given multiple constraints and goals. By providing such constraints such as total bandwidth and frequency max values, this approach provides a data-driven way to enforce optimal resource usage without degrading essential system observability.

Context Awareness through ROS2 Integration: The artifact adapts topic configurations based on real-time robot position and task zones, offering fine-grained control in dynamic environments like a warehouse. This enhances both performance and resource conservation by focusing effort where it's most needed.

However this artifact is not perfect, and can't satisfy each and every edge case. From the data gathered from the simulation and literature the following weakness and limitations have been identified:

Static Knowledge and Rule Dependence: The Knowledge component relies on a fixed zone map and pre-defined rules. While powerful, this limits flexibility in dynamic or unknown environments where zone definitions may change or be unknown at runtime. Being the artifact still in a young stage, the DSL is still up for modifications and upgrades.

Complexity in Rule Management: As the number of topics, zones, and quality dimensions grows, managing the rule set may become error-prone or difficult to scale. Conflicts or unintended interactions between rules are possible, especially in overlapping zones or edge conditions.

Optimization Overhead and Black-Box Nature: While optimization adds precision, it introduces some runtime overhead and may obscure how decisions are made unless well-visualized. Users without optimization knowledge may find it hard to tune weights or interpret results. In the current state of the artifact, applying the optimizer with the way the QUPER models weights a balanced in, could lead to user input error and may be harmful on this procedure.

Limited Learning or Adaptation Beyond Rules: The current architecture lacks any learning-based adaptation. All behaviors are explicitly specified in rules or derived via optimization, meaning the system does not adapt or improve based on feedback

or performance history over time. This limitation can be addressed in future work, as such complex task can be achieved by a bigger development team [40] with more resources in hand.

Having addressed the strengths and weaknesses of SmartTestudo, the main result from this analysis is clear. The architectural design of SmartTestudo blends the structured reasoning of MAPE-K with the expressiveness of QUPER based trade-offs and precision from mathematical optimization, all specifiable within an accessible, rule-based DSL. This combination results in a flexible and extensible adaptive monitoring system for ROS 2. However, it also introduces challenges in scalability, runtime adaptation, and managing rule complexity. Future improvements could focus on dynamic knowledge updates, conflict resolution mechanisms, and incorporating machine learning for better long-term adaptability.

8.2 Limitations

This research utilized the Gazebo simulator [59] to evaluate the performance of the TurtleBot3 rover in various industrial scenarios with Adaptive Monitoring enabled. However, certain factors introduce limitations that may affect the validity of the results.

8.2.1 Threats to internal validity

The main antagonists to the internal validity of this research, are mainly regarding the *human factors*, the *TurtleBot3 Gazebo simulator*, and the *code base of ROS2*.

Due to the nature of the architecture of the artifact, human factor is dependent on the phase of writing the monitor parameters and context settings, these can pose a threat by having a big impact on the performance of the adaptive monitoring.

The Gazebo simulator, while robust, does not fully replicate the physical and environmental conditions of the industrial scenario, which can leading to potential inaccuracies in the data.

The limitations regarding the code base have been found and analyzed in Section 5. These issues are still an open question and gap in the development of ROS2 and are seen as a constraint that can impact the performance of the artifact.

8.2.2 Threats to external validity

The threats to external validity are mainly regarding the findings of this research. The threats identified are regarding *context specific findings* and the *simulation environment*.

This research focuses on applying this strategy on a proof of concept using the ROS2 based CPS TurtleBot3. This decision can create a threat to external validity by producing results that can be valid in this particular ROS2 based CPS. This research is generic to the different ROS2 system, but to validate it furthermore the adaptive monitoring should be installed on another ROS2 platform to be tested and

evaluated.

By relying on the Gazebo simulator [59], this research's findings are based on a simulated environment rather than a physical system. This introduces certain limitations, as external factors that exist in real-world industrial settings may not be accurately replicated in simulation.

The Gazebo simulator does not fully capture real-world external influences such as sensor noise, hardware degradation, or unexpected environmental changes. Additionally, factors like battery performance over time, temperature variations, and real-world wear-and-tear effects are not modeled in the simulation, potentially affecting the applicability of the results to real systems.

The industrial scenarios tested in Gazebo may not encompass the full range of possible conditions encountered in real-world settings. Unexpected obstacles, and human interactions could introduce challenges not accounted for in the simulation, limiting the generalization of the findings.

While the adaptive monitoring system performs well in the tested scenarios, its scalability to more complex environments, larger robotic fleets, or highly dynamic settings remains uncertain. Performance may degrade when applied to multi-robot systems or more demanding industrial applications.

The sensors modeled in Gazebo do not always fully replicate real-world sensor behavior. Noise levels, calibration drift, and hardware failures can affect perception in ways that are not accurately reflected in the simulation, potentially leading to overestimated system performance.

The study does not account for long-term operational challenges such as hardware fatigue, software degradation, or changes in environmental conditions over extended periods. In real-world deployments, these factors could influence the reliability and effectiveness of the adaptive monitoring system.

8.2.3 Threats to construct validity

The main challenges identified regarding construct validity are two, *measurement accuracy* and *monitoring prioritization*.

Measurement accuracy proves crucial in identifying the effectiveness of the adaptive monitoring system, any inaccuracies or oversimplifications could affect the study's conclusions.

Monitoring prioritization is regarding the bias in the selection of what metrics need to be adapted to reduce or increase the frequency, certain assumptions made in data prioritization and monitoring strategies, such as the classification of "critical data," may not align perfectly with real-world priorities, and need to be validated by conducting interviews with experts.

8.2.4 Delimitations

This research is deliberately scoped as follows to maintain focus and feasibility. Four delimitations have been identified for this research, *simulator based*, *ROS2 TurtleBot3*, *literature scope* and *narrowed methodological approach*.

This research is based on the Gazebo simulator and not involving the use of physical

hardware to make tests on the artifact, this is a delimitation that proves critical to the results.

The artifact is valid for all ROS2 applications but is limited as a proof of concept in a TurtleBot3 environment simulator, this can be a limitation on the validity of the results.

The literature review in this research centers on adaptive monitoring systems, utility function design, and ROS2 architecture. Topics such as the detailed mechanical design of the CPS or other industrial scenario contexts are excluded.

The last delimitation is regarding the narrowed methodological approach. While this study introduces a framework for adaptive monitoring, it does not aim to create a universally optimal solution but rather explores one possible approach to address the problem.

9

Conclusion

This chapter closes the research by presenting the main takeaways and unresolved issues.

9.1 Takeaways

This thesis set out to investigate the current limitations of monitoring systems within Cyber-Physical Systems (CPS), especially in the context of ROS2 based robotic platforms. Through a structured literature review and architectural analysis, it became evident that many existing monitoring approaches suffer from rigidity, lack of scalability, and inefficient resource usage, particularly in bandwidth and topic frequency regulation. In response to these findings, this research introduced *SmartTestudo: an adaptive monitoring artifact* built upon the MAPE-K feedback loop architecture, enhanced by rule-based contextual adaptation through a custom Domain Specific Language (DSL), and guided by trade-offs defined using the QUPER model and mathematical optimization. Empirical evaluation via a Gazebo simulated warehouse scenario using TurtleBot3 demonstrated the effectiveness of *SmartTestudo* in reducing bandwidth usage and dynamically adjusting topic frequency based on environmental context. The main highlights from the results show a clear bandwidth reductions exceeding 80–90% on several key topics, a smart adaptation to high-priority zones via frequency increases and a minimal intervention needed from users, thanks to the DSL’s ease of use.

The research confirms that a structured and flexible monitoring approach driven by MAPE-K, rule-based logic, and optimization can significantly improve monitoring efficiency without compromising observability or responsiveness in ROS2 systems. While the artifact shows promise, it also reveals limitations in rule scalability, dynamic adaptability, and knowledge generalization paving the way for further research and future work.

9.2 Future Work

The results of this research highlighted the benefits of using an adaptive monitoring system however it may not always be the optimal solution for the problem, some fields may require a deep level of monitoring that can’t be achieved with this strat-

egy.

This research still leaves some open challenges and some future work that could aim to improve and validate the adaptive monitoring system presented in the artifact. The main points being *testing on other platforms, simulate in real environment, interview experts, improve the requirement elicitation tool* and *evaluate different strategies to understand the monitoring parameters*. These points are the main wounds left open by this research.

Regarding the SmartTestudo artifact, more work has to be put, as the adaptive monitoring strategy has proven succesful, the proof of concept should go a level deeper in the ROS and not be a meta-monitoring tool. New requirements should be elicited and the impact on the CPU and computational power must be evaluated. Regarding the ADIRF Ruler tool, experts opinion is needed to improve the current design. A key part in designing a tool is interviews to understand the needs of the monitoring engineer. Other points of interest for future development would be analyzing the weaknesses identified in RQ3.

To address this future work a researcher can build on this research and following design science research, structure it into different iterations.

Bibliography

- [1] Abdulrahman Al-Batati, Anis Koubaa, and Mohamed Abdelkader. Ros 2 in a nutshell: A survey. *Preprints*, October 2024.
- [2] Robin Amsters and Peter Slaets. Turtlebot 3 as a robotics education platform. In *Proceedings of Robotics in Education (RiE 2019)*, pages 170–181, 2019. First Online: 07 August 2019.
- [3] Paolo Arcaini, Elvinia Riccobene, and Patrizia Scandurra. Modeling and analyzing mape-k feedback loops for self-adaptation. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 13–23, 2015.
- [4] Pradeepkumar Ashok, Ganesh Krishnamoorthy, and Delbert Tesar. Guidelines for managing sensors in cyber-physical systems with multiple sensors. *International Journal of Automation and Computing*, 2012.
- [5] Nigan Bayazit. Investigating design: A review of forty years of design research. *Design Issues*, 20(1):16–29, December 2004.
- [6] Richard Berntsson Svensson and Björn Regnell. A case study evaluation of the guideline-supported quper model for elicitation of quality requirements. In Samuel A. Fricker and Kurt Schneider, editors, *Requirements Engineering: Foundation for Software Quality*, pages 230–246, Cham, 2015. Springer International Publishing.
- [7] Tobias Betz, Maximilian Schmeller, Harun Teper, and Johannes Betz. How fast is my software? latency evaluation for a ros 2 autonomous driving software. In *2023 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7, Chengdu, China, June 2023. IEEE. Conference Date: 04-07 June 2023, Added to IEEE Xplore: 27 July 2023.
- [8] Andrea Bonci, Francesco Gaudeni, Maria Cristina Giannini, and Sauro Longhi. Robot operating system 2 (ros2)-based frameworks for increasing robot autonomy: A survey. *Applied Sciences*, 13(23):12796, November 2023.
- [9] Christophe Bédard, Ingo Lütkebohle, and Michel Dagenais. ros2_tracing: Mul-

- tipurpose low-overhead framework for real-time tracing of ros 2. *IEEE Robotics and Automation Letters*, 7(3):6511–6518, July 2022.
- [10] Volkan Cagdas and Erik Stubkjær. Design research for cadastral systems. *Computers, Environment and Urban Systems*, 35:77–87, 2011.
- [11] Musab Talha Cakin. Ros 2 data flow scheduling: Design and implementation of a cross-process real-time executor. Master’s thesis, University of Applied Sciences Technikum Wien, Wien, Austria, September 2023.
- [12] Betty H. C. Cheng, Robert Jared Clark, Jonathon Emil Fleck, Michael Austin Langford, and Philip K. McKinley. AC-ROS: Assurance Case Driven Adaptation for the Robot Operating System. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS ’20)*, pages 102–113. ACM, 2020.
- [13] Vu Dang Khoa Chiem. Implementing slam and autonomous navigation on a turtlebot3. Course report, Vanier College, Faculty of Science and Technology, Department of Physics, May 2024. Presented to Prof. Jicai Pan, Statics and Engineering Physics (203-HTK-05 sect. 00002).
- [14] Jane Cleland-Huang, Ankit Agrawal, Michael Vierhauser, Michael Murphy, and Mike Prieto. Extending mape-k to support human-machine teaming. In *Proceedings of the 17th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS ’22)*, pages 120–131, New York, NY, USA, 2022. ACM.
- [15] COIN-OR Foundation. Cbc (coin-or branch and cut). <https://github.com/coin-or/Cbc>, 2025. Accessed: 2025-04-18.
- [16] ROS Community. rqt: Ros gui toolkit. <http://wiki.ros.org/rqt>, 2025. Accessed: 2025-03-07.
- [17] ROS Community. rqt_graph: Ros graph visualization tool. http://wiki.ros.org/rqt_graph, 2025. Accessed: 2025-03-07.
- [18] ROS 2 Contributors. rosbag2: Ros 2 bag reader and writer. <https://github.com/ros2/rosbag2>, 2025. Accessed: 2025-03-07.
- [19] Pedro Medeiros de Assis Brasil, Fabio Ugalde Pereira, Marco Antonio de Souza Leite Cuadros, Anselmo Rafael Cukla, and Daniel Fernando Tello Gamarra. A study on global path planners algorithms for the simulated turtlebot 3 robot in ros. In *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*. IEEE, 2020.
- [20] Igor Dejanović. textx — a python tool for dsls. <https://textx.github.io/textX/index.html>, 2015. Accessed: 2025-04-18.

-
- [21] Aline Dresch, Daniel Pacheco Lacerda, and José Antônio Valle Antunes. *Design Science Research*, pages 67–102. Springer International Publishing, Cham, 2015.
- [22] Christien Enzing, Chiel Scholten, Joost van Barneveld, Adriana Tapus, Michael Henshaw, Bram Vanderborght, Eldert J. van Henten, Fred van Houten, Stamatios Karnouskos, and Haydn Thompson. Ethical aspects of cyber-physical systems - annex 1: Technical horizon scan of cps in seven areas of application. Technical report, STOA - Science and Technology Options Assessment, 2023. Scientific Foresight Study.
- [23] Teresa Ferrer-Mico, Miquel Àngel Prats-Fernàndez, and Albert Redo-Sanchez. Impact of scratch programming on students’ understanding of their own learning process. *Procedia - Social and Behavioral Sciences*, 46:1219–1223, 2012. Under a Creative Commons license.
- [24] Fraunhofer Institute for Manufacturing Engineering and Automation. Ros-industrial: Open source for industrial robotics, May 2014. Accessed: 2024-11-26.
- [25] Farnaz Fotrousi and Theocharis Tavantzis. Reqgenie: Gpt-powered conversational-ai for requirements elicitation. In Dietmar Pfahl, Javier Gonzalez Huerta, Jil Klünder, and Hina Anwar, editors, *Product-Focused Software Process Improvement*, pages 352–359, Cham, 2025. Springer Nature Switzerland.
- [26] J. J. Gertler. *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker, New York, NY, USA, 1998.
- [27] Pardis Ghazi and Martin Glinz. Challenges of working with artifacts in requirements engineering and software engineering. *Requirements Engineering*, 22(3):359–385, September 2017.
- [28] GNU Project. Gnu linear programming kit (glpk). <https://www.gnu.org/software/glpk/>, 2025. Accessed: 2025-04-18.
- [29] Elias E. Hartmark and Tage Andersen. Runtime verification of autonomous robotic systems in ros 2. Master’s thesis, Norwegian University of Life Sciences, Norway, TBD 2024. 30 ECTS, Applied Robotics.
- [30] Amjad Hudaib, Raja Moh’d Taisir Masadeh, Mais Haj Qasem, and Abdullah Issa Alzaqebah. Requirements prioritization techniques comparison. *Modern Applied Science*, 12(2):62–71, January 2018.
- [31] IBM Corporation. Ibm ilog cplex optimizer. <https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer>, 2025. Accessed: 2025-04-18.

- [32] Jeffrey Ichnowski, Kaiyuan Chen, Karthik Dharmarajan, Simeon Adebola, Michael Danielczuk, and Víctor Mayoral-Vilches. FogROS2: An Adaptive Platform for Cloud and Fog Robotics Using ROS 2. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11163–11170. IEEE, 2022.
- [33] Southwest Research Institute. The ros 1 vs ros 2 transition, 08 October 2022. Accessed: 2024-11-26.
- [34] MdNafee Al Islam, Jane Cleland-Huang, and Michael Vierhauser. Adam: Adaptive monitoring of runtime anomalies in small uncrewed aerial systems. In *Proceedings of the 2024 IEEE/ACM 19th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE/ACM, 2024. MdNafee Al Islam and Jane Cleland-Huang (University of Notre Dame, USA); Michael Vierhauser (University of Innsbruck, Austria).
- [35] Mohd Javaid, Abid Haleem, Ravi Pratap Singh, and Rajiv Suman. An integrated outlook of cyber–physical systems for industry 4.0: Topical practices, architecture, and applications. *Green Technologies and Sustainability*, page 100001, 2022. Open-access, Under a Creative Commons license.
- [36] Hyeong Ryeol Kam, Sung-Ho Lee, Taejung Park, and Chang-Hun Kim. Rviz: a toolkit for real domain data visualization. *Telecommunication Systems*, 60:337–345, April 2015.
- [37] Jongkil Kim, Jonathon M. Smereka, Calvin Cheung, Surya Nepal, and Marthie Grobler. Security and performance considerations in ros 2: A balancing act. *arXiv preprint arXiv:1809.09566*, 2018. 6 pages, 6 figures.
- [38] Eric Knauss. Constructive master’s thesis work in industry: Guidelines for applying design science research. *arXiv*, 2012.04966, 2020.
- [39] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2149–2154, Sendai, Japan, 2004. IEEE/RSJ. Robotics Research Labs, University of Southern California.
- [40] Sophia Kolak, Afsoon Afzal, Claire Le Goues, Michael Hilton, and Christopher Steven Timperley. It takes a village to build a robot: An empirical study of the ros ecosystem. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 430–440, 2020.
- [41] Tomaz Kosar, Pablo E. Martinez Lopez, Pablo A. Barrientos, and Marjan Mernik. A preliminary study on various implementation approaches of domain-specific language. *Information and Software Technology*, 50(5):390–405, 2008. Available online 21 April 2007.

- [42] Edward A. Lee. The past, present and future of cyber-physical systems: A focus on models. *Sensors*, 15:4837–4869, 2015. Open-access.
- [43] Yixiao Li, Yutaka Matsubara, Hiroaki Takada, Satoru Funahashi, and Hiroki Kawashima. Monitor and analyze rare ros2 performance issues with a unified tracing framework. In *Proceedings of the 2024 The 6th World Symposium on Software Engineering (WSSE)*, pages 95–104. ACM, 2024. Published: 08 December 2024.
- [44] Yongxin Liao, Eduardo Rocha Loures, Fernando Deschamps, Guilherme Brezinski, and André Venâncio. The impact of the fourth industrial revolution: A cross-country/region comparison. *Systematic Review*, 28, 2018. Open-access.
- [45] Yang Liu, Yu Peng, Bailing Wang, Sirui Yao, and Zihe Liu. Review on cyber-physical systems. *IEEE/CAA Journal of Automatica Sinica*, 4(1):27–40, January 2017.
- [46] Yang Lu. Cyber physical system (cps)-based industry 4.0: A survey. *Journal of Industrial Integration and Management*, 2(3):1750014, 2017.
- [47] Matt Luckcuck, Marie Farrell, Louise Dennis, Clare Dixon, and Michael Fisher. Formal specification and verification of autonomous robotic systems: A survey. *arXiv preprint arXiv:1807.00048*, 2019. Last revised 1 May 2019, version 3.
- [48] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7, 2022.
- [49] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, 10(4):1–15, 2010.
- [50] Yuya Maruyama, Shinpei Kato, and Takuya Azumi. Exploring the performance of ros2. In *2016 International Conference on Embedded Software (EMSOFT)*, pages 1–10, 2016.
- [51] Víctor Mayoral. ros-robotics-companies: A list of companies using or supporting ros in their products or services. <https://github.com/vmayoral/ros-robotics-companies>, 2023. Accessed: 2025-05-14.
- [52] Víctor Mayoral-Vilches and Giulio Corradi. Adaptive computing in robotics, leveraging ros 2 to enable software-defined hardware for fpgas. *arXiv preprint arXiv:2109.03276*, 2021. Submitted on 30 Aug 2021.
- [53] Marjan Mernik, Jan Heering, and Anthony M. Sloane. When and how to develop domain-specific languages. *ACM Computing Surveys (CSUR)*, 37(4):316–344, December 2005.

- [54] Metro Robots. Ros companies map, 2023. Accessed: 2025-05-14.
- [55] Audrey Moui, Thierry Desprats, Emmanuel Lavinal, and Michelle Sibilla. Information models for managing monitoring adaptation enforcement. In *ADAPTIVE 2012: The Fourth International Conference on Adaptive and Self-Adaptive Systems and Applications*, Toulouse, France, 2012. International Academy, Research, and Industry Association (IARIA), IARIA.
- [56] NASA. Fret : Formal requirements elicitation tool, 2021. Accessed: 2024-11-26.
- [57] NASA. Nasa’s ogma runtime verification tool, 2022. Accessed: 2024-11-26.
- [58] M. Ohadi, M. R. Jahed Motlagh, and B. S. Ghanbarzadeh. Ros 2 application to the safety monitoring of collaborative robots. In *Proceedings of the 2024 4th International Conference on Robotics and Control Engineering (RobCE)*, pages 99–103. IEEE, 2024.
- [59] Open Robotics. Gazebo simulator, 2024. Accessed: 13-Feb-2025.
- [60] Open Source Robotics Foundation. The ros 2 project - metrics. <https://docs.ros.org/en/rolling/The-R0S2-Project/Metrics.html>, 2024. Accessed: 2025-05-14.
- [61] Open Source Robotics Foundation. Ros discourse forum. <https://discourse.ros.org/>, 2024. Accessed: 2025-05-04.
- [62] Deep Patel, Chayan Maiti, and Sreekumar Muthuswamy. Real-time performance monitoring of a cnc milling machine using ros 2 and aws iot towards industry 4.0. In *IEEE EUROCON 2023 - 20th International Conference on Smart Technologies*, pages 776–781, 2023.
- [63] Ian Peake, Joseph La Delfa, Ronal Bejarano, and Jan Olaf Blech. Simulation components in gazebo. In *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, pages 1–8. IEEE, 2021.
- [64] Jonas Peeck, Johannes Schlatow, and Rolf Ernst. Online latency monitoring of time-sensitive event chains in ros2. Technical Report 202101271521-0, Institute of Computer and Network Engineering, TU Braunschweig, January 2021.
- [65] Ivan Perez, Anastasia Mavridou, Tom Pressburger, Alexander Will, and Patrick J. Martin. Monitoring ros2: From requirements to autonomous robots. In *Proceedings of FMAS2022 ASYDE2022*, volume 371, pages 208–216, 2022.
- [66] Ola Pettersson. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems*, 2005. Received: 20 September 2004; Revised: 1 September 2005; Accepted: 13 September 2005.
- [67] ProMods. Euro truck simulator 2 - music (#11 hurry up 2), 2021. Accessed: 2025-05-18.

-
- [68] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. Ros: An open-source robot operating system. In *ICRA Workshop on Open Source Software*, Kobe, Japan, 2009. IEEE.
- [69] P. L. Ringold, B. Mulder, J. Alegria, R. L. Czaplewski, T. Tolle, and K. Burnett. Establishing a regional monitoring strategy: the pacific northwest forest plan. *Environmental Management*, 23(2):179–192, 1999.
- [70] Sean Rivera, Antonio Ken Iannillo, Sofiane Lagraa, Clément Joly, and Radu State. Ros-fm: Fast monitoring for the robotic operating system (ros). In *Proceedings of the 25th International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 187–196. IEEE, 2020.
- [71] M.A. Roach, J. Pennney, and B.H. Jared. Exploring a supervisory control system using ros2 and iot sensors. Technical report, University of Texas at Austin, 2023. Presented at the 2023 International Solid Freeform Fabrication Symposium.
- [72] Open Robotics. Nvidia ros 2 projects. <https://docs.ros.org/en/jazzy/Related-Projects/Nvidia-R0S2-Projects.html>, 2025. Accessed: 2025-05-15.
- [73] ROBOTIS. Turtlebot3, 2024. Accessed: 13-Feb-2025.
- [74] ROBOTIS. Turtlebot3 features - robotis e-manual, 2024. Accessed: 13-Feb-2025.
- [75] ROS Visualization. Rviz, 2024. Accessed: 13-Feb-2025.
- [76] ROS Wiki. Rviz, 2024. Accessed: 13-Feb-2025.
- [77] Nayan B. Ruparelia. Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes*, 35(3):8–13, May 2010.
- [78] Maryam Ghaffari Saadat, Angelo Ferrando, Louise A. Dennis, and Michael Fisher. Rosmonitoring 2.0: Extending ros runtime verification to services and ordered topics. *Electronic Proceedings in Theoretical Computer Science (EPTCS)*, 411:38–55, November 2024.
- [79] Jan A. Snyman and Daniel N. Wilke. *Practical Mathematical Optimization: Basic Optimization Theory and Gradient-Based Algorithms*, volume 133 of *Springer Optimization and Its Applications*. Springer, Cham, Switzerland, second edition, 2018.
- [80] Marco Stadler, Michael Riegler, and Johannes Sametinger. Towards increasing safety in collaborative cps environments. In *Proceedings of the 34th DEXA Conferences and Workshops at IWCFs’2023*, 2023.

- [81] Peter N. Stearns. *The Industrial Revolution in World History*. Routledge, New York, 5th edition, 2020.
- [82] Richard Berntsson Svensson, Tony Gorschek, and Björn Regnell. Quality requirements in practice: A study of requirements specification at QUPER. In *13th International Conference on Requirements Engineering (RE)*, pages 275–284. IEEE, 2009.
- [83] Richard Berntsson Svensson, Thomas Olsson, and Björn Regnell. Introducing support for release planning of quality requirements — an industrial evaluation of the quper model. In *Proceedings of the International Workshop on Software Product Management (IWSPM)*, 2008.
- [84] Richard Berntsson Svensson, Björn Regnell, and Aybuke Aurum. Towards modeling guidelines for capturing the cost of improving software product quality in release planning. In *Proceedings of the 11th International Conference on Product Focused Software, PROFES '10*, page 20–23, New York, NY, USA, 2010. Association for Computing Machinery.
- [85] The PuLP Developers. Pulp: A python library for linear programming. <https://coin-or.github.io/pulp/>, 2025. Accessed: 2025-04-18.
- [86] Dirk Thomas. Changes between ros 1 and ros 2, September 2015. Accessed: 2024-11-26.
- [87] Vijay Vaishnavi, Bill Kuechler, Stacie Petter, and Gerard De Leoz. Design science research in information systems, 2025. Accessed: 2025-02-18.
- [88] Gijs M.C. van den Hoven. Introducing a performance observation framework to ros2. Master’s thesis, Mathematics and Computer Science, March 2024.
- [89] Michael Vierhauser, Rebekka Wohlrab, Marco Stadler, and Jane Cleland-Huang. Amon: A domain-specific language and framework for adaptive monitoring of cyber–physical systems. Affiliations: a. Johannes Kepler University Linz, LIT Secure and Correct Systems Lab, Linz, 4040, Austria; b. Chalmers | University of Gothenburg, Department of Computer Science and Engineering, Gothenburg, 41296, Sweden; c. University of Notre Dame, Department of Computer Science and Engineering, Notre Dame, 46617, IN, United States.
- [90] Juan Vizcarrondo, Jose Aguilar, Ernesto Exposito, and Audine Subias. Mapek as a service-oriented architecture. *IEEE Latin America Transactions*, 15(6):1163–1175, 2017.
- [91] Rebekka Wohlrab, Javier Cámara, David Garlan, and Bradley Schmerl. Explaining quality attribute tradeoffs in automated planning for self-adaptive systems. *Journal of Systems and Software*, page 111538, 2022. Open access under a Creative Commons license.

- [92] Xin-She Yang. *Introduction to Mathematical Optimization: From Linear Programming to Metaheuristics*. Cambridge International Science Publishing, Cambridge, United Kingdom, 2008.
- [93] Liao Yuan. Introduction of robot operating system 2: Ros2. *H2020-MSCA-ITN SAS Program - ESR2*, 2020.
- [94] Edith Zavala, Xavier Franch, and Jordi Marco. Adaptive monitoring: A systematic mapping. *Information and Software Technology*, 105:161–189, 2019.

A

Appendix

A.1 ReqGenie requirement elicitation

As mentioned in the AI Use Statement section and explained in the Artifact Requirement section, to perform the requirement elicitation phase, the tool ReqGenie [25] has been used.

This link shows the conversation <https://chatgpt.com/share/6823012a-8584-8011-afe4-976e35f4dd88>.

A.2 ADIRF Ruler Tool Screenshots

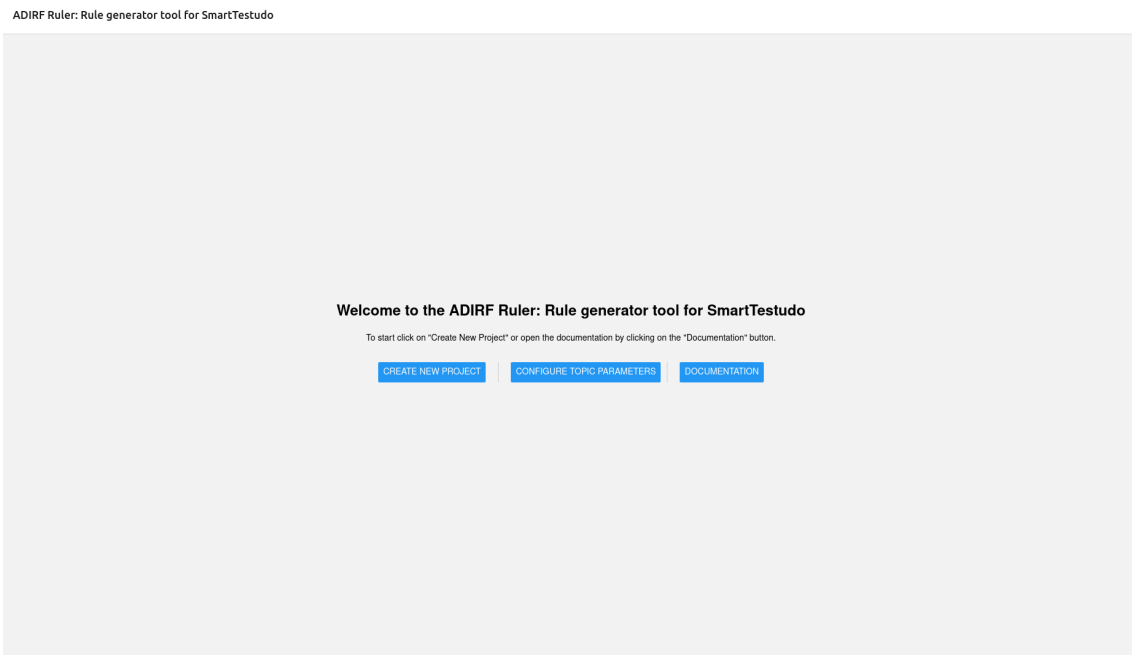


Figure A.1: ADIRF Tool Home screen.

A. Appendix

← ADIRF Ruler: Topic Setup

Topic	Msg Size (KB)	Unacceptable	Marginal	Acceptable	Desirable	Competitive	Excessive
/cmd_vel	0.052	5	10	15	18	20	25
/odom	0.72	10	15	20	25	30	35
/camera/ camera_info	0.38	10	15	20	25	30	35
/camera/ image_raw/ compressed	420	5	10	15	20	30	35
/camera/ image_raw	6220	2	5	10	15	20	25
/imu	0.32	50	75	100	150	200	250
/joint_state	0.12	2	4	6	8	10	15
/scan	2.94	2	4	6	8	10	15
/tf	0.17	2	10	20	30	40	45

Total Bandwidth (KBps):

[SAVE TOPIC PARAMETERS](#)

Figure A.2: ADIRF Tool Topic parameter setup for QUPER model OBI optimizer.

```
project(9).dsl
~/Downloads

1 begin
2 // DSL Generated File
3
4 // Global frequencies
5 SetFrequency /cmd_vel 5.0
6 SetFrequency /odom 10.0
7 SetFrequency /camera/camera_info 10.0
8 SetFrequency /camera/image_raw/compressed 5.0
9 SetFrequency /camera/image_raw 2.0
10 SetFrequency /imu 50.0
11 SetFrequency /joint_state 2.0
12 SetFrequency /scan 2.0
13 SetFrequency /tf 2.0
14
15 // Conditional block 1
16 If(
17   CheckValue /odom pose_x > 11 AND
18   CheckValue /odom pose_x < 15 AND
19   CheckValue /odom pose_y > 7 AND
20   CheckValue /odom pose_y < 11
21 ) Then
22   SetFrequency /imu 200.0
23   SetFrequency /camera/image_raw 35.0
24 end
```

Figure A.3: Output from ADIRF Tool.

A.3 Findings Graphs

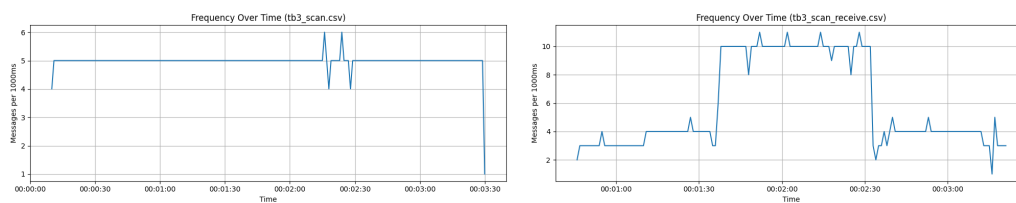


Figure A.4: /scan Topic Frequency Analysis.

A. Appendix

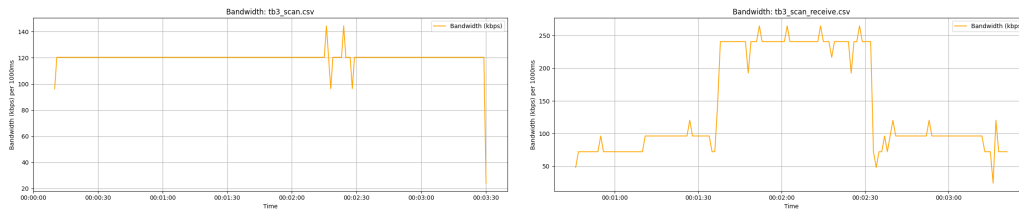


Figure A.5: /scan Topic Bandwidth Analysis.

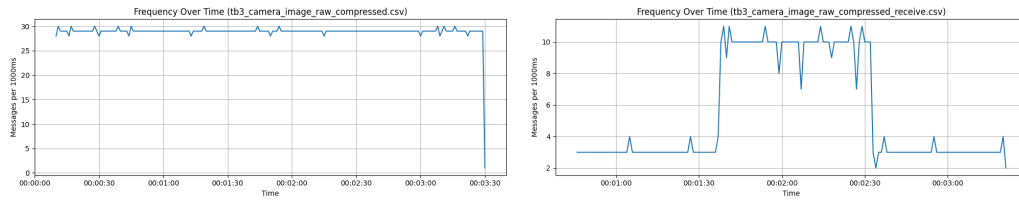


Figure A.6: /camera/image_raw/compressed Topic Frequency Analysis.

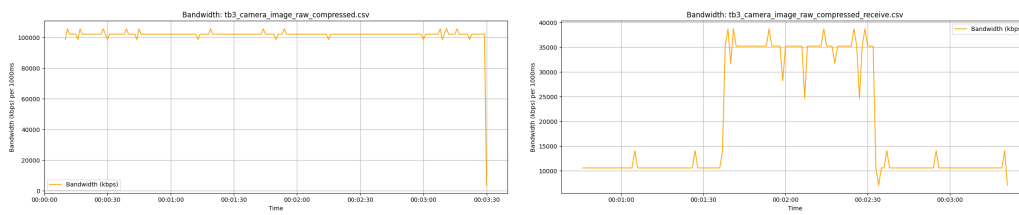


Figure A.7: /camera/image_raw/compressed Topic Bandwidth Analysis.

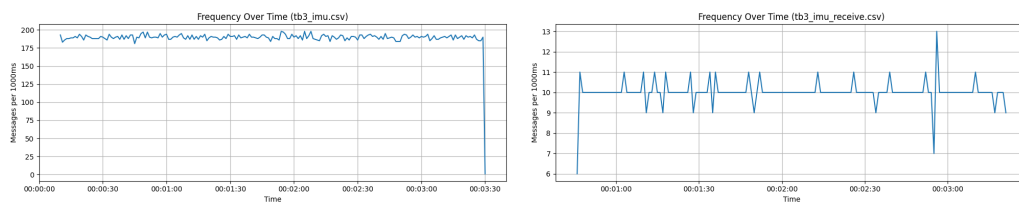


Figure A.8: /imu Topic Frequency Analysis.

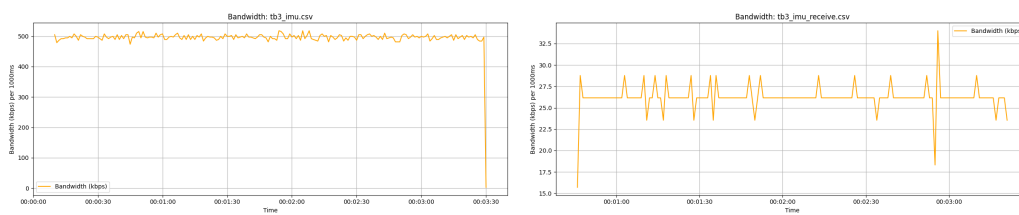


Figure A.9: /imu Topic Bandwidth Analysis.

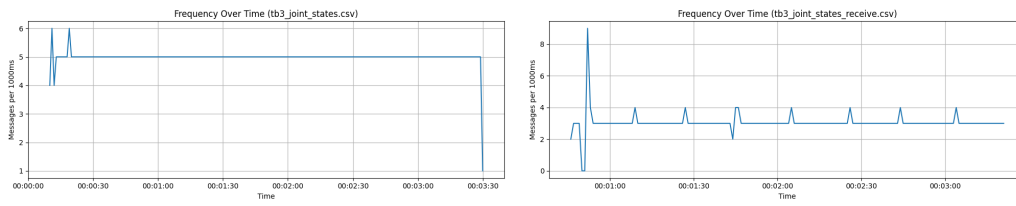


Figure A.10: /joint_state Topic Frequency Analysis.

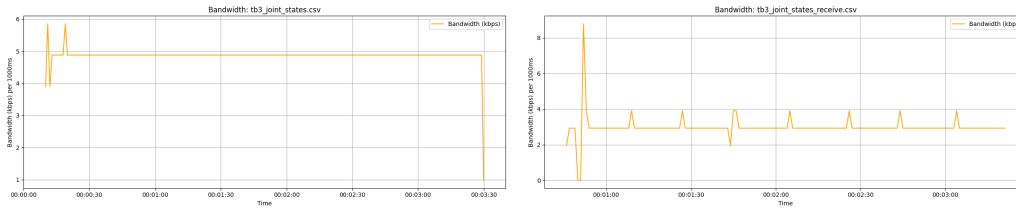


Figure A.11: /joint_state Topic Bandwidth Analysis.

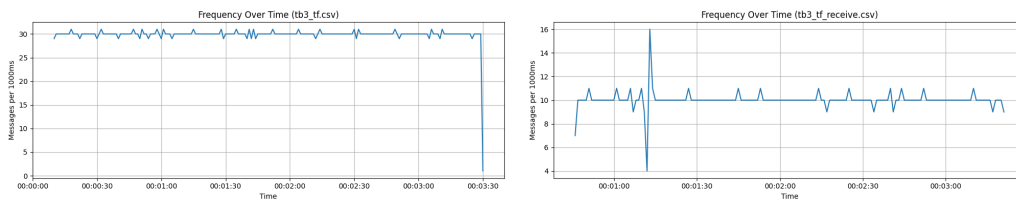


Figure A.12: /tf Topic Frequency Analysis.

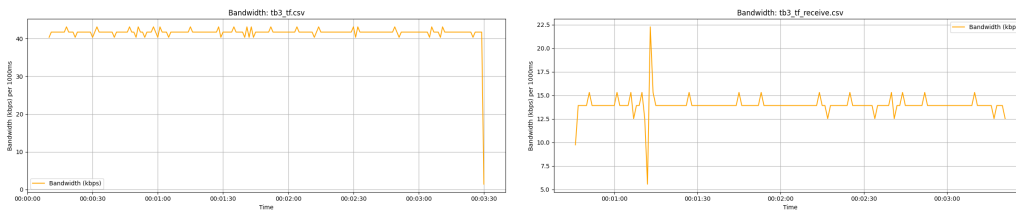


Figure A.13: /tf Topic Bandwidth Analysis.