A Preliminary Systematic Mapping on Software Engineering for Robotic Systems: A Software Quality Perspective

Marcela G. dos Santos
Bianca M. Napoleão
Fabio Petrillo
marcela.santos1@uqac.ca
bianca.minetto-napoleao1@uqac.ca
fabio@petrillo.com
Université du Québec à Chicoutimi
Chicoutimi, Québec, Canada

Darine Ameyed
darine.ameyed.1@ens.etsmtl.ca
Synchromedia Laboratory,
École de Technologie Supérieure,
Université du Québec
Montreal, Québec, Canada

Fehmi Jaafar
jaafarfe@crim.ca
Computer Research Institute of
Montréal
Montreal, Québec, Canada

ABSTRACT

Robotic systems have been increasingly employed in everyday tasks. Considering that software plays a crucial point in robot systems, to investigate how software engineering concepts in a software quality perspective can improve robotic systems. In this work, we present a systematic mapping to identify and classify the state-of-art of software engineering for robotic systems in a quality software perspective. We selected and systematically analyzed a final set of 35 primary studies extracted from an automated search on Scopus digital library.

This work presents three main contributions. Firstly, we organize a catalogue of research studies about software engineering, more specifically software quality applied in robotic systems. Next, we systematically analyze software quality areas used in robotic systems. Finally, we discuss insights into research opportunities and gaps in software engineering to robotic systems for future studies.

As a result, we observed that there are studies in the robotic systems area, addressing in a combined way, software engineering approaches and software quality aspects. The less investigated software quality aspect is security. Due to this fact, we presented an overview of the state-of-art on blockchain applying in robotics systems. Blockchain brings opportunities for changing the ways that robots interact with humans. Finally, we identify research opportunities and gaps in software quality on robotic systems, presenting an overview for future studies.

CCS CONCEPTS

• **Software and its engineering → Embedded software**; • **Computer systems organization → Embedded systems**; *Robotics.*

KEYWORDS

Robotic systems, software quality, systematic mapping, software engineering

SERP4IoT'20, July, 2020, Virtual

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

ACM Reference Format:
Marcela G. dos Santos, Bianca M. Napoleão, Fabio Petrillo, Darine Ameyed, and Fehmi Jaafar. 2020. A Preliminary Systematic Mapping on Software Engineering for Robotic Systems: A Software Quality Perspective. In . ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The use of robotic systems in everyday tasks has grown in the last years. The robotic applications can be industrial operations, surgical procedures, infotainment, and home service tasks or mission-critical. The main reason for this phenomenon is that robots became "smarter" and cost-effective. The idea is that the robotic systems can do the same activity more efficiently and effectively than a human [1].

According to Ford [20], we are in a crucial moment for robotics because we are living a transition movement. Specialized robots, built to operate in highly controlled environments on a specific task, will be replaced to general-purpose robots that can work in a heterogeneous environment, intermixed with humans, and perform a broad spectrum of functions. It is expected that the robots start to replace or to assist human. The robotics systems are becoming more sophisticated, distributed, and integrated [11].

Using software concepts as Software Engineering (SE) methods, techniques, and tools through application of systematic, disciplined, and quantifiable approaches to the development, operation, and maintenance of software, software engineering has contributed to improving software development [17]. This fact leads us to affirm that performing studies on SE for robotic systems is not only essential but also strategic. As benefits, we can cite: to develop bigger, faster, cheaper robotic software systems, to make it possible to build and evolve new robotic software systems [11].

Nevertheless, the main goal of SE is to produce quality software. Software quality management is an SE area focused on the investigation of practices that can improve quality aspects of a produced software [45]. Considering that software plays a crucial point in robot systems, efforts to improve the quality of robotic systems is important since quality characteristics such as integrity, efficiency, usability, reliability, or maintainability [26] impact robotic systems directly.

The **goal** of this study is to identify, classify, and evaluate state-of-art on software quality aspects in robotic systems. More specifically, we intend to (i): identify software quality aspects that have been investigated by SE and robotic community; and (ii) detect and classify

SE approaches used to address software quality aspects on robotic systems. To achieve this goal, we performed a systematic mapping study on software engineering for robotic systems on software quality perspective. In other words, we systematically extracted data and analyzed these studies to establish the relation between software quality and robotic systems.

The main **contributions** of this study are: (i) a catalogue offering research studies about software engineerin, more specifically, software quality applied in robotic systems, (ii) a systematic analysis of software engineering and software quality areas applied in robotic systems; and (iii) insights of research opportunities and gaps in software engineering to robotic systems for future studies.

The **audience** of this study are (i) researchers interested in having an overview and contributing to the research areas addressed in this study; and (ii) practitioners interested to understand the research on the application of software engineering and software quality concepts for robotic systems.

The organization of this work is as follows. Section 2 defines the concepts of robotic systems, software engineering and software quality. Section 3 describes in detail the adopted study design. Section 4 presents the results from our analysis. Section 5 discusses the main findings. Section 8 synthesizes the final remarks and future work.

2 BACKGROUND

2.1 Robotic Systems

A robotic system is a combination of hardware and software components as two distinct layers that can be integrated to build a robot [27]. According to Craig [15], robots can be classified into two major classes: industrial and mobile robots.

An industrial robot is a robot which is automatically controlled, re-programmable, multipurpose manipulator programmable in three or more axes, which can be either fixed in place or mobile for use in industrial automation applications [24].

Some numbers can show the impact and importance of the global market of industrial robots. In 2017, 381,000 units of industrial robots were shipped on a global level. This is an increase of 30% compared to the previous year. In addition, from 2013 to 2017, the annual sales volume of industrial robots increased by 114%. Also, in 2017, the sales value increased by 21% compared to 2016. [41].

Mobile robots are machines which are controlled by software and sensors to identify their surroundings and move around in their environment. The mobility brings to the robots an improvement in the operative capabilities, but on the other hand, the complexity increase and brings additional challenges concerning safety [4].

The interactions that happen during the execution of the tasks, a mobile robot can need to interact with the environment, with other robots or with the human operator. The collaboration with the human operator is often desirable for complex robotic tasks, such as navigation and manipulation of objects in hazardous environments.

However, this standard classification has been changed. According to Ford [20], nowadays, there is a transition movement in robotics for special-purpose robots aiming for built robots to operate in highly controlled environments on a specific task, to general-purpose robots that can work in a heterogeneous environment, that interact more and more with humans. These robots, also called smart robots, are equipped with sensors and intelligent software and promise to bring a new industrial revolution [4].

In other words, the number of mobile robots (or smart robots) that have been applied in industrial environments has effectively increased [37]. The number of total smart robots in the market is expected to reach USD $7.85 billion by 2020, at an estimated Compound Annual Growth Rate (CAGR) of 19.2 % between 2015 and 2020 [11].

For our study, we considered all the robotic systems, the industrial manipulators, the mobile robots, and the mobile robots that are applied in the industry.

2.2 Software Quality

SE provides methods, techniques and tools to produce quality software, the SE essential and basic goal. International Standard ISO/IEC 9126 [25] defines software quality as the totality of characteristics of a software product that meet implicit and explicit needs. Explicit software needs are defined in software requirements documents. It also regards to the quality of software development process since the final product must meet client expectations. Implicit software needs are external factors that can be considered subjective by users, but it can lead to serious consequences; for example, effectiveness, safety and satisfaction in a specified use context.

Demand for software quality has motivated a number of researchers to develop software quality models [3, 12, 26]. The McCall model [26] was developed during the 1970s and continues to be applied nowadays. This model purposes eleven main factors (quality characteristics) which have a significant impact on software quality. They are: Correctness, Reliability, Efficiency, Integrity, Usability, Maintainability, Testability, Flexibility, Portability, Reusability, and Interoperability.

3 STUDY DESIGN

This section covers the study design method conducted to understand the state-of-art pertaining software engineering, more specifically, software quality and robot systems.

This study employed a Systematic Mapping (SM) method. According to Kitchenham & Charters [29], a SM study provides a wide overview of a research area seeking to organize and summarize a quantity of existing evidence regarding a topic of interest. Accordingly, a SM was appropriate to gather and summarize existing contributions broadly from SE, including software quality and robotic systems. We followed well-established guidelines [29, 43] to perform our SM study.

3.1 Research question

We refined our research goal into one Research Question (RQ):

RQ1: What are software engineering approaches used to address software quality aspects in the robotic system domain?

The intent of this question is to detect and analyse software engineering approaches applied to improve software quality aspects in robotic systems.

3.2 Search strategy and study selection

The search strategy adopted is automated search. According to [29], automated search is the most common adopted search strategy to identify relevant studies for a SM or a Systematic Literature Review (SLR). In order to perform an automated search, the first step is the creation of a search query [29]. Our search query is:

```
((robot*) AND ("software engineering"))
```

We executed our search query on *Scopus* digital library considering three metadata fields: title, abstract and keywords. *Scopus* is one of the most commonly used digital library in Computer Science [35, 59] with more than 60 million records. Scopus digital library includes papers from several international publishers, including Cambridge University Press, Institute of Electrical and Electronics Engineers (IEEE), Nature Publishing Group, Springer, Wiley-Blackwell and Elsevier.

Following, we define our study selection criteria. The study selection criteria consist in Inclusion Criteria (IC) and Exclusion Criteria (EC) elaborated in order to filter our set of studies to answer our RQs.

The inclusion criteria are:

IC1: The study must be a primary study;

IC2: The paper must be a conference paper or article;

IC2: The study must address software engineering and software quality aspects applied in robotic systems.

The exclusion criteria are:

EC1: The study published as an abstract;

EC2: The study is not written in English;

EC3: The study is a short paper, keynote, tutorial, challenge and showcase;

EC4: The study does not present an overview of software engineering and software quality concerning robotics systems.

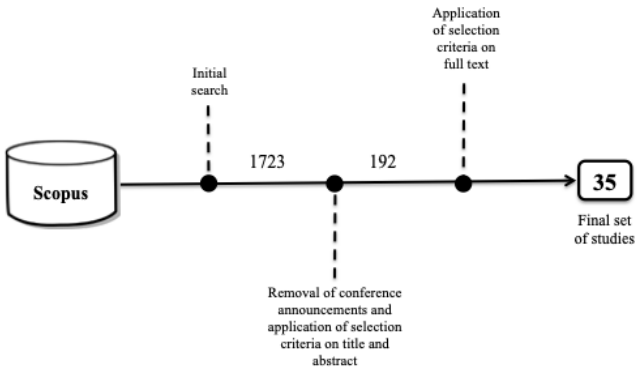

Figure 1: Overview of the search strategy and study selection processes

As illustrated in Figure 1, a total of 1723 studies were returned from the automated search execution. Following we removed conferences announcements and applied the selection criteria (IC and EC) on title and abstract leading to 192 candidate studies. Next, the selection criteria were applied on studies full text, resulting in a final set of 35 included studies.

3.3 Data extraction and synthesis

We aim to identify the relation between software quality and the development of robotic systems. To answer our Resarch Question we created a data extraction form. This form was created as a spreadsheet with specific fields to support our data extraction activity.

Firstly, we extracted from each included study in the concepts that regard the main topic addressed by the contribution provided by each study. It is important to highlight that the fact that in some cases, there is more than one software quality addressed by the study, but we decided to highlight the main topic in our study.

Secondly, we combined the set of concepts to obtain representative categories. We used the McCall quality model [26] and Pressman book [45] to guide the major categorization step; both renowned references in SE field.

As a result from our categorization, "Software Quality for Robotic Systems" was related to: security, usability, modularity, availability, resilience, maintainability, safety and reliability. "Software Engineering Approaches for Robotic Systems" were related to: architecture/design (analysis or presentation of an architecture), coding(coding or analysis of an implementation), framework/pattern (coding, analysis or presentation of a framework), model/method (analysis or presentation of methodology that focuses on creating and exploiting models) and test (analysis or presentation of verification and validation).

It is worth to mention that we are not interested in all software engineering approaches that are applied in robotic systems, but only in the approaches used together software quality aspects on the robotic systems. The final list of included studies, their main addressed topics and software quality aspects are presented in Table 1.

4 RESULTS

This section answers the research question proposed in Section 3 aiming to achieve our study goal.

RQ1: *What are software engineering approaches used to address software quality aspects in the robotic system domain?*

We classified our set of studies using the quality software aspects that are investigated by each study. We used as a basis McCall [26] software quality model to guide our classification. This analysis resulted in 9 software quality aspects as shown Figure 2.

Concerning **RQ1**, it is important to highlight that all studies that addressed any software quality aspect were also applying at least one of the fundamental approaches of software engineering as in shown Figure 3.

We analyzed one by one, the software qualities mapped in our study to map what main approaches in software engineering are applied to improve the specific software quality.

4.1 Availability

According to Pressman [45], software availability is the probability that a program is operating according to requirements at a given point in time. Two studies in our set addressed **Availability**. Natale *et al.* [40] show the software architecture of the humanoid robot and the software engineering best practices used in the research. The main objective is to address the requirements of the users better

Table 1: List of included studies

Reference	Title	Quality Aspect	Main topics
[40]	The iCub software architecture: Evolution and lessons learned.	Availability	Architecture/Design
[32]	OpenGRASP: A toolkit for robot grasping simulation.	Availability	Coding
[47]	The ROCS framework to support the development of autonomous robots.	Modularity	Framework/Pattern
[56]	A reliability evaluation model of distributed autonomous robotic system architectures.	Reliability	Architecture/Design
[30]	Development of real-time control software for autonomous mobile robot.	Reliability	Coding
[5]	Fault avoidance in development of robot motion-control software by modeling the computation.	Reliability	Model/Method
[53]	Modelling Autonomous Resilient Multi-robotic Systems.	Resilience	Model/Method
[52]	Formal development and quantitative assessment of a resilient multi-robotic system.	Resilience	Model/Method
[13]	A knowledge centric approach to conceptualizing robotic solutions.	Reusability	Architecture/Design
[57]	Robot behavior and service-based motion behavior structure design in formation control.	Reusability	Architecture/Design
[48]	Reusability quality metrics for agent-based robot systems.	Reusability	Architecture/Design
[16]	A family of domain-specific languages for specifying civilian missions of multi-robot systems.	Reusability	Architecture/Design
[49]	Design abstraction and processes in robotics: From code-driven to model-driven engineering.	Reusability	Architecture/Design
[60]	Runtime models for automatic reorganization of multi-robot systems.	Reusability	Framework/Pattern.
[18]	A software engineering approach for the development of heterogeneous robotic applications.	Reusability	Framework/Pattern
[2]	A framework-based approach for fault-tolerant service robots.	Reusability	Framework/Pattern
[8]	Stable analysis patterns for robot mobility.	Reusability	Framework/Pattern
[7]	The robotics experience: Beyond components and middleware.	Reusability	Middleware
[50]	Software reuse across robotic platforms: Limiting the effects of diversity.	Reusability	Model/Method
[28]	UML-based service robot software development: A case study.	Reusability Maintainability	Architecture/Design
[39]	Mechatronic objects for real-time control software development.	Reusability Maintainability	Architecture/Design
[6]	Orca: A component model and repository.	Safety	Architecture/Design
[38]	A formal approach to AADL model-based software engineering.	Safety	Architecture/Design
[36]	A UML-based method for risk analysis of human-robot interactions	Safety	Architecture/Design
[23]	Safety oriented software engineering process for autonomous robots.	Safety	Architecture/Design
[14]	Modularity and mobility of distributed control software for networked mobile robots.	Safety	Architecture/Design
[31]	Model-driven interactive system design for therapy. robots.	Safety	Architecture/Design
[22]	A modeling framework for software architecture specification and validation.	Safety	Framework/Pattern
[33]	An XML-driven component-based software framework for mobile robotic applications	Reusability Modularity	Framework/Pattern
[46]	Formal Specification of Robotic Architectures for Experimental Robotics.	Safety	Framework/Pattern
[51]	Simulation and testbeds of autonomous robots in harsh environments.	Safety	Test
[21]	A testing-based approach to ensure the safety of shared resource concurrent systems.	Safety	Test
[42]	Private cloud deployment model in open-source mobile robots ecosystem.	Security	Framework/Pattern
[34]	Evaluating the usability of robot programming toolsets.	Usability	Coding
[55]	Identifying organizational barriers - A case study of usability work when developing software in the automation industry.	Usability	Coding

and improve the system availability. The main approach in software engineering used is Architecture.

The second study that addresses Availability, is the study performed by León *et al.* [32]. It presents a new simulation toolkit that addresses not only availability but also extensibility and, interoperability. The main objective is a tool proposal based on a modular architecture and a designated editor. In this case, the main approach is Coding.

4.2 Modularity

Modularity is the concept that allows software to be divided into separately named and addressable components, sometimes called modules that are integrated to satisfy problem requirements [45].

In our set, just one study addresses the Modularity aspect explicitly. In [47], Ramos *et al.* implemented the RoCS (Robotics and Cognitive Systems) framework for autonomous robots in order to improve the modularity in robotic systems.

4.3 Reliability

Reliability is evaluated by measuring the frequency and severity of failure, the accuracy of output results, the mean-time-to-failure, the ability to recover from failure, and the predictability of the program [45].

In our set, three studies investigated Reliability on the robotic systems and applied a software engineering to improve this aspect. In [56], Xin *et al.* evaluated the reliability of Distributed Autonomous

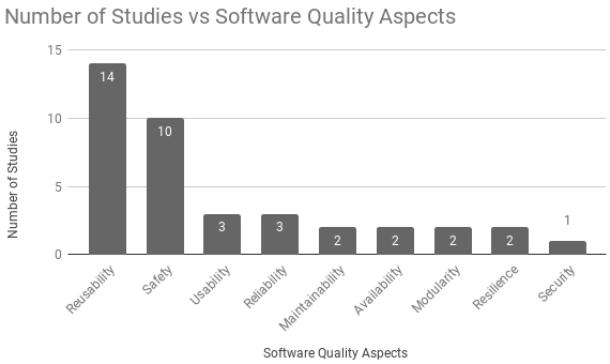

Figure 2: Software quality in robotic Systems

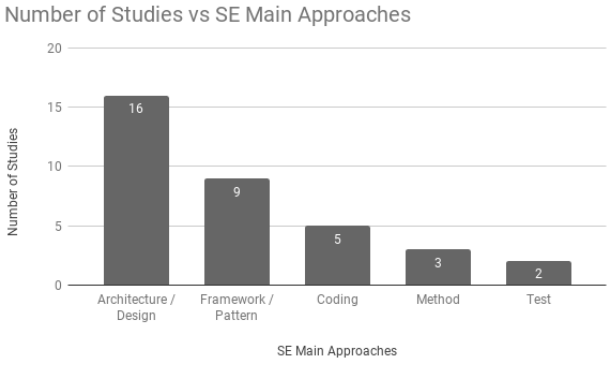

Figure 3: SE main approaches in robotic systems

Robotic System (DARS) architectures systematically, and provided a reliability evaluation model of DARS architectures. In the other two studies ([30], [5]) the reliability is improved using the main approach classified as a Coding in our study.

4.4 Resilience

Resilience is an ability of the system to deliver its services in a dependable way despite the changes [53]. In our set of studies, we have two ([53], [52]) that improve the resilience aspect using Method as a software engineering approach.

4.5 Reusability

The software quality aspect most addressed is **Reusability** with 14 studies that investigated this aspect in robotic systems. In the development of software system for robots, the software reuse is conceived as cut and paste of code lines from program to program.

This practice might work for the development of simple robotic systems (e.g. for educational purposes) or for unique systems (e.g. a research prototype) but not for a complex systems, for example, industrial robots or mobile robots with a high number of sensors [10]. The SE approaches that addressed were Architecture/Design (8/14 studies), Framework/Pattern (5/14 studies) and Method(1/14 studies).

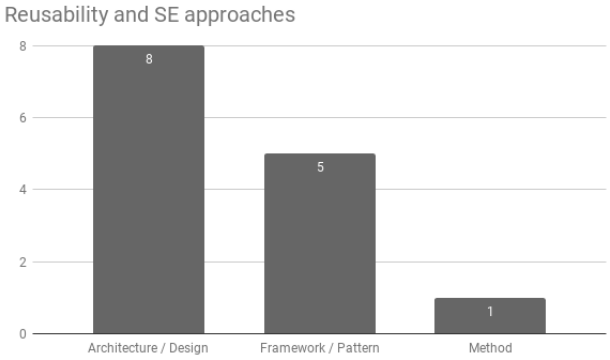

Figure 4: Reusability X Main approaches in Robotic Systems

4.6 Safety

Our study shows that safety has been investigated on the robotic system domain (industrial and mobile robots). According to Bozhinoski et al. [4], one the most important reasons for the success of industrial robotics is its assurance of a high degree of safety.

Robotic systems became smarter and started being integrated in various aspects of everyday life. Therefore, the assurance of safety in robotic system in everyday tasks means ensuring that the robot can move in undiscovered environments and the interaction among robots/robots and robots/humans happened without physical injury of people and loss or damage to equipment/property [4].

The majority of the studies that investigated or addressed this software quality aspect, as in the **Safety** aspect, has as principal SE approach Architecture/Design, but also Framework/Pattern and Test (Figure 5). One of the emergent technology that can improve safety in robotic systems is blockchain. There are several studies about the applying of blockchain approaches. For example, [19], proposed solutions using SC that may provide an infrastructure for ensuring that robotic swarm systems follow specified legal and safety regulations.

4.7 Security

One aspect that was drawn to our attention was the fact that only one study from our set of analysed studies addressed the **Security** aspect. Security and quality are entirely and completely linked [54]. Previous work pointed out that it is necessary to consider the security, reliability, availability, and dependability all through the software life cycle [45].

According to [45], to build a secure system, developers must focus on the quality of the software code, and this focus must begin during the design phase. Indeed, systems with a high level of quality are more difficult to attack.

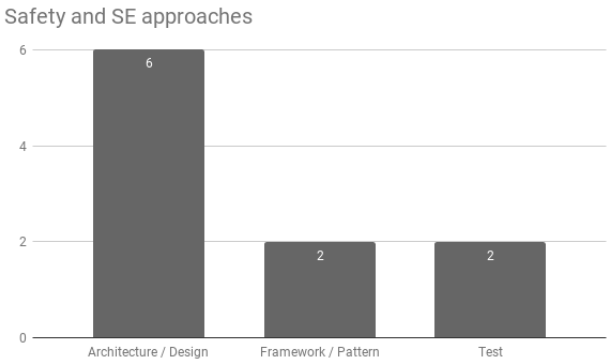

Figure 5: Safety *versus* Main approaches in Robotic Systems

4.8 Usability

The **Usability** software quality has concerning human factors, overall aesthetics, consistency, and documentation. In our study, we could analyze two studies that improve the usability in robotic systems. In [34], Mackenzie *et al.* explore the issues of evaluating such tool-sets as to their usability. And, in [55], Winter *et al.* investigate the connections between usability efforts and organizational factors.

5 DISCUSSION

Software Quality is one area that yet needs being explored for the researchers in Robotic Systems as well as the application of main approaches of Software Engineering. When we initiated our analysis, we found 192 potential studies that addressed SE. After applied one of EC that concerning SQ aspects, only 18% of these studies had the preoccupation of investigating a software quality concept in robotic systems explicitly. This fact leads us to affirm that there is this gap in the intersection of software quality and robotic systems.

Software test, a sub-area in software quality, is investigated only in 2 of the included studies. As stated by Chung et al. [58], many robot weak points and problems were discovered by the tests. This fact emphasizes the need to investigate the software test for robotic systems further.

Only three studies analyzed or presented methods to develop software for robotic systems. Vistbakka and Troubitsyna [53] proposed a multi-agent-based formal outlook on ensuring the resilience of multi-robotic systems. Tarasyuk et al. [52] presented a formal development and quantitative assessment of a resilient multi-robotic system. Smith et al. [50] demonstrated a method for supporting software reuse across robotic platforms and hence facilitating improved software engineering practices. Others methods can be proposed to support any processing activity during software development.

Software architecture is a high-level view of the software system in terms of architectural components as computational elements and connectors that enable interconnections between components [1]. According to [9], the architecture-centric software development increases the quality, modularity and reusability, which confirms our results.

Finally, five studies addressed coding with a focus in coding development or coding implementation analysis.

6 THREATS TO VALIDITY

Threats to validity usually happen in a mapping study, and it was not different in our study. We highlight some of those threats and the mechanism that we applied to address it.

First, the main limitation of this work are the established categories. We could identify other categories based on other quality models. However, we opted to follow the McCall model because it is a renewed quality model in SE. Besides, we could answer other RQs or even go deeper into our results, but we intend with this SM to provide a broad overview regarding SE in quality software on robotic systems.

The second threat to validity is the bias created by the fact that we executed our search query only on Scopus digital library. However, because of this, our mapping is preliminary work; consequently, we decided to start the study only with one digital library and afterward as a future work, by performing the same query in other libraries.

7 RELATED WORK

A systematic review of applying modern software engineering techniques to developing robotic systems was performed by Pons *et al.* on a set of 67 primary studies [44]. The authors identified a growth in the use of approaches, for example, component-based development as well as service-based architecture and model-driven software development. The main difference between our study and [44] is the designing of the research string. We design our string to find all studies about robotic and software engineering; we did not use specific software engineering terms. Pons *et al.* have used the terms MDD (Model Driven Development), MDE (Model Driven Software Engineering), Domain Specific Language, Code Generation, generative programming, CBD (Component Based Development), component based, service based, SOA (service oriented architectures) and Web service.

In [17], Feitosa *et al.* presented a systematic mapping study on software engineering in the embedded software and mobile software development. Their research focuses on to clarifying how software engineering is currently applied in embedded software development. The work performed by Feitosa *et al.* and in our study can be considered similar, each cutting the topic of software engineering in robotic systems from different perspectives, but both performing categorization of the primary studies using fundamental activities of software engineering. The main difference between these two studies is that ours considers software quality aspects and different research questions, thus leading to different results, findings, and future work.

Ahmad *et al.* performed a systematic mapping study for robotic systems on a set of 56 peer-reviewed papers [1]. They did a taxonomically and classified the existing research and systematically mapped the solutions, frameworks, notations and evaluation methods to highlight the role of software architecture in robotic systems. Our study differs from theirs because (i) we specifically focus on

how software engineering approaches are used to improve software quality aspects; (ii) the objective of our study is to characterize existing research on the intersection between software quality and software engineering approaches.

A systematic mapping study on a set of 58 primary studies was performed by Bozhinoski *et al.* [4]. It is state of the art from a software engineering perspective on existing solutions aiming at managing safety for mobile robotic systems. The main contributions of their study are a classification framework for methods or techniques for managing safety, a map of current software methods or techniques for software safety for MRSs, an overview about the emerging challenges and implications for future research, and a replication package for independent replication and verification of this study. Our study had a goal to analyzed the intersection between software quality and software engineering. One of the software aspects analyze was Safety; for this reason, we can consider both studies to be complementary.

8 CONCLUSIONS

We performed a preliminary SM on software engineering for robotic systems on a software quality perspective.

As the aim of a systematic mapping is to provide mechanism representative of the entire domain studied, we believe that results presented in this work show a preliminary state-of-art of software quality in robotic systems. The significant contribution of this work is the identification of research areas that need to be investigated in future work to improve the quality of this study.

As a result, we observed that there are studies in the robotic systems area addressing in a combined way, software engineering approaches and software quality aspects. The less investigated software quality aspect is security.

As future work, we intend to perform our research query in three other software engineering digital libraries: ACM Digital Library, IEEE Xplore, and Web of Science. Besides that, we aim to investigate the problems related to software quality aspects mapped in this study and suggest other approaches to improve each software quality.

REFERENCES

[1] Aakash Ahmad and Muhammad Ali Babar. 2017. Software Architectures for Robotics Systems: A Systematic Mapping Study. *CoRR* abs/1701.05453 (2017). arXiv:1701.05453 http://arxiv.org/abs/1701.05453

[2] Heejune Ahn, Woong-Kee Loh, and Woon-Young Yeo. 2012. A Framework-Based Approach for Fault-Tolerant Service Robots. *International Journal of Advanced Robotic Systems* 9, 5 (2012), 200. https://doi.org/10.5772/54023 arXiv:https://doi.org/10.5772/54023

[3] B. W. Boehm, J. R. Brown, and M. Lipow. 1976. Quantitative Evaluation of Software Quality. In *Proceedings of the 2Nd International Conference on Software Engineering* (San Francisco, California, USA) *(ICSE '76)*. IEEE Computer Society Press, Los Alamitos, CA, USA, 592–605. http://dl.acm.org/citation.cfm?id=800253.807736

[4] Darko Bozhinoski, Davide Di Ruscio, Ivano Malavolta, Patrizio Pelliccione, and Ivica Crnkovic. 2019. Safety for mobile robotic systems: A systematic mapping study from a software engineering perspective. *Journal of Systems and Software* 151 (2019), 150 – 179. https://doi.org/10.1016/j.jss.2019.02.021

[5] Yury Brodskiy, Robert Wilterdink, Stefano Stramigioli, and Jan Broenink. 2014. Fault Avoidance in Development of Robot Motion-Control Software by Modeling the Computation. In *Simulation, Modeling, and Programming for Autonomous Robots*, Davide Brugali, Jan F. Broenink, Torsten Kroeger, and Bruce A. MacDonald (Eds.). Springer International Publishing, Cham, 158–169.

[6] Alex Brooks, Tobias Kaupp, Alexei Makarenko, Stefan Williams, and Anders Orebäck. 2007. *Orca: A Component Model and Repository.* Springer Berlin Heidelberg, Berlin, Heidelberg, 231–251.

[7] G Broten, D Mackay, Simon Monckton, and J Collier. 2009. The Robotics Experience: Beyond Components and Middleware. *IEEE Robotics and Automation Magazine* 16 (03 2009), 46–54.

[8] Davide Brugali. 2005. Stable Analysis Patterns for Robot Mobility. In *PPSDR@ICRA*.

[9] Davide Brugali, Alex Brooks, Anthony Cowley, Carle Côté, Antonio C. Domínguez-Brito, Dominic Létourneau, François Michaud, and Christian Schlegel. 2007. *Trends in Component-Based Robotics.* Springer Berlin Heidelberg, Berlin, Heidelberg, 135–142.

[10] Davide Brugali, Luca Gherardi, Andrea Luzzana, and Alexey Zakharov. 2012. A Reuse-Oriented Development Process for Component-Based Robotic Systems. *Lecture Notes in Computer Science.* https://doi.org/10.1007/978-3-642-34327-8_33

[11] D. Brugali and M. Reggiani. 2005. Software stability in the robotics domain: issues and challenges. In *IRI -2005 IEEE International Conference on Information Reuse and Integration, Conf, 2005.* 585–591. https://doi.org/10.1109/IRI-05.2005.1506537

[12] H. Kaspar M. Lipow G.J. Macleod B.W. Boehm, J.R. Brown and M.J. Merrit. 1984. *Specification of software quality attributes.* Prepared by Boeing for RADC.

[13] Subhrojyoti Roy Chaudhuri, Amar Banerjee, N. Swaminathan, Venkatesh Choppella, Arpan Pal, and P. Balamurali. 2019. A Knowledge Centric Approach to Conceptualizing Robotic Solutions. In *Proceedings of the 12th Innovations on Software Engineering Conference (Formerly Known As India Software Engineering Conference)* (Pune, India) *(ISEC'19)*. ACM, New York, NY, USA, Article 12, 11 pages. https://doi.org/10.1145/3299771.3299782

[14] Liam Cragg, Huosheng Hu, and Norbert Voelker. 2007. *Modularity and Mobility of Distributed Control Software for Networked Mobile Robots.* Vol. 30. 459–484. https://doi.org/10.1007/978-3-540-68951-5_26

[15] John J Craig. 2018. *Introduction to Robotics: Mechanics and Control.* Hoboken, NJ.:Pearson.

[16] Davide Di Ruscio, Ivano Malavolta, and Patrizio Pelliccione. 2014. A family of domain-specific languages for specifying civilian missions of multi-robot systems. 1319 (01 2014), 16–29.

[17] Daniel Feitosa, Katia Felizardo, Lucas Oliveira, Denis Wolf, and Elisa Nakagawa. 2010. Software Engineering in the Embedded Software and Mobile Robot Software Development: A Systematic Mapping.. In *SEKE 2010 - Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering.* 738–741.

[18] Juan-Antonio Fernández-Madrigal, Cipriano Galindo, Javier González, Elena Cruz-Martín, and Ana Cruz-Martín. 2008. A software engineering approach for the development of heterogeneous robotic applications. *Robotics and Computer-Integrated Manufacturing* 24, 1 (2008), 150 – 166. https://doi.org/10.1016/j.rcim.2006.10.002

[19] Eduardo Castelló Ferrer. 2016. The blockchain: a new framework for robotic swarm systems. *CoRR* abs/1608.00695 (2016). arXiv:1608.00695 http://arxiv.org/abs/1608.00695

[20] Martin Ford. 2016. *Rise of the Robots: Technology and the Threat of a Jobless Future.* Basic Books; Reprint edition.

[21] Lars-Ake Fredlund, Julio Marino, Raul Alborodo, and Angel Herranz. 2015. A Testing-Based Approach to Ensure the Safety of Shared Resource Concurrent Systems. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability* 8938 (11 2015). https://doi.org/10.1177/1748006X15614231

[22] Nicolas Gobillot, Charles Lesire, and David Doose. 2014. A Modeling Framework for Software Architecture Specification and Validation. In *Simulation, Modeling, and Programming for Autonomous Robots*, Davide Brugali, Jan F. Broenink, Torsten Kroeger, and Bruce A. MacDonald (Eds.). Springer International Publishing, Cham, 303–314.

[23] V. Gribov and H. Voos. 2013. Safety oriented software engineering process for autonomous robots. In *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA).* 1–8. https://doi.org/10.1109/ETFA.2013.6647969

[24] ISO Central Secretary. 2012. *Robots and robotic devices — Vocabulary.* Standard ISO/IEC ISO 8373:2012. International Organization for Standardization, Geneva, CH. https://www.iso.org/standard/55890.html

[25] ISO/IEC. 2001. *ISO/IEC 9126. Software engineering – Product quality.* ISO/IEC.

[26] P.K. Richards J.A. McCall and G.F. 1978. *Factors in Software Quality (3 volumes).* RADC reports.

[27] J. Jackson. 2007. Microsoft robotics studio: A technical introduction. *IEEE Robotics Automation Magazine* 14, 4 (Dec 2007), 82–87. https://doi.org/10.1109/MRA.2007.905745

[28] Minseong Kim, Suntae Kim, Sooyong Park, Mun-Taek Choi, Munsang Kim, and Hassan Gomaa. 2008. *UML-Based Service Robot Software Development: A Case Study.* Vol. 2006. https://doi.org/10.1145/1134285.1134360

[29] B. Kitchenham and S Charters. 2007. Guidelines for performing Systematic Literature Reviews in Software Engineering.

[30] Jong-Hyuk Lee, Eu-Teum Jo, Hyeon-A Jeong, and Hyungshin Kim. 2011. Development of Real-Time Control Software for Autonomous Mobile Robot. *Journal of Institute of Control, Robotics and Systems* 17 (04 2011). https://doi.org/10.5302/J.ICROS.2011.17.4.336

[31] Moonhee Lee, Hussein Abdullah, and Otman Basir. 2004. Model-Driven Interactive System Design for Therapy Robots. *Journal of Intelligent and Robotic Systems*

39 (04 2004), 345–363. https://doi.org/10.1023/B:JINT.0000026089.28518.b0

[32] Beatriz León, Stefan Ulbrich, Rosen Diankov, Gustavo Puche, Markus Przybylski, Antonio Morales, Tamim Asfour, Sami Moisio, Jeannette Bohg, James Kuffner, and Rüdiger Dillmann. 2010. OpenGRASP: A Toolkit for Robot Grasping Simulation. In *Simulation, Modeling, and Programming for Autonomous Robots*, Noriaki Ando, Stephen Balakirsky, Thomas Hemker, Monica Reggiani, and Oskar von Stryk (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 109–120.

[33] X. Li, Y. Jin, and X. Hu. 2006. An XML-Driven Component-Based Software Framework for Mobile Robotic Applications. In *2006 2nd IEEE/ASME International Conference on Mechatronics and Embedded Systems and Applications*. 1–6. https://doi.org/10.1109/MESA.2006.296924

[34] Douglas MacKenzie and Ronald Arkin. 2001. Evaluating the Usability of Robot Programming Toolsets. *The International Journal of Robotics Research* 17 (09 2001). https://doi.org/10.1177/027836499801700405

[35] D. Maplesden, E. Tempero, J. Hosking, and J.G. Grundy. 2015. Performance analysis for object-oriented software: A systematic mapping. *IEEE Transaction on Software Engineering* 41, 7 (2015), 691–710.

[36] Damien Martin-Guillerez, Jérémie Guiochet, David Powell, and Christophe Zanon. 2010. A UML-based method for risk analysis of human-robot interactions. *Proceedings of the 2nd International Workshop on Software Engineering for Resilient Systems, SERENCE 2010*. https://doi.org/10.1145/2401736.2401740

[37] M. Merten and H. Gross. 2008. Highly Adaptable Hardware Architecture for Scientific and Industrial Mobile Robots. In *2008 IEEE Conference on Robotics, Automation and Mechatronics*. 1130–1135. https://doi.org/10.1109/RAMECH.2008.4681459

[38] Hana Mkaouar, Bechir Zalila, Jérôme Hugues, and Mohamed Jmaiel. 2019. A formal approach to AADL model-based software engineering. *International Journal on Software Tools for Technology Transfer* (07 Mar 2019). https://doi.org/10.1007/s10009-019-00513-7

[39] Patrick F. Muir and Jeremy W. Horner. 1998. Mechatronic objects for real-time control software development. In *Sensors and Controls for Intelligent Machining, Agile Manufacturing, and Mechatronics*, Patrick F. Muir, Peter E. Orban, and Patrick F. Muir (Eds.), Vol. 3518. International Society for Optics and Photonics, SPIE, 251 – 265. https://doi.org/10.1117/12.332802

[40] Lorenzo Natale, Ali Paikan, Marco Randazzo, and Daniele E. Domenichelli. 2016. The iCub Software Architecture: Evolution and Lessons Learned. *Frontiers in Robotics and AI* 3 (2016), 24. https://doi.org/10.3389/frobt.2016.00024

[41] The International Federation of Robotics. 2018. World Robotics - Industrial Robot Report 2018 published. https://ifr.org/ifr-press-releases/news/global-industrial-robot-sales-doubled-over-the-past-five-years

[42] Petri Oksa and Pekka Loula. 2016. Private Cloud Deployment Model in Open-Source Mobile Robots Ecosystem. In *Towards Autonomous Robotic Systems*, Lyuba Alboul, Dana Damian, and Jonathan M. Aitken (Eds.). Springer International Publishing, Cham, 239–248.

[43] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology* 64 (2015), 1 – 18. https://doi.org/10.1016/j.infsof.2015.03.007

[44] Claudia Pons, Roxana Giandini, and Gabriela Arévalo. 2012. A systematic review of applying modern software engineering techniques to developing robotic systems. *Ingeniería e Investigación* 32 (04 2012), 58–63.

[45] Roger Pressman. 2010. *Software Engineering: A Practitioner's Approach* (7 ed.). McGraw-Hill, Inc., New York, NY, USA.

[46] Arunkumar Ramaswamy, Bruno Monsuez, and Adriana Tapus. 2020. *Formal Specification of Robotic Architectures for Experimental Robotics*. Springer International Publishing, Cham, 15–37.

[47] Leonardo Ramos, Gabriel Divino, Esther Colombini, Leonardo Montecchi, and Breno Bernard França. 2019. The RoCS Framework to Support the Development of Autonomous Robots.

[48] Cailen Robertson, Ryoma Ohira, Jun Jo, and Bela Stantic. 2019. Reusability Quality Metrics for Agent-Based Robot Systems. In *Robot Intelligence Technology and Applications 5*, Jong-Hwan Kim, Hyun Myung, Junmo Kim, Weiliang Xu, Eric T Matson, Jin-Woo Jung, and Han-Lim Choi (Eds.). Springer International Publishing, Cham, 121–134.

[49] Christian Schlegel, Andreas Steck, Davide Brugali, and Alois Knoll. 2010. Design Abstraction and Processes in Robotics: From Code-Driven to Model-Driven Engineering. 324–335. https://doi.org/10.1007/978-3-642-17319-6_31

[50] G Smith, R. Smith, and Aster Wardhani. 2005. Software reuse across robotic platforms: limiting the effects of diversity. 252– 261. https://doi.org/10.1109/ASWEC.2005.42

[51] Richard S. Stansbury, Eric L. Akers, Hans P. Harmon, and Arvin Agah. 2007. *Simulation and Testbeds of Autonomous Robots in Harsh Environments*. Springer Berlin Heidelberg, Berlin, Heidelberg, 71–92.

[52] Anton Tarasyuk, Inna Pereverzeva, Elena Troubitsyna, and Linas Laibinis. 2013. Formal Development and Quantitative Assessment of a Resilient Multi-robotic System. In *Software Engineering for Resilient Systems*, Anatoliy Gorbenko, Alexander Romanovsky, and Vyacheslav Kharchenko (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 109–124.

[53] Inna Vistbakka and Elena Troubitsyna. 2019. Modelling Autonomous Resilient Multi-robotic Systems. In *Software Engineering for Resilient Systems*, Radu Calinescu and Felicita Di Giandomenico (Eds.). Springer International Publishing, Cham, 29–45.

[54] Huaiqing Wang and Chen Wang. 2003. Taxonomy of security considerations and software quality. *Commun. ACM* 46 (06 2003), 75–78. https://doi.org/10.1145/777313.777315

[55] Jeff Winter, Kari Rönkkö, and Mikko Rissanen. 2014. Identifying organizational barriers—A case study of usability work when developing software in the automation industry. *Journal of Systems and Software* 88 (2014), 54 – 73. https://doi.org/10.1016/j.jss.2013.09.019

[56] H. Xin, X. Changbin, Z. Qing, and L. Kexin. 2017. A reliability evaluation model of distributed autonomous robotic system architectures. In *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA)*. 203–210. https://doi.org/10.1109/SERA.2017.7965729

[57] Fan Yang, Shirong Liu, and Deguo Dong. 2012. Robot Behavior and Service-based Motion Behavior Structure Design in Formation Control. *Robot* 34 (01 2012), 120. https://doi.org/10.3724/SPJ.1218.2012.00120

[58] Yun Koo Chung and Sun-Myung Hwang. 2007. Software testing for intelligent robots. In *2007 International Conference on Control, Automation and Systems*. 2344–2349. https://doi.org/10.1109/ICCAS.2007.4406752

[59] H. Zhang, B.A. Muhammad, and T. Paolo. 2011. Identifying Relevant Studies in Software Engineering. *Information and Software Technology* 53, 6 (2011), 625–637.

[60] Christopher Zhong and Scott A. DeLoach. 2011. Runtime Models for Automatic Reorganization of Multi-robot Systems. In *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems* (Waikiki, Honolulu, HI, USA) *(SEAMS '11)*. ACM, New York, NY, USA, 20–29. https://doi.org/10.1145/1988008.1988012