IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 14, 2021, accepted November 4, 2021, date of publication November 8, 2021,
date of current version December 9, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3126681

A Systematic Literature Review on Software Maintenance for Cyber-Physical Systems

NADHIRA KHEZAMI[1], MAROUANE KESSENTINI[2], AND THIAGO DO N. FERREIRA[3]
[1]Laboratory of Advanced Systems, Tunisia Polytechnic School, University of Carthage, Tunis 1054, Tunisia
[2]Department of Computer and Information Science, and the Dearborn Artificial Intelligence Research (DAIR) Center, University of Michigan–Dearborn, Dearborn, MI 48128, USA
[3]College of Innovation and Technology, University of Michigan–Flint, Flint, MI 48502, USA

Corresponding author: Nadhira Khezami (nadhira.khezami@fsb.ucar.tn)

ABSTRACT Cyber-physical systems (CPS) are widely used in almost every sector of our modern life. They are also changing the way how systems are designed and maintained as CPS represent a combination of hardware and software components. Thus, the maintenance of CPS is challenging due to the various components that are involved, including embedded software technologies, internet of things (IoT), machine to machine interactions, connectivity and wireless networks. In this paper, we performed a systematic literature review of the existing studies related to software maintenance of CPS starting from January 2006 until December 2020. After extensive manual analysis and filtering, we identified a total of 109 primary studies that we deeply analyzed through different criteria to answer four main research questions about software maintenance activities, techniques, types and evaluation methods used in CPS. Based on the data collected from this survey, we created a taxonomy to classify the existing research works, identified research trends, and highlighted gaps in the literature and avenues for further research in the field.

INDEX TERMS Systematic literature review, cyber-physical systems, software maintenance.

I. INTRODUCTION

In 2006, Dr Helen Gill, the director of the Embedded & Hybrid Systems program at the National Science Foundation (NSF) in the United States, announced a new research project on ''cyber-physical systems'' (CPS). That was the first time that this term has been used. CPS are not traditional embedded/real-time systems. They are intelligent entities which dynamically integrate computation and software with physical processes in a way that is considered as an intersection between the cyber and the physical more than just a union, which requires a deep understanding of the dynamics among computers, networking, and physical systems.

Ever since 2006, large amounts of funds have been awarded for research on CPS, making them as national research priorities in several countries around the world in a way that, today, CPS are considered as the key element to ensure a successful transition towards the upcoming industrial revolution: industry 4.0.

With this rising research interest on CPS, comes the need of appropriate robust maintenance strategies that enable to increase the useful life, the productivity and the reliability of the cyber-physical systems. Obviously, the maintenance of CPS covers two different parts: the machinery maintenance, which involves repairing the equipment (e.g., by replacing the non-functional parts), and the software maintenance that is about the computing side of CPS.

In this paper, we are interested in the software maintenance of cyber-physical systems. In fact, this type of maintenance could lead to predict and prevent the machinery failures of these systems which would ensure their good performance in today's highly competitive environments.

A. PROBLEM DESCRIPTION AND MOTIVATION

The definition of ''software maintenance'' as presented in the IEEE standard [1] is as follows: *''Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment''*.

In today's world where computing devices (such as Internet of Things (IoT) devices) are embedded in so many equipment around us, and where technology is rapidly evolving from day to day; software maintenance became a mandatory requirement. The purpose of software maintenance is not only to

The associate editor coordinating the review of this manuscript and approving it for publication was Mahmoud Elish.

ensure the good performance and operation of these systems by sustaining the software product throughout its life cycle, but also to add and adapt new technologies to the existing environments.

The importance of software maintenance became even higher with the introduction of cyber-physical systems. While each CPS component (either hardware, software or network component) plays an important role in the proper functioning of the CPS, the software is considered as the conductor and the most important of all of the CPS components. Indeed, implementing the right software on CPS allows to optimize the use of each component, to create a high level of synergy between the different elements and to control the whole system effectively. On the other hand, a faulty software could interrupt the interaction of each CPS component with the others and with the surrounding environment which leads to a total failure of the system.

This fact leads us to affirm that performing studies on software maintenance for CPS is essential and even strategic for future research works. Several papers address the maintenance of cyber-physical systems, but only few covered the software maintenance of these systems.

We conducted this study using a systematic literature review (SLR) by following a defined protocol established by Kitchenham and Charters [2]. SLRs are considered as a powerful tool to summarize existing research works and to understand the state-of-the-art about a specific subject. Hence, it helps researchers to identify future directions for that specific research field.

B. CONTRIBUTIONS

Cyber-physical systems represent today a multidisciplinary research field with a very increasing interest worldwide. Software maintenance is one of the pillars to ensure an effective and efficient operation for CPS. Our primary purpose with this systematic literature review is to deeply go through all papers related to this field to study them and propose future orientations and, in order to ensure the quality and the validity of our study's output, this SLR follows a protocol defined in the literature [2]–[4]. We used various electronic databases and a large number of articles to comprise all the possible candidate studies and cover more works than existing SLRs. We identified a final set of 109 studies related to software maintenance for CPS published between 2006 and 2020, fulfilling the quality assessment criteria. These studies can be used by the research and industry communities as a reliable basis and help them conduct further research on software maintenance for CPS.

This SLR contributes to the existing literature in the following ways:

- We present a comprehensive qualitative and quantitative synthesis reflecting the state-of-the-art in software maintenance for CPS with data extracted from those 109 high-rigor studies. Our synthesis covers the following themes: software maintenance activities, techniques, types and evaluation methods applied in the CPS field.
- We establish a common ground towards a unified taxonomy that classifies and categorizes the research works used in our study which may help the researchers to easily reach the desired class of studies based on the category of the software maintenance to be applied on CPS and the best tools and approaches to use for their research purpose.
- We provide guidelines and recommendations based on our findings to support further research in the area by identifying the research gaps and axis that need more investigations and by proposing future research directions on several aspects of the software maintenance of the CPS.

C. RELATED SURVEYS

Few articles in literature have dealt with surveys or reviews on software maintenance aspects for CPS. Existing systematic literature reviews examine findings in very specific aspects of software maintenance applied on CPS. Security challenges are one of the biggest concerns for CPS maintenance and attracted widespread attention of researchers from all around the world. Geismann and Bodden [5] conducted a systematic literature review on CPS dealing with this particular aspect of maintenance. The aim of this paper was to find out which model-driven approaches for secure CPS do exist that cover explicitly both cyber and physical layers of CPS. In the same context, Nazarenko and Safdar in their survey [6] presented an insight on vulnerabilities and attack types on CPS with distinction between security and safety challenges. Another aspect of CPS maintenance was studied in the SLR conducted by Zhou *et al.* [7] and which focused on investigating several testing methods for CPS. In [8], Santos *et al.* conducted a mapping review on software quality aspects on robotic systems and made a classification of the software engineering approaches used to address software quality aspects on robotic systems.

On the other hand, an almost complete survey on software maintenance was carried out in [9]. In this paper, Malhotra *et al.* presented several metrics, types and activities about software maintenance and maintainability. However, it didn't deal with a specific area of research, such as the CPS [10], [11].

At the time this SLR was conducted, there is no survey in literature that deals with the overall aspects of the software maintenance applied on CPS as we did it in this review. For this, we tried to make our SLR the most comprehensive and complete as possible to be a good reference for researchers interested in this subject.

D. ORGANIZATION

The remainder of the paper is structured as follows. First, we outline in section II the research methodology we followed and the underlying protocol for the systematic literature review. Then, we conducted our survey and reported

and discussed its outcome in section III. Finally, and before we conclude in section VI, we state future directions and perspectives to our study in section V.

II. RESEARCH METHODOLOGY

This section details the performed research steps and the protocol of our systematic literature review. We start in subsection II-A by describing the research questions emphasized in our survey. Then, in subsection II-B, we present the literature search steps that we considered. In subsection II-C, we highlight the inclusion and exclusion criteria to filter our final data set. The data pre-processing step and our proposed taxonomy are described in subsection II-D. The quality assessment criteria are defined in subsection II-E.

A. RESEARCH QUESTIONS

We present here-after the main research questions we seek to answer in this literature review, taking into consideration the aforementioned objectives:

- **RQ1.** What are the software maintenance activities for CPS?
- **RQ2.** What are the techniques used for the automation of software maintenance for CPS?
- **RQ3.** What are the common evaluation methods used to validate software maintenance techniques for CPS?
- **RQ4.** What are the main types of software maintenance used for CPS?

B. LITERATURE SEARCH STRATEGY

We conducted our search in several scientific literature sources in order to collect the maximum of papers that are dealing with our subject and to make our review as comprehensive as possible. The databases we used are described as follows:

- **Digital Libraries:** ACM Library, IEEE Xplore, Science-Direct, SpringerLink.
- **Citation Databases:** Web of Science (formerly ISI Web of Knowledge), Scopus.
- **Citation Search Engines:** DBLP, Google Scholar.

For our search, we needed to cover two different areas: software maintenance and cyber-physical systems. For that, we defined a set of keywords relative to each area that could be considered as scientific synonyms:

- **Set 1:** software maintenance, software testing, software quality
- **Set 2:** cyber-physical system, cyber physical system, embedded, robot, robotic, industry 4.0, autonomous, internet of things, iot, smart

We put as many related keywords as possible aiming not to exclude any relevant approach during our search. These keywords were combined using logical operators **ANDs** and **ORs** to create search terms. The goal of our generated search strings is to collect all papers having at least one keyword from the first set and one keyword from the second set either in their title, or abstract or keywords. The final list of search strings is shown in Table 1.

TABLE 1. Final list of search strings.

Group	keywords
Software Maintenance	`software maintenance` **OR** `software testing` **OR** `software quality`
AND	
Cyber-physical Systems	`cyber-physical system` **OR** `cyber physical system` **OR** `embedded` **OR** `robot` **OR** `robotic` **OR** `industry 4.0` **OR** `autonomous` **OR** `internet of things` **OR** `iot` **OR** `smart`

After that, we refined the resulting set of papers by following a multi-stage model (as shown in Figure 1) in order to keep the most relevant publications as much as possible.

Figure 1 shows the number of articles kept at each step along with the total returned publications. The different steps of our systematic review are as follow:

- **Step 1:** Executing the search queries on the selected databases mentioned above. A total of 7592 references were found.
- **Step 2:** Applying a set of clearly defined inclusion and exclusion criteria (explained in subsection II-C) to our resulting papers, which reduced the list of candidate papers to 5167.
- **Step 3:** Removing the duplicates. A total 4783 articles were kept.

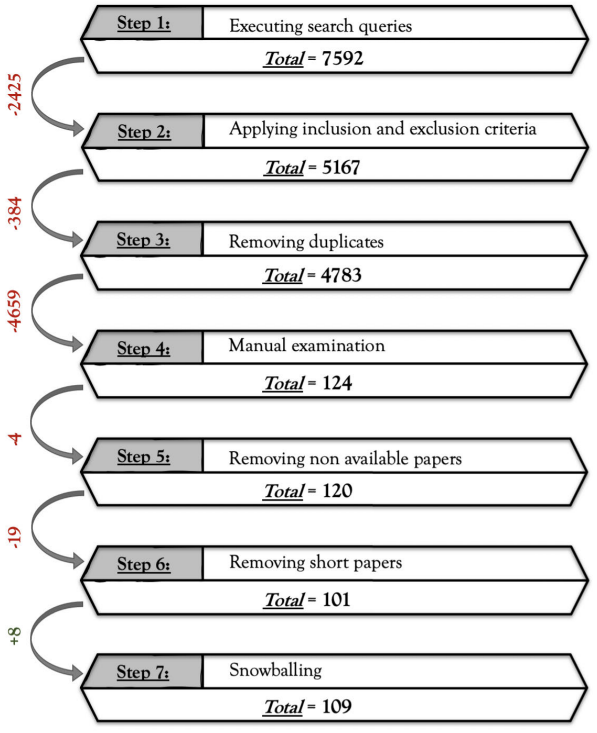

FIGURE 1. SLR steps.

Step 4: Performing a manual examination of titles and abstracts to discard irrelevant publications and exclude those that describe unrelated domains, such as Blockchain for example, or papers talking about the use of CPS for maintenance. We also looked at the body of the paper whenever necessary to make sure that we only keep papers that deal with software maintenance applied to cyber-physical systems. At this step, we also applied a study quality assessment (which will be described in subsection II-E). This has drastically reduced our set of papers to 124 articles.

Step 5: Downloading the obtained set of articles. Four papers were not available online, which made us to remove them from our list. At the end of this step, we kept a list of 120 articles.

Step 6: We removed all short papers having less than 5 pages. Our list at this stage is limited to 101 articles.

Step 7: Finally, applying forward as well as backward snowballing as recommended by Wohlin [3], until no additional papers were found. This allowed us to add 8 new papers satisfying our selection criteria for this SLR, and to get a final set of 109 relevant papers.

C. INCLUSION AND EXCLUSION CRITERIA

We defined a set of inclusion and exclusion criteria that we followed to filter out irrelevant papers as explained in Stage 2 of our search methodology.

1) INCLUSION CRITERIA

All of the following criteria must be satisfied in the selected primary studies:

1) **Date range:** we limited our search in a range between 2006 -the year when the term ''cyber-physical systems'' has first appeared- and 2020 since this SLR was conducted at the beginning of 2021.
2) **Subject area:**The article must be related to computer science and engineering fields and propose software maintenance techniques, methods and tools applied for CPS.
3) **Language:** The paper must be written in English.
4) **Paper types:** We selected articles from journals, conference proceedings and books. In case a conference paper has a journal extension with the same title, authors and abstract, that would be considered as a duplicated article and we would include only one of them.
5) **Availability:** The paper must be available in an electronic format.
6) **Quality assessment:**The paper must pass the quality assessment criteria that are elaborated in subsection II-E.

2) EXCLUSION CRITERIA

If an article selected in **Step 1** of our search methodology holds at least one of the following criteria, it is excluded from our final list.

1) **Publication stage:** We focused in our SLR on collecting mature research works. For that, we excluded all papers that are explicitly marked as ''work-in-progress'', as well as papers published in workshops, symposiums or poster sessions only.
2) **Subject areas:** We excluded all papers that are not related to computer science nor engineering fields, and studies that do not focus on software maintenance applied for CPS.
3) **Grey Literature:** that *''stands for manifold document types produced on all levels of government, academics, business, and industry''* [12] and covers all research works that have been published in non-commercial nor academic forms.

D. STUDY CLASSIFICATION

To address our research questions formulated in subsection II-A, we conducted our systematic literature review in order to seek answers to the following questions:

- Software maintenance activities for CPS (related to RQ1).
- Automation techniques of software maintenance for CPS (related to RQ2).
- Evaluation methods of software maintenance for CPS (related to RQ3).
- Software maintenance types used for CPS (related to RQ4).

We outline in Table 2 the keywords we defined for our research questions as follows to identify relevant papers:

- **RQ1: Software maintenance activities for CPS:**
  - **Testing**: by maintaining control over software modification and adapting programs so that different hardware, software, system features, and telecommunications facilities can be used.
  - **Maintenance quality**: by preventing software performance from degrading to unacceptable levels.
  - **Bugs repairing**: by correcting faults and maintaining control over the software's day-to-day functions,
  - **Security**: by identifying security threats and fixing security vulnerabilities.
- **RQ2:Automation techniques of software maintenance for CPS:**
  - **Formal/ conventional methods**: it includes all conventional tools and methodologies that support software maintenance based on a rigorous mathematical process.
  - **Machine Learning**: it includes unconventional new methods used for software maintenance based on machine learning (ML) and artificial intelligence (AI), and where failures should be self-avoided and self-managed by the system,
  - **Static analysis**: it is about analysing the source code without actually executing the program but mainly collecting information from the structure of the code,

  - **Dynamic analysis**: it is about analysing execution traces to assess the system's behavior through running the system against a specified or intended behavior.
- **RQ3: Evaluation methods of software maintenance for CPS:**
  - **Automated** or **manual** approaches:
    * the manual evaluation approach is based on testing, and there are several testing techniques that can be used manually.
    * the automated evaluation approach uses automated processes to test the software.
  - **Academic** or **industrial** approaches:
    * the academic evaluation approach uses open source tools.
    * the industrial evaluation approach is based on projects with industrial partners.
- **RQ4: Software maintenance types used for CPS:**
  - **Corrective maintenance**: it is about taking actions to rectify bugs and errors observed while the system is in use.
  - **Adaptive maintenance**: it aims to keep the software product up-to date with any new related technology, or to adapt it to new platforms or operating systems based on user's request.
  - **Perfective maintenance**: it includes modifications and updates needed to support new features or to change some functionalities that the user could ask for in order to keep the software usable over a long period of time.
  - **Preventive maintenance**: its goal is to anticipate and prevent problems of the software which might occur in the future in order to improve its future maintainability and provide a basis for future enhancements.

As shown in Table 2, we extracted as well a set of keywords for each answer we provided which helped us to properly classify the resulting papers for our study. For these keywords, we just put the root, and removed all the possible affixes (suffixes or prefixes) in order to cover the maximum forms and derivatives of the word. For example the word *automat* will cover, in our search, words such *automation, automatic, automatically, automate, automated, etc.*.

E. STUDY QUALITY ASSESSMENT

The quality of publications is assessed in parallel at the time of data extraction. After having applied the inclusion and exclusion criteria, we performed a measurement of quality of the primary studies we collected. We followed the quality checklist given by Kitchenham and Charters in [2]. We chose a set of questions (described in Table 3) to which the answers could be a ''Yes'', ''Partially'', or ''No'', which correspond to a score of 1, 0.5, or 0, respectively. If any of the previous answers could not be applied to a question in a specific study, we don't evaluate that study for that particular question. If a study does not meet the quality assessment criteria, it is removed from our final list.

III. RESULTS

We will present in this section the results we obtained from our 109 articles in order to answer the research questions defined earlier. For each article from our last set of papers, we implemented a Python code to go through the body of this paper and count the number of appearances of all the keywords defined in Table 2 for every potential answer to that particular RQ. If this sum of the keywords per paper is superior to 10, the answer will be automatically validated for that RQ and that particular paper. If not, and if the sum is not null, we will go to a manual examination to be able to confirm or not that answer as a way to check the relevance of the paper to a particular research question.

Table 4 summarizes these results based on the taxonomy described in subsection II-D. We will present as well in this section the observations and insights that can be derived from the classification results. Based on existing studies related to software maintenance for CPS, we found four main activities: testing, quality assessment, software repair and security issues detection and repair. The testing of software in CPS received the highest attention from the research community which is a consistent result with existing systematic reviews on software maintenance in general. Furthermore, it is important that security is considered as an important topic for the maintenance of software in cyber-physical systems. Indeed, the vulnerability of software running in CPS can cause severe issues in critical domains. The assessment of the quality of CPS systems and their automated repair received the least number of existing studies of around 17% of selected papers. It is interesting that the detection of quality issues is receiving less attention from the community than the security issues. One main reason could be that performance issues may not be as costly as security vulnerabilities.

To address the different software maintenance problems in CPS, we found that mainly four categories of techniques are used (RQ2). As described in Table 4, both formal methods and machine learning techniques are the most popular techniques used by existing CPS studies with respectively 32.73% and 50.91%. Indeed, most of CPS devices have limited memory and CPU. Thus, the use of formal methods is adequate to verify the correctness of the software and its requirements in CPS. Meanwhile, the recent advances in IoT enabled the distribution of the learning from the CPS data between embedded devices and IoT devices. Thus, the use of machine learning techniques to deal with software maintenance issues in CPS received much attention from the research community. The use of static and dynamic analysis of the code in CPS is still limited with only 12.73% and 3.64% respectively. One possible explanation is the limited focus of the research community in addressing quality issues as described in RQ1. It is also interesting to note that most of existing proposed tools and approaches for software maintenance in CPS are automated and half of them validated in an industrial setting (RQ3). Finally, Table 4 shows that the majority of existing studies focused on both corrective, preventive and adaptive maintenance while very few studies

TABLE 2. List of keywords used to detect the different categories.

Category	Keywords
Software Maintenance activities for CPS (RQ1)	
Testing	test, regression testing, test case, unit test
Quality Issues Detection	software quality, detect, priorit,
Bugs repairing	repair, correct, fix, refactoring
Security	secur, safe, attack surface, virus, hack, vulnerability, vulnerable, spam
Automation techniques of software maintenance for CPS (RQ2)	
Formal or Conventional Methods	model check, formal method
Machine Learning	machine learning, deep learning, reinforcement learning, learning, artificial intelligence, neural networks, regression,
Static analysis	static analysis
Dynamic analysis	dynamic analysis
Evaluation methods of software maintenance for CPS (RQ3)	
Automated	automat
Manual	manual
Academic	academ, theor, open source
Industrial	proprietary, industrial, industry, collaborator, collaboration
Software maintenance types used for CPS (RQ4)	
Corrective	corrective maintenance, repair, correct, fix
Adaptive	adaptive maintenance, adapt
Perfective	perfective maintenance, refactoring
Preventive	preventive maintenance, predictive maintenance, predict, prevent, detect

TABLE 3. PS quality assessment questions [13].

Quality Assessment	Questions
Design	Are the automation techniques of software maintenance for CPS clearly described?
	Are the maintenance activities considered clearly stated and defined?
	Was the sample size justified?
	Are the evaluation measures fully defined?
Conduct	Are the data collection methods adequately described?
Analysis	Are the results of applying the identification techniques evaluated?
	Are the data sets adequately described? (size, programming languages, source)
	Are the study participants or observational units adequately described?
	Are the statistical methods described?
	Are the statistical methods justified?
	Is the purpose of the analysis clear?
	Are the scoring systems (performance evaluation) described?
Conclusion	Are all study questions answered?
	Are negative findings presented?
	Are the results compared with previous reports?
	Do the results add to the literature?
	Are validity threats discussed?

(less than 1%) address issues related to perfective maintenance such as the refactoring of software in CPS. It is understandable that bugs in general cost more than performance issues thus developers are in general reluctant to make their code better as they focus more on fixing existing issues/bugs.

Figure 2 shows the distribution of existing studies across the globe. In order to create a classification per country, we went through the institutions of the authors' affiliations of each article where we extracted all the countries mentioned in these affiliations for each article. Since each paper could have more than one author, it is possible consequently that it corresponds to more than 1 country. Our papers are distributed over 33 countries and 4 continents only, which are Europe (68 articles), America (44 articles), Asia (20 articles) and Oceania (1 article) as presented in Figure 2. None of our final set of articles related to the field of software maintenance for CPS is from Africa.

TABLE 4. Papers references for all categories.

Category	Percentage	Papers
Software maintenance activities for CPS (RQ1)		
Testing	40.41%	[S1]–[S99]
Quality	16.73%	[S5], [S6], [S11]–[S13], [S17]–[S19], [S22], [S24]–[S26], [S28], [S37], [S41]–[S43], [S46], [S47], [S51], [S54], [S56], [S60]–[S62], [S67], [S73], [S77], [S80], [S84], [S87], [S88], [S92]–[S94], [S96], [S98], [S100]–[S103]
Repairing	13.88%	[S3], [S4], [S8], [S12], [S14], [S15], [S17], [S19], [S22], [S23], [S32], [S35], [S37], [S39], [S42], [S43], [S47], [S50], [S62], [S63], [S66], [S71], [S80], [S84], [S87], [S91]–[S96], [S98], [S100], [S101]
Security	28.98%	[S2]–[S6], [S9], [S11], [S14], [S15], [S17]–[S19], [S21], [S22], [S24]–[S27], [S29], [S31]–[S35], [S39], [S41], [S42], [S44], [S45], [S47]–[S49], [S51], [S53], [S54], [S56], [S58]–[S63], [S68]–[S70], [S72], [S73], [S75], [S80]–[S83], [S85], [S87]–[S89], [S91]–[S97], [S99], [S102]–[S107]
Software maintenance techniques for CPS (RQ2)		
Formal methods	32.73%	[S8], [S14], [S26], [S35], [S42], [S47], [S53], [S62], [S66], [S68], [S72]–[S74], [S82], [S91], [S93], [S101], [S107]
Machine Learning	50.91%	[S2], [S5], [S6], [S9], [S12], [S14], [S19], [S24], [S26], [S32], [S35], [S41], [S46], [S48], [S49], [S52], [S54], [S59], [S61], [S62], [S70], [S74], [S90], [S93], [S94], [S96]–[S98]
Static Analysis	12.73%	[S17], [S44], [S47], [S53], [S82], [S91], [S102]
Dynamic Analysis	3.64%	[S36], [S47]
Evaluation methods of software maintenance for CPS (RQ3)		
Automated	64.71%	[S1]–[S24], [S26]–[S39], [S41]–[S64], [S66]–[S74], [S76], [S77], [S79], [S80], [S82], [S84]–[S87], [S89]–[S106], [S108]
Manual	35.29%	[S3]–[S6], [S8], [S9], [S11], [S15], [S16], [S18]–[S20], [S22]–[S26], [S32], [S35], [S37], [S38], [S41], [S43]–[S45], [S48], [S51], [S53], [S56], [S57], [S61], [S62], [S65], [S67], [S69], [S72], [S74], [S75], [S77], [S78], [S80], [S82], [S83], [S86]–[S88], [S90]–[S94], [S102], [S104], [S106]
Academic	49.66%	[S2]–[S10], [S12], [S13], [S18]–[S24], [S26], [S28], [S29], [S32], [S33], [S35]–[S37], [S39], [S42], [S45]–[S48], [S51], [S52], [S54], [S55], [S57]–[S60], [S62], [S63], [S66]–[S74], [S76], [S77], [S79]–[S81], [S83], [S84], [S86], [S87], [S89], [S91]–[S94], [S96]–[S98], [S101], [S102], [S104], [S107]
Industrial	50.34%	[S2]–[S4], [S7]–[S14], [S16]–[S22], [S24]–[S27], [S29], [S31], [S36]–[S39], [S41], [S42], [S44], [S46], [S48], [S50]–[S61], [S63], [S65], [S67], [S71]–[S74], [S76], [S77], [S79]–[S81], [S84], [S87], [S88], [S91]–[S94], [S96], [S98], [S99], [S101]–[S103], [S105], [S106], [S109]
Software maintenance types used for CPS (RQ4)		
Corrective Maintenance	35.19%	[S3]–[S6], [S8], [S10], [S12], [S14], [S15], [S17]–[S19], [S21]–[S23], [S25]–[S30], [S32]–[S37], [S39], [S41]–[S48], [S50]–[S54], [S56], [S58], [S60]–[S67], [S69]–[S73], [S76]–[S82], [S84], [S87], [S88], [S90]–[S104], [S107]
Adaptive Maintenance	28.76%	[S2], [S3], [S5]–[S10], [S13], [S14], [S16]–[S19], [S21]–[S24], [S26]–[S30], [S32], [S34], [S36], [S39], [S40], [S43], [S45]–[S49], [S51], [S52], [S58], [S59], [S61], [S62], [S64]–[S68], [S70]–[S73], [S77], [S80], [S83]–[S85], [S87], [S88], [S90], [S92]–[S97], [S99], [S100], [S102], [S106]
Perfective Maintenance	0.86%	[S27], [S100]
Preventive Maintenance	35.19%	[S2]–[S6], [S9], [S11]–[S13], [S15]–[S26], [S28], [S29], [S31]–[S35], [S37]–[S39], [S41]–[S43], [S45]–[S51], [S53], [S54], [S56]–[S62], [S64], [S67], [S68], [S70], [S73], [S74], [S76], [S77], [S80]–[S85], [S87], [S88], [S90]–[S105]

We extracted in Figure 3 the top 10 countries publishing papers about software maintenance for CPS. The United States of America comes on the top of the list of countries with almost the third of all our final publications (33.03% of all articles), followed by Germany with 12.84% and China with 9.17%. The announcement of the NSF back in 2006 about CPS seems to have given the United States a head start over other countries in the world with a considerable increase in interest over the years in this research field especially that CPS were listed as the number one research priority by the U.S. President's Council of Advisors on Science and Technology.

Figure 4 shows the yearly evolution of the scientific production in the field of software maintenance for CPS between 2006 and 2020. This curve shows the increasing interest of the scientific community about this thematic, in spite of the small drops in 2016 (−33.33% of research interest compared to 2015) and in 2020 (−22.72% of research interest compared to 2019). Although the word "cyber-physical system" officially appeared in 2006, it is

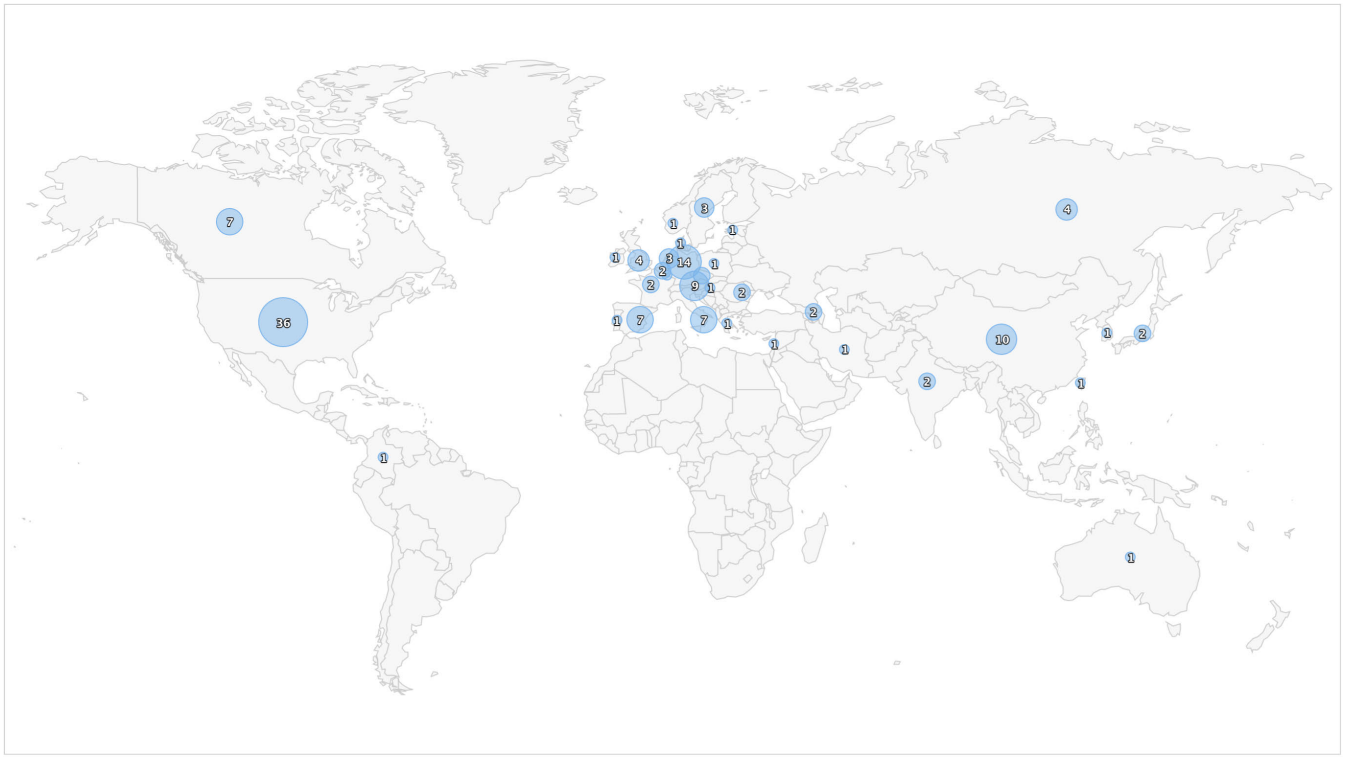

FIGURE 2. Distribution of the publications around the world.

only in 2009 that the first paper dealing with the software maintenance of CPS was published.

A. SOFTWARE MAINTENANCE ACTIVITIES FOR CPS

The need to software maintenance for CPS aims to satisfy the user requirements by maintaining good quality, efficiency, operability, and safety of that software for the CPS. It leads this study to spot four main categories of software maintenance activities:

- Testing
- Quality issues detection
- Bugs repairing
- Security

Figure 5 illustrates the percentage of the papers related to each software maintenance activity. The majority of the papers (40.41%) deal with testing operations. This big interest of the researchers in testing comes to satisfy the ultimate need of ensuring software reliability. To avoid bugs in the program that could lead to challenging debugging sessions, it is important to run the appropriate tests on the software. Security is also a topic of major interest with 28.98% rate among all our final papers. Indeed, it is very important for CPS – as they involve various interconnected systems – to avoid any vulnerability that can be compromised by a hacker attack directly affecting the operation of these systems and to have a software that is robust and resilient to attacks and cyber-security risks.

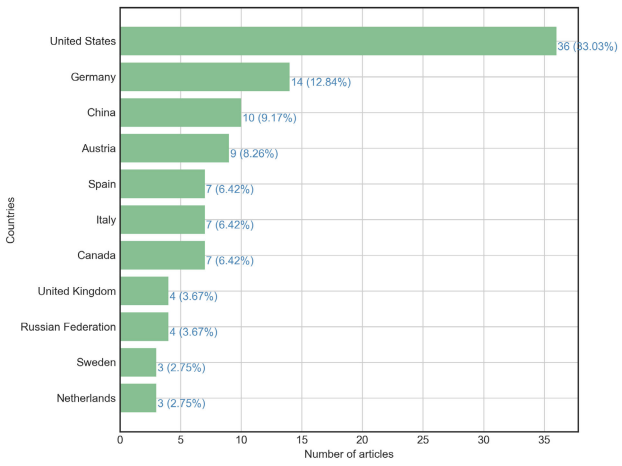

FIGURE 3. Number of publications in the top 10 most active countries.

About 17% of the articles subject of this study are related to quality issues. One of the biggest user requirements has always been to develop good quality products. But it is also equally if not more important to preserve the quality of these products as they are maintained.

Bugs repairing represents 13.88% of our papers. This is about correcting and fixing problems within the software, which is the core of the software maintenance in general. But when it comes to CPS environments, this topic –in spite of its

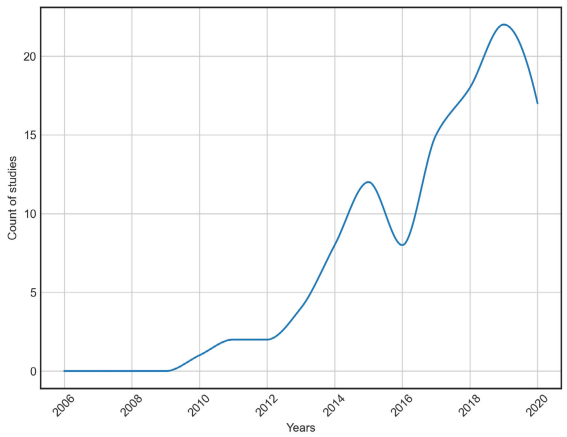

FIGURE 4. Evolution of the research articles from 2006 till 2020.

huge importance– becomes less prioritizing for researchers compared to the other 3 software maintenance activities.

This is more obvious when we look at the yearly evolution of the publications per software maintenance activities for CPS as shown in Figure 6, where we can see that the research interest on software bugs repairing for CPS has only increased for the last three years (2018, 2019 and 2020).

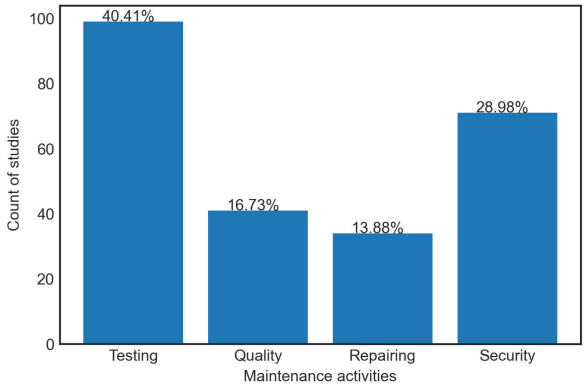

FIGURE 5. Histogram illustrating the count of publications on software maintenance activities for CPS.

B. SOFTWARE MAINTENANCE TECHNIQUES FOR CPS

Several techniques can be used for software maintenance. In this study, we divided these techniques into four essential categories:

- Formal/conventional methods
- Machine learning
- Static analysis
- Dynamic analysis

Figure 7 shows the percentage of publications about software maintenance techniques for CPS. At the top of the used techniques in the articles subject of this study, we find machine learning with a percentage of 50.91% of all articles.

This research interest into Machine Learning has increased since 2016 as shown in Figure 8. This increasing interest in unconventional software maintenance techniques such as ML was expected since CPS are the core of a new industrial revolution that is based on AI, ML and data analysis. Hence, it was normal to expect such techniques to ensure their maintenance.

Besides, formal methods for software maintenance are still interesting researchers in the field of CPS, as they made the subject of 18 articles out of a total of 109 between 2013 and 2020. Static and dynamic analysis seem not to be subjects of big interest for researchers about software maintenance of CPS with only rates of 12.73% and 3.64% respectively from our selected set of papers. Two papers only dealt with dynamic analysis and were both published in 2018.

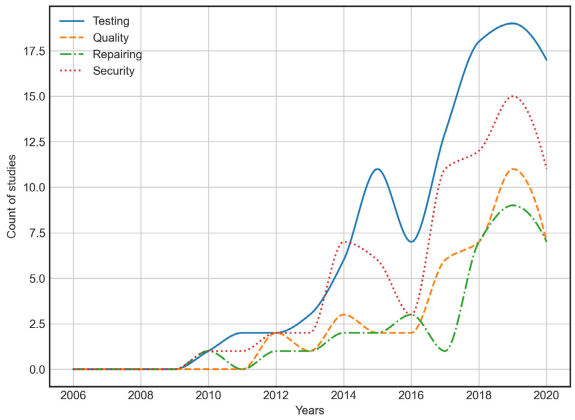

FIGURE 6. Yearly evolution of the publications per software maintenance activities for CPS.

C. EVALUATION METHODS ON SOFTWARE MAINTENANCE FOR CPS

To ensure the effectiveness of software maintenance, it is mandatory to use evaluation methods and tools. The objective of the evaluation phase is to assess whether software maintenance activities fulfill the maintenance requirements or not. In our survey, we considered two major points of view:

- Automated or Manual
- Academic or Industrial

As shown in Figure 9, about 65% of the papers of this study deal with automated evaluation approaches. When we recall that the most used maintenance technique in our papers is ML (see Figure 7), this high research interest on automated approaches seems natural. The manual approaches are still interesting many researchers in the field of CPS as almost one third (35.29%) of our papers deal with this subject. Seventy three of the studies dealt with evaluation methods based on industrial projects, slightly more than those with open-source approaches (72 articles). When it comes to software maintenance for CPS, academic and industrial tools are equally important.

Figure 10 presents the yearly evolution of the publication per evaluation method. The four methods follow almost the same momentum with a slightly higher interest on automated tools.

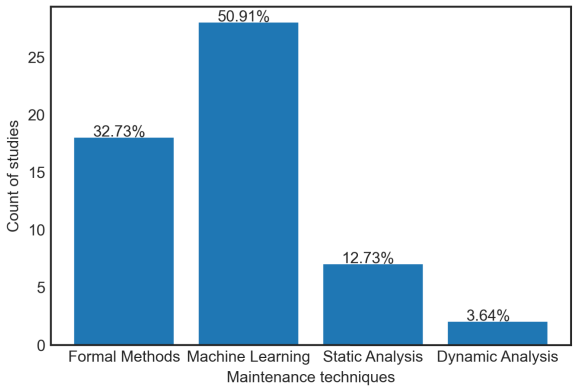

FIGURE 7. Histogram illustrating the count of publications on software maintenance techniques for CPS.

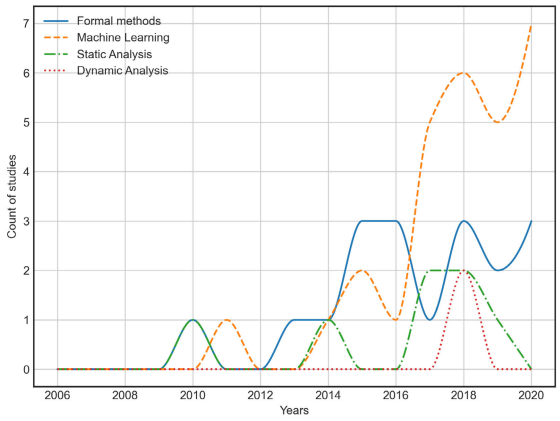

FIGURE 8. Yearly evolution of the publications per software maintenance activities for CPS.

D. SOFTWARE MAINTENANCE TYPES FOR CPS

Software maintenance has four big categories:

- Corrective
- Adaptive
- Perfective
- Preventive

As shown in Figure 11, the preventive maintenance is at the top of researchers' interests in CPS field with a percentage of 35.19% of our selected papers. In fact, this particular type of maintenance is the key element for enterprises risk management as it prevents sudden failures of the system by systematic testing. This was expected in our survey due to the distinct system features of CPS that need advanced types of maintenance.

The need to corrective maintenance will always remain high. This is what explains the big rate (35.19%, equally to preventive maintenance) of articles dealing with this type of maintenance, following almost the same momentum as the preventive maintenance as shown in Figure 12.

More than one quarter of our selected articles (28.76%) deal with the adaptive maintenance for CPS. However, only two articles (0.86%) published in 2019 (see Figure 12), dealt with the perfective maintenance for CPS. This was a bit surprising for us, as there is always a big need to change or add new features and new functionalities to cyber-physical systems. Since these two articles were published recently, we believe that this research area for CPS will be the subject of more future scientific works.

IV. THREATS TO VALIDITY

The assessment of Threats to Validity is critical to secure the quality of a systematic literature review. In this section, we present some threats to validity of our results, according to Wohlin *et al.* [3].

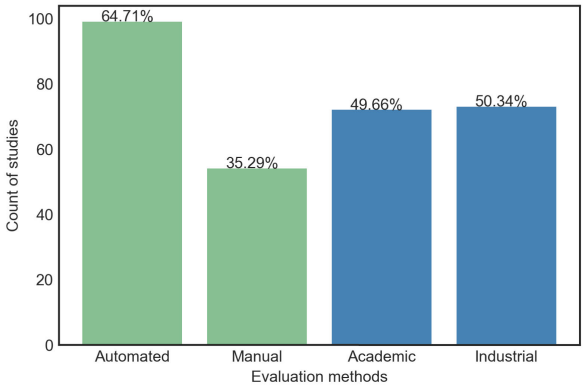

FIGURE 9. Histogram illustrating the count of publications per evaluation method.

A. CONSTRUCT VALIDITY

It refers to the relation between the theory behind the experiment and the observation(s). The main threats in this category are related to the research questions, search string and electronic databases used. With respect to the research questions, we had several discussions about them and goals of our mapping. Also, the databases used in our experiment are well-known sources and used in the literature, where these sources return papers that were published in conferences and journals of the field, thus including the most relevant studies.

There are some threats related to the search strings that can impact the results, and consequently, the number of papers found. For example, some papers may not explicitly mention the keywords we specified. To minimize such threat, we included in the adopted method a snowballing step. In this step, eight additional studies were identified and included. Even though this is a small number, we tried to understand why the papers found in the snowballing step were not selected by our search, and then we prefer to refine our search string and re-run it.

B. CONCLUSION VALIDITY

It relates to issues that affect the ability to draw the correct conclusions from the extracted data in our study. In this category, a threat is related to our classification schemes that was made automatically through a Python code.

To ensure the correctness of the obtained results, a manual verification phase by the three authors has been added after

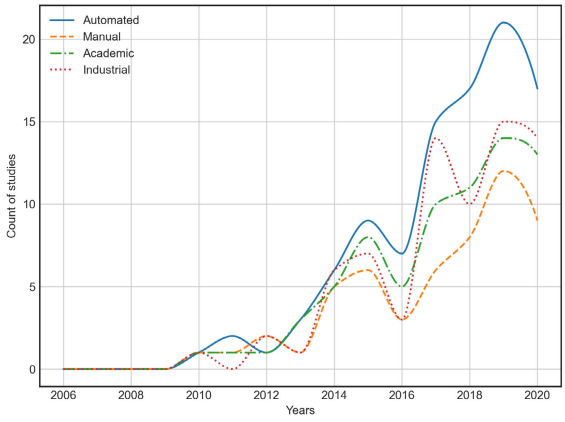

FIGURE 10. Yearly evolution of the publications per software maintenance evaluation method.

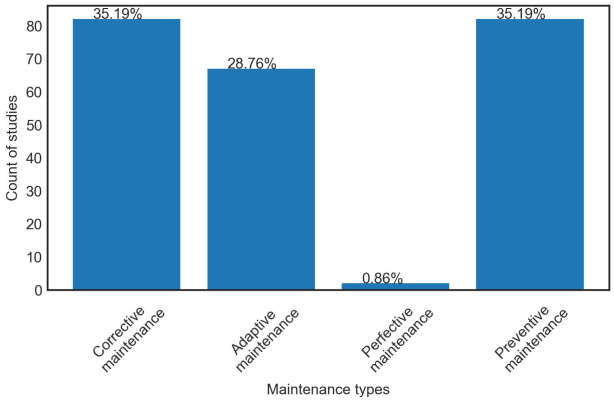

FIGURE 11. Histogram illustrating the count of publications per software maintenance type.

running the code where we went through the articles checking whether the appearance of certain keywords is relevant or not to the RQ answers we were looking for. Moreover, a cross-check was necessary among the authors to make sure that the obtained conclusions address properly the research questions.

C. INTERNAL VALIDITY

It evaluates the relationship between the treatment and the output. In this paper, the treatment is the set of papers included and the outcome is the analysis reported. In this category, a possible threat to internal validity concerns the data extraction using the list of keywords to detect the different categories, since some relevant studies may not be selected. To address this threat, we elaborated well-defined inclusion and exclusion criteria, applying them by carefully analyzing the papers. Besides, a snowballing reading was performed in the primary studies and relevant studies already known by us were selected. Moreover, we had many meetings and discussions during the extraction and selection of studies, because in some of them software maintenance applied to CSP was not so clear to the reader and, in some situations, some subjective decisions were taken.

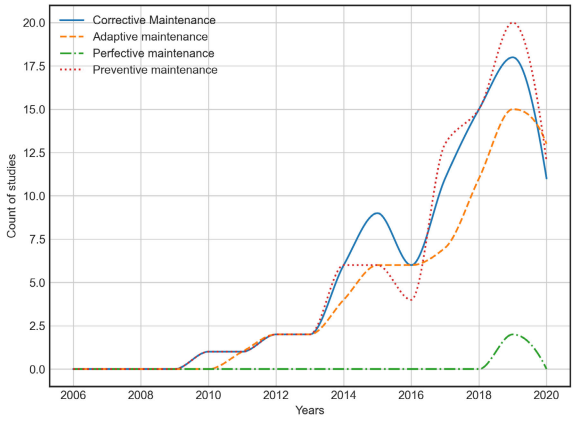

FIGURE 12. Yearly evolution of the publications per software maintenance activities for CPS.

V. IMPLICATIONS AND FUTURE RESEARCH DIRECTIONS

In this section, we identify new opportunities for future research directions related to software maintenance for CPS based on the outcomes of our systematic literature review.

A. DOCKERIZING SOLUTIONS FOR SOFTWARE IN CPS

Most of cyber-physical systems, such as autonomous/ connected vehicles, switched to the use of software containers (e.g., Docker). In fact, several studies show the benefits of containers to improve the reusablility, deployment and modularity of software systems for control systems in vehicles. However, the migration into the use of software containers is bringing many challenges related to their continuous monitoring and integration into embedded systems where limited resources are available. Software Containers have their dependencies packaged together to provide the flexibility of moving containerized features/applications to other compatible environments without the need to re-integrate or re-flash the ECUs in cyber-physical systems. This ability has not been available to the embedded environment and still under-explored. With the availability of such flexibility, an interesting research area can be designed to gain from this benefit for the embedded ECUs.

We searched in our 109 final studies for the words "*docker*" and "*container*". However, none of them contained any of these two words. This gap could lead to an important new research axis that needs to be taken into consideration in the future.

B. MANAGING THE SOFTWARE WORKLOAD IN CPS

The current literature is still lacking a solution and strategy to enable dynamic workload balancing and intelligent scheduling of the software running in CPS (such as embedded ECUs). This enables the CPS to move critical applications to other ECUs in case of a failure or if the CPS switches to power conservation mode. This approach can be fully functional independent of cloud connectivity and prioritizing standard solutions. The current schedulers proposed in the literature for balancing and scheduling the workload of containers is not

optimized for embedded systems. Most of existing scheduler algorithms try to spread number of containers equally to all hosts which can be suitable for cloud environments but not limited resources such as CPS. An intelligent scheduler is needed for instance in embedded devices for autonomous cars, unlike cloud environments, to optimize the memory usage based on requests and priorities. Thus, it is critical to design an optimization-based scheduler algorithm to dynamically balance the software workload in CPS based on different conflicting objectives/scenarios (e.g. memory usage, CPU usage, intra-communications, number of images, etc.).

C. CLOUD AND IoT QUALITY OF SERVICE (QoS) IN CPS

With the emergence of new trends such as the Internet of Things (IoT), the software parts of CPS are distributed from edge devices and fog computing nodes to powerful cloud servers. Thus, finding the best deployment of software components is an extremely challenging and computationally hard problem for software engineers. One of the main problems is that testing and validating deployments are much harder for CPS as for classical software systems, sometimes even impossible to be done on the physical level. Thus, digital twins may be a strategy to test and validate (re)-deployments on a virtual level.

In existing research, researchers worked on dedicated architectural languages for moving software containers from on-premise computing infrastructures to the cloud infrastructures and platforms. However, such architectural language as well as more general system modeling languages miss the ability to cover the current spectrum of having edge/ fog/ cloud architectural layers which may be even dynamically switched if required during the runtime of a system. Thus, it is critical to extend previous developments on finding good deployments of software running in CPS for given requirements. This can be done before deploying the system for its first time, but also during runtime to consider specific needs in case some unexpected disturbance is occurring or finding a better trade-off for a specific situation. Such new paradigms need continuous optimization and learning processes running behind the scenes for having CPS appropriately embedded in our society.

VI. CONCLUSION

In this paper, we have conducted a systematic literature review on software maintenance for cyber-physical systems accompanied by meta-analysis to answer the defined research questions. After a comprehensive search that follows a systematic series of steps and assessing the quality of the studies, 109 publications were identified.

Based on the data extracted from these selected papers, we derived a comprehensive synthesis on the state-of-the-art on software maintenance for cyber-physical systems. We were able to cover four different axis of this research subject: software maintenance activities, techniques, types and evaluation methods applied on CPS. We analyzed the primary studies and displayed the results using charts, graphs and maps to ease the understanding of the collected data.

We were also able to identify three potential research themes related to software maintenance for CPS and that we believe were not –at all or sufficiently– covered by researchers in the field until the date this review was carried. These potential research works are mainly about the virtualization solutions for software using docker containers, managing the software workload in cyber-physical systems when dealing with their software maintenance, and the cloud and IoT quality of service issues.

The results of our systematic review will help both researchers and practitioners to understand the current status of the field, structure it, and identify potential gaps for their future research directions.

REFERENCES

[1] D. V. Edelstein, "Report on the IEEE STD 1219–1993—Standard for software maintenance," *ACM SIGSOFT Softw. Eng. Notes*, vol. 18, no. 4, pp. 94–95, 1993.

[2] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," School Comput. Sci. Math. Univ. Durham, Durham U.K., Tech. Rep. EBSE-2007-01, Jul 2007.

[3] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proc. 18th Int. Conf. Eval. Assessment Softw. Eng.*, 2014, pp. 1–10.

[4] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering—A systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, 2009.

[5] J. Geismann and E. Bodden, "A systematic literature review of model-driven security engineering for cyber–physical systems," *J. Syst. Softw.*, vol. 169, Dec. 2020, Art. no. 110697.

[6] A. A. Nazarenko and G. A. Safdar, "Survey on security and privacy issues in cyber physical systems," *AIMS Electron. Elect. Eng.*, vol. 3, no. 2, pp. 111–143, 2019.

[7] X. Zhou, X. Gou, T. Huang, and S. Yang, "Review on testing of cyber physical systems: Methods and testbeds," *IEEE Access*, vol. 6, pp. 52179–52194, 2018.

[8] M. G. dos Santos, B. M. Napoleão, F. Petrillo, D. Ameyed, and F. Jaafar, "A preliminary systematic mapping on software engineering for robotic systems: A software quality perspective," in *Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng. Workshops*, Dec. 2020, pp. 647–654.

[9] R. Malhotra and A. Chug, "Software maintainability: Systematic literature review and current trends," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 26, no. 8, pp. 1221–1253, 2016.

[10] A. Ghannem, G. El Boussaidi, and M. Kessentini, "Model refactoring using, examples: A search-based approach," *J. Softw., Evol. Process.*, vol. 26, no. 7, pp. 692–713, 2014.

[11] R. Almhana, W. Mkaouer, M. Kessentini, and A. Ouni, "Recommending relevan classes for bug reports using multi-objective search," in *Proc. 31st IEEE/ACM Int. Conf. Automat. Software Eng. (ASE)*, Dec. 2016, pp. 286–295.

[12] J. Schöpfel, "Towards a Prague definition of grey literature," in *Proc. 25th Int. Conf. Grey Literature, Transparency Grey Literature, Grey Tech Approaches High Tech Issues*, Prague, Czech Republic, Dec. 2010, pp. 11–26.

[13] J. Al Dallal, "Identifying refactoring opportunities in object-oriented code: A systematic literature review," *Inf. Softw. Technol.*, vol. 58, pp. 231–249, 2015.

PRIMARY STUDIES

[S1] T. Bijlsma, A. Buriachevskyi, A. Frigerio, Y. Fu, K. Goossens, P. J. Van Der Perk, A. Terechko, and B. Vermeulen, "A distributed safety mechanism using middleware and hypervisors for autonomous vehicles," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2020, pp. 1175–1180.

[S2] I. Aldalur, M. Illarramendi, F. Larrinaga, T. Perez, F. Saenz, G. Unamuno, and I. Lazkanoiturburu, "Advantages of arrowhead framework for the machine tooling industry," in *Proc. 46th Annu. Conf. Ind. Electron. Soc.*, Oct. 2020, pp. 4511–4518.

[S3] H. Kausch, M. Pfeiffer, D. Raco, and B. Rumpe, "An approach for logic-based knowledge representation and automated reasoning over under-specification and refinement in safety-critical cyber-physical systems," in *Proc. Softw. Eng. Workshops*, 2020, pp. 1–8.
[S4] S. Cejka, F. Kintzler, L. Müllner, F. Knorr, M. Mittelsdorf, and J. Schumann, "Application lifecycle management for industrial IoT devices in smart grid use cases," in *Proc. 5th Int. Conf. Internet Things, Big Data Secur.*, vol. 1, 2020, pp. 257–266.
[S5] Y. Qin, T. Xie, C. Xu, A. Astorga, and J. Lu, "CoMID: Context-based multiinvariant detection for monitoring cyber-physical software," *IEEE Trans. Rel.*, vol. 69, no. 1, pp. 106–123, Mar. 2020.
[S6] H. Choi, S. Kate, Y. Aafer, X. Zhang, and D. Xu, "Cyber-physical inconsistency vulnerability identification for safety checks in robotic vehicles," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 263–278.
[S7] X. Liu-Henke, S. Jacobitz, M. Göllner, J. Zhang, S. Scherler, and A. and O. Yarom, "Cyber-physical industry 4.0 laboratory test field to simulate self-optimizing intralogistics," in *Proc. 19th Int. Conf. Mechatronics*, 2020, pp. 1–6.
[S8] T. Liakh and A. Grivtsova, "Dynamic verification of process-oriented control software by the case of crossroad control," in *Proc. Int. Russian Autom. Conf. (RusAutoCon)*, Sep. 2020, pp. 1037–1041.
[S9] H. Noguchi and S. Sugano, "Ephemeral-cyber-physical system: A cloud-like CPS using shared devices in open IoT," *IEEE Syst. J.*, vol. 14, no. 4, pp. 5176–5186, Dec. 2020.
[S10] K. N. Nikolaevich, Z. S. Aleksandrovich, P. U. Nikolaevna, S. M. Petrovich, B. V. Vladimirovich, T. D. Anatolievich, M. A. Ivanovich, and K. V. Aleksandorovich, "Instrumental system of temporal analysis of models of concurrent computing systems constructed using theory of temporary finite state automata," in *Proc. Moscow Workshop Electron. Netw. Technol. (MWENT)*, 2020, pp. 1–8.
[S11] R. Johari, I. Kaur, R. Tripathi, and K. Gupta, "Penetration testing in IoT network," in *Proc. 5th Int. Conf. Comput., Commun. Secur. (ICCCS)*, Oct. 2020, pp. 1–7.
[S12] P. K. Mahato and A. Narayan, "QMine: A framework for mining quantitative regular expressions from system traces," in *Proc. IEEE 20th Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C)*, Dec. 2020, pp. 370–377.
[S13] C. Kruger, A. Narayan, F. Castro, B. Hage Hassan, S. Attarha, D. Babazadeh, and S. Lehnhoff, "Real-time test platform for enabling grid service virtualisation in cyber physical energy system," in *Proc. 25th IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2020, pp. 109–116.
[S14] H. Guissouma, C. P. Hohl, H. Stoll, and E. Sax, "Variability-aware process extension for updating cyber physical systems over the air," in *Proc. 9th Medit. Conf. Embedded Comput. (MECO)*, Jun. 2020, pp. 1–8.
[S15] D. S. Fowler, J. Bryans, M. Cheah, P. Wooderson, and S. A. Shaikh, "A method for constructing automotive cybersecurity tests, a can fuzz testing example," in *Proc. IEEE 19th Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C)*, Mar. 2019, pp. 1–8.
[S16] J. Rodriguez-Guerra, C. Calleja, I. Elorza, A. M. Macarulla, A. Pujana, and I. Azurmendi, "A methodology for real-time HiL validation of hydraulic-press controllers based on novel modeling techniques," *IEEE Access*, vol. 7, pp. 110541–110553, 2019.
[S17] M. Portolan, A. Savino, R. Leveugle, S. Di Carlo, A. Bosio, and G. Di Natale, "Alternatives to fault injections for early safety/security evaluations," in *Proc. IEEE Eur. Test Symp. (ETS)*, May 2019, pp. 1–10.
[S18] R. Ahmadi and J. Dingel, "Concolic testing for models of state-based systems," in *Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2019, pp. 4–15.
[S19] B. Sheehan, F. Murphy, M. Mullins, and C. Ryan, "Connected and autonomous vehicles: A cyber-risk classification framework," *Transp. Res. A, Policy Pract.*, vol. 124, pp. 523–536, Jun. 2019.
[S20] B. Meyers, S. Van Mierlo, D. Maes, and H. Vangheluwe, "Efficient software controller variant development and validation (ECoVaDeVa) overview of a flemish icon project," in *Proc. STAF*, 2019, pp. 49–54.
[S21] I. Raghupatruni, T. Goeppel, M. Atak, J. Bou, and T. Huber, "Empirical testing of automotive cyber-physical systems with credible software-in-the-loop environments," in *Proc. IEEE Int. Conf. Connected Veh. Expo. (ICCVE)*, Nov. 2019, pp. 1–6.
[S22] Y. Zheng, A. Davanian, H. Yin, C. Song, H. Zhu, and L. Sun, "FIRM-AFL: High-throughput greybox fuzzing of iot firmware via augmented process emulation," in *Proc. 28th USENIX Secur. Symp.*, Santa Clara, CA, USA, Aug. 2019, pp. 1099–1114.
[S23] A. Altuntas and J. Baugh, "Hybrid theorem proving as a lightweight method for verifying numerical software," in *Proc. IEEE/ACM 2nd Int. Workshop Softw. Correctness HPC Appl.*, 2018, pp. 1–8.
[S24] R. Ankele, S. Marksteiner, K. Nahrgang, and H. Vallant, "Requirements and recommendations for IoT/IIoT models to automate security assurance through threat modelling, security analysis and penetration testing," in *Proc. 14th Int. Conf. Availability, Rel. Secur.*, Aug. 2019, pp. 1–8.
[S25] B. Scherer, "RTOS aware non-intrusive testing of cyber-physical systems in HIL (hardware in the loop) environment," in *Proc. 16th Int. Meas. Confederation IMEKO Tech. Committee 10 Conf.* Berlin, Germany: IMEKO & EUROLAB, 2019, pp. 20–25.
[S26] M. Eckhart, K. Meixner, D. Winkler, and A. Ekelhart, "Securing the testing process for industrial automation software," *Comput. Secur.*, vol. 85, pp. 156–180, Aug. 2019.
[S27] B. Brenner, E. Weippl, and A. Ekelhart, "Security related technical debt in the cyber-physical production systems engineering process," in *Proc. 45th Annu. Conf. Ind. Electron. Soc.*, Oct. 2019, pp. 3012–3017.
[S28] T. Sondej, K. Sieczkowski, R. Olszewski, and A. Dobrowolski, "Simultaneous multi-site measurement system for the assessment of pulse wave delays," *Biocyber. Biomed. Eng.*, vol. 39, no. 2, pp. 488–502, Apr. 2019.
[S29] K. Hrabovska, N. Simkova, B. Rossi, and T. Pitner, "Smart grids and software testing process models," in *Proc. Smart City Symp. Prague (SCSP)*, May 2019, pp. 1–6.
[S30] J. Malenfant, "Stochastic hybrid systems meet software components for well-founded cyber-physical systems software architectures," in *Proc. 13rd Eur. Conf. Softw. Archit.*, vol. 2, 2019, pp. 132–138.
[S31] J. Kim, S. Chon, and J. Park, "Suggestion of testing method for industrial level cyber-physical system in complex environment," in *Proc. IEEE Int. Conf. Softw. Test., Verification Validation Workshops (ICSTW)*, 2019, pp. 148–152.
[S32] S. Breese, F. Kopsaftopoulos, and C. Varela, "Towards proving runtime properties of data-driven systems using safety envelopes," in *Proc. 12th Int. Workshop Struct. Health Monit. (IWSHM)*. Stanford, 2019, pp. 1–10.
[S33] T. Koch, D. P. F. Moller, and A. Deutschmann, "A Python-based simulation software for monitoring the operability state of critical infrastructures under emergency conditions," in *Proc. IEEE Int. Conf. Electro/Inf. Technol. (EIT)*, May 2018, pp. 290–295.
[S34] A. Brokalakis, N. Tampouratzis, A. Nikitakis, S. Andrianakis, I. Papaefstathiou, and A. Dollas, "An open-source extendable, highly-accurate and security aware CPS simulator," in *Proc. 13rd Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, Jun. 2017, pp. 81–88.
[S35] I. Pill and F. Wotawa, "Automated generation of (F)LTL oracles for testing and debugging," *J. Syst. Softw.*, vol. 139, pp. 124–141, May 2018.
[S36] J. Sini, M. Violante, and R. Dessi, "Computer-aided design of multi-agent cyber-physical systems," in *Proc. IEEE 23rd Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, vol. 1, Mar. 2018, pp. 677–684.
[S37] J. Inoue, F. Kanehiro, M. Morisawa, and A. Mori, "Detecting errors in a humanoid robot," in *Proc. IEEE Int. Conf. Softw. Qual., Rel. Secur. (QRS)*, Mar. 2018, pp. 163–170.
[S38] C. A. González, M. Varmazyar, S. Nejati, L. C. Briand, and Y. Isasi, "Enabling model testing of cyber-physical systems," in *Proc. 21st ACM/IEEE Int. Conf. Modeling Driven Eng. Lang. Syst.*, Oct. 2018, pp. 176–186.
[S39] S. D. Hitefield, M. Fowler, and T. C. Clancy, "Exploiting buffer overflow vulnerabilities in software defined radios," in *Proc. IEEE Int. Conf. Internet Things*, Jul. 2018, pp. 1921–1927.
[S40] M. Perez-Hernandez, B. Alturki, and S. Reiff-Marganiec, "FABIoT: A flexible agent-based simulation model for IoT environments," in *Proc. IEEE Int. Conf. Internet Things*, Jul. 2018, pp. 66–73.
[S41] G. Falco, C. Caldera, and H. Shrobe, "IIoT cybersecurity risk modeling for SCADA systems," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4486–4495, Dec. 2018.
[S42] Y. Fu, W. Choosilp, and Z. Dong, "Model-based test-driven cyber-physical system design," in *Proc. SoutheastCon*, 2018, pp. 1–6.
[S43] I. Drave, S. Hillemacher, T. Greifenberg, B. Rumpe, A. Wortmann, M. Markthaler, and S. Kriebel, "Model-based testing of software-based system functions," in *Proc. 44th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2018, pp. 146–153.
[S44] N. Mansoor, J. A. Saddler, B. Silva, H. Bagheri, M. B. Cohen, and S. Farritor, "Modeling and testing a family of surgical robots: An experience report," in *Proc. 26th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2018, pp. 785–790.

[S45] A. Turlea, "Search based model in the loop testing for cyber physical systems," in *Proc. IEEE 16th Int. Conf. Embedded Ubiquitous Comput. (EUC)*, Oct. 2018, pp. 22–28.

[S46] J. Zenisek, J. Wolfartsberger, C. Sievi, and M. Affenzeller, "Streaming synthetic time series for simulated condition monitoring," *IFAC-Papers Line*, vol. 51, no. 11, pp. 643–648, 2018.

[S47] T. Avgerinos, D. Brumley, J. Davis, R. Goulden, T. Nighswander, A. Rebert, and N. Williamson, "The mayhem cyber reasoning system," *IEEE Secur. Privacy*, vol. 16, no. 2, pp. 52–60, Mar. 2018.

[S48] M. Alenazi, N. Niu, W. Wang, and J. Savolainen, "Using obstacle analysis to support sysml-based model testing for cyber physical systems," in *Proc. IEEE 8th Int. Model-Driven Requirement Eng. Workshop (MoDRE)*, Aug. 2018, pp. 46–55.

[S49] A. Lauber, H. Guissouma, and E. Sax, "Virtual test method for complex and variant-rich automotive systems," in *Proc. IEEE Int. Conf. Veh. Electron. Saf. (ICVES)*, Sep. 2018, pp. 1–7.

[S50] S. Schreiner, R. Gorgen, K. Gruttner, and W. Nebel, "A quasi-cycle accurate timing model for binary translation based instruction set simulators," in *Proc. Int. Conf. Embedded Comput. Syst., Archit., Modeling Simulation (SAMOS)*, Jul. 2016, pp. 348–353.

[S51] F. Tacliad, T. D. Nguyen, and M. Gondree, "DoS exploitation of Allen-Bradley's legacy protocol through fuzz testing," in *Proc. 3rd Annu. Ind. Control Syst. Secur. Workshop*, Dec. 2017, pp. 24–31.

[S52] M. Golagha, A. Pretschner, D. Fisch, and R. Nagy, "Reducing failure analysis time: An industrial evaluation," in *Proc. IEEE/ACM 39th Int. Conf. Softw. Eng., Softw. Eng. Pract. Track (ICSE-SEIP)*, May 2017, pp. 293–302.

[S53] L. Pike, J. Sharp, M. Tullsen, P. C. Hickey, and J. Bielman, "Secure automotive software: The next steps," *IEEE Softw.*, vol. 34, no. 3, pp. 49–55, May 2017.

[S54] B. Yerkes, B. Ramsey, M. Rice, J. Pecarina, and S. Dunlap, "Security analysis of a software-defined radar," in *Proc. 5th Int. Conf. Manage. Leadership Governance*. New York, NY, USA: Academic, 2017, p. 386.

[S55] R. Aggarwal and C. Smidts, "Task allocation in geo-distributed cyber-physical systems," Idaho Nat. Lab., Idaho Falls, ID, USA, Tech. Rep. INL/CON-17-41438, 2017.

[S56] P. E. Weerathunga and A. Cioraca, "The importance of testing smart grid IEDs against security vulnerabilities," in *Proc. 69th Annu. Conf. Protective Relay Eng. (CPRE)*, Apr. 2016, pp. 1–21.

[S57] A. Arrieta, U. Markiegi, and L. Etxeberria, "Towards mutation testing of configurable simulink models: A product line engineering perspective," *J. Ingeniera oftw. Bases Datos*, Mar. 2017.

[S58] H. Choi, S. Kate, Y. Aafer, X. Zhang, and D. Xu, "Cyber-physical inconsistency vulnerability identification for safety checks in robotic vehicles," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 263–278.

[S59] A. Furfaro, L. Argento, A. Parise, and A. Piccolo, "Using virtual environments for the assessment of cybersecurity issues in IoT scenarios," *Simul. Model. Pract. Theory*, vol. 73, pp. 43–54, Apr. 2017.

[S60] S. A. P. Kumar and B. Xu, "Vulnerability assessment for security in aviation cyber-physical systems," in *Proc. IEEE 4th Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)*, Jun. 2017, pp. 145–150.

[S61] U. Adhikari, T. Morris, and S. Pan, "WAMS cyber-physical test bed for power system, cybersecurity study, and data mining," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2744–2753, Nov. 2017.

[S62] J. Straub, "Accelerated stress & reliability testing for software and cyber-physical systems," in *Proc. IEEE Accelerated Stress Test. Rel. Conf.*, Dec. 2016, pp. 1–6.

[S63] F. Ferracuti, A. Freddi, A. Monterié, and M. Prist, "An integrated simulation module for cyber-physical automation systems," *Sensors*, vol. 16, no. 5, p. 645, May 2016.

[S64] A. Estebsari, E. Pons, E. Patti, M. Mengistu, E. Bompard, A. Bahmanyar, and S. Jamali, "An IoT realization in an interdepartmental real time simulation lab for distribution system control and management studies," in *Proc. IEEE 16th Int. Conf. Environ. Electr. Eng. (EEEIC)*, Jun. 2016, pp. 1–6.

[S65] V. Hazlewood, K. Benninger, and G. Peterson, "Developing applications with networking capabilities via end-to-end SDN (DANCES)," in *Proc. XSEDE16 Conf. Diversity, Big Data, Sci. Scale*, 2016, pp. 1–7.

[S66] G. Wainer, "DEVS modelling and simulation for development of embedded systems," in *Proc. Winter Simulation Conf. (WSC)*, Dec. 2015, pp. 73–87.

[S67] A. Arrieta, S. Wang, G. Sagardui, and L. Etxeberria, "Search-based test case selection of cyber-physical system product lines for simulation-based validation," in *Proc. 20th Int. Syst. Softw. Product Line Conf.*, Sep. 2016, pp. 297–306.

[S68] Z. Li and R. Kang, "Strategy for reliability testing and evaluation of cyber physical systems," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage. (IEEM)*, Dec. 2015, pp. 1001–1006.

[S69] R. Cohen, R. Jobredeaux, and E. Feron, "A credible autocoding application within a rocket and its payload," in *Proc. IEEE/AIAA 34th Digit. Avionics Syst. Conf. (DASC)*, Sep. 2015, pp. 1–4.

[S70] C. Berger, "Accelerating regression testing for scaled self-driving cars with lightweight virtualization—A case study," in *Proc. IEEE/ACM 1st Int. Workshop Softw. Eng. Smart Cyber-Phys. Syst.*, Mar. 2015, pp. 2–7.

[S71] M. Prist, A. Freddi, S. Longhi, and A. Monteriu, "An integrated simulation module for wireless cyber-physical system," in *Proc. IEEE 15th Int. Conf. Environ. Electr. Eng. (EEEIC)*, Jun. 2015, pp. 1397–1402.

[S72] Z. Jiang and R. Mangharam, "High-confidence medical device software development," *Found. Trends Electron. Des. Autom.*, vol. 9, pp. 309–391, 2015.

[S73] C. Elks, B. W. Johnson, R. Williams, N. George, and M. Reynolds, "Lessons learned from conducting large scale fault injection experiments on safety critical digital I&C systems," *Int. Top. Meeting Nucl. Plant Instrum., Control, Hum.-Mach. Interface Technol.*, vol. 3, pp. 1944–1959, Jan. 2015.

[S74] Z. Song, P. Labalette, R. Burger, W. Klein, S. Nair, S. Suresh, L. Shen, and A. Canedo, "Model-based cyber-physical system integration in the process industry," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2015, pp. 1012–1017.

[S75] S. Tan, W.-Z. Song, S. Yothment, J. Yang, and L. Tong, "ScorePlus: An integrated scalable cyber-physical experiment environment for smart grid," in *Proc. 12nd Annu. Int. Conf. Sens., Commun., Netw. (SECON)*, 2015, pp. 381–389.

[S76] B. Peischl, "Software quality research: From processes to model-based techniques," in *Proc. IEEE 8th Int. Conf. Softw. Test., Verification Validation Workshops (ICSTW)*, Apr. 2015, pp. 1–6.

[S77] A. Arrieta, G. Sagardui, and L. Etxeberria, "Test control algorithms for the validation of cyber-physical systems product lines," in *Proc. 19th Int. Conf. Softw. Product Line*, Jul. 2015, pp. 273–282.

[S78] A. Dayal, Y. Deng, A. Tbaileh, and S. Shukla, "VSCADA: A reconfigurable virtual SCADA test-bed for simulating power utility control center operations," in *Proc. IEEE Power Energy Soc. Gen. Meeting*, Jul. 2015, pp. 1–5.

[S79] J. O. Blech, M. Spichkova, I. Peake, and H. Schmidt, "Cyber-virtual systems: Simulation, validation & visualization," in *Proc. 9th Int. Conf. Eval. Novel Approaches Softw. Eng. (ENASE)*, 2014, pp. 1–8.

[S80] H. Dantas, Z. Erkin, C. Doerr, R. Hallie, and G. V. D. Bij, "EFuzz: A fuzzer for DLMS/COSEM electricity meters," in *Proc. 2nd Workshop Smart Energy Grid Secur.*, 2014, pp. 31–38.

[S81] C. W. Axelrod, "Reducing software assurance risks for security-critical and safety-critical systems," in *Proc. IEEE Long Island Syst., Appl. Technol. (LISAT) Conf.*, May 2014, pp. 1–6.

[S82] M. W. Whalen, A. Murugesan, S. Rayadurgam, and M. P. Heimdahl, "Structuring simulink models for verification and reuse," in *Proc. 6th Int. Workshop Model. Softw. Eng.*, 2014, pp. 19–24.

[S83] H. Gao, Y. Peng, Z. Dai, and H. Li, "Techniques and research trends of network testbed," in *Proc. 10th Int. Conf. Intell. Inf. Hiding Multimedia Signal Process.*, Aug. 2014, pp. 537–541.

[S84] L. Zhang, H. Sun, X. Ma, C. Xu, and J. Lu, "Challenges in developing software for cyber-physical systems," in *Proc. 5th Asia–Pacific Symp. Internetware*, 2013, pp. 1–10.

[S85] K. Piotrowski and S. Peter, "Sens4u: Wireless sensor network applications for environment monitoring made easy," in *Proc. 4th Int. Workshop Softw. Eng. Sensor Netw. Appl. (SESENA)*, 2013, pp. 37–42.

[S86] C. Huang, B. Miller, F. Vahid, and T. Givargis, "Synthesis of networks of custom processing elements for real-time physical system emulation," *ACM Trans. Design Autom. Electron. Syst.*, vol. 18, no. 2, pp. 1–21, Mar. 2013.

[S87] Z. Jiang, M. Pajic, and R. Mangharam, "Cyber–physical modeling of implantable cardiac medical devices," *Proc. IEEE*, vol. 100, no. 1, pp. 122–137, Jan. 2011.

[S88] X. Zhang, Y. Liu, F. Zhang, J. Ren, Y. L. Sun, Q. Yang, and H. Huang, "On design and implementation of neural-machine interface for artificial legs," *IEEE Trans. Ind. Informat.*, vol. 8, no. 2, pp. 418–429, May 2012.

[S89] J. Wan, H. Suo, H. Yan, and J. Liu, "A general test platform for cyber-physical systems: Unmanned vehicle with wireless sensor network navigation," *Proc. Eng.*, vol. 24, pp. 123–127, Dec. 2011.
[S90] K.-C. Chuang, C.-S. Shih, and S.-H. Hung, "User behavior augmented software testing for user-centered GUI," in *Proc. ACM Symp. Res. Appl. Comput.*, 2011, pp. 200–208.
[S91] A. Boydston and W. Lewis, "Certification challenges of mixed critical cyber-physical rotorcraft systems," in *Proc. 36th Eur. Rotorcraft Forum (ERF)*. Assoc. Aeronautique et Astronautique de France, 2010, pp. 1–11.
[S92] J. Geismann and E. Bodden, "A systematic literature review of model-driven security engineering for cyber–physical systems," *J. Syst. Softw.*, vol. 169, p. 110697, 2020.
[S93] J.-P.-A. Yaacoub, O. Salman, H. N. Noura, N. Kaaniche, A. Chehab, and M. Malli, "Cyber-physical systems security: Limitations, issues and future trends," *Microprocess. Microsyst.*, vol. 77, Sep. 2020, Art. no. 103201.
[S94] S. Ruiz-Arenas, I. Horváth, R. Mejía-Gutiérrez, and E. Z. Opiyo, "Towards the maintenance principles of cyber-physical systems," *Strojniáki vestnik-J. Mech. Eng.*, vol. 60, no. 12, pp. 815–831, Dec. 2014.
[S95] S. A. Asadollah, R. Inam, and H. Hansson, "A survey on testing for cyber physical system," in *Proc. IFIP Int. Conf. Test. Softw. Syst. (ICTSS)*, Sharjah, United Arab Emirates, 2015, pp. 194–207.
[S96] X. Zhou, X. Gou, T. Huang, and S. Yang, "Review on testing of cyber physical systems: Methods and testbeds," *IEEE Access*, vol. 6, pp. 52179–52194, 2018.
[S97] T. Bures, "Software engineering for smart cyber-physical systems: Challenges and promising solutions," *ACM SIGSOFT Softw. Eng. Notes*, vol. 42, no. 2, pp. 19–24, 2017.
[S98] M. Shcherbakov, A. Glotov, and S. Cheremisinov, "Proactive and predictive maintenance of cyber-physical systems," in *Cyber-Physical Systems: Advances in Design & Modelling*. Springer, Jan. 2020, pp. 263–278, doi: 10.1007/978-3-030-32579-4_21.
[S99] M. G. dos Santos, B. M. Napoleão, F. Petrillo, D. Ameyed, and F. Jaafar, "A preliminary systematic mapping on software engineering for robotic systems: A software quality perspective," in *Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng. Workshops*, Jun. 2020, pp. 647–654.
[S100] T. Krismayer, M. Vierhauser, R. Rabiser, and P. Grunbacher, "Comparing constraints mined from execution logs to understand software evolution," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2019, pp. 491–495.
[S101] E. Jharko, "Formalizing the safety functions to assure the software quality of NPP safety important systems," in *Proc. 16th Int. Conf. Informat. Control, Autom. Robot.*, 2019, pp. 637–644.
[S102] J. Edwards and A. Kashani, "Identifying security vulnerabilities early in the ecu software development lifecycle," SAE, Warrendale, PA, USA, Tech. Rep., 2017.
[S103] V. Skormin, A. Dolgikh, and Z. Birnbaum, "The behavioral approach to diagnostics of cyber-physical systems," in *Proc. IEEE AUTOTEST*, Mar. 2014, pp. 26–30.
[S104] M. J. Fernandez, P. J. Sanchez-Cuevas, G. Heredia, and A. Ollero, "Securing uav communications using ros with custom ecies-based method," in *Proc. Workshop Res., Educ. Develop. Unmanned Aerial Syst. (RED UAS)*, 2019, pp. 237–246.
[S105] V. Kuts, M. Sarkans, T. Otto, and T. Tähemaa, "Collaborative work between human and industrial robot in manufacturing by advanced safety monitoring system," in *Proc. DAAAM*, vol. 28, 2017, pp. 1–6.
[S106] P. Kalb and R. Breu, "Tool support for collaborative software quality management," in *Proc. ACM/IEEE 17th Int. Conf. Modeling Driven Eng. Lang. Syst.*, 2014, p. 6.
[S107] M. Rong, "A model for CPS software system trustworthiness evaluation based on attributes classifying," in *Proc. 8th Int. Conf. Comput. Sci. Educ.*, Apr. 2013, pp. 1309–1314.
[S108] T. S. Letia and A. O. Kilyen, "Method for approaching the cyber-physical systems," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, Oct. 2016, pp. 757–766.
[S109] A. C. Becerra, W. Zeng, M.-Y. Chow, and J. J. Rodríguez-Andina, "Green city: A low-cost testbed for distributed control algorithms in smart grid," in *Proc. 41st Annu. Conf. Ind. Electron. Soc.*, 2015, pp. 1948–1953.

NADHIRA KHEZAMI received the Ph.D. degree in electrical engineering, automation and industrial computing from the Central School of Lille, France, in 2011. She is currently an Associate Professor with the University of Carthage, Tunisia, and also a Visiting Professor and a Researcher at the University of Michigan–Dearborn, USA. Her research interests include advanced automatic control, renewable energies, embedded systems, and recently artificial intelligence and cyber-physical systems.

MAROUANE KESSENTINI received the Ph.D. degree from the University of Montreal, Canada, in 2012. He is currently a Tenured Associate Professor and leading a research group on software engineering intelligence. He received several grants from both industry and federal agencies and published over 110 papers in top journals and conferences. He has several collaborations with industry on the use of computational search, machine learning, and evolutionary algorithms to address software engineering and services computing problems. He was a recipient of the prestigious 2018 President of Tunisia distinguished Research Award, the University Distinguished Teaching Award, the University Distinguished Digital Education Award, the College of Engineering and Computer Science Distinguished Research Award, four best paper awards. His AI-based software refactoring invention, licensed, and deployed by industrial partners, is selected as one of the top eight inventions at the University of Michigan, in 2018 (including the three campuses), among over 500 inventions, by the UM Technology Transfer Office.

THIAGO DO N. FERREIRA received the Ph.D. degree in computer science from the Federal University of Parana, in 2019. He is currently an Assistant Professor with the College of Innovation and Technology (CIT), University of Michigan–Flint. His research interests include user preferences, optimization algorithms, and artificial intelligence techniques to address several software engineering problems, such as software requirements, software testing, and software refactoring.