Technical Report

October 2018

# Review of Open Source Simulators in ICS/IIoT Security Context

Jeremy M Watson CBE, Uchenna D Ani

PETRAS INTERNET OF THINGS RESEARCH HUB

**PETRAS Project:**

Analytical Lenses for Internet of Things Threats

(ALIoTT)

PETRAS

UCL DEPARTMENT OF SCIENCE, TECHNOLOGY, ENGINEERING AND PUBLIC POLICY

## Details of document preparation and issue:

| Version no. | Prepared | Checked | Reviewed | Approved | Issue date | Issue status |
|---|---|---|---|---|---|---|
| 1.1 | Uchenna D Ani | Jeremy Watson | Jason Nurse | Jeremy Watson | 26/06/18 | Draft 1 |
| 1.2 | Uchenna D Ani | Jeremy Watson | Al Cook | Jeremy Watson | 06/08/18 | Draft 2 |
| 1.3 | Jeremy Watson & Uchenna Ani | Uchenna Ani | Jeremy Watson | Jeremy Watson | 15/10/18 | Final |
| | | | | | | |

# Executive Summary

---

## Overview

In industrial control systems (ICS), simulation has found widespread use during system design and in tuning process control parameters or exploring the effects of new control algorithms. Simulation enables the assessment of performance at scale and allows research to be conducted by those with limited access to real physical infrastructures. However, as ICSs are often no longer isolated from other networks and the internet, hence are subject to security and safety issues, simulation is also required to understand the issues and their solution. To foster transparent, collaborative and cost-effective studies, demonstrations, and solution development, and attract the broadest interest base, simulation is indeed critical and Open Source is a good way to go since simulators in this category are less expensive to access, install, and use, and can be run with general purpose (non-proprietary) computing equipment and setups.

## Findings

This research presents the following key findings:

1. A lot of Open Source simulation tools exist and span applications areas such as communications and sensor networks (C&WSNs), ICS/SCADA, and IIoT.
2. The functional structures and characteristics that appear common in Open Source simulators include: supported licence types, programming languages, operating systems platforms, user interface types, and available documentation and types.
3. Typical research around Open Source simulators is built around modelling, analysis and optimisation of operations in relations to factors such as flexibility, mobility, scalability, and active user support. No single Open Source simulator addresses all conceivable characteristics. While some are strong in specific contexts relative to their development, they are often weak in other purpose-based research capabilities, especially in the context of IoT.
4. Most of the reviewed Open Source tools are not designed to address security contexts. The few that address security such as SCADASim only consider very limited contexts such as testing and evaluating Denial-of-Service (DoS), Man-in-the-middle (Mitm), Eavesdropping, and Spoofing attacks.

## Recommendations

The following key recommendations are presented:

1. Future developments of Open Source simulators (especially for IIoT) should explore the potential for functionalities that can enable the integration of diverse simulators and platforms to achieve an encompassing setup.
2. Developers should explore the capabilities of generic simulators towards achieving architectures with expansible capabilities into multi-class domains, support easier and faster modelling of complex systems, and which can attract varied users and contributors.
3. Functional characteristics such as; ease of use, degree of community acceptance and use, and suitability for industrial applications, should also be considered as selection and development criteria, and to emphasise simulator effectiveness. This can support consistency, credibility, and simulation system relevance within a domain that is continually evolving.
4. Future Open Source simulation projects developments should consider and adopt the more common structural attributes including; Platform Type, Open Source Licence Type, Programming Language, User Interfaces, Documentation, and Communication Types. These should be further complemented by appropriate editorial controls spanning quality coding, revision control and effective project disseminations and management, to boost simulation tool credibility and wide acceptance.
5. The range of publication dates (earliest to latest) for: citations, code commits, and number of contributors associated to Open Source simulator projects can also support the decision for interests and adoption of specific Open Source projects.
6. Research objectives for ICS/IIoT Open Source simulators should also include security performance and optimisation with considerations towards enhancing confidentiality, integrity and availability.
7. Further studies should explore the evaluation of security topics which could be addressed by simulation – more specifically, proposing how this may be achieved and identifying what can't be addressed by simulation. Investigations into simulation frameworks that can allow multi-mode simulations to be configured and operated are also required. Research into Industry 4.0 System-of-Systems (SoS) security evaluations, dependency, and cascading impacts method or analysis is another area of importance.

# Contents

## List of Figures

# 1. Introduction

## 1.1 Background

Simulation may be defined as the practice of demonstrating or imitating the behaviour or characteristics of a system using another (often small-scale) system. This process has long been a key tool in understanding the impact of design choices in systems in which direct experimentation is expensive or infeasible (for example because the system does not yet exist). In terms of industrial control systems, simulation has found widespread use during system design and in tuning process control parameters or exploring the effects of new control algorithms. At the same time, much of the research on general networked systems, particularly wireless networked systems, has been undertaken using modelling and simulation to allow for the assessment of performance-at-scale and to allow research to be conducted by those with limited access to the internals of routers, wireless nodes, etc.

Leading up to the emergence of Industry 4.0[1], there has been increasing synergy between the networking technologies and process control, motivated by reasons of cost, flexibility and performance amongst other benefits [1]. Bus-based communication systems are increasingly being replaced by network-based communication systems, typically those based on proven internet technologies modified to provide strong synchronization and appropriate latency guarantees. As a consequence, there is considerable opportunity for the integration of the corresponding simulation systems to reflect this new reality for industrial control. This has been happening, and such integrated systems, and the research that uses them, unsurprisingly focus largely on issues of performance.

Whilst performance is indeed critical, networked systems, particularly those that are no longer truly isolated from the wider Internet, may be subject to other very serious issues related to safety and security. Systems within this category are being exposed to new forms of risk typically related to attacks, particularly those that are cyber-related with consequences that can be physical as well as virtual. Whilst standard commercial-off-the-shelf (COTS) internet security technologies help in protecting against such attacks, the behavioural complexity of coupled feedback systems offers new opportunities to the attacker. Researchers need experimental playgrounds where they can assess the feasibilities and modes these attacks can take, the impacts of these attacks, and the efficacy of proposed controls/defences. This is doubly important because the numbers of forensically analyzed attacks on industrial control systems is low and so there is little experience on which to draw; for those situations in which there is evidence, attacks launched by nation-state actors have a high degree of sophistication.

## 1.2 Open Source for Security Simulations

Given that most commonplace simulation exercises are related to objectives of performance testing and evaluation, it is important to define the added objective of security as a new simulation context, as well as considering the motivations for the focus on Open Source simulators.

[1] The digital transformation of industrial markets, involving the evolution to cyber-physical systems, embodying the fourth industrial revolution on the path to an end-to-end value chain with Industrial IoT and decentralized intelligence in manufacturing, productions, logistics and supply chain.

### 1.2.1 Security Context in Simulation

Security may be viewed from the perspective of implementing technical, physical and administrative measures and controls to provide confidentiality, integrity and availability [2]. In the context of simulation, it then points to the practice of testing the existence or lack, of the described implementation objectives. Thus, the 'security' goal for a simulator could be one that enables any implications (or sensitivity) to be assessed that could be caused by an inability to sufficiently address a primary or associated ancillary security requirement. Ancillary security requirements indicate more specific security-related contexts or applications of focus. This can encompass a broad range of potential requirements including vulnerability evaluation, threat analysis, device tampering, remote interference, unauthorised access, role abuse, resilience evaluation, criticality evaluation, defence mechanism evaluation, etc. Describing the applicable security context(s) in relation to the capabilities of simulators will help to clearly clarify on the strengths and limitations associated with the Open Source simulators under study.

It is essential for national security that we have a robust mechanism for identifying and addressing vulnerabilities proactively. The industrial sector is critical to the economic health of the nation and an approach to security that relies on the blind hope that attacks hardly reach or affect ICS operations, and when they do, they will not be sufficiently severe or widespread to cause lasting damage is imprudent at best.

At present, although networking technologies are increasingly being used in process control, the research communities are largely distinct. For the networking community, who have considerable experience of research in network security, to engage more fully with the process control community, there must be experimental platforms that they can use with ease. Such experimental platforms are ultimately likely to range from purely software-based simulation systems through to cyber-physical testbeds, increasing in fidelity and cost.

### 1.2.2 Why Open Source Simulators / Projects

Simulation is critical, and commercial (proprietary) and Open Source simulation tools are often adopted as the preferred instruments by the scientific and engineering community. Commercial simulation tools are typically available through individual licensing agreements for software simulation applications with monetary benefits attached to their use. They do not typically grant open access to the actual code of the software simulation platforms. Of course, this endows the vendors alone with the legal rights to copy or modify the codes. Thus, full control over program codes remain solely with the code authors or creators [3]. In using simulators and of course other software under the commercial licenses, users are subject to licensing conditions – consenting not to use the software in ways not explicitly signed-off by the authors. However, couple of constraints are making commercial tools quite unattractive, some of which include: the limiting of customization and adaptability (which are dependent on the author(s) for updates, fixes, and support), and most importantly, the high cost of license purchase and maintenance. For instance, a US$1.2 billion spending is reported as Gartner's estimate of India's public and private sectors spending on proprietary IT products and service in the healthcare sectors for 2015 [4], a cost which might be significantly reduced if Open Source tools were considered.

In contrast, Open Source simulation approaches draw from the open-source philosophy promoted by free developers' communities referred to as the "open-source way" [5]. The philosophy advocates that any open-source initiative, project or product is meant to be created in the spirit of collaborative participation, free exchange, transparency, rapid prototyping, meritocracy, and community support [3]. More succinctly, the Open Source initiative [6] outlines the criteria for Open Source license compliance, some of which includes: no restriction to software tool sales and sharing, software must include source code and allow for distributions in both raw and compiled forms, license must not be specific to a product, must allow modifications and derived works, no discrimination against fields of endeavours, persons or groups. It is these criteria that drive Open Source tool benefits which do not exist or come at huge financial cost with proprietary simulation tools. Some of these benefits include: continual evolution and less susceptibility to bugs, no vendor lock-in, adaptability to suit personalized business requirements, open access to community support, increased modularity and scalability [7]

A steady increase in interest is noted in the adoption of Open Source software tools and approaches, which includes simulation applications. An online news article [8] reporting on Black Duck Software and North Bridge's survey, found Open Source software to be witnessing massive penetration into very many businesses. From the article, findings indicated that 78% of respondents surveyed indicated that their respective organisations ran part or all of their operations on Open Source tools. This indicated a significant increase from a similar survey in 2010, where 42% of respondents said that they used Open Source in the running of their business. 93 % indicated their organization's use of Open Source had increased or remained the same in the past year, as against commercial tool usage. 64% confirmed participation in Open Source projects, which indicated an increase from 50% in the previous year (2014). The projection was for an anticipated increase to 88% by 2018 for organizations' contributions to Open Source projects. Interestingly, the report also had 66% indicating they consider Open Source tools first before other options. A more recent report – "2018 TIDELIFT Professional Open Source Survey" [9] indicates that 92% of applications in 703 respondents' organisations contained Open Source libraries. Supporting this, two-third of the surveyed respondents indicated that all of their applications made use of some Open Source elements. This suggests a relatively widespread adoption.

In investigating the motivation for adopting open-source software against commercial (proprietary) software tools, Dhir and Dhir [4] found that better security, free support, and ease of software development were the prominent drivers for the adoption of open-source solutions. Based on their study, researchers [10] also suggested an arguable claim that for every successful commercial tool/software, there is potentially an equivalent or sometime better Open Source software. If this is true, there may not be a strong justification to recourse to the large financial expense of commercial tools, given comparable Open Source alternatives.

Notwithstanding the above, it is apparent that from a licensing perspective, the main differences between Open Source and commercial software (including simulators), are associated with the cost and the conditions of using the software. In this era of collaborative workings, these drivers support capacities to achieve the much-desired aggregation or contribution of ideas and concepts, which are open to critiques, reviews and refinements to achieve more robust outcomes and solutions – and at a significantly reduced, or no cost. Potentially, there can be more refined outcomes from a well-coordinated multitude of contributions. To foster transparent, collaborative and cost-effective studies, demonstrations, and solution development, as well as to attract the widest interest base, simulation is indeed critical and open-source is perhaps the better way to go since such simulators are less expensive to access, install, and use, and can be run with general purpose (non-proprietary) computing equipment and setups.

This report lists and discusses commonly used Open Source simulators across the networking and industrial control communities as well as fusions between them. It is the first part of a programme of work intended to examine the use and utility of simulation, particularly Open Source simulator/projects in the context of industrial control system security. It is expected to result in recommendations for the construction of new simulators that address gaps that can be identified on the basis of both paper analysis and the engagement of governmental and industrial stakeholders.

### 1.3 Research Questions

This report focuses on the use of Open Source simulators to capture or represent ICS domain functionalities and processes. By functionalities and processes, we imply production and service control, management, and optimisation. In order to understand the current state of play with respect to the use of simulation, we believe it is important to address a number of research questions which include:

1. What are the common Open Source pure simulation tools used in Networks, ICS, and IIoT research and development projects?
2. What are the functional structures and characteristics that encompass these Open Source simulators? These include the purpose, compatible operating system platforms (e.g. Linux) that the tools run on, underlying generic programming languages (e.g. Python) that the projects use, the usage license (e.g. open) of the tools, and the degree of credibility for the simulation tools
3. How widely used are the Open Source tools (e.g. count of key references to the tools and their adoptions for research)
4. What kind of research has built on these Open Source simulators? (e.g. operational, security analysis performance).
5. What are the strengths, limitations, constraints, and restrictions attributed to these tools in relations to suitability for IIoT simulation?
6. Have any of these simulators been designed with capability to explore industrial-related security analysis?
7. What aspects of security analysis and modelling have been explored using these tools, and what aspects can be explored for future (further) analysis?

This report aims to address the first three of these questions from a generalized perspective, and the remainder with respect to the most important simulators covering the context under study.

## 2. Methodology

The methodology for carrying out this study/review is described in this section. As indicated in Figure 1, the study starts with explaining how the relevant literature was obtained from academic databases and internet, and how the materials were examined to identify Open Source simulation projects and(or) tools. This is followed by the outline and definition of the criteria that provide bases for the analysis and comparison of the Open Source projects and tools.

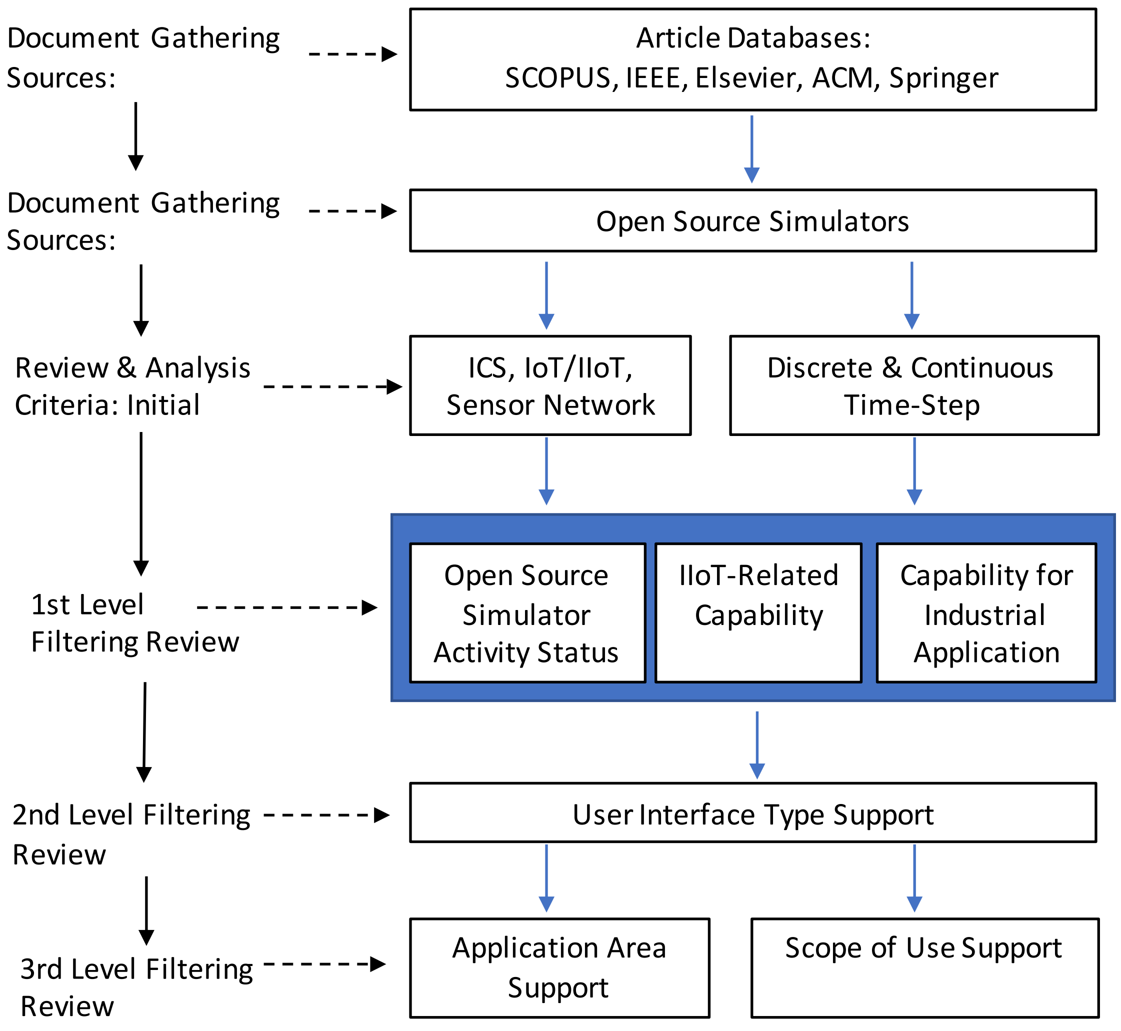
Document Gathering Sources:
Article Databases: SCOPUS, IEEE, Elsevier, ACM, Springer
Document Gathering Sources:
Open Source Simulators
Review & Analysis Criteria: Initial
ICS, IoT/IIoT, Sensor Network
Discrete & Continuous Time-Step
1st Level Filtering Review
Open Source Simulator Activity Status
IIoT-Related Capability
Capability for Industrial Application
2nd Level Filtering Review
User Interface Type Support
3rd Level Filtering Review
Application Area Support
Scope of Use Support

*Figure 1: Review Approach*

## 2.1 Document Gathering

To gain the relevant insights for addressing the research questions defined in the previous section, appropriate contextual boundaries were delineated for the review activities to ensure that the scope of study and results specification was maintained. In line with this, the following review constrictions were adopted:

1. Only Open Source simulation systems that are distributed with a type of free or Open Source license were examined. These include simulation tools and software whose source codes have been made freely and openly available for use and modification by the project developers.
2. Open Source project and simulation tools in areas related to industrial/productions networks including industrial control systems (ICS), supervisory control and data acquisition (SCADA), distributed control systems (DCS), process control systems (PCS), communication and sensor networks, and IoT simulators and projects were considered. This is to help capture as much as possible the entire spectrum of industrial internet-of-things (IIoT).

3. Open Source simulators that cover batch processes, continuous processes, or both, were included since these exemplified varied modes of industrial processes implementable in ICS and IIoT environments.

An initial search and aggregation of simulation tools was undertaken from academic literature. The keywords involved Boolean combinations of ‘Open Source’ OR ‘Open-Source’ AND ‘ICS Simulators’ OR ‘Network Simulators’ OR ‘Simulator for Industrial Control’ OR ‘Simulators for Industrial Internet of Things’. Key academic databases such as SCOPUS, IEEE Xplore, Elsevier, Springer, and ACM DL, were used to search for related articles. Searches were also performed in popular Open Source projects and software repositories such as Source-Forge (http://sourceforge.net/), GitHub Open Source portal (https://github.com/open-source). In addition, similar searches using the search phrases listed above were performed on Google with the aim of broadening the view and outcomes of the study beyond academic domain. This was to avoid omitting some new and emerging simulators that may be useful to the industrial community but have not enjoyed formal documentation in research and publications. This study is also intended to bring to academic light (where necessary) new Open Source simulation tools, systems, and platforms with the capability to support research and developments in IIoT with perspectives on security.

## 2.2 Review Criteria for Open Source (OS) Simulators

Several academic papers have discussed the topic of software selection and review [11]–[16]. Specifically, the selection of simulation software (simulators) is an area that has witnessed considerable research attention from several researchers. Key research drivers in this direction have been to obtain knowledge that can guide informed-selection or adoption of suitable system and network simulators for certain use case application areas. These works have provided a compact, generalised notional basis for establishing this study with insights into the key requirements chiefly considered in simulators that often inform their selection and adoption for system replication. For example, one key factor refers to Fidelity, i.e., the degree to which a simulator can capture or re(present) real system scenarios [16]. This of course can qualify an aspect of suitability or fitness for use. In the context of IIoT, a couple of prior works have discussed similar and related contexts, such as review of Open Source simulators for wireless sensor networks [17], [18], Open Source simulators for ICS/SCADA-related systems [19], [20]. However, we were unable to find any papers that have reviewed Open Source simulators in relation to the extensions to IIoT systems and applications, and with special perspectives on security evaluations. Example studies cited above mostly focus on performance for specific sub-system applications such as wireless sensors and traditional network communications on industrial or enterprise domains, rather than the larger IIoT.

Typically, OS has certain peculiar characteristics that make some of the criteria outlined in such existing works inapplicable. For example, simulator cost is discussed in [21]. Not repudiating the existence of other significant Open Source simulator characteristics, simulator price is also important; this is not necessarily visible in the COTS domains [21]. Accordingly, an example is discussed which involves not questioning the choice of methodology for documenting or communicating project updates and utilisation, even though both characteristics are significant for Open Source projects and typically form licence conditions that reflect either the origin of the Open Source software or a specific view of what its Open Source nature is (what it means to be open). It is not usual to question how developers of a proprietary software project communicate even though it is an important attribute of an Open Source project.

A number of publications [22]–[24] describe essential characteristics for evaluating Open Source simulators. Others [18], [20], [25], [26] reviewed Open Source simulators from specific domains such as wireless sensor networks (WSNs), ICS, SCADA, Industrial Operations, etc., based on self-selected evaluation attributes and characteristics. Even though these reviews do not directly refer to IIoT specifically, the focus often represents a facet or sub-system of what is considered as the larger IIoT, thus, providing useful knowledge and insights about sound Open Source simulator development. This work has been used to provide inspiration and guidance for the criteria for review and evaluation that characterize our study.

To manage complexity, IIoT is typically viewed/constituted as a system of systems (SoSs). Each subsystem may involve the use of numerous sets of different Open Source simulators. The chances are that this may result in a large number of simulators when fused together, introducing a complexity that may further affect the practicality and depth of achieving acceptable fidelity. Should the subsystem simulation system be developed by different parties, there is hardly any guarantee that the same simulator system would be used to capture all parts of a system scenario. Of course, this will lead to an explosion in the number of fused simulators, and even more; questionable outcomes that result from differing assumptions built into supposedly similar functional units in different subsystems, or the interactions amongst them.

### 2.3 Open Source Simulator Evaluation Criteria

We adopt a method of defining common dimensions and factors gathered from prior works and literature to help manage the volume of contents, analysis, and results. In [27] some factors have been suggested as contributory to a good open-source software project. These include: development status, distribution mechanisms, version control, communication channels, developer guidelines, documentation, open source licence type, and code/commit reviews. Some of these factors have also been used by Dagkakis & Heavey [21] to evaluate open-source discrete event simulation software for operations research. The above works provide inspiration and basis for the relevant criteria/factors adopted in this study. These include: *Language, Open-source Licence Type, Simulation Technique/Mode, Documentation, Communications, Version Control, Development Status*. In addition, other factors were included that related to the key aspects of interest in this research. For example: the interest in learning the trend surrounding the use reviewed open source tools led to the inclusion of '*Purpose*'**.** The interest to learn about the operating system compatibility and user-centric considerations of the open source projects brought about the inclusion of *'Platform'* and *'User Interface'* as factors for consideration. The interest to learn the degree of actual representativeness teach open source tool provides led to including *'Level of Realism'*. Needing to learn how common each open-source simulator/project is in knowledge and use by the simulation research/application communities led to including *'Usage Citation Count'* as a potential criterion. Lastly, security is a primary focus and interest in this study of open-source simulators. More specifically; to learn which of the existing open source tools are characterised to handle security-related simulation scenarios, as well as understand the specific security contexts covered. This led to introducing the *'Security-oriented Application'* criterion. It is believed that the information these new factors will provide when combined with existing factors identified, can enable even newer insights that can support effective decision-making when it comes to the evaluation and selection of open-source simulators, or considering attributes of focus for future open-source simulator developments.

Eventually, the number of criteria adopted in this study for evaluating open-source simulation tools in the ICS/IIoT environment increased. The combined criteria include: *Purpose, Language, Platform, Open-source Licence, User Interface, Usage Citation Count, Simulation Modes, Documentation, Communications, Version Control, Level of Realism, Development Status, Security-oriented application*. We note that the outcomes of initial search represent only a sample of the true state of affairs and that, in particular, academic simulators often have useful lifetimes that correlate strongly with the support provided by a time-limited funding stream. For all but a few simulators that are so widely used and supported that they have achieved a critical mass of users, open-source simulators tend to have a lifecycle and it is therefore important to identify those that are emerging as well as those that are established.

## 2.4 Open Source Simulator Review Guidelines

In this section, the results of the review are presented based on the dimensions and characteristics of simulation tools described in the preceding section. However, it is worth emphasising our initial forecast of the likelihood of a huge number of results from an initial search for simulators with the inclusion of multiple related system aspects and applications - ICS/SCADA, WSNs, and IoT/IIoT. As anticipated, a significant number of results were obtained, and this introduced a significant review challenge: directly applying all-characteristic evaluations to the aggregated set of Open Source simulators at once, and then presenting the results in a way that is both comprehensible and that illuminates underlying similarities and differences. Consequently, we adopted a multi-level evaluation approach. This involved splitting the set of characteristics into groups, and applying incremental evaluations using successive groups of characteristics. Open Source simulators are then eliminated from the list if they failed to satisfy any of the criteria within each group of characteristics. This way, a multi-level filtering of simulators can be achieved while being guided by the objectives for performing the review and application context. A similar approach was adopted in [21] in reviewing Open Source discrete event simulation software for operations research. This enabled the researchers achieve a review that was focused on the more promising DES simulation software. The full results of both initial and subsequent levels of the review are presented in the Appendix A (Due to space constraints, results are broken into tables).

Sub-section 4.1 provides the initial generalised considerations outlining the statistical dispositions of the results. This covers the multi-level filtering of simulators based on certain grouped criteria. The objective of this is to provide an initial generalised overview of up-to-date in Open Source simulators, which enable some important features or capabilities of the IIoT. This is followed by successive phases and levels of reviews, subsequently yielding new subsets of Open Source simulator outcomes, which are derived from filtering out non-compliant simulators based on the related-level group of criteria.
The initial set of simulators herein referred to as the ‘complete set’ were aggregated along with their corresponding attributes and in relation to the review dimensions described in Appendix A. The ‘First-level filtering’ was then performed using three criteria:

*(i.* *Open Source simulators (projects) that were considered Non-active or Inactive*
*(ii.* *Open Source simulators (projects) that were considered too narrow or outside the immediate scope of IIoT*
*(iii.* *Open Source simulators (projects) that lacked the capacity for extension to real/industrial application.*

This filtering addressed the above-mentioned characteristics of '*purpose*' and '*development status*' and yielded a narrowed subset of Open Source simulators.

The 'Second-level filtering' was performed on the narrowed subset with respect to 'user interface support', eliminating simulators that had no form of graphical support. As expected, an even narrower subset of Open Source simulators resulted.

We adopted this multi-level approach to limit consideration to the more promising Open Source simulators for ICS and IIoT security. For example, simulators may provide documentation, but their documentation formats, multiplicity-support, and update protocols may differ. Communications support amongst simulation tools serving similar purposes may also vary in channel modes and traffic frequency. Equally, some tools may well lag behind in capability updates provided by developers but may have a more active user community than more recently updated simulation tools. Thus, we also provide a purpose-driven group-level comparative review of the narrowest subset of results. This can provide a deeper sense of advantage or comparative strength of some simulators over others within the same purpose group.

## 3. Results and Analysis

The summed result of the Open Source simulators reviewed as presented in the appendices provides concise details about each simulator in respect of the characteristics defined. From the results, some valuable generalised inferences can be drawn about the state-of-the-art in ICS/WSN/IIoT Open Source simulations.

### 3.1 Programming Languages

From Figure 2 C++ and Java appear to be the most common computer programming languages supported and(or) adopted in most simulator development projects. This result is consistent with the findings in [21]. As observed, the Figure 2 referenced indicates that C++ is supported by 27 simulators, while Java is supported by 22. The two languages sum up to 68.1% of the aggregated programming language supported by all the reviewed Open Source simulators. Some of the Open Source simulators reviewed indicated to support multiple languages, for example, Hase [28] provides implementations for C++ and Java, as is OOSimL [29], whilst SimGrid [30] provides support for C, Java, and Ruby.

The preferences for C++ and Java by Open Source simulator researchers and developers may be linked or influenced by the speed of performance of the two programming languages. The authors in [21] thought the same. In comparing the execution speed of programming languages, a study [31] showed that C, C++, and Java were faster-executing languages than others. The research further clarified that C++ is especially faster than Java. Nanz et al [32] acknowledge the speed superiority of C over other languages in their study. However, C presents a significant challenge in that it strictly provides a command-line interfaces (CLI) which is generally viewed as non-user-friendly, hence more difficult to learn or use [33]. This is an issue in the industrial environment where stakeholders might not necessary be expert programmers, nor be open to the rigours of learning and interpreting code instructions. This makes C quite weak, and may also explain the reason for its minimal adoption in software developments in general, and especially simulation projects, and the preferred adoption of the simplified, more intuitive, GUI-oriented C++. C#

and Python are supported by 4 and 6 simulators respectively, Visual Basic .NET, Scala, JavaScript, Ruby, FORTRAN, Objective-c, and Modelica all have one simulator application each.

## 3.2 Operating Systems

The statistical distribution regarding operating system platforms supported is presented in Figure 3. Most of the OS simulation tools are developed for Linux platforms. Indeed, 52 out of the 60 Open Source simulators support execution on the Linux platform. There is a link between the choice of programming language for Open Source simulators and the Linux platform. Linux is developed based on C and C++ programming language [34], and is made available free for download on an Open Source licence. We assume that the common language of development can foster easy and faster interaction between simulator and platform. Moreover, simulation systems often involve complex and multi-tasking functions. Linux seem to provide this well, and is a more flexible platform for wider infrastructure and applications compatibility, also providing better capability for run-time error management. From the perspective of developers, Linux has been judged above other platforms in providing greater convenience, capability, security, interface, and recovery [35]. Developers view Linux to be friendlier [36] and not requiring updated hardware resources – it can run or interact with older infrastructures without significant lags. It is a better lightweight system which is more robust in terms of crashes, and has better capabilities for security [37]. Above all, it inexpensive to own and use, with free regular updates; features that come without expense unlike other platforms like Windows and Mac OS. These latter platforms were used by 35 and 18 Open Source simulators respectively. UNIX is used by 3 projects, FreeBSD, Solaris, and Android each host 2 projects, while POSIX is the OS for one simulator project.

## 3.3 Licence Types

From the OS Licence criteria results presented in Figure 4, the GPL (in 15) - and LGPL (in 11) Open Source licence types seem to be mostly adopted in majority of the simulators, however, dissimilar versions of each of these two licences type seem to be adopted. Both licence types provide a good way for freely accessing and using Open Source simulators. However, these licences enforce restrictions that deter modification and re-distribution of Open Source software in a proprietary manner. This raises significant concerns that may discourage industrial sectors from adopting them [21]. It suffices to say that most applications of Open Source simulators may well be tailored towards research and development studies, and few to real infrastructure applications.

Some of the simulators offer support for multiple licences. For example, OpenModelica is licenced under the GPL v3 and OSMC-PL which is more of a customised license - allowing for proprietary extensions to be licensed under different conditions to the core OpenModelica code. This type of licence scenario (multiple coverage) occurs when a simulator is covered by a licence different from its supporting operating system. Most often, the platform licence would have to be satisfied as well. BSD and APL had 5 simulators, AL had 4, MIT had 2, ASL and CCA had 1 simulator each. 6 simulators had custom licences specific to them and described in their licence files. As noted in [38], a key challenge for developers would be to easily and aptly identify if such customised licences are indeed GPL-compatible, to allow for free integration with GPL applications. We were unable to determine the defined open licence types for seven other simulators such as: EMSO, Coco, and VTB. The assumption is that these might have some type of Open Source licence definitions that are not easily available to developers. Again, this is not good for an

Open Source project or simulator. It is expected that interested developers will be able to obtain without difficulty, relevant details for Open Source projects for the purpose of driving improvements [21].

## 3.4 Simulation Modes

Based on the results from Figure 5 for simulation mode analysis, 34 out of the 60 OS simulation tools supported only discrete time-step event simulation (DES) processing modes. This number covers most of the tools reviewed. However, we note that 4 other tools supported Parallel DES; allowing for multiple DES processes to be executed at the same time. Some of these include: YetiSim [39], GloMoSim [40], and PARSEC [41]. 5 simulators cover both DES and CS (Continuous Simulations), for example Coco [42], DWSim [43], and OpenModelica [44] supported both DES and CS. Other uncommon simulation modes captured by simulators include TDS (Trace-driven simulation) in 1 simulator, ABS in 1 simulator, DS in 2 simulators, DEVS in 3 simulators, and 6 other simulators supporting multiple simulation modes.

In the industrial domain, discrete time-step event processes also characterise batch process flows that typically involve processing bulk products or services in groups through each step of a desired process. Subsequent processes must wait until current batch is completed [45]. This is the situation in a majority of current non-continuous industrial processes. This is different from the emerging continuous time-step event simulations where processes are modelled continuously – no waits involved. It is good for a tool to adopt a generic or multiple-process mode capability to allow it to be tendered for different application contexts. However, we note a contrasting argument in favour of domain-specific DES environments which are said to facilitate easier model development. A trade-off seems to loom and open for debate between the depth of domain-specific application and the width of a generalised application area for industrial environment simulators.

## 3.5 Documentation and Communication

Based on the documentation and communication modes characteristics of reviewed Open Source simulators, some useful results were obtained and presented in Figure 6.

For communication, the results showed that most (52) of the Open Source simulation projects adopted Internet Relay Chat (IRC) as medium for the Open Source community interactions, support and interventions. 35 employed Mailing lists, 18 used websites, while Forum and Wiki is adopted by 2 projects each. It is perhaps surprising that some (3) projects did not have any dedicated means for communications. The assumption is that such projects are either inactive based on updates and have very low community interest and usage. Communication features are valuable in supporting Open Source simulator project developers to promote trust and confidence on the part of users, as well as monitoring their use of the simulator. We concur with earlier depositions [12] that the continuous availability and release of project updates, version improvements and modifications, provision of support to functional issues, and maintenance information are key services that should come packed with Open Source simulation projects.

For documentation, 25 projects provide project documentation records in HTML formats, 18 use manuals in varied formats, e.g. DOC, PDFs, and PPTs for documenting their development and usability guides, 8 projects use other unconventional but interactive forms such as videos and technical course, while 5 projects use APIs. What is worrisome is the number of projects (19) for which we were unable to identify any dedicated form of documentation. Good documentary support is arguably a key feature promoting adoption and use of Open Source simulators. Documentation that provides guides to installation, use and maintenance, can help free users from complete dependence on developers to answer or resolve even the most minor errors. This in turn reduces the support pressure on the project developers [12]. The availability of helpful documentation in forms of user manuals and tutorials can greatly help users learn how to correctly use or apply the features of Open Source simulators and adopt updates. This can significantly save time spent on troubleshooting and error management.

Generally, we observe a lack adoption of some of the relevant OS project features that can boost the credibility of Open Source simulators. Features like easy and Open Source project documentation and communications. We assume this is a potential reason that makes some of the projects quite unattractive to the OS community.

### 3.6 First Level Filtering Review (FL-1)

This first level filtering activity was aimed at discarding Open Source simulators that matched the three important criteria:

(i. *Open Source simulators where their development status indicated them to be inactive in terms of current support.*

(ii. *Open Source simulators where their functionality scopes were in domains not related to ICS, WSNs, or IIoT.*

(iii. *Open Source simulators where their application did not include practical real-world situations, including Open Source simulators that were solely designated for educational training and practice purposes.*

Out of the initial 'complete simulator' set of 60 projects, 35 simulators were discarded for falling into one or more of the three listed filtering criteria. Generally, many of the simulators discarded seemed to be inactive – without any active OS community activities or discussions, not getting updates in the past 2 years, or lacked any clear information from developers about the active status of the project, especially relating to still being under development. For example, JARPROSIM [46], [47] had its most recent version update in 2014. We checked the log of the commit history of JARPROSIM where available to determine the existence of a more recent update on the code base. There was none. Again, Avrora [48] had its latest version series – Avrora 1.7.X released in 2008. A last modified version of the series was done in 2013. On the other hand, GeneSim [49] was last updated in 2012, but the project website indicated that the project was still under continuous development.

Other simulator projects were also discarded because they were not within the scope of the context studied. For example, Hase [28] was developed to strictly simulate computer architectures. We assume this to not capture the scope simulation scope of IIoT since ICS/SCADA, WSNs, and IIoT domain components were beyond computer architectures alone. There are communication gateways, Control automation components, sensors, actuators, etc., that need to be simulated too. TerraME [50] is another simulator that was discarded because it was purposed on simulating Terrestrial systems, which evidently is outside the scope of evaluation. JSL is an example of a simulator that was discarded because its use is constrained for educational purpose only. From the project website, it was discovered that OOSimL [29] was developed "to support simulation education to students of computing" (http://ksuweb.kennesaw.edu/~jgarrido/oosiml.html), this was also discarded due to constraints in the licence description. After the first level filtering, a total of 25 simulators were left, and constituted the first-level subset.

**3.7 Second Level Filtering (FL-2)**

The second level filtering was performed on the result from the FL-1 subset which included 25 Open Source simulators. This was aimed at discarding simulators that may be difficult to use by a (wider) generalised community of users due to a lack of user-friendly interfaces. Thus, the important criterion was:

*(i. Open Source simulators that lacked user-friendly operating interface(s)*

Although the debate on user-friendly usage preferences between Graphical User Interfaces (GUI) and Command-Line Interfaces (CLI) is still open in the computing community, at the advanced (expert) developer levels; there is wider preference, support and adoption for CLI than GUI in view of faster execution, easy and more consistent portability of instructions [51]. Another study [52] finds expert users indicating that CLI provides compact, quicker and fewer ways of executing a task associated with a good understanding of deeper instruction interactions. This may relate to the assumption for the want of complete control over application functionalities and behaviour, a capacity that is difficult to achieve in similar scale for GUI compared to CLI.

However, the same is not the case for non-expert users (novices) who seem to get more done and better using GUI [51]. This disposition only reaffirms findings from prior works [53] that explored the differences between Mac GUI and CLI for a file directory and structure utility tasks performance. The outcome indicated a significant upscale in the capability of non-expert users to learn and perform with the GUI. Other prior researches have also slightly favoured GUI over CLI even for expert users. Rauterberg [54] reported that users were generally faster with GUI than with CLI. According to their study, experts required 51% less time to complete tasks. Another work [55] compared spreadsheet and word-processing task activities using a GUI and CLI, and found that users had less frustration and fatigue with GUI than with CLI, and completed 35% more tasks, with 17% more accuracy.

It is important to grasp these varied viewpoints in order to appreciate the necessary trade-offs and balance between GUI and CLI when implementing user interfaces for Open Source simulators. In the context of operational industrial control systems, GUIs are typically the norm. This is especially significant since ICS and now IIoT domains would not only consist of expert system programmers, but also include operations users and analysts who may not necessarily retain the expert skills of coding or interpreting instructions needed for a CLI. Thus, with respect to simulation, we back the argument that whilst expert system programmers may prefer CLI-based systems, the need to consider operational experience to enhance the credibility of results would suggest that the GUI support is also important.

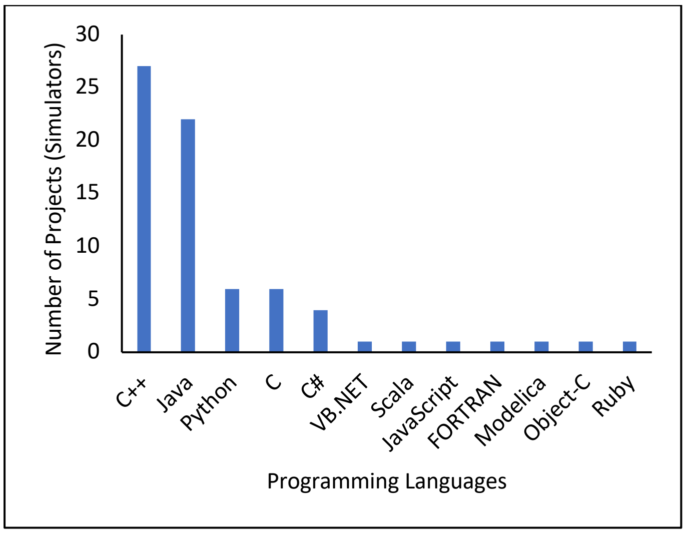

*Figure 2: Programming Languages supported by Open Source simulators*

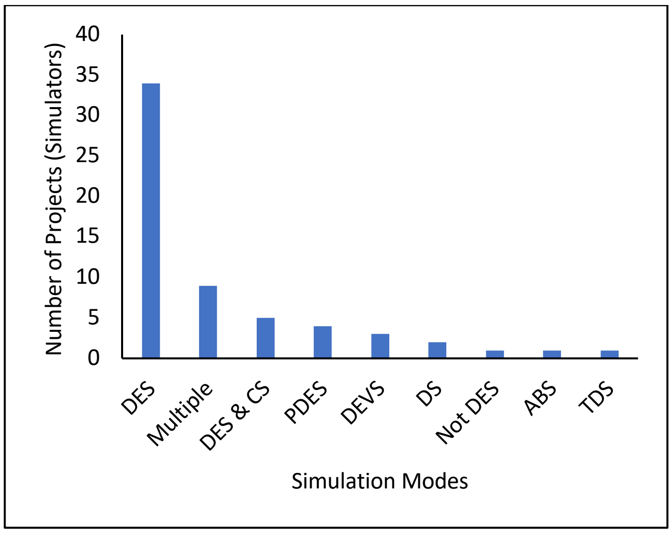

*Figure 5:Simulation Modes supported by Open Source simulator*

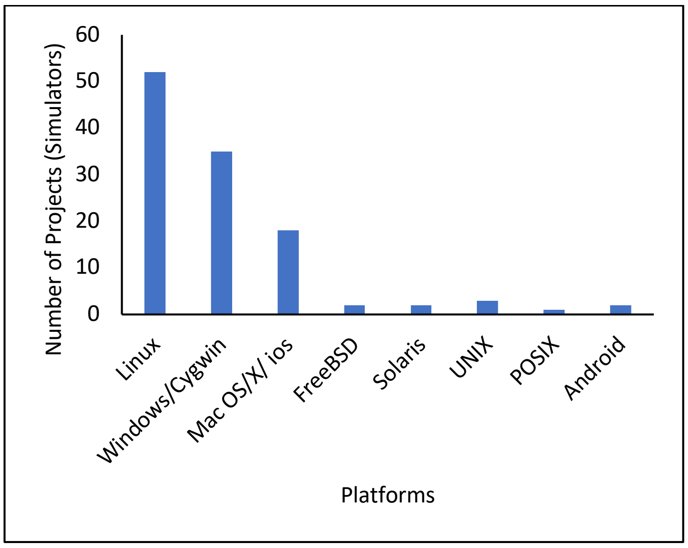

*Figure 3: Operating System Platforms supported by Open Source simulator*

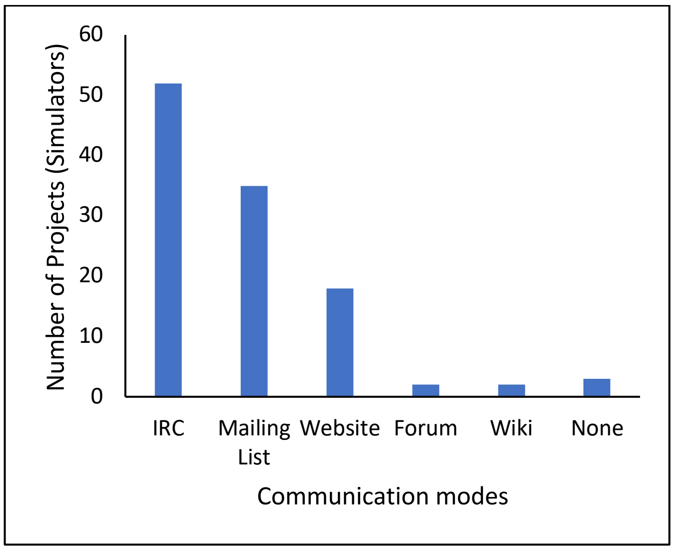

*Figure 6: Communication Modes supported by Open Source simulator*

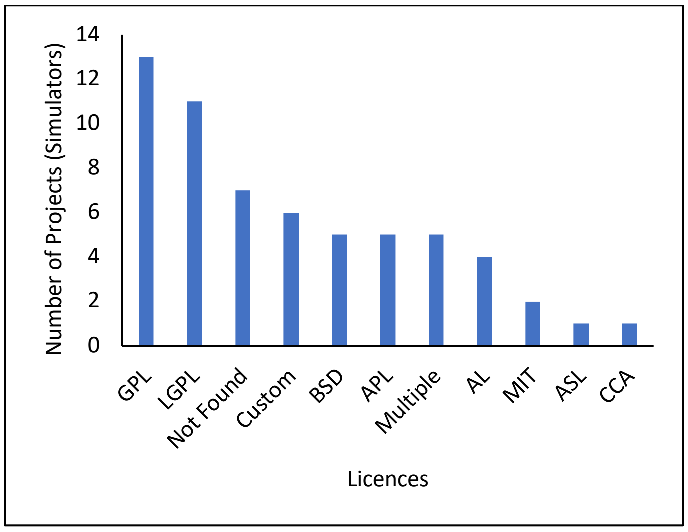

*Figure 4 Licence Types supported by Open Source simulator*

Furthermore, the complexities within the ICS and IIoT domains call for simulators with more intuitive, user-friendly interfaces so they are better suited to accommodate easy understanding of the interplay amongst system components and their interactions. Thus, the second-level filtering was done from the perspective of ease of use. Simulators with only CLI are less likely to provide good intuitiveness, easy comprehension, less stress, and visualisation capabilities, as such are less likely to attract interest, especially from the wider group of non-expert users. These advantages put simulators with GUI ahead. However, simulators that support both interface features provide even greater advantages, retaining a combined strength of the two interfaces in one system. We assume that such would be more likely to attract a wider population of users spanning both experts and non-experts since the preferences of both are supported. On this basis, we filtered out simulators that strictly supported CLI modes with no consideration for GUI.

Figure 7 presents the results of the interface classification. Out of the 25 Open Source simulators in the FL-1 subset, four were discarded because they supported only CLI user interfaces. These include; Simpy [56], DEV-C++ [57], ScipySim [58], and TOSSIM [59]. An interesting observation is that these simulators were either developed or compatible with C++ and Python programming languages, which both have GUI capabilities, yet this was not included in the design architectures. We assume that the purpose and scope of application of the simulators (mostly generic) may have influenced their limitations to CLI user interfaces. The FL-2 yielded 21 simulators that either supported GUI solely or combined GUI and CLI interface supports. We were unable to identify clear information about the interface support for SimGrid [30], however, this simulator was not discarded because of considerations for some of its other attributes which may give ideas on the likely interface type. For example, SimGrid claims to support multiple programming languages - Java, C, and Ruby. Its application area includes simulations for distributed systems (Cloud, Grid, Fog, etc). Thus, it is likely that GUI support is available for this simulator since most software and simulators with similar properties seem to provide GUI support.

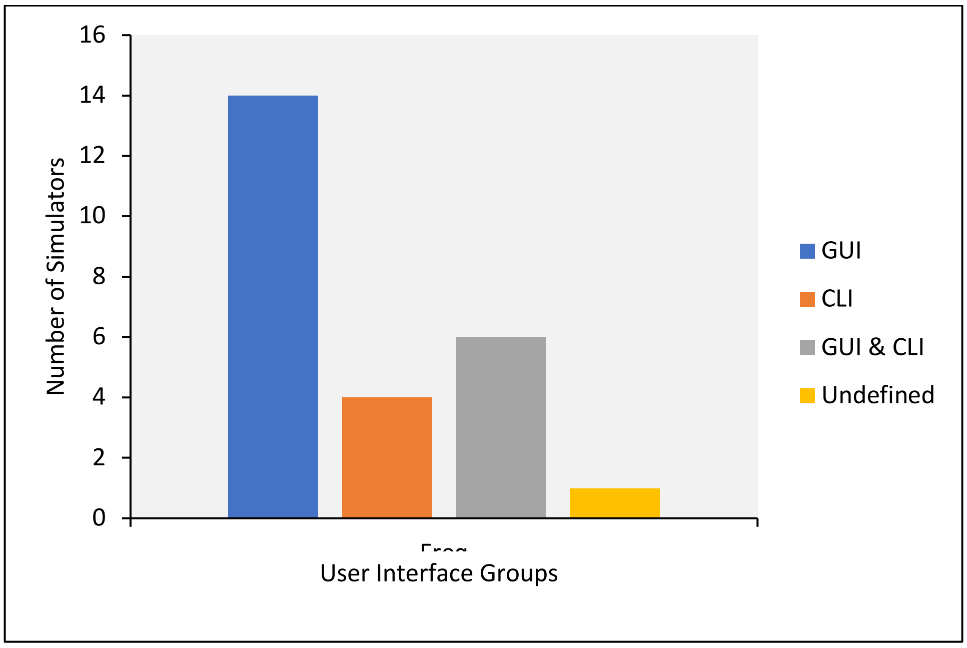

*Figure 7::User Interface supported by Open Source simulator*

The point being that CLI cannot be totally done away with as it constitutes the foundation and building block of every computing and application platform. CLI also plays very significant roles in the development of GUIs. However, integrating GUIs introduces a significant capacity for shortcuts, such as using a click to activate actions that would ordinarily be achieved using multiple lines of written codes on a CLI. GUIs also bring in the benefits of visualising results and logs, simplifying usability, and improving understanding and interpretation. These are quite significant in the evolving domain of big data-driven IIoT.

### 3.8 Third Level Filtering and Evaluation (FL-3)

The remaining 21 Open Source simulators were further analysed based on:

*(i.* *Application area groups: Communications and wireless sensor networks, Operations, ICS/SCADA, IoT/IIoT, and Multiple*

*(ii.* *Within each group, their degree of fidelity, scope of usage, the kind of researches built around them, their strengths, limitations, restrictions, and their specific use for security analysis studies.*

Using these, we aim to evaluate the capability of each Open Source simulation tool, determine the capability gaps, and identify areas for potential improvements that can incorporate security analysis. We also consider the potential customisability of Open Source simulators for IIoT applications that focus on security.

### 3.9 Scope of Adoption and Application

To assess how widely used the selected simulators are, and to understand the kind of research that has built on the simulator and (or) their applications, a systematic literature review approach [60] was adopted. To achieve this, the period of 2010-2018 was adopted to obtain key research results that focus on individual simulator workplaces. In [61], this is termed 'research fronts'; referring to clusters of papers that share a common intellectual base. In this review, the common intellectual base refers to research and intellectual outputs (peer-reviewed) related to the selected simulation tools. Accordingly, Liang et al [61] identified two key metric indicators for research fronts: (i) Usage Count, (ii) Times Cited, with specific application to the Web of Science (WoS) database of Articles. The latter metric (Times Cited) is the more common metric adopted for analysing citation fronts, however, the authors in [61] identified limitations of Times Cited – mainly related to elongated time lag in seeing the effects of research fronts, and an inability to reflect current interests of the research community. Thus, they investigated the potential for adopting 'Usage Count' and made a preliminary case for its adoption as a better alternative or complement for research fronts analysis.

For this review, the 'Times cited' indicator is used as the criterion for evaluating the top Ten (10) articles from the keyword search results on Web of Science (WoS) database for each of the selected simulators. The keyword format used include: "Simulation" AND "Simulator-Name" where simulator-name was changed for actual simulators e.g NS-2 for each of the analysis. As a specific example, in the case of NS-2, the keyword "simulation" AND "NS-2" was used, and within the earlier defined time range. The results/documents were restricted to peer-reviewed articles (journals, conferences, and books). The summary results obtained is shown in Appendix B.

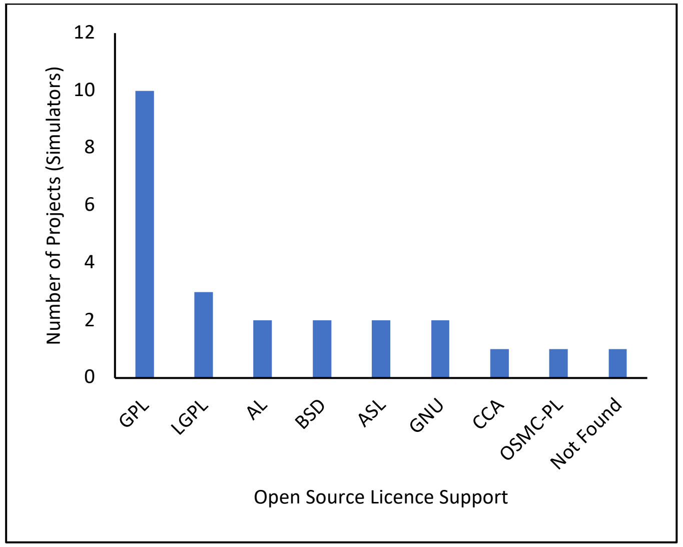

*Figure 8: Licence Types*

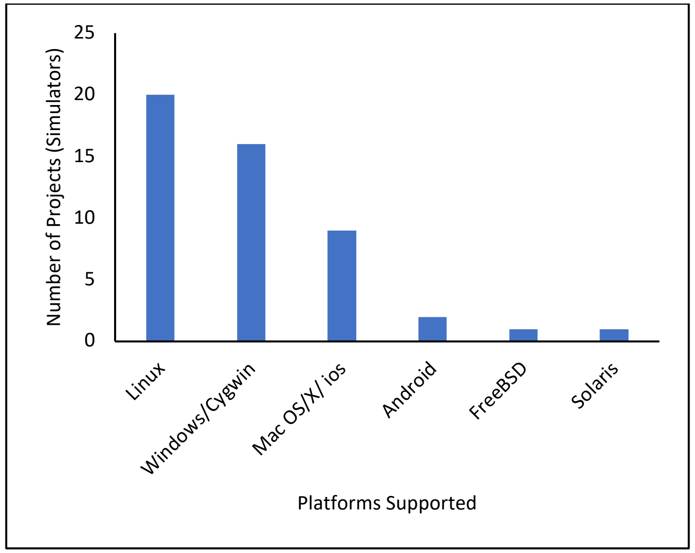

*Figure 9: Platform Types*

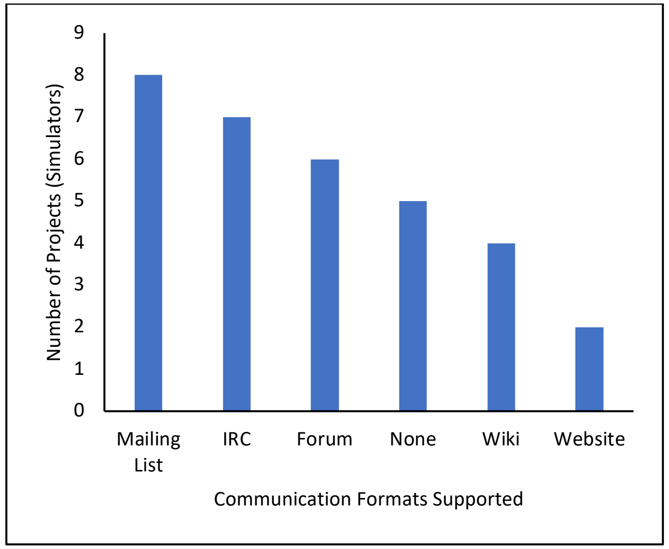

*Figure 11: Communication Types*

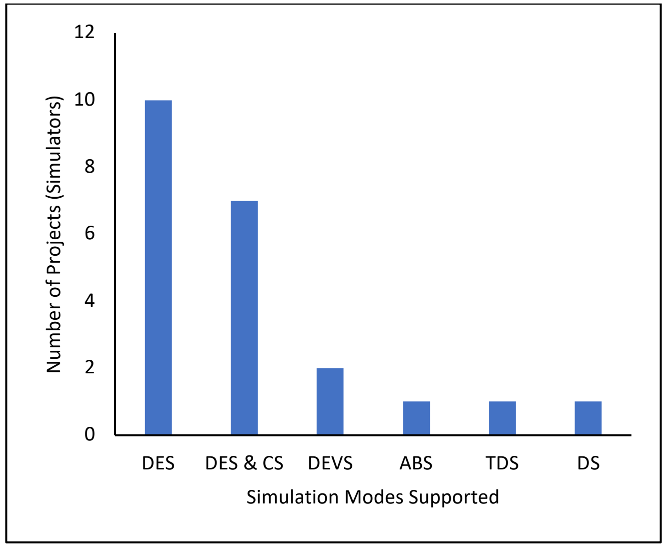

*Figure 10: Simulation Modes*

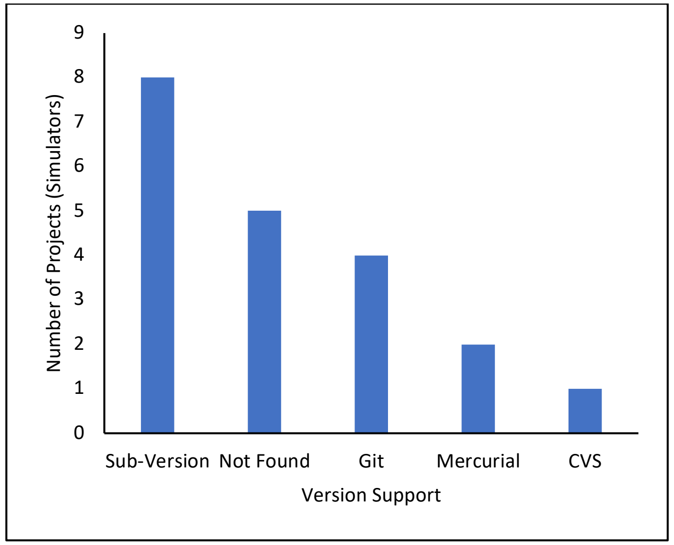

*Figure 12: Version Support Types*

From Figure 13 certain insights can be drawn from the analysis of the refined and filtered subsets of the 21 simulators. The results of the broad classification showed that many (7 out of 21) of the simulators in the new subset of Open Source simulators claimed to possess generic architectural capacities. These include: JaamSim, DESMO-J, PowerDEVS, JavaSim, CD++, RePast, and PySimulator. Four simulators: OMNET++, NS-2, NS-3, and OverSim were classed under communications/wireless sensor networks. Three were classed as suitable for more generalised operations that did not encompass direct physical hardware infrastructure. Three each of the Open Source simulators were classed under more specific areas such as ICS/SCADA (OpenModelica, SCADASim, RapidSCADA) and IoT/IIoT, while one simulator (SimGrid) was claimed to cover multiple architectures. The scope involved was not clearly outlined in any of the project demonstrations. Further research on the use of the simulator only indicated application areas in parallel and distributed computing system simulations and studies. It is observed that the results of this updated analysis aligned or showed similar patterns to the initial results of analysing the entire 60 OS simulation projects. For examples, C++ (10) and Java (8) still topped the list of programming languages supported by the subset of Open Source simulators as indicated in Figure 2. For the type of OS licences supported, GPL (10) and LGPB (3) also topped the list as seen in Figure 8. Linux (20) and Windows (16) were the operating platforms most commonly supported by the new subset as shown in Figure 9. The most widely adopted simulation mode is DES (discrete event simulation – Figure 10). Most (12) of the Open Source simulators were supported using online documentation. This is closely followed by manual-type documentation, supporting (10) simulators as seen in Figure 11.

Concerning user interfaces, fourteen of the simulators had only GUI’s, while six offered both GUI and CLI interfaces. We were unable to find clear descriptions of the user interface of one simulator, i.e., SimGrid. Version support which indicates how the Open Source simulation projects were being managed in terms of subsequent improvements and product updates, showed that eight of the simulators used ‘Sub-version’; four projects used Git, two used mercurial support for updating their projects, while one simulator used ‘CVS’ (Figure 12). We were unable to clearly determine the type for five simulator projects. The initial generalised results of analysing all (21) of the Open Source simulators based on ‘Time Cited’ (Figure 15) revealed quite enlightening insights on how popular this OS were amongst the academic and industrial community of Open Source users.

From a simulator-level analysis, NS-2 (683), OMNET++ (626), and NS-3 (338) research-related works and studies were the mostly cited documents within the review parameters described earlier (Figure 14 and Figure 15). In not exactly the same order of ranking, the same simulators NS-2, OMNET++, and NS-3 research-related works produced the most publications amongst all the simulators, also within the review parameters earlier described. Thus, from a generalised perspective, that these simulators had the widest interest and utilisation by the Open Source community of users. Based on ‘Time Cited’ parameter, it appears that NS-2 was more widely utilised than the other two simulators. In a similar pattern, the order of utilisation scale of the next ranked simulators based on the ‘Times Cited’ criteria indicated thus; OMNET++ - second, NS-3 – third, Repast – fourth, SimGrid – fifth, PowerDEVs – sixth, SCADASim –

seventh, Cooja – eighth, OpenModelica – ninth, and Manpy -tenth (Figure 14). The order can be followed from Figure 14 and Figure 15.

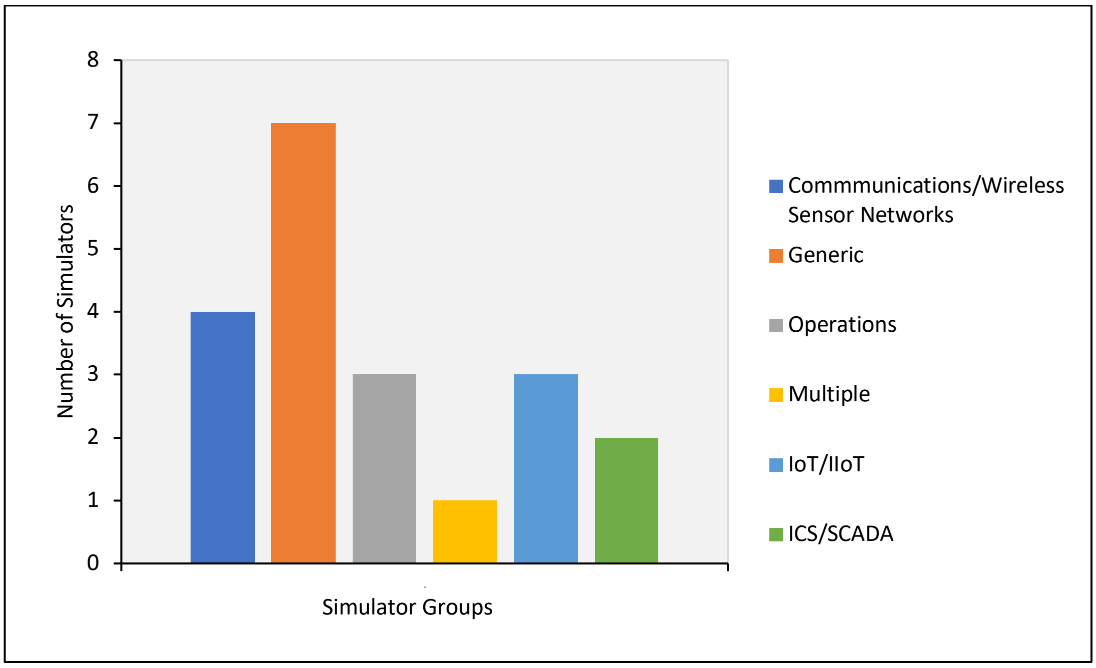

*Figure 13: Purpose-driven Groups*

From a group-level classification, we aimed to determine which of the simulators is more commonly used by the OS community for research and development. For the simulators within the communications and wireless sensor network group, NS-2 (683 times of cited articles) appear to be the most widely used simulator, followed by OMNET++ (627 times of cited articles), and then NS-3 (338 times of cited articles). The least used simulator seems to be OverSim. For Open Source simulators within the ICS/SCADA group, SCADASim (45 times cited articles) seems to be the most widely used (Figure 18). This is followed by OpenModelica (35). In the group of Open Source simulators for IoT/IIoT applications as indicated in Figure 17, Cooja appears to be the most commonly used tool at the moment. Proview and KAA do not appear to have any growing interest within the OS community currently. For the generic group of Open Source simulators, Repast appears to be the currently most commonly used tool, this is followed by PowerDEVs, and JaamSim and DESMOJ. For the operations group of simulators, ManPy is the most cited (Figure 16).

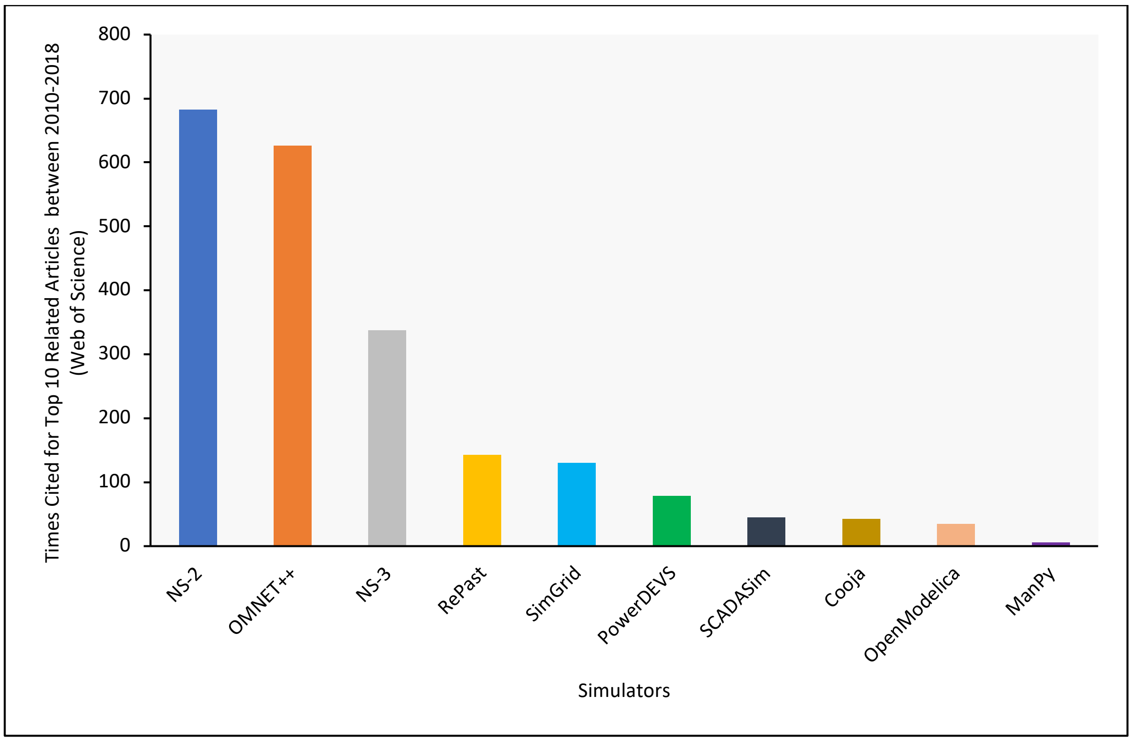

*Figure 14: First 10 most cited Open Source simulators*

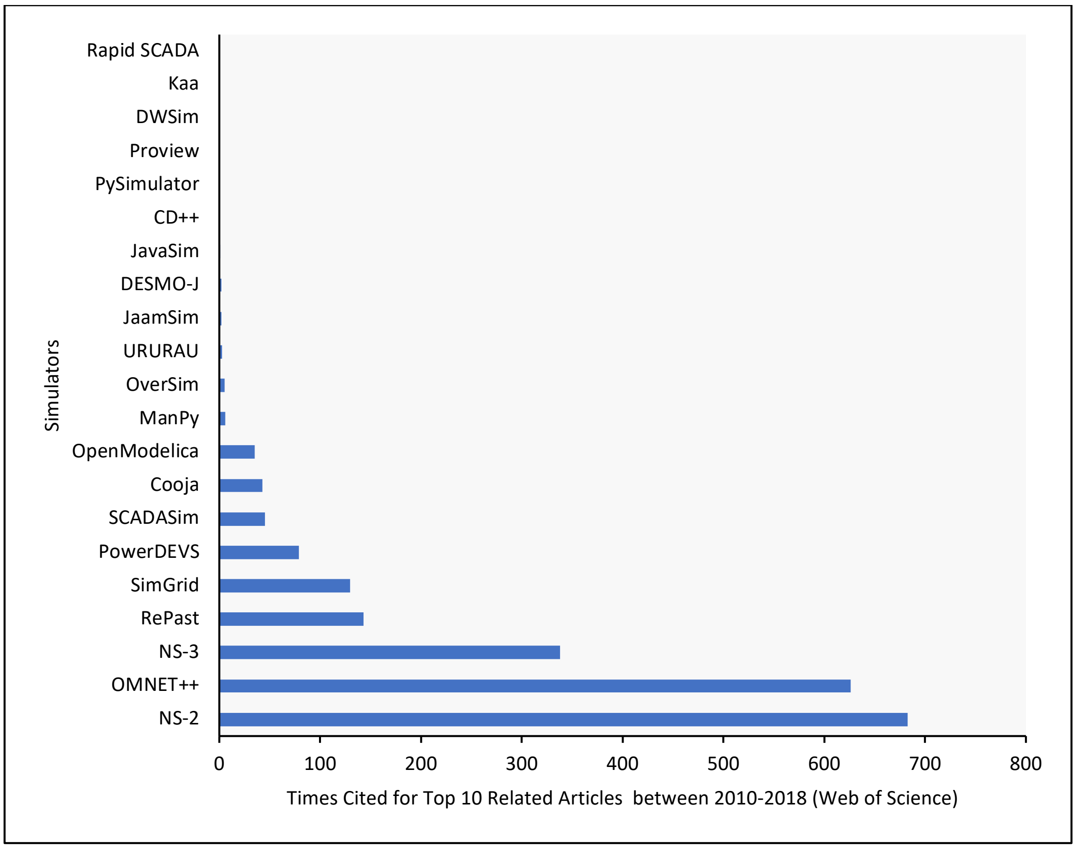

*Figure 15: Ranking based on most cited articles*

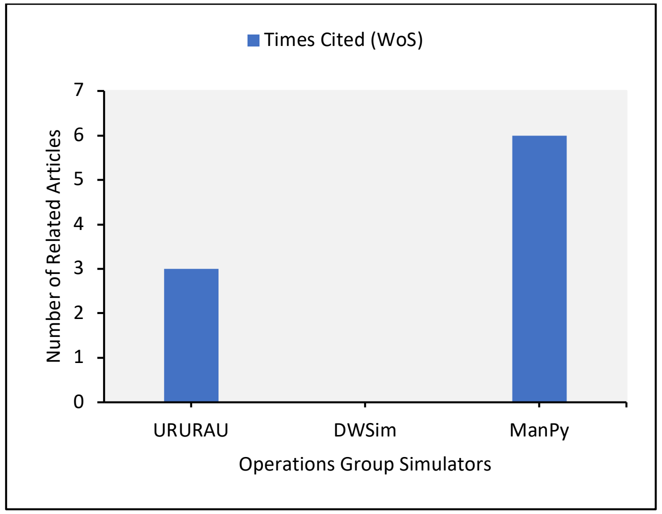

Figure 16:Operations Group of OS usability Ranking

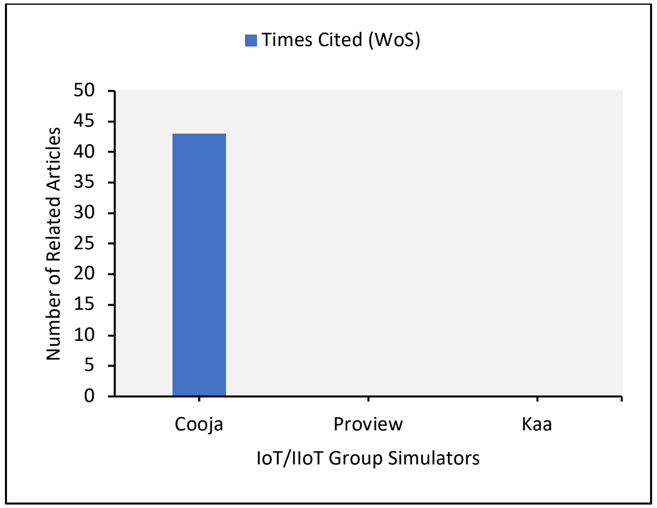

Figure 17:IoT/IIoT Group of OS usability Ranking

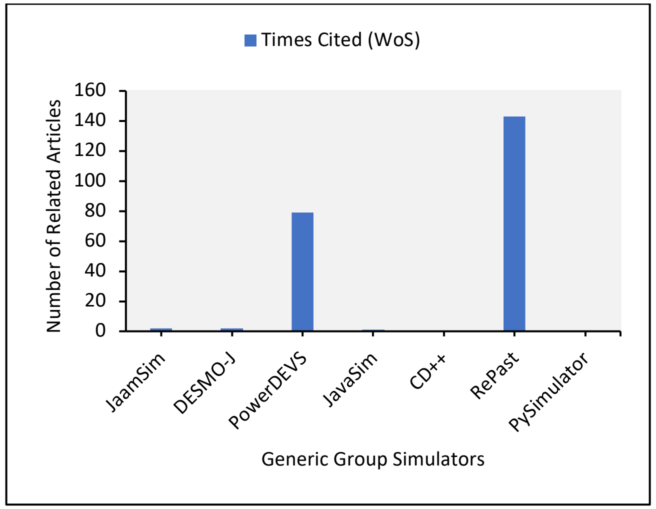

Figure 19:Generic Group of OS usability Ranking

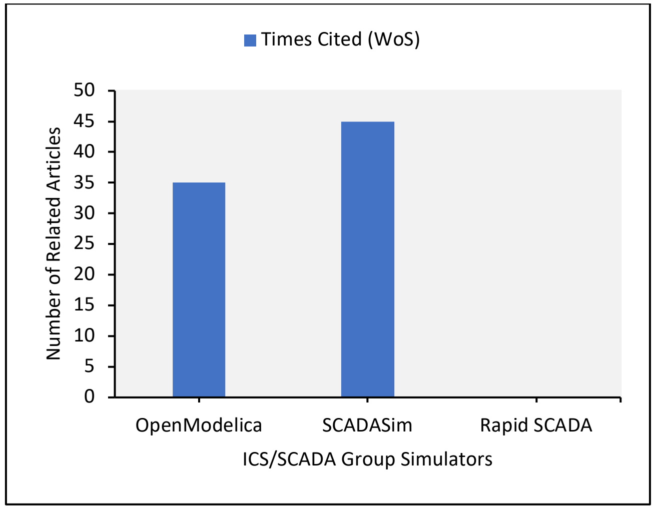

Figure 18 : ICS/SCADA Group of OS usability Ranking

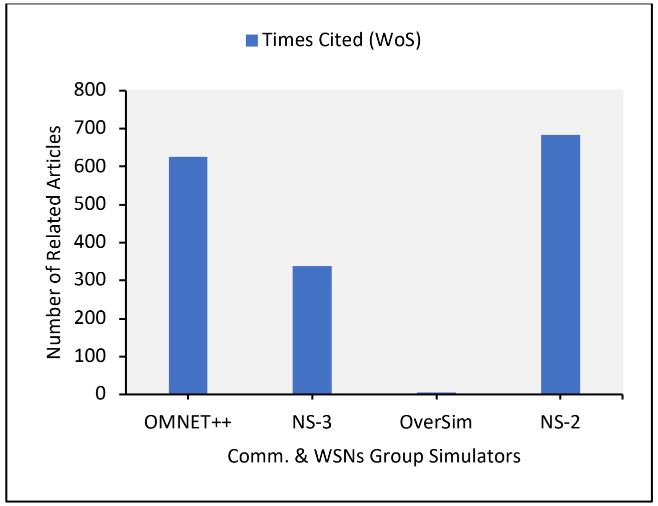

Figure 20: Comm & WSNs Group of OS usability Ranking

## 4. Open Source Simulators Overview

The Open Source simulators in the level 3 filtering subset are briefly described according to their grouped categories below.

### 4.1 Communications and Wireless Sensor Networks Simulators

In this group, 4 simulators emerged, these include: OMNET++, NS-2, NS-3, and OverSim. We briefly describe these simulators below:

#### 4.1.1 OMNET++

OMNeT++ is short for Objective Modular Network Tested in C++ [17]. It is a component-based, modular-capable discrete event simulator for both wired and wireless networks. It is mostly used for simulations for research and educational purposes as well as in the global scientific community. OMNeT++ is a general-purpose simulator with capability for replicating any component-based network interactions. It allows for free reusability of models, and supports extensive GUI features [21].

***Strengths:*** (i) Supports strong GUI features and environment. (ii) Provides a more simplified tracing and debugging than other simulators. (iii) Provides accurate modelling of physical phenomenon including hardware.

***Limitations:*** (i) Is quite limited in the number of protocols supported. (ii) Performance analysis and management are unclear (iii) Covers only partial extensions for mobility.

--------------------------------------------------------------------------------------------------------------

#### 4.1.2 NS-2

NS-2 is short for network simulator version-2, which supports both wired and wireless simulation capabilities and protocols such as TCP, UDP etc. It was developed to simulate the realism of actual systems [17]  It is one of the most commonly used Open Source simulators because of its flexibility and modular capability. NS-2 can simulate various issues such as; protocol interaction, scalability, effect of network dynamics, and congestion control, etc. It also supports execution of varied scenarios including network topology size, density distribution, traffic generation, membership distribution, real-time variance of membership, network dynamics etc. NS-2 results can be viewed in either text-based or animation-based formats [18]

***Strengths:*** (i) NS-2 retains many models including realistic mobility, robust and flexible scripting and simulation setups. (ii) NS-2 possesses a large user community with diverse support, and continuous development. (iii) NS-2 simplifies traffic and movement pattern via an efficient energy model. (iv) Complex scenarios can be easily tested and common for its modularity

***Limitations:*** (i) NS-2 requires recompilation with each change in user codes. (ii) High complexity when modelling real systems. (iii) It is often slow when simulating highly populated scenarios.

--------------------------------------------------------------------------------------------------------------

### 4.1.3 NS-3

NS-3 is short for network simulator version 3 and is a completely new simulator that is not an extension of NS-2. It can be used to model packet data networks, especially non-internet-based systems. NS-3 supports C++ and python languages. NS-3 enables device models for wired network Ethernet using CSMA/CD protocol scheme with cumulative back-off to contend for the shared transmission medium. A set of 802.11 models are also enabled that strive to provide a precise MAC level implementation of the 802.11 specification and a PHY-level model of the 802.11a specification [18].

Some features of NS-3 include: reduces memory footprint for simulation and does not assign memory for virtual zero bytes. No requirement for mobility model since wired devices do not require knowledge of node position. NS-3 protocols are designed to achieve closer realism of systems, and integration with several other open-source networking applications; reducing the requirement for simulation model redevelopment. The NS-3 project endeavours to sustain an open environment for researchers and developer to contribute and share their software [17].

***Strengths:*** (i) Supports better modularity than NS2, (ii) Support simulation for TCP, UDP, ICMP, IPv4, multicast routing, P2P and CSMA protocols. (iii) Supports ported code which potential simplifies model validation and amplifies credibility. (iv) Much more flexible than any other simulators [18]. (v) Wide range of use in both optimization and expansion of the existing networks.

***Limitations:*** (i) Python bindings do not work on Cygwin. (ii) Supports only IPv4 [17]. (iii) NS3 require dedicated maintainers to benefit from its merits. (iv) Active maintainers would need to consistently respond to the user questions, bug reports, tests and validations.

---

### 4.1.4 OverSim

OverSim [62] is a discrete event simulators that attempts to replicate the transmission, exchange of network-oriented messages via gate connections. OverSim is built on the modular-oriented OMNET++ and is GUI-driven.

***Strengths:*** (i) OverSim has efficient event scheduler and strong GUI support. (ii) Supports reuse of protocol implementations in real-world networks. (iii) Supports deeper inspection of message for bug identification.

***Limitations:*** (i) Supports limited number of protocols. (ii) Performance analysis and management are usually unclear.

---

## 4.2 ICS/SCADA

In this group, three simulators were reviewed. These include: OpenModelica, SCADASim, and Rapid SCADA. We briefly describe these simulators below:

### 4.2.1 OpenModelica

An Open Source modelling and simulation package designed for academic and industrial applications.

***Strengths:*** (i) Well documented projects. (ii) Has considerable OS community support with active forums for users and developers

***Limitations:*** (i) Yields challenges and irregular results when used for optimisation tasks.

------------------------------------------------------------------------------------------------------------

### 4.2.2 SCADASim

SCADASim [63] is an Open Source simulator designed to enable flexible and efficient SCADA system replication. SCADASim can support connections with real industrial hardware, thus possess emulation capabilities. SCADASim is built on top of OMNET++.

***Strengths:*** (i) Supports modularity. (iii) Supports real hardware integrations. (iii) Supports security tests and analysis.

***Limitations:*** (i) Small user community and contributions. (ii) Poor documentation

------------------------------------------------------------------------------------------------------------

### 4.2.3 RapidSCADA

RapidSCADA is an Open Source simulator for SCADA applications related to automated systems such as: Industrial automation systems, automation systems for home, fire alarms and security, access controls, and any controller-driven systems.

***Strengths:*** (i) Updated project documentation and communications. (ii) Supports hardware integrations. (iii) Web-enabled and Customisable

***Limitations:*** (i) Very little adoption for formal research and development. (ii) Not quite popular yet. (iii) No certainty of capability since there are very few available community usage results

------------------------------------------------------------------------------------------------------------

## 4.3 IoT/IIoT

In this group, three simulators were reviewed. These include: Cooja, Proview, and Kaa. We briefly describe these simulators below:

### 4.3.1 Cooja

Cooja [64] is an Open Source network simulator interface for IoT applications. Cooja is the Contiki network simulator and can support both small and large networks of Contiki motes.

***Strengths:*** (i) Flexible simulation control

***Limitations:*** (i) Slow execution while emulating hardware

---------------------------------------------------------------------------------------------------------------

### 4.3.2 Proview

Proview is an Open Source simulator for general process control systems which has evolved into a complete, integrated and low-cost solution that runs on standard PCs under the Linux operating system [65]. Proview retains all the typical functions for driving sequential control, adjustment, data acquisition, communication, supervision, etc.

***Strengths:*** (i) Good documentation. (ii) Support multiple protocol implementations

***Limitations:*** Not Found

---------------------------------------------------------------------------------------------------------------

### 4.3.3 Kaa

Kaa [66] is a new multi-purpose middleware simulator platform for the Internet of Things. It enables the development of complete end-to-end IoT solutions, connected applications, and smart products. Kaa is compatible with nearly any type of connected devices, sensors, and gateways. Kaa also provides a clear organisation of IoT features, can be extended for varied IoT applications, and supports plug-and-play. Kaa also retains nearly unlimited options for connectivity protocols and analytics integration [66].

***Strengths:*** (i) Support is hardware-agnostic, i.e. can connect to a huge variety of devices. (ii) Manages unlimited device integrations. (iii) Remote configurations. (iv) Support for cloud services, sensor data analytics, and user behavioural analysis

***Limitations:*** Not Found

---------------------------------------------------------------------------------------------------------------

## 4.4 Operations

In this group, three simulators were reviewed. These include: URURAU, DWSim, and ManPy. We briefly describe these simulators below:

### 4.4.1 URURAU

URURAU [67] is an Open Source discrete event simulator for simulating supply chain and logistics applications. It is GUI-oriented and is said to still be under development.

***Strengths:*** (i) Allows flexible construction of simulation models at varied layers of a software structure. (ii) Enables the flexible inclusion of new process commands to accommodate complex systems. (iii) Multi-platform support.

***Limitations:*** (i) Limited user support

---------------------------------------------------------------------------------------------------------------

### 4.4.2 DWSim

**DWSim** [43] is an Open Source multi-platform CAPE-OPEN compliant chemical process simulator. It is built on the top of the Microsoft .NET 4.5 as well as Mono Platforms and features a rich Graphical User

Interface (GUI). It can support easier simulations, analysis and understanding of the behaviours of chemical system.

***Strengths:*** Not Found

***Limitations:*** (i) It is restricted to chemical process simulation applications.

---------------------------------------------------------------------------------------------------------------------

### 4.4.3 ManPy

ManPy is short for 'Manufacturing in Python' and is an Open Source simulator project [68] purposed towards providing a structured layer of precise manufacturing objects that build on SimPy [56]. ManPy attempts to provide a collection of OS DES objects linkable like "black boxes" to form a model [68]. The collection is structured such that it is expansible via developer customisation and the addition of entirely new objects as needed. Although originally intended for manufacturing, ManPy's architecture is scaled to allow the development of objects relevant to other applications.

***Strengths:*** (i) Provides applications for an array of industrial case studies. (ii) Allows for modification and addition of new objects

***Limitations:*** (i) Limited to the simulation of procedural processes. (ii) Is restricted to discrete event simulation modes alone

---------------------------------------------------------------------------------------------------------------------

## 4.5 Generic

In this group, seven simulators emerged and were reviewed. These include: JaamSim, DESMO-J PowerDEVS, JavaSim, CD++, RePast, PySimulators. We briefly describe these simulators below:

### 4.5.1 JaamSim

JaamSim is an Open Source simulator developed by Ausenco (http://www.ausenco.com/) a company in Canada [69]. JaamSim provides a 3D user interface for discrete event simulations. JaamSim benefits from a generic architecture that enables developers to easily modify and add functionality palettes. Although fully open-source, JaamSim is developed and maintained by a commercial organisation, this may influence external support and funding.

***Strengths:*** (i) Flexible support for the addition of new objects. (ii) Can be tested and used by novice users with ease. (iii) Can be accessed and downloaded as a standalone executable for varied supported operating systems.

***Limitations:*** (i) Uncertainty about its credibility. (ii) No diversified demonstration of its applications. (iii) Not actively used by a wider community

---------------------------------------------------------------------------------------------------------------------

### 4.5.2 DESMO-J

DESMO-J [70] is an Open Source simulator that supports both event-driven and process-oriented modelling and simulation scenarios. DESMO-J provides readily usable classes for common model components. The user interface also has a capacity both 2D and 3D features. It is developed and maintained by the University of Hamburg computer science department, which provides online tutorials as well as API documentations for how to use the Open Source simulator. DESMO-J simulator has been used to perform research in the manufacturing sector.

***Strengths:*** (i) There is some degree of updating on the project.

***Limitations:*** (i) DESMO-J project seems to have no active community. (ii) No active support and communications available.

--------------------------------------------------------------------------------------------------------------------

### 4.5.3 PowerDEVS

PowerDEVS [71] is an Open Source simulator for implementing the DEVS formalism. Its main focus is for simulating hybrid systems. The interface allows atomic DEVS models in C++ language to be defined, which can be graphically joined in hierarchical block diagrams to create more complex systems [57].

***Strengths:*** (i) Active and frequent commits and contributions.

***Limitations:*** (i) No formal project documentation for third party users.

--------------------------------------------------------------------------------------------------------------------

### 4.5.4 JavaSim

JavaSIM [72] is a java-based Open Source package that provides discrete event process-based simulation similar to SIMULA's simulation class and libraries. JavaSim also provides complete examples and test routines illustrations.

***Strengths:*** (i) JavaSim possesses the Java built-in internet awareness features that enables its support potential for distributed simulation. (ii) JavaSim architecture enables developers to extend capabilities for customised needs.

***Limitations:*** (i) Limited in providing discrete event simulation models. (ii) Incomplete in comparison to commercial simulation environments. (iii) Very little documentation.

--------------------------------------------------------------------------------------------------------------------

### 4.5.5 CD++

CD++ [73] is a toolkit for Discrete-Event modelling and simulation which is based on the DEVS (Discrete-Event systems Specifications) formalism. CD++ has the simulation of Cell DEVS models. This is its main feature, though normal DEVS models can be simulated too. DEVS models can also be coupled to Cell DEVS models. Atomic models are written in C++ and are linked to the simulation tool [57].

***Strengths:*** (i) Well updated documentation for user support.

***Limitations:*** Not Found.

------------------------------------------------------------------------------------------------------------------------

**4.5.6 RePast**

The Repast Suite [74] is a Java-based Open Source modelling system that is designed for use on workstations and small computing clusters. Currently, RePast come in two versions: Repast Simphony and Repast for High Performance Computing.

***Strengths:*** (i) Active user community and consistent updates.

***Limitations:*** Not Found.

------------------------------------------------------------------------------------------------------------------------

**4.5.7 PySimulator**

PySimulator[75] is an Open Source simulator which a GUI for simulating different model types; including Functional Mockup Units, Modelica Models and SimulationX Models. These involve plotting result variables and applying simulation result analysis tools such as Fast Fourier Transform.

***Strengths:*** (i) Designed as a plugin system and can be extended by developer contributions. (ii) Can cope with large system representations.

***Limitations:*** Sparsely supported by OS community.

------------------------------------------------------------------------------------------------------------------------

## 4.6 Multiple

In this group, only one simulator was reviewed. The simulator is SimGrid, and some of its features are presented below:

### 4.6.1 SimGrid

SimGrid [30] is an Open Source package for simulating distributed applications on distributed platforms. SimGrid is scalable enough to support varied model implementations and adopts higher-level user interface designs. SimGrid also comes with APIs for both 'simulation' and 'real-world' modes to support applications development by distributed computing developers.

***Strengths:*** (i) Scalability and Speed of execution.

***Limitations:*** Not Known.

------------------------------------------------------------------------------------------------------------------------

## 5. Discussion

In this section, a discussion is presented based on the results of the review of Open Source simulators and projects carried out in the previous sections. The findings of the reviews are presented in the line with the research questions defined in section 2.

In answering the first research question: **Q1-** ***What are the common Open Source simulation tools used in Networks, ICS/SCADA, and IIoT research and development projects?***

We find that there are a lot of Open Source simulators that can be and are being used for the broad purposes described. This, in itself is a positive thing. Which means that developers and researchers are open to several options for the choice of Open Source simulator for their works depending on the nature of work being undertaken. It is also noted that some of the tools reviewed are categorised within specifics such as Networks, ICS/SCADA, and IIOT. This means that simulators within each class may only support replicating functions and capabilities within their respective domains and may not support functionalities in the other domain classes. However, there is the potential to benefit from the integration of functionalities provided by these varied platforms.

We also find that a larger proportion of the tools are classed as generic, a feature we consider significantly useful (Figure 19). This provides potential for developing architectures and infrastructures with expansible capabilities, into multi-class domains. It is beneficial to have such infrastructures as they can enable easier and faster modelling and simulation of complex systems or functionalities. Essentially, flexible and modular-capable Open Source simulator infrastructure presents a viable solution towards achieving such simulation capacity [21]. Open Source simulators that claim to have generic architecture models, should be able to present or demonstrate capabilities for such expansion and functionality features. As necessary, these need to be clearly identified and classified.

Quite unlike the generic domain, simulators within the communications and wireless sensor network domain (NS-2, NS-3, OMNET++, and OverSim) do have some level of expanded capabilities at varied levels within the network and communications landscape (Figure 20). These simulators do have a large interest from the scientific community. Open Source simulators such as NS-2, OMNET++, and NS-3 appear to attract substantial interest in the Open Source community as evidenced by the huge numbers of peer-reviewed articles that relate to these projects, and the number of times references are made to the articles that have used or explored these tools to further research goals. These simulators also appear in papers that have compared Open Source simulators within this domain [17], [18], [21], [25]. We think that this is good for the IIoT simulation context for a couple of reasons. First is the realisation that the domain of communications and sensors networks is an important part of the range of interests and knowledge needs of developers. Secondly, this domain also forms part of the IIoT development trend currently experiencing a great deal of hype. Hence, these simulators can provide a good understanding of the possible behaviour of similar architecture within the larger IIoT.

The above phenomenon also applies to the ICS/SCADA group of simulators (Figure 18). Although there are fewer Open Source infrastructures within the domain, as there is much proprietary support. SCADASim appears to have a growing community of users as seen from review results, as well as the existence of a few papers that have used or reviewed the simulator for study or test purposes, besides the authors of the simulator [20]. In other domains (IoT/IIoT and the general operations) – (Figure 17), usability is currently limited, and mostly originates from the authors of the respective Open Source simulators. There are fewer external contributors or users. We assume that this is connected to the trends in development in these areas or the Open Source simulators themselves. At least this is true for the IoT/IIoT domain. IoT and IIoT development is not as old as that of networks for instance. Open Source simulators like Cooja, Proview, and KAA have only emerged within the last few years to reflect a new class of tools for simulating/hosting and managing IoT/IIoT based on cloud or remote deployment and an array of features to allow system level deployment. These platforms can (or could) be run as simulators in ways that could be considered more representative of deployed systems, but do not purposefully address security. Unfortunately, these Open Source tools and yet not widely known. Their proprietary counterparts such as AWS, Google, ThingWorx, Mindsphere, and Bosch IoT are those seemingly having large impact currently. Perhaps, this may be because IoT/IIoT itself is currently still immature, and investigations and explorations on how effective and standardised this can be are still underway. The concept of simulating IoT/IIoT is also an idea under conception and testing, and common architecture is yet to be achieved [76]. Only Cooja appear to have gained some measure of attention by Open Source users, and it is expected that a steady growth in interest and utilisation will be seen in the future. This is also true for counterpart OS tools like Proview and KAA, which promise quite remarkable capabilities for simulating IIoT capabilities.

The existence of a community of developers and users is crucial for Open Source simulators. However, it is not sufficient on its own to ensure that the simulators achieve the necessary fidelity. Having a community of developers whose engagements are not properly controlled can adversely affect the project, where interests and contributors begin to withdraw their support due to cases of rudeness, stereotyping, harassment, and other negative and discouraging behaviours [77]. Notwithstanding, in the multitude of code/design contributors there is great potential to achieve bug reduction and better systems structures and capabilities [21]. However, in any simulation, it is necessary to make simplifying assumptions. If those simplifications are not well articulated or understood, then the chances are that codes and design structures will be reused either simply because they are available or because unmodified reuse can support direct comparison with existing published results whether validated or not. In either case, and especially the latter, it is possible that broadly invalid assumptions are unknowingly baked into projects spanning many years. For example, the plethora of simulations that characterized early work in ad hoc networking were later shown to be based on assumptions that were so unfounded as to render much of the considerable standard of work of no practical value.

In addition, since the industrial domain is chiefly characterised by a more complex combination of user agents with varied expertise, it is pertinent to have Open Source simulators that can attract developers as well as varied users. This can be achieved from quality coding that is readable, and an ability to disseminate and effectively manage OS projects [21]. Several of the Open Source simulators reviewed seem to lag in employing valuable Open Source development essentials that can drive success. Some of the essentials toward achieving this are presented in the guidelines by [38]. Consequently, a critical community and appropriate editorial control are essential precursors to the development of credible simulation platforms. At present, there is much work to be done in reaching this point for application domains like IIoT in which complex manufacturing-oriented feedback systems with uncertainties and loss are common.

While attempting to resolve answers for the second and third research questions:
**Q2 - *What are the functional structures and characteristics that encompass these open-source simulators? These include the purpose, compatible operating system platforms (e.g. Linux) that the tools run on, underlying generic programming languages (e.g. Python) that the projects use, the usage license (e.g. open) of the tools, and the degree of credibility for the simulation tools*?**

**Q3 - *How widely used are the Open Source tools (e.g. count of key references to the tools and their adoptions for research*)?**

We find that there are a number of earlier reviews for some of the designated field areas. More common are reviews of wireless sensor network simulators. Key functional structures and characteristics that appear common in these reviews as well as reviews in the other domain areas (ICS/SCADA, Operations, etc) include: licence types supported by the Open Source simulators, programming languages supported, operating systems supported, user interfaces support, and documentation types supported. These characteristics speak volumes about the degree of effectiveness of an Open Source simulator, although they may rarely provide the exhaustive basis for selecting one or other particular Open Source simulator(s).

We think that learning the level of Open Source usability or how widely the simulators are used by the open-source community also brings to light the attributed potential for future and consistent development and relevance of the simulator. For example, we assume that for additional reasons NS-2, OMNET++, and NS-3 are still relevant aside from having common standard Open Source features (Fogel, 2005), because of how widely and consistently the simulators are currently used to further studies, research, and test purposes. All three simulators are being updated quite frequently following the work of contributors that span broad areas of applications. We assume that subsequent utilisations often build upon or are influenced by the work, reports, and recommendations of prior works, and publishers, especially concerning real-world applications. It may be a cogent view that most Open Source simulators start from research institutes, and then strive to penetrate the academic environment [21]. This is expected to translate to applications not restricted to educational domains but to extend to real/industrial environments.

In summary, to validate simulation packages as being Open Source, functional structures and characteristics such as platform, Open Source licence, programming language, user interfaces, documentation, and communication types need to be carefully considered, selected, and well-articulated with their relevant and common attributes. This should be complemented by clear indications of a widely interested and contributing community via related publications. These should indicate success stories on the use of the simulators as well as their applications in real-world applications.

In providing answer to the fourth research question: **Q4 *What kind of research has built on these Open Source simulators? (e.g. operational, security analysis performance)?***

We find that the strengths of reviewed Open Source simulators vary according to the purposes for which they can be used. Most of which are based on performance analysis and optimisations. For example, the strengths of Open Source simulators within the communication and sensor network group are generally epitomised by *flexibility, modularity, mobility, scalability, fidelity,* and *active user/community support.* Limitations often exist due to the lack of one or more of the above characteristics in individual simulators. For example, OMNET++ is noted to cover only partial extensions for mobility and is also limited in the protocols it supports. In contrast, some of the characteristics appear as limitations in the other groups (ICS/SCADA, IoT/IIoT, Operations, and Generic) of simulators as well. A common limitation prevalent in these groups involve Open Source simulators that have very low or virtually no active community users/developers, and restricted capabilities in demonstrating aspects of IIoT outside their immediate focus. This may simply be a reflection of their maturity, but it may also have influenced the degree of interest in using them.

In providing answer to the fifth research question: **Q5 - *What are the strengths, limitations, constraints, and restrictions attributed to these tools in relations to suitability for IIoT simulation?***

We refer to section 4 where these concerns have been addressed. To be suited for the IIoT environment, Open Source simulators need to incorporate features such as flexibility, modularity, scalability, interoperability, and mobility to their generalised architectures. This is to support typical IoT characteristics attributed to devices, their interactions, and their services capabilities. Open Source simulators should be able to provide for the modelling of the dynamism of IIoT and the interplay between components and layers of hierarchical control in such a way that reflects real world scenarios.

In providing answer to the sixth and seventh research questions: **Q6 - *Has any of the simulators been designed with capability to explore industrial-related security analysis?***

**Q7 - *What aspects of security analysis and model have been explored using these tools, and what aspects can be explored for future (further) analysis?***

We find that very few (only three) of the Open Source simulators provide positive answers to these questions. Most of the simulators appear to focus on simulating functional and operational performance instead. For example, open simulators in the communications and wireless sensor networks group such as NS-2, NS-3, OMNET++, are typically used to simulate network performance and to measure quality of service (QoS) metrics – bandwidth, throughputs, latency, jitter [78], within certain predefined configurations. To incorporate and evaluate security functionalities or features, simulators in this category typically require the creation or modification of packet formats [79]. Security features are not normally automatically built into the simulators. For example, researchers in [79] defined new packet formats to represent new protocols in NS-2. This protocol-building enabled the addition of encryption and decryption capabilities in the data packets to ensure confidentiality of data. It also allowed for the implementation of a message digest generation function for data integrity of packets during transmission. Since mobile adhoc and sensor networks are typically characterised by resource-constrained nodes relative to power, computational capacities, memory, and communications bandwidth [80], [81], other aspects of security considered often include determining the impact of certain security implementations on the quality of service parameters. An example include the work in [82] involving the development of an independent single security layer to in NS-2 to manage the majority of security mechanisms distributed over other network layers on an IoT use case. In [83] a collaborative approach is presented for improving quality of services in wireless sensor network by using SAFEQ and the Watchdog algorithm. These examples show that service-oriented security features can be studied and analysed within some contexts related to IIoT, for example the sensor part. These are not embedded into the simulators, but solely depend on the ability to build new protocols and algorithms that introduce the desired security context. Security simulations with these tools are often limited by the knowledge and ability of the users involved.

Only JavaSim, SCADASim, and KAA simulators were found to have characterised inbuilt capabilities for some security modelling and analysis. In particular, SCADASim is said to be specifically developed for security-related analysis for ICS/SCADA systems. The authors in [63] presented and demonstrated the capability of the simulator for testing and evaluating denial-of-service (DoS), Man-in-the-middle, Eavesdropping, and Spoofing attacks. We also found a publication [72] where JavaSim was used to simulate injection prevention for web applications. KAA is a relatively new Open Source simulator for the IIoT, although the authors claim that the simulator supports security capabilities such as authentication and encryption, there are no independent narratives relating to the effectiveness of this simulator to replicate the described security features. It would be interesting to look at other types of attacks such the black hole, worm hole, session hijacking, impersonation, and traffic analysis using some of these Open Source simulators.  It is interesting to note that none of the Open Source simulators covered in this review indicates a capability to address very well the broad range of ancillary security simulation objectives outlined in section 1, as the requirements were potentially not present at the inception of the Open Source projects.

We also assume that the lack of security-related work in publications related to most of the Open Source simulators is an indication that security has not been a predominant issue or interest within the community of users and developers. Rather, interests and emphases appear to focus on functionality, performance analysis and optimisation. This focus needs to change from performance-only to security-inclusive modelling (architectures) if the ongoing deployment of IIoT systems is not to result in such a widespread distribution of individual vulnerabilities that there is national exposure to those who would wish us harm.

## 6. Conclusion

There is a long history of Open Source development for simulators and much research has been undertaken using them. Nevertheless, this is not an endeavour to be undertaken lightly: the prospect of developing a single simulator for IIoT systems is somewhat daunting given the degree of complexity resulting from the close interdependence of rather disparate systems [6]. However, the failure of the academic community to fully engage in researching IIoT security means that the race to deploy such systems is likely to give rise to vulnerabilities that are poorly understood and that, if exploited, could result in widespread physical and economic damage, injury, or even death. To engage that wider academic community, simulators provide an essential foundation since it is possible to conduct work using them readily and cheaply and, as the experience from network simulators shows, the time to critical mass is significantly reduced in comparison to domains in which these tools are less well developed.

From both ICS and IIoT perspective, it is obvious that there are Open Source simulation projects that are exploring the capabilities of representing these systems and environments to a degree that becomes useful for research. Even though this is the case the big challenge of configuring and managing these simulators for the desired (security-related) scenario remains. Ideally, if the scenario is to be useful, it should model the context that meets the real-world challenge that the simulator is intended to address. This leaves the networking community and indeed other simulation communities/affiliates with the need to demonstrate credibility in simulations (recognizing the risks of not doing so). Credibility comes, in part, from carefully designed simulators, that must, in the case of IIoT, be built with the active engagement of experts from interdisciplinary teams along with user contributions that are integrated under tight editorial control. But it also comes, in part, from the use of testbeds or operational systems to populate simulation cases and to validate simulation results, at least in part, so that there is evidence to support belief in them.

Again, this topic overlaps with the modelling and management of these systems themselves. This whole area does not seem to be well addressed, and it needs to be underpinned by good information management. Interesting aspects needing attention include; investigations into data models to represent ICS/IIoT systems to enable information integration, management and (where required) simulation. This presents a big gap in the current Industry 4.0, and also makes the simulation topic challenging. Further work is also required in the in-depth analysis and evaluation of the simulation capacities and fidelity of cloud-based Open Source IIoT simulators, in relation to security, given that most of the tools in this class are not typically developed to address security in the first place. It would be worthwhile to also engage studies looking into the evaluation of security topics that could be addressed by simulation – more specifically, suggesting how this could be achieved and identifying what can't be addressed by simulation. Investigations into simulation frameworks that can allow multi-mode simulations to be configured and operated are also required. Researches into Industry 4.0 system-of-systems (SoS) security evaluations, dependency, and cascading impacts method or analysis are relevant. This of course include explorations into model-driven optimisation of systems, and the opportunity and the challenges of doing this.

At present, there are gaps in all of the above areas and an exigency in needing to fill them. Consequently, this work is intended to inform understanding of these research gaps and needs, and to form part of a programme in which we expect the next step to be the commissioning of a simulator system that is designed by a team that includes industrial and governmental stakeholders. The intention would be to release this to the academic community using a flexible Open Source license and to promote it (and the research potential it embodies) across a variety of currently disjoint academic themes – from networking to process control – using the combined credibility of the contributors and stakeholders to further both adoption and interest in the published outcomes of using such a simulator experimentally.

## References

[1] A. Gilchrist, “Introducing Industry 4.0,” in *Industry 4.0: The Industrial Internet of Things*, First., Apress, 2016, pp. 195–215.

[2] Siemens, “Securing Industrial Control Systems. The Challenge and Common-sense,” Germany, 2018.

[3] R. Crahmaliuc, “Open-Source or Proprietary Software - What is Best for Users?,” *SIMSCALE Blog Website*, 2018. .

[4] S. Dhir and S. Dhir, “Adoption of open-source software versus proprietary software: An exploratory study,” *Strateg. Chang.*, vol. 26, no. 4, pp. 363–371, 2017.

[5] The Open Source Way Community, “The Open Source Way,” *Open Source Way Community Website*, 2017. [Online]. Available: http://www.theopensourceway.org. [Accessed: 22-Sep-2018].

[6] Opensource.com, “What is the open source way?,” *Open Source Community Website*, 2018. [Online]. Available: https://opensource.com/open-source-way. [Accessed: 22-Sep-2018].

[7] Vanson Bourne Ltd, “Open Source vs Proprietary: What organisations need to know,” 2017.

[8] S. J. Vaughan-Nichols, “It’s an open-source world: 78 percent of companies run open-source software,” *ZDNet Online Magazine*, 16-Apr-2015.

[9] TIDELIFT, “How to make open source work better for everyone,” 2018.

[10] A. Singh, R. . Bansal, and N. Jha, “Open Source vs. Proprietary Software,” *Int. J. Comput. Appl. (0975 – 8887)*, vol. 114, no. 18, pp. 26–31, 2015.

[11] A. S. Jadhav and R. M. Sonar, “Evaluating and selecting software packages: A review,” *Inf. Softw. Technol.*, vol. 51, no. 3, pp. 555–563, 2009.

[12] A. Arisha and M. El Baradie, “On the Selection of Simulation Software for Manufacturing Application,” in *Nineteenth International Manufacturing Conference (IMC-19)*, 2002, pp. 495–507.

[13] A. Gupta, “How to select a Simulation Software,” *Int. J. Eng. Res. Dev.*, vol. 10, no. 3, pp. 35–41, 2014.

[14] J. Nikoukaran, V. Hlupic, and R. J. Paul, “Criteria for Simulation Software Evaluation,” in *Proceedings of the 1998 Winter Simulation Conference*, 1998, pp. 399–406.

[15] J. Nikoukaran, “Software selection for simulation in manufacturing : a review. In Simulation Practice and Theory,” *Proceeding 1999 Winter Simul. Conf.*, vol. 7, pp. 1–14, 1999.

[16] H. Kavak, J. J. Padilla, D. Vernon-Bido, R. J. Gore, and S. Y. Diallo, “A Characterization of Cybersecurity Simulation Scenarios,” in *19th Communications and Networking Simulation Symposium (CNS’16)*, 2016, pp. 1–8.

[17] P. Chhimwal, D. S. Rai, and D. Rawat, “Comparison between Different Wireless Sensor Simulation Tools,” *IIOSR J. Electron. Commun. Eng.*, vol. 5, no. 2, pp. 54–60, 2013.

[18] M. H. Kabir, S. Islam, H. Javed, and S. Hossain, “Detail Comparison of Network Simulators,” *Int. J. Sci. Eng. Res.*, vol. 5, no. 10, pp. 203–218, 2014.

[19] A. Almadhor, “A Survey on Generic Scada Simulators.pdf,” *Int. J. Comput. Appl.*, vol. 128, no. 8, pp. 38–43, 2015.

[20] K. Mathioudakis, N. Frangiadakis, A. Merentitis, and V. Gazis, “Towards generic SCADA simulators: A survey of existing multi-purpose co-simulation platforms, best practices and use-cases,” in *Conf-Scoop.Org*, 2013.

[21] G. Dagkakis and C. Heavey, “A review of open source discrete event simulation software for operations research,” *J. Simul.*, vol. 10, no. 3, pp. 193–206, Aug. 2016.

[22] K. R. Lakhani and E. Von Hippel, “How open source software works: ‘free’ user-to-user assistance,” *Res. Policy*, vol. 32, no. 6, pp. 923–943, 2003.

[23] D. Margan and S. Čandrlić, “The success of open source software: A review,” in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2015 - Proceedings*, 2015, no. May, pp. 1463–1468.

[24] S. Bajracharya, J. Ossher, and C. Lopes, “Sourcerer: An infrastructure for large-scale collection and analysis of open-source code,” *Sci. Comput. Program.*, vol. 79, pp. 241–259, 2014.

[25] A. Kellner, K. Behrends, and D. Hogrefe, “Simulation Environments for Wireless Sensor Networks,” 2010.

[26] E. Egea-López, J. Vales-Alonso, a. S. Martínez-Sala, P. Pavón-Mariño, and J. García-Haro, “Simulation Tools for Wireless Sensor Networks,” in *Summer Simulation Multiconference - SPECTS’05*, 2005, pp. 2–9.

[27] K. Fogel, *How To Run A Successful Free Software Project - Producing Open Source Software*, 2nd ed. Karl Fogel, 2005.

[28] A. R. Robertson and R. N. Ibbett, “HASE: a flexible high performance architecture simulator,” in *Twenty-Seventh Hawaii International Conference on System Sciences,* 1994.

[29] J. M. Garrido, *Object Oriented Simulation: A Modeling and Programming Perspective*. New York: Springer, 2009.

[30] M. Quinson, “SimGrid: a Generic Framework for Large-Scale Distributed Experiments,” in *Proceedings of the Tenth International Conference on Computer Modeling and Simulation*, 2008, pp. 126–131.

[31] L. Prechelt, “An empirical comparison of seven programming languages,” *Computer (Long. Beach. Calif).*, vol. 33, no. 10, pp. 23–29, 2000.

[32] S. Nanz and C. A. Furia, “A comparative study of programming languages in rosetta code,” in *Proceedings - International Conference on Software Engineering*, 2015, pp. 778–788.

[33] B. Smith, “Object-Oriented Programming,” in *Advanced Action Script 3.0: Design Patterns*, New York, USA: Apress, 2015, pp. 1–23.

[34] C. Centioli *et al.*, “Open source real-time operating systems for plasma control at FTU,” *IEEE Trans. Nucl. Sci.*, vol. 51, no. 3 I, pp. 476–481, 2004.

[35] H. A. Alhassan and C. Bach, "Operating System and Decision Making," in *ASEE 2014 Zone I Conference*, 2014.

[36] K. G. Srinivasa, H. C. S. Raddi, M. S. Krishna, and N. Venkatesh, "MeghaOS: Cloud based Operating System and a Framework for Mobile Application Development," in *2011World Congress on Information and Communication Technologies*, 2011, pp. 858–863.

[37] M. Jasiunas, A. Chakraborty, and D. Kearney, *A distributed operating system supporting strong mobility of reconfigurable computing applications in a swarm of unpiloted airborne vehicles*. 2009.

[38] K. Fogel, "Participating as a Business, Non-Profit, or Government Agency," in *How To Run A Successful Free Software Project - Producing Open Source Software*, 2nd ed., 2005.

[39] A. Guillon and D. Loach, "YetiSim: a C++ simulation library with execution graphs instead of coroutines," in *DBLP Conference: Conference: Proceedings of the 2008 Spring Simulation Multiconference, SpringSim 2008*, 2008.

[40] K. Jaiswal and O. Prakash, "Simulation of MANET using GloMoSim Network Simulator," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 4, pp. 4975–4980, 2014.

[41] R. Bagrodia *et al.*, "Parsec: A Parallel Simulation Environment for Complex Systems," *Computer (Long. Beach. Calif).*, vol. 31, no. 10, pp. 77–85, 1998.

[42] R. K. Herz, "Getting started with the COCO chemical process simulator," 2012.

[43] Kannan M Moudgalya, "Home | DWSIM," *Project Website*, 2017. [Online]. Available: https://dwsim.fossee.in/. [Accessed: 16-Apr-2018].

[44] OSMC, "Welcome to OpenModelica - OpenModelica," *Project Website*. [Online]. Available: https://openmodelica.org/. [Accessed: 16-Apr-2018].

[45] S. Chen, "Comparison of Batch versus Continuous process in the Pharmaceutical Industry based on safety consideration.," Texas A&M University, 2017.

[46] B. Abdelhabib and B. Brahim, "JAPROSIM: A Java framework for process interaction discrete event simulation," *J. Object Technol.*, vol. 7, no. 1, pp. 103–119, 2008.

[47] B. Belattar and A. Bourouis, "Yet another java based discrete-event simulation library," *J. Softw.*, vol. 9, no. 1, pp. 82–88, 2014.

[48] B. L. Titzer, D. K. Lee, and J. Palsberg, "Avrora- scalable sensor network simulation with precise timing," in *4th International Symposium on Information Processing in Sensor Networks, IPSN 2005*, 2005, pp. 477–482.

[49] J. R. Doner, "GENESIM : generic network simulator," *IEEE J. Sel. AREAS Commun.*, vol. 6, no. 1, 1988.

[50] T. G. de S. Carneiro, P. R. de Andrade, G. Câmara, A. M. V. Monteiro, and R. R. Pereira, "An extensible toolbox for modeling nature–society interactions," *Environ. Model. Softw.*, vol. 46, pp. 104–117, Aug. 2013.

[51] J.-W. Chen and J. Zhang, "Comparing Text-based and Graphic User Interfaces for Novice and Expert Users," in *AMIA Annual Symposium proceedings*, 2007, pp. 125–129.

[52] Olofsson Robin and S. Hultstrand, "Git - CLI or GUI Which is most widely used and why ?," Blekinge Institute of Technology, Sweden, 2015.

[53] S. Davis and R. Bostrom, "An experimental investigation of the roles of the computer interface and individual characteristics in the learning of computer systems," *Int. J. Human–Computer Interact.*, vol. 4, no. 2, pp. 143–172, 2009.

[54] M. Rauterberg, "An empirical comparison of menu-selection (((CUI) and desktop (GUI) computer programs carried out by beginners and experts," *Behav. Inf. Technol.*, vol. 11, no. 4, pp. 227–236, 1992.

[55] I. Temple, Barker & Sloane, "The Benefits of the Graphical User Interface: A Report on New Primary Research.," Redmond; Wash, 1990.

[56] Team Simpy, "Overview — SimPy 3.0.10 documentation," *Simpy documentation*, 2017. [Online]. Available: https://simpy.readthedocs.io/en/latest/. [Accessed: 16-Apr-2018].

[57] Y. Van Tendeloo and H. Vangheluwe, "An evaluation of DEVS simulation tools," *Simulation*, vol. 93, no. 2, pp. 103–121, 2017.

[58] A. McInnes and B. Thorne, "ScipySim: towards distributed heterogeneous system simulation for the SciPy platform (Work-in-Progress)," in *Theory of Modeling & Simulation: DEVS ...*, 2011.

[59] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 126–137.

[60] M. J. Grant and A. Booth, "A typology of reviews: An analysis of 14 review types and associated methodologies," *Health Info. Libr. J.*, vol. 26, pp. 91–108, 2009.

[61] G. Liang, H. Hou, Z. Hu, F. Huang, Y. Wang, and S. Zhang, "Usage Count: A New Indicator to Detect Research Fronts," *J. Data Inf. Sci.*, vol. 2, no. 1, pp. 89–104, 2017.

[62] I. Baumgart, B. Heep, and S. Krause, "OverSim: A Flexible Overlay Network Simulation Framework," in *2007 IEEE Global Internet Symposium*, 2007, pp. 79–84.

[63] C. Queiroz, M. Abdun, and Z. Tari, "SCADASim—A Framework for Building SCADA Simulations," *IEEE Trans. Smart Grid*, vol. 2, no. 4, pp. 589–597, 2011.

[64] T. Mehmood, "COOJA Network Simulator: Exploring the Infinite Possible Ways to Compute the Performance Metrics of IOT Based Smart Devices to Understand the Working of IOT Based Compression &amp; Routing Protocols," 2017.

[65] Mandator and SSAB Oxelösund, "About ProviewR," *www.proview.se* , 2018. [Online]. Available: http://www.proview.se/v3/index.php/about-proview-leftmenu-27. [Accessed: 16-Apr-2018].

[66] KaaIoT, “Kaa IoT Overview — IoT Development product. Kaa open-source Internet of Things platform.,” *kAAPROJECT.ORG*, 2018. [Online]. Available: https://www.kaaproject.org/overview/. [Accessed: 16-Apr-2018].
[67] T. A. Peixoto, J. J. De, A. Rangel, Í. De, and O. Matias, “FREE AND OPEN-SOURCE SIMULATION SOFTWARE " URURAU ",” in *XLVII Brazilian Symposium on Operational Research*, 2015, pp. 3161–3173.
[68] O. Olaitana *et al.*, “Implementing ManPy, a semantic-free open-source discrete event simulation package, in a job shop,” *Procedia CIRP*, vol. 25, pp. 253–260, 2014.
[69] D. H. King and H. S. Harrison, “Open-Source Simulation Software ‘JaamSim,’” in *Proceedings of the 2013 Winter Simulation Conference*, 2013, pp. 2163–2171.
[70] B. Gehlsen and B. Page, “A FRAMEWORK FOR DISTRIBUTED SIMULATION OPTIMIZATION,” in *Proceedings of the 2001 Winter Simulation Conference*, 2001, pp. 508–514.
[71] F. Bergero and E. Kofman, “PowerDEVS: A tool for hybrid system modeling and real-time simulation,” *Simulation*, vol. 87, no. 1–2, pp. 113–132, 2011.
[72] D. Larson, J. Liu, and Y. Zuo, “Performance Analysis of JavaScript Injection Detection Techniques,” in *IEEE International Conference on Electro Information Technology*, 2014, pp. 140–148.
[73] G. Wainer, “CD++: A toolkit to develop DEVS models,” *Softw. - Pract. Exp.*, vol. 32, no. 13, pp. 1261–1306, 2002.
[74] Argonne National Laboratory, “Repast Suite,” *Repast Simulator Website*, 2017. [Online]. Available: https://repast.github.io/. [Accessed: 16-Apr-2018].
[75] A. Pfeiffer, M. Hellerer, S. Hartweg, M. Otter, and M. Reiner, “PySimulator - A Simulation and Analysis Environment in Python with Plugin Infrastructure,” in *Proceedings of the 9th International Modelica Conference*, 2012, pp. 523–536.
[76] S.-W. Lin *et al.*, “The Industrial Internet of Things Volume G1: Reference Architecture,” The Industrial Internet of Things Volume G1: Reference Architecture IIC:PUB:G1:V1.80:20170131, 2017.
[77] R. S. Geiger, “Summary Analysis of the 2017 GitHub Open Source Survey,” 2017.
[78] N. Kumar, “Mobile Ad hoc Network : Issue and Challenges Related to QoS and Solutions,” *Int. J. Eng. Technol. Sci. Res.*, vol. 4, no. 10, pp. 415–419, 2017.
[79] N. Hegde and S. S. Manvi, “Simulation of Wireless Sensor Network Security Model Using NS2,” *Int. J. Latest Trends Eng. Technol.*, vol. 4, no. 1, pp. 113–119, 2014.
[80] M. Asif, S. Khan, R. Ahmad, D. Singh, and M. Sohail, “Quality of Service of Routing Protocols in Wireless Sensor Networks: A Review,” *IEEE Access*, vol. 5, pp. 1846–1871, 2017.
[81] A. Dorri and S. R. Kamel, “Security Challenges in Mobile Ad Hoc Networks: A Survey,” *Int. J. Comput. Sci. Eng. Surv.*, vol. 6, no. 1, pp. 15–29, 2015.
[82] H. M. Aldosari, V. Snasel, and A. Abraham, “A New Security Layer for Improving the security of internet of things ( IoT ),” *Int. J. Comput. Inf. Syst. Ind. Manag. Appl.*, vol. 8, pp. 275–283, 2016.
[83] A. R. Karare, S. V Sonekar, and K. Akanksha, “Improving the Quality of Services in Wireless Sensor Network by Improving the Security,” in *International Journal of Engineering Research and Applications*, 2014, no. April, pp. 43–47.

## Appendix A

### Overview of the complete set of Open Source simulators

| S/N | Simulators | Language | Open Source Licence Type | Interface | Platform | Purpose (Simulation Application Area) | Documentation | Communications | Version Control | Simulation Modes | Type | Development Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | SimPy | Python | MIT | CLI | Linux, Windows | Generic | Online | Mailing list | Mercurial | DES | | |
| 2 | OMNET++ | C++ | ASL v2.0 | GUI & CLI | Linux, Windows, Mac OS | Computer & Wireless Sensor Networks Computer & Communication Networks | PDF, Online | Mailing list | x | DES | Simulator | Active |
| 3 | JaamSim | Java | GPL v3 | GUI | Linux, Windows, Mac OS X | Generic | Manuals | Forum | Git | DES | Simulator | Active |
| 4 | NS-3 | C++/Python | GPL v2 | GUI | Linux, FreeBSD, Mac OS | Computer & Wireless Sensor Networks Computer & Communication Networks | Several Formats | Wiki, IRC | Mercurial | DES | Simulator | Active |
| 5 | JiST | Java | Custom | CLI | Linux | Scalable Wireless Ad hoc Communication Networks | Manuals | x | cvs | DES | Simulator | Inactive |
| 6 | JAPROSIM | Java | LGPL v3 | GUI | Linux | Generic | HTML API | x | Subversion | DES & CS | Simulator | Inactive |
| 7 | DESMO-J | Java | ASL v2.0 | GUI | Linux, Solaris, Windows, Mac OS | Generic | PDF | x | Subversion | DES | Simulator | Active |
| 8 | Facsimile | Scala | LGPL v3 | GUI | Linux, BSD, Windows, Mac OS, UNIX | Engineering & Manufacturing | Online | x | Git | DES | Simulator | Inactive |
| 9 | SharpSim | C# | GPL v2 | GUI | Windows | Generic | PDF tutorial | Forum | x | DES | Simulator | Inactive |
| 10 | PowerDEVS | C++ | GPL v3 | GUI | Linux, Windows | Generic | x | x | Subversion | DEVS | Simulator | Active |
| 11 | DEV-C + + | C++ | X | CLI | Linux, UNIX | Generic | x | x | x | DEVS | Simulator | Active |
| 12 | Spades/JAVA | Java | X | CLI | Linux, Windows, Mac OS | Generic | Manual (DOC) | x | x | PDES | Simulator | Inactive |
| 13 | J-Sim | Java | BSD | GUI | Linux, Windows, Mac OS | Biomedicine | Online and videos | x | x | Multiple | Simulator | Inactive |
| 14 | Hase | C++/Java | Custom | GUI | Linux, Windows, Mac OS X | Computer Architecture | Online manual | x | x | DES | Simulator | Active |

| S/N | Simulators | Language | Open Source Licence Type | Interface | Platform | Purpose (Simulation Application Area) | Documentation | Communications | Version Control | Simulation Modes | Type | Development Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | SimJava | Java | Custom | GUI | UNIX, Windows NT | Generic | Online manual | x | x | DES | Simulator | Inactive |
| 16 | JavaSim | Java | LGPL v2.1 | GUI | Linux | Generic | DOC, Online, API, PDF | Mailing lists | x | DES | Simulator | Active |
| 17 | C++ Sim | C++ | LGPL v2.1 | GUI | Linux, Windows | Generic | PDF | x | x | DES | Simulator | Inactive |
| 18 | CD++ | C++ | X | GUI | Linux, Windows | Generic | DOC, videos | x | x | DEVS | Simulator | Active |
| 19 | SystemC | C/C++ | Custom | CLI | Linux | Systems on Chip | Video, Technical Courses | Forum | x | DES | Simulator | Unknown |
| 20 | PARSEC | C++ | Multiple | CLI | Linux | Parallel simulation | x | x | x | PDES | Simulator | Inactive |
| 21 | Turtuga | Java | LGPL | GUI | Linux, BSD, Windows, Mac OS, UNIX | Generic | x | x | x | DES | Simulator | Inactive |
| 22 | SIM.JS | JavaScript | LGPL | GUI | Cross-Platform | Generic | Online | Mailing list | Subversion | DES | Simulator | Inactive |
| 23 | GeneSim | C++ | GPL v2 | GUI | Linux | Code Generator | x | x | cvs | DES | | |
| 24 | OOSimL | C++/Java | Custom | GUI | Linux, Windows | Simulation Education | Manual (pdf—2 parts) | x | x | DES | Simulator | Inactive |
| 25 | Root-Sim | C | GPL v3 | CLI | POSIX | Parallel simulation | Online | x | Subversion | PDES, DS | Simulator | Inactive |
| 26 | libFAUDES | C++ | LGPL | GUI | Linux, Windows, Mac OS | Manufacturing | Online, API | x | x | Not DES | Simulator | Inactive |
| 27 | TerraME | C++ | LGPL | GUI | Linux, Windows, Mac OS | Terrestrial Systems | HTML API, DOC, PDF | x | Subversion | Multiple | Simulator | Active |
| 28 | RePast | C++ | New BSD | GUI | Linux, Windows, Mac OS | Generic | Online | Mailing lists | CVS | ABS | Simulator | Active |
| 29 | MGSim | C++ | X | GUI | Not Found | Systems on Chip | x | x | Git | DES | Simulator | Inactive |
| 30 | OSSim | Java | BSD | GUI | Linux | Educational | x | x | x | DES | | |
| 31 | URURAU | Java | GPL | GUI & CLI | Windows | Supply chain | Youtube videos | x | Mercurial | DES | | |
| 32 | SimGrid | Java/C/Ruby | GPL | Undefined | Linux | Distributed Systems (Fog, Cloud, MPI, Grid, etc) | Online | Mailing list, IRC | Subversion | DS | Simulator | Active |
| 33 | SIMCAN | C++ | Multiple | GUI | Linux | Computer & Communication Networks | x | Wiki | x | DES | Simulator | Inactive |
| 34 | OverSim | C++ | CCA v3 | GUI | Linux, Windows, Mac OS X | Computer & Communication Networks | Online | Mailing list | x | DES | Simulator | Active |

| S/N | Simulators | Language | Open Source Licence Type | Interface | Platform | Purpose (Simulation Application Area) | Documentation | Communications | Version Control | Simulation Modes | Type | Development Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 35 | JGPSS | Java | Custom | GUI | Linux | Simulation Education | x | x | Subversion | DES | Simulator | Inactive |
| 36 | JSL | Java | GPL | CLI | Linux | Simulation Education | PDF | x | Mercurial | DES | Simulator | Unknown |
| 37 | YetiSim | C++ | GPL v3 | GUI | Linux | Generic | x | Mailing lists | x | PDES | Simulator | Inactive |
| 38 | FreeSML | Java | LGPL | CLI | Linux | Generic | x | x | x | DES | Simulator | Inactive |
| 39 | Jades | Java | AL v2.0 | GUI | Windows | Generic | x | x | Git | DES | Simulator | Inactive |
| 40 | XGDESK | C# | MIT | GUI | Windows | Generic (Game-oriented) | x | x | x | DES | Simulator | Inactive |
| 41 | ScipySim | Python | GPL v3 | CLI | Linux, Windows, Mac OS X | Generic | x | Wiki | Mercurial | Multiple | Simulator | Active |
| 42 | PySimulator | Python | LGPL | GUI | Linux, Windows | Generic | x | x | Git | Multiple | Simulator | Active |
| 43 | TOSSIM | Python/C++ | BSD | CLI | Linux, Windows | Wireless Sensor Networks Communication | Online , PDF | Wiki, IRC | X | DES | Emulator | Active |
| 44 | Castalia | C++ (OMNET++) | APL | CLI | Linux, Mac OS | Wireless Sensor Networks, BAN, and Low power embedded devices Communication | Online , PDF | Forum, Wiki | Git, Subversion | DES | Simulator | Inactive |
| 45 | GloMoSim | C | APL | GUI | Linux, Windows | Computer & Wireless Sensor Networks Computer & Communication Networks | PDF | X | X | PDES | | Inactive |
| 46 | EmStar | C | APL | GUI | Linux | Wireless Sensor Networks Communication | X | Forum | X | TDS | Simulator, Emulator, Real | Inactive |
| 47 | ATEMU | C | APL | GUI | Linux, Solaris | Wireless Sensor Networks Communication | Online , PDF | Mailing list | X | DES | Emulator | Inactive |
| 48 | Avrora | Java | APL | CLI | Linux | Wireless Sensor Networks Communication | Online , PDF | Website | Subversion | DES | Simulator | Inactive |
| 49 | Cooja | Java | BSD | GUI | Linux, Cygwin/Windows, Mac OS X | IoT Network Simulator | X | IRC, Mailing List, Website, Wiki | Git | Multiple | Simulator, Emulator, Real | Active |
| 50 | Proview | C/C++/Java/FORTRAN | GNU/GPL | GUI | Linux, Windows | General Process Control and IoT | Online | Forum, Wiki | Subversion | Multiple | Emulator, Real | Active |

| S/N | Simulators | Language | Open Source Licence Type | Interface | Platform | Purpose (Simulation Application Area) | Documentation | Communications | Version Control | Simulation Modes | Type | Development Status |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 51 | VTB | C++ | Open Source | GUI | Linux | Electric and Power Control Systems (Naval) | X | X | Subversion | Multiple | Simulator | Inactive |
| 52 | Coco | C++ | Open Source | GUI | Windows | Steady-State Chemical Engineering Process Systems | Online | Forum, Mailing Lists | Subversion | DES & CS | Simulator | Inactive |
| 53 | DWSim | Visusl Basic .NET, C# | GNU/GPL v3 | GUI & CLI | Linux, Windows, MacOS, Android, iOS | Steady-State Chemical Engineering Process Systems (CAPE-OPEN Complaint) | Online, API, Youtube | Forum, Mailing Lists | Subversion | DES & CS | Simulator | Active |
| 54 | EMSO | C++ | Open Source | GUI | Linux, Windows | Chemical Process Dynamic Systems (Equation-oriented) | Online | Website, Wiki, | Subversion | DES & CS | Simulator | Inactive |
| 55 | OpenModelica | Modelica | GPL v3/OSMC-PL | GUI & CLI | Linux, Windows, MacOS | Industrial System Modelling | Online, Book, PPT Tutorial | Forum, Mailing Lists | Subversion | DES & CS | Simulator | active |
| 56 | SCADASim | OMNET++ | GPL v2 | GUI | Linux | ICS/SCADA System Simulation | X | IRC | X | DES | Simulator, Emulator, Real | inactive |
| 57 | ManPy | Python | LGPL | GUI & CLI | Linux | Manufacturing processes | PDF | IRC | | DES | Simulator | Active |
| 58 | Kaa | C/C++/Java/Objective-C | ASL v2.0 | GUI | Linux, Windows, Android, etc. | IoT Environment Simulation with Cloud Capabilities | Online | Forum, IRC, Blog | Subversion | multiple | Simulator, Emulator, Real | Active |
| 59 | Rapid SCADA | C# | ASL v2.0 | GUI | Linux, Windows | Control Automations (sensor, relays, controllers) | Online, Youtube, PDF | Forum, Website | Subversion | Multiple | Simulator, Emulator, Real | Active |
| 60 | NS-2 | C++ | GPL v2 | GUI & CLI | Linux, Cygwin/Windows | Computer & Wireless Sensor Networks Computer & Communication Networks | Several Formats | Wiki, IRC | Mercurial | DES | Simulator | Active |

## Appendix B: Overview of the Revised 21 Open Source simulators

| S/N | Simulators | Language | Open Source Licence Type | Interface | Times Cited (WoS) - Related Articles | Total No. of Articles (WoS) | Use for Security Analysis | Type | Development Status | Security Aspects Considered | Application Area Group |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | OMNET++ | C++ | ASL v2.0 | GUI & CLI | 626 | 549 | - | Simulator | Active | - | Communications & Wireless Sensor Networks |
| 2 | JaamSim | Java | GPL | GUI | 2 | 1 | - | Simulator | Active | - | Generic |
| 3 | NS-3 | C++, Python | GPL | GUI | 338 | 597 | - | Simulator | Active | - | Communications & Wireless Sensor Networks |
| 4 | DESMO-J | Java | ASL | GUI | 2 | 2 | - | Simulator | Active | - | Generic |
| 5 | PowerDEVS | C++ | GPL | GUI | 79 | 9 | - | Simulator | Active | - | Generic |
| 6 | JavaSim | Java | LGPL | GUI | 1 | 2 | Yes | Simulator | Active | JavaScript Injection Prevention for Web Pages | Generic |
| 7 | CD++ | C++ | Not Found | GUI | 0 | 0 | - | Simulator | Active | - | Generic |
| 8 | RePast | C++ | BSD | GUI | 143 | 99 | - | Simulator | Active | - | Generic |
| 9 | URURAU | Java | GPL | GUI & CLI | 3 | 4 | - | Simulator | Active | - | Operations |
| 10 | SimGrid | Java, C, Ruby | GPL | Not Found | 130 | 43 | - | Simulator | Active | - | Multiple |
| 11 | OverSim | C++ | CCA | GUI | 5 | 18 | - | Simulator | Active | - | Communications & Wireless Sensor Networks |
| 12 | PySimulator | Python | LGPL | GUI | 0 | 0 | - | Simulator | Active | - | Generic |
| 13 | Cooja | Java | BSD | GUI | 43 | 68 | - | Simulator, Emulator, Real | Active | - | IoT/IIoT |
| 14 | Proview | C, C++, Java, FORTRAN | GPL | GUI | 0 | 1 | - | Emulator, Real | Active | - | IoT/IIoT |

| S/N | Simulators | Language | Open Source Licence Type | Interface | Times Cited (WoS) - Related Articles | Total No. of Articles (WoS) | Use for Security Analysis | Type | Development Status | Security Aspects Considered | Application Area Group |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | DWSim | Visusl Basic .NET, C# | GPL | GUI & CLI | 0 | 1 | - | Simulator | Active | - | Operations |
| 16 | OpenModelica | Modelica | GPL/OSMC-PL | GUI & CLI | 35 | 49 | - | Simulator | Active | - | ICS/SCADA |
| 17 | SCADASim | (C++), OMNET++ | GPL | GUI | 45 | 2 | Yes | Simulator, Emulator, Real | Active | DoS, MitM, Eavesdropping, and Spoofing | ICS/SCADA |
| 18 | ManPy | Python | LGPL | GUI & CLI | 6 | 6 | - | Simulator | Active | Manufacturing (workflow, job shop, capacity planning), business processes, logistics and supply chain Mgt | Operations |
| 19 | Kaa | C, C++, Java, Objective-C | AL | GUI | 0 | 0 | Yes | Simulator, Emulator, Real | Active | Authentication, Encryption, persistence, IoT Capabilities enabling PnP, IIoT-Enabled | IoT/IIoT |
| 20 | Rapid SCADA | C# | AL | GUI | 0 | 0 | - | Simulator, Emulator, Real | Active | - | ICS/SCADA |
| 21 | NS-2 | C++ | GPL | GUI & CLI | 683 | 1398 | - | Simulator | Active | - | Communications & Wireless Sensor Networks |