

A review of the use and utility of industrial network-based open source simulators: functionality, security, and policy viewpoints

Journal of Defense Modeling and Simulation: Applications, Methodology, Technology
1–24

© The Author(s) 2020



DOI: 10.1177/1548512920953499

journals.sagepub.com/home/dms



Uchenna Daniel Ani¹ , Jeremy McKendrick Watson¹, Madeline Carr¹, Al Cook², and Jason RC Nurse³

Abstract

Simulation can provide a useful means to understand issues linked to industrial network operations. For transparent, collaborative, cost-effective solutions development, and to attract the broadest interest base, simulation is critical and open source suggested, because it costs less to access, install, and use. This study contributes new insights from security and functionality characteristics metrics to underscore the use and effectiveness of open source simulators. Several open source simulators span applications in communications and wireless sensor networks, industrial control systems, and the Industrial Internet of Things. Some drivers for their use span are as follows: supported license types; programming languages; operating systems platforms; user interface types; documentation and communication types; citations; code commits; and number of contributors. Research in these simulators is built around performance and optimization relative to flexibility, scalability, mobility, and active user support. No single simulator addresses all these conceivable characteristics. In addition to modeling contexts that match real-world scenarios and issues, an effective open source simulator needs to demonstrate credibility, which can be gained partly through actively engaging experts from interdisciplinary teams along with user contributions integrated under tight editorial controls. Government-led policies and regulations are also necessary to support their wider awareness and more productive use for real-world purposes.

Keywords

Open source simulators, industrial control system simulations, Industrial Internet of Things simulation, security simulations, open source tools, system simulations

1. Introduction

Simulations have 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). With respect to industrial control systems (ICSs), simulation use has become pervasive during system design and in tuning process control parameters or exploring the outcomes of new control algorithms. At the same time, much of the research on general networked systems, particularly sensors and wireless networked systems, has been done using modeling 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, there has

¹Department of Science Technology Engineering and Public Policy, University College London, UK

²Critical Insights Security Ltd, UK

³School of Computing, University of Kent, UK

Corresponding author:

Uchenna Daniel Ani, Department of Science Technology Engineering and Public Policy, University College London, 11-20 Capper Street, London, WC1E 6JA, UK.

Email: u.ani@ucl.ac.uk

been increasing synergy between the networking technologies and process control, motivated by reasons of cost, flexibility, and performance, amongst other benefits.¹ Consequently, there is sizeable opportunity to integrate corresponding simulation systems to reflect the new reality of industrial control. Systems that have explored this integration and the research that uses them, unsurprisingly, focus largely on issues of performance.

Whilst performance is undeniably 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 security and safety. This category of system is being exposed to new forms of risks typically related to cyber-attacks, with both physical and virtual consequences. For example, the cyber-attack incident that targeted the Ukrainian electric system critical infrastructure interfered with supervisory control and data acquisition (SCADA) automation functions, causing the affected substations to shut down and a move to manual operations. This affected 225,000 customers across three different distribution-level service territories.² Thus, aside from the initial objective of testing performance using simulations, it has also become crucial to consider the added objective of testing for security, aimed at exploring how to implement technical, physical, and administrative measures and controls to provide confidentiality, integrity, and availability.³ A simulator's security goal typically includes enabling an understanding of sensitivities caused by an inability to sufficiently address certain relevant security requirements. Clarifying the applicable security context(s) and characteristics of simulators can help provide a clear understanding of the strengths and limitations of open source simulators under consideration.

Although standard commercial-off-the-shelf (COTS) security technologies are helping to provide a level of security, the behavioral complexity of coupled feedback systems with the COTS technology still offers new opportunities to attackers. Researchers need experimental platforms to serve as testbeds they can use with ease, where they can assess the feasibilities and modes these attacks can take, the impacts of the attacks, and the effectiveness of proposed controls/defenses. This is especially important because the numbers of forensically analyzed attacks on ICSs is low due to certain challenges.⁴ Firstly, there is less availability for static forensic procedures due to system criticality – ICSs cannot be powered-off for acquisition. Secondly, there is a likelihood of losing forensic evidence due to the volatility of information in ICSs.⁵ Thus, there is little experience on which to draw. For those incident scenarios that have evidence, attacks launched by nation-state actors also have a high degree of sophistication.

In this era of collaborative work, the motivation of which remains to achieve the contribution and collection of ideas

and concepts, which can be open to critiques, there are reviews and refinements to achieve a more robust solution, and at significantly reduced or no cost. Potentially, there can be more refined outcomes from a well-coordinated ensemble of contributions. To foster transparent, collaborative, and cost-effective studies, demonstrations, and development of solutions, and 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. In addition, open source simulators can be run with general purpose (non-proprietary) computing equipment and setups. Simulation through open source provides a means to address some of the mentioned challenges while benefiting from collaborative participation, free exchange, transparency, rapid prototyping, meritocracy, and community support.⁶ These help to address such issues as license use, modification and share restrictions, vendor lock-ins, and high cost of maintenance attributed to using commercial simulators.⁷ The open source philosophy enables users to read source/development codes, tailor generic simulator changes to suit specific needs, and enable on-the-fly program debugging. These capacities also place open source simulators (just like other open source software) side-by-side with proprietary simulators in user communities and market niches where tractability is crucial, and shared development is effective and supported. In such markets, as is typical today, conflicts and market pressure play a part in establishing a level playing field for the creation of incentives from both open source simulators and their commercial counterparts. Potentially, government has a role in driving a balance and avoiding undesirable market failures.^{8,9}

This paper presents an analytic study of commonly used open source simulators across the computer networking and industrial control communities. The study aims at evaluating the use and utility of open source simulators to represent ICS domain functionality and security-related attributes, and to highlight the relevant factors, attributes, and policy directions that can support future open source ICS/Industrial Internet of Things (IIoT) tool developments. This is explored through the following research objectives: (a) investigate the common open source simulation tools that are used in industrial networks and Internet of Things (IoT) research and development projects; (b) analyze the functional characteristics of identified open source simulators; (c) investigate the interest and usage level of existing open source simulators, (d) identify the kind of research that has built on these open source simulators (e.g., operational performance, optimizations, or security analysis); (e) identify if any of the reviewed simulators have been designed with capability for security analysis; (f) analyze the aspects of security analysis and modeling that have been explored using open source tools, and aspects open for future (further) analysis; and (g) analyze how policy

and regulations can support the growth and development of open source simulators. By addressing these objectives, this study contributes new insights drawing from metrics related to security inclusions and usability coverage to underscore the effectiveness of open source simulators in the ICS context. Aspects of policy and regulation that can support such effectiveness are discussed with attribute recommendations for constructing simulators that address identified gaps. This study is intended to reveal (where necessary) new open source simulation tools, systems, and platforms with the capability to support research and developments in the emerging industrial control and Internet with perspectives on security.

The rest of the paper is outlined as follows: Section 2 presents a review of related works. Section 3 presents a description of the research methodology adopted in this study, and the assessment criteria used. In Section 4, the results of evaluations are presented and analyzed. Section 5 presents a discussion of results, and Section 6 presents the conclusion and future works.

2. Related work

A number of publications^{10–12} describe some valuable characteristics of successful open source projects. An open source project can be successful if it can reward contributions through boosting the reputation of contributors, or providing answers or solutions to lingering questions.¹⁰ A successful project needs to possess an ability to collect large-scale data from a community of users and enable a robust evaluation and analysis of source codes.¹² Margan and Čandrlić¹¹ assert that the success of an open source software project depends on a number of factors, including the ability to attract a sufficient number of highly motivated and skilled contributors, having an excellent code base, a thriving communities of users, an independent and powerful leadership and control with clearly defined goals and visions, plus good communication mechanisms.

Others^{13–16} have reviewed open source simulators from specific domains, such as wireless sensor networks (WSNs), ICSs, SCADA, industrial operations, etc., based on self-selected evaluation attributes and characteristics such as domain-specific applications, critical infrastructure sectors of adoption, license distribution type, open source programming language supported, availability of documentation, degree of ease of use, and interface support type. Even though these reviews do not directly refer to the IIoT specifically, their focus often represents a facet or sub-system of what is considered as the larger IIoT, thus providing useful knowledge and insights about open source software characteristics that are useful for analysis, and sound open source project development. These works

provide information and guidance for the criteria for review and evaluation that characterizes our study.

Typically, open source projects have certain peculiar characteristics that make some of the criteria outlined in existing works inapplicable. For example, simulator cost is discussed by Dagkakis and Heavey.¹⁷ Not repudiating the existence of other significant open source simulator characteristics, simulator cost is also important, but this is not necessarily visible in open source.¹⁷ It is more relevant in the COTS domains. Accordingly, an example is discussed that involves not questioning the choice of methodology for documenting or communicating project updates and utilization, although both characteristics are significant and often form license 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 also unusual to question how developers of a proprietary software project communicate, although it is an important attribute of an open source project.

In the context of the IIoT, a couple of prior works have discussed similar and related contexts, such as reviews of open source simulators for WSNs^{13,18} and open source simulators for ICS/SCADA-related systems.^{16,19} In this, similar open source simulation characteristics are seen to be used for evaluating and comparing a number of simulation projects. For example, graphical user interface (GUI)-support characteristics, license distribution types, application domain, and supported programming languages appear in multiple works that analyze WSNs simulators.^{13,17,18} However, we were unable to find any papers that have reviewed open source simulators considering IIoT applications, and with perspectives on security. In addition, no prior works captures the broad range and category of simulators represented in this work. The 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.

To manage complexity, the IIoT is typically viewed as a system of systems (SoS). Each sub-system 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 of achieving acceptable fidelity. Should the sub-system simulation ensemble be developed by different parties, there is no 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 worse, questionable outcomes that result from differing assumptions built into supposedly similar functional units in different sub-systems, or the interactions amongst them.

3. Methodology

This study followed a systematic analysis process from data collection to analysis²⁰ comprising a two-phase approach as proposed by Webster and Watson,²¹ which supports following rigorous and relevant research techniques, and guaranteeing the quality and veracity of selected articles.²² This approach has been used by Lu²³ to analyze Industry 4.0 technologies, applications, and associated open research issues. The processes followed are presented in Figure 1 and described in the next section.

3.1. Document gathering and selection

Relevant literature was gathered and selection made based on the chosen search phrase “Open Source Simulators” from SCOPUS, IEEE Xplore, ScienceDirect, Springer, ACM Digital Library (DL), and Web of Science (WoS)

literature databases. In addition, the keyword “network simulators” was also used for searches on two popular software project repositories: Source-Forge (<http://sourceforge.net/>) and GitHub (<https://github.com/open-source>). For both searches, the date range for articles was from 2003 to April 2018. As anticipated, this search yielded a total of 767 related articles/projects, as shown in Table 1.

The second phase involved extracting the document meta-data from the 767 related articles. From these, titles or contexts were checked to identify the articles that covered open source simulator project characterizations relative to ICSs, communications and sensor networks, and the IoT. This was carefully performed, and the unrelated articles were discarded. For open source project articles that appeared in multiple databases, only a single instance was recorded, while the other instances were discarded to avoid duplication of the same records and results. At the end of the process, a total of 60 related open source

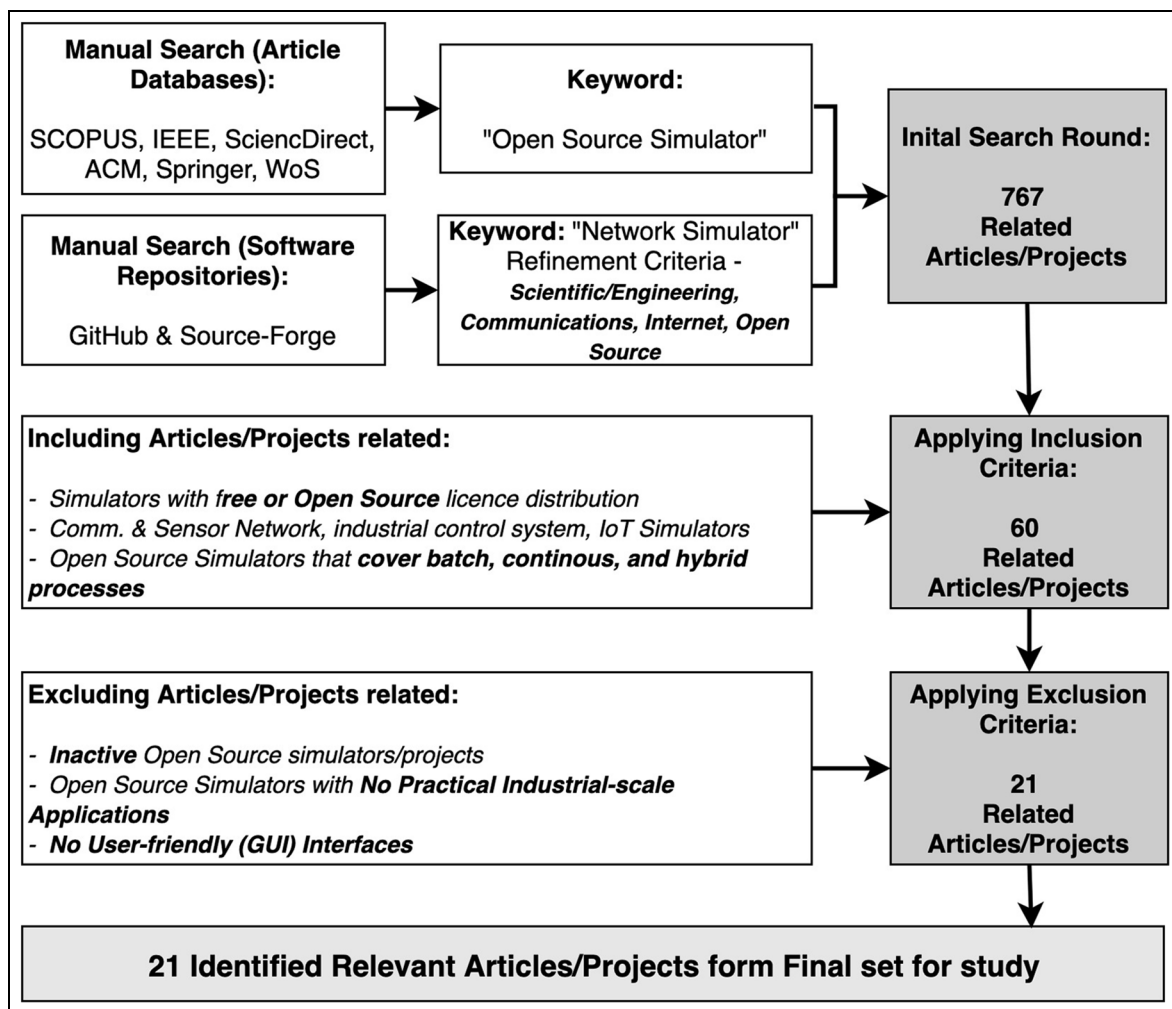


Figure 1. Article review process/approach. IoT: Internet of Things; GUI: graphical user interface.

Table 1. Database source results for open source simulators article survey.

Databases/repositories	No. of articles/projects
SCOPUS	149
IEEE Xplore	66
ScienceDirect	163
ACM Digital Library	47
Web of Science (WoS)	75
Springer	174
GitHub Repository	32
Source-Forge Repository	61
Total	767

simulator articles/projects were selected. Another process was adopted involving further review of abstracts/project overviews for the 60 projects with considerations for some inclusion and exclusion criteria. These formed the boundaries for evaluations, to ensure maintenance of the study scope and alignment with the study objectives. The following review restrictions were made.

1) Inclusions

- i. Only 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.
- ii. Open source project and simulation tools in areas related to industrial/production networks, including ICSs, SCADA, distributed control systems (DCSs), process control systems (PCSs), communication and sensor networks, and the IoT, were considered.
- iii. Open source simulators that cover batch processes, continuous processes, or both, were included since these exemplified varied modes of industrial processes implementable in digital networks.

2) Exclusions

- iv. Open source simulators found to be inactive in terms of use for research and support and(or) development status were not examined.
- v. Open source simulators without practical industrial or real-world applications, for example, those solely designated for educational training and practice purposes were not examined.
- vi. Open source simulators that lacked any form of user-friendly operating or visualization interface(s) were not examined.

After applying the above inclusion and exclusion criteria, a total of 39 articles/projects were discarded.

Twenty-one articles/projects were found to be relevant and were used to further the defined research objectives and scope of the study. These selected articles/projects, listed in Table 2, formed the basis of analysis in this study.

3.2. Evaluation criteria for open source simulators

Evaluating and selecting simulation software is a subject that has been addressed in some prior works.^{24–26} More relevant to this research, we find that some publications^{17,27–29} have tried to address the selection of open source simulation tools, most of which do not directly relate to ICSs. Nevertheless, the publications have helped to establish a solid conjectural background on the types of open source licenses,²⁹ the varied license attributes, how choices may be rationalized,²⁶ and how to successfully run an open source project (including simulator-based projects).²⁷ The publications also provide reference to some important factors for evaluating existing open source tools/projects, some of which can also pass as criteria for selecting open source simulation tools/projects and be pointers for improvements.

For example, open source simulator development status, distribution mechanisms, version control, communication channels, developer guidelines, documentation, open source license type, and code/commit reviews are attributes that can contribute to a good open source project.²⁷ Some of these attributes, in addition to latest release, language, domain, and simulation technique, have been used to evaluate open source discrete event simulation (DES) software for operations research.¹⁷ The prior works provide inspiration and reference guide for considering and selecting the factors believed to best suit the need of this study: programming language, open source license type, simulation mode, documentation, communications, and version control. Considering suitability based on the information that can be produced, the selected attributes elaborate on the structure of open source simulators and shed light on how they are built and how they function. These details, when aggregated, can help address the research objectives outlined. According to research needs, other factors were included to capture key aspects of research interests. The interest to understand the trend around the use of reviewed open source tools led to the inclusion of “Purpose-Driven Group.” Keenness to learn about operating system compatibilities and user-centric considerations of open source projects led to including “Operating Platform” and “User Interface” as factors for consideration.

To assess how widely the selected simulators are used, and to understand the kind of research built around them and their applications, a systematic literature review³⁰ of related works was done spanning the period of 2010–2018

Table 2. Characterization of open source simulators (part 1).

Simulators	Programming language	Open source license type	Interface	Platform	Times cited (WoS)	No. of articles (WoS)	Inbuilt security capacity	Purpose (simulation application area)	Simulation techniques
OMNET++	C++	ASL v2.0	GUI & CLI	Linux, Windows, Mac OS	626	549	-	CWSNs	DES
JaamSim	Java	GPL v3	GUI	Linux, Windows, Mac OS X	2	1	-	Generic	DES
NS-3	C++, Python	GPL v2	CLI with CLI + GUI-like visualization tool	Linux, FreeBSD, Mac OS	338	597	-	CWSNs	DES
DESMO-J	Java	ASL v2.0	GUI	Linux, Solaris, Windows, Mac OS	2	2	-	Generic	DES
PowerDEVS	C++	GPL v3	GUI	Linux, Windows	79	9	-	Generic	DEVS
JavaSim	Java	LGPL v2.1	GUI	Linux	1	2	Yes	Generic	DES
CD++	C++	Not found	GUI	Linux, Windows	0	0	-	Generic	DEVS
RePast	C++	New BSD	GUI	Linux, Windows, Mac OS	143	99	-	Generic	ABS
URURAU	Java	GPL	GUI & CLI	Windows	3	4	-	Supply chain	DES
SimGrid	Java, C, Ruby	GPL	Not found	Linux	130	43	-	Distributed Systems (fog, cloud, MPI, grid, etc.)	DS
OverSim	C++	CCA v3	GUI	Linux, Windows, Mac OS X	5	18	-	CWSNs	DES
PySimulator	Python	LGPL	GUI	Linux, Windows	0	0	-	Generic	DES & CS
Cooja	Java	BSD	GUI	Linux, Cygwin/Windows, Mac OS X	43	68	-	IoT network simulator	DES & CS
Proview	C, C++, Java, FORTRAN	GNU/GPL	GUI	Linux, Windows	0	1	-	General process control and IoT	DES & CS
DWSim	Visual Basic .NET, C#	GNU/GPL v3	GUI & CLI	Linux, Windows, MacOS, Android, iOS	0	1	-	Steady-state chemical engineering process systems (CAPE-OPEN compliant)	DES & CS

(continued)

Table 2. (Continued)

Simulators	Programming language	Open source license type	Interface	Platform	Times cited (WoS)	No. of articles (WoS)	Inbuilt security capacity	Purpose (simulation application area)	Simulation techniques
OpenModelica	Modelica	GPL v3, OSMC-PL	GUI & CLI	Linux, Windows, MacOS Linux	35	49	-	Industrial system modeling ICS/SCADA system simulation	DES & CS
SCADASim	(C++) OMNET++	GPL v2	GUI	Linux	45	2	Yes	Manufacturing processes IoT environment simulation with cloud capabilities	DES
ManPy	Python	LGPL	GUI & CLI	Linux	6	6	-	Control automations (sensor, relays, controllers)	DES
Kaa	C, C++, Java, Objective-C	ASL v2.0	GUI	Linux, Windows, Android, etc.	0	0	Yes	CWSNs	DES & CS
Rapid SCADA	C#	ASL v2.0	GUI	Linux, Windows	0	0	-	Control automations (sensor, relays, controllers)	DES & CS
NS-2	C++	GPL v2	GUI & CLI	Linux, Cygwin/ Windows	683	1398	-	CWSNs	DES

WoS: Web of Science; ASL: Apache Software License; GPL: General Public License; LGPL: Lesser General Public License; BSD: Berkeley Software Distribution; CCA: Consumer Credit Act; OSMC-PL: Open Source Modelica Consortium Public License; GUI: graphical user interface; CLI: command-line interface; CWSN: computer & wireless sensor networks; MPI: message passing interface; IoT: Internet of Things; ICS: industrial control system; SCADA: supervisory control and data acquisition; DES: discrete event simulation; DEVS: Discrete Event System Specification; ABS: agent-based simulation; DS: dynamic simulation; CS: continuous simulation.

to obtain key research results that focus on individual simulator workspaces. This is termed “research fronts,”³¹ referring to clusters of papers that share a common intellectual base. In this study, the common intellectual base refers to peer-reviewed research and knowledgeable outputs related to selected simulation tools. Two key metric indicators for research fronts include “Usage Count” and “Times Cited,” with specific application to the WoS database of articles.³¹ “Times Cited” seems to be the more common metric of the two adopted for analyzing citation fronts, despite its limitations of characterizing elongated time-lags in demonstrating the effects of research fronts and an inability to reflect current interests of the research community. With the case for using “Usage Count” as a viable alternative or complement still open in the research community, the “Times Cited” indicator is adopted in this study as the criterion for evaluating the top 10 articles from the keyword search results on the WoS database for each of the selected simulators. The keyword format used included “Simulation” AND “Simulator-Name,” where Simulator-Name was changed for the actual simulator name, for example, NS-2, for each analysis. As a specific example, in the case of NS-2, the keywords “simulation” AND “NS-2” was used, and within the defined time range. The WoS has a reputation for supervised selection and inclusion of materials based on high-quality and high-impact indexing by humans, consistent and structured documentation, better accuracy of results, and reduced duplicates and false positives.³² Also, the WoS seems preferred by organizations as a standard.³³ Search results were restricted to peer-reviewed articles (journals, conferences, and books) in order to ensure the quality and credibility of outcomes.

Lastly, considering security as a key focus, the study also aimed to determine the open source tools that are characterized to address security-related simulation scenarios, and to understand the specific security contexts covered. This led to introducing the “Security-oriented Application” criterion. Thus, as presented in Figure 2, the criteria adopted in this study include programming language, open source license type, operating platform, user interface, simulation mode, documentation, communications, version control, purpose-driven group, times cited, and security-oriented application. Potentially, combining the information from all the factors listed can enable new insights that can support effective decision-making in evaluating and selecting open source simulators, and for considering attributes of focus for future open source simulator developments. The results of initial searches offer a sense of the true state; in particular, it indicates that 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. Thus, it is essential to identify those that are emerging as well as those that are established.

4. Evaluation

In this section, the evaluation outcomes of the open source simulator analysis are presented based on the criteria described in the previous section.

4.1. Initial results and filtering

A total of 60 open source simulation tools were identified. However, 39 open source tools were discarded for not satisfying the defined inclusion and(or) exclusion criteria.

Many of the discarded open source project simulators seemed inactive for either of a number of reasons: not having significant usage in the simulation community; lacking active community activities or forums; not receiving updates in the past 2 years; and lacking any clear information from developers about the active status of the project, especially related to still being under development. For example, JARPROSIM^{34,35} did not seem to have wide usage, and had its most recent version update in 2014. A check on the log of its code commit history yielded null results. Again, Avrora³⁶ had its latest version series – Avrora 1.7.X released in 2008 – and a last modified version in the development series was done in 2013. NS-2¹⁸ was not discarded because it had quite significant usage for research although its last update was in November 2011. Hase³⁷ was discarded because it was developed to solely simulate computer architectures without considering network components. The TerraME³⁸ simulator was discarded because it was purposed for simulating terrestrial systems, which was clearly outside the scope of evaluation. JSL and OOSimL³⁹ simulators were excluded because their uses were limited to educational purposes.

Simpy,⁴⁰ DEV-C++,⁴¹ ScipySim,⁴² and TOSSIM⁴³ simulators were excluded on the grounds of supporting only command-line interfaces (CLIs). Considering the drivers for ease of use, simulators with only CLIs and without GUIs or any form of graphic visualization are less likely to provide good intuitiveness, easy comprehension, less stress, and visualization capabilities, and as such are less likely to attract interest, especially from the wider group of non-expert users. These advantages put open source simulators with a GUI ahead. However, simulators that support both GUI and CLI features provide even greater advantages, retaining the combined strength of the two interface forms in one system. Presumably, this would be more likely to attract a wider population of users spanning both experts and non-experts, since the preferences of both are supported. Based on this advantage, NS-3 was not discarded, because it had a GUI-like visualization tool linked to its CLI.

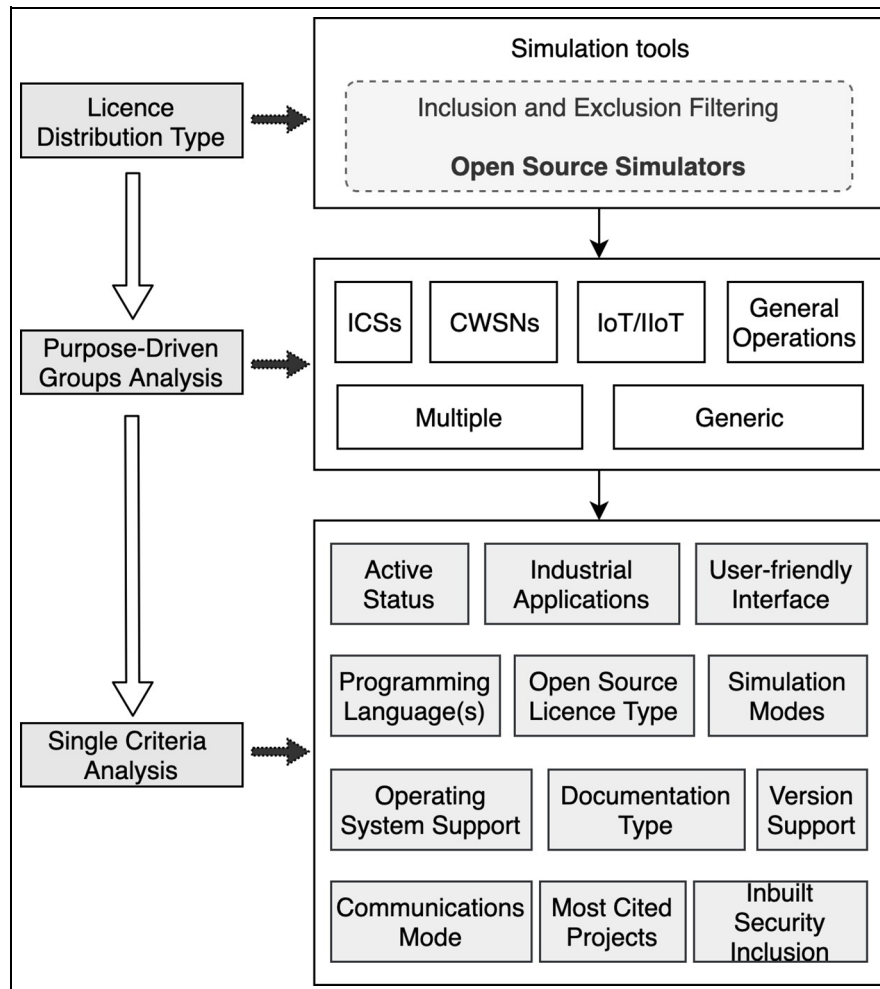


Figure 2. Open source simulator evaluation criteria. ICS: industrial control system; CWSN: computer & wireless sensor networks; IoT: Internet of Things; IIoT: Industrial Internet of Things.

A total of 21 open source simulators remained for further analysis based on the criteria described in Section 4.1. The results are summarized in Section 4.2 (Subsections 4.2.1 to 4.2.10).

4.2. Results

Valuable inferences can be drawn from the results of analyzing the selected open source simulators.

4.2.1. Purpose-driven groups. A purpose-based classification of the 21 open source simulators showed that a third (seven) of the simulators claimed to possess generic architectures. These are JaamSim, DESMO-J, PowerDEVS, JavaSim, CD++, RePast, and PySimulator. Four simulators, OMNET++, NS-2, NS-3, and OverSim, were categorized under communications/WSNs. Three simulators

were categorized as suitable for more generalized operations that did not encompass direct physical hardware infrastructure. Three simulators each were classed under more specific areas, namely ICS/SCADA (OpenModelica, SCADASim, RapidSCADA) and IoT/IIoT (Cooja, Proview, Kaa), while one simulator (SimGrid) claims to cover multiple architectures. The usage scope of SimGrid was not clearly outlined in any of the project demonstrations. Further research on the use SimGrid indicated application areas only in parallel and distributed computing system simulations and studies.

4.2.2. Programming languages. From Figure 3, C++ and Java are the most common programming languages supported in open source simulator projects. This outcome is consistent with the findings of other related works.¹⁷ C++ is supported by 10 simulators, while Java is

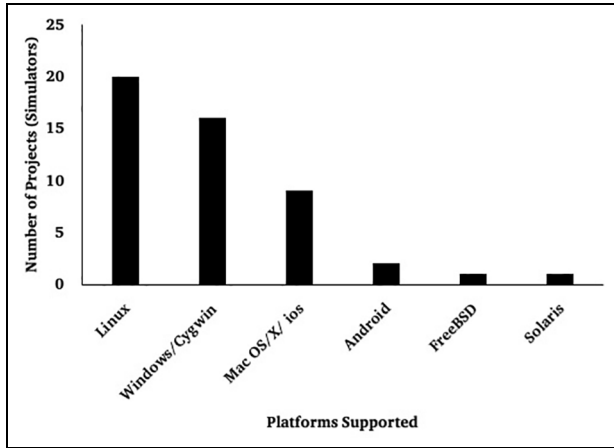


Figure 3. Programming language.

supported by eight. The two languages sum up to 58.06% of the aggregate programming languages support by the sample simulators. Some of the simulators indicate support for multiple programming languages. For example, NS-2¹⁸ supports C++ and Python, Proview⁴⁴ provides implementations for C, C++, and Java, as does Kaa,⁴⁵ while SimGrid⁴⁶ provides support for C, Java, and Ruby.

The preferences for C++ and Java by open source simulator researchers and developers may be linked to the performance speed of the two programming languages. Other authors¹⁷ thought the same. Comparing the execution speed of programming languages, C, C++, and Java are shown to be faster-executing than others, and C++ is especially faster than Java.⁴⁷ Although C programming language may be thought of as slightly faster than C++ and others,⁴⁸ it presents a usability challenge in that it only provides a CLI. Moreover, C is often viewed as being difficult to learn and use, hence non-user-friendly.⁴⁹ This can be an issue in the industrial environment where operators might not necessarily be expert programmers, nor be open to the rigors of learning and interpreting code instructions. This makes C limited and may also explain why it currently has a marginal role in software developments in general, and especially simulation projects. The simplified, more intuitive, GUI-oriented C++ is often preferred. Python and C# are supported by three and two simulators, respectively. Visual Basic .NET, Ruby, FORTRAN, Objective-C, and Modelica all have one simulator application each.

4.2.3. License types. From Figure 4, the GPL (General Public License) open source distribution type is the most widely adopted by the sample open source simulators, with 47.62% adoption (10 simulators). This is followed by the LGPL (Lesser General Public License) type in three

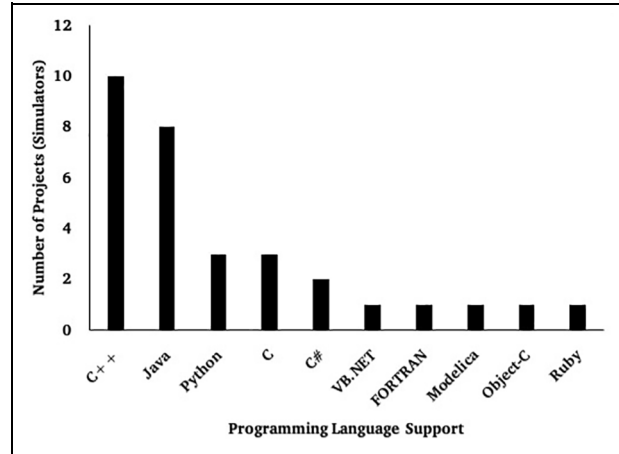


Figure 4. License distribution type.

simulators. Different versions of each of these two licenses seem to have been adopted. Both license types provide a good way to freely access and use open source simulators; however, the licenses enforce restrictions that deter modification and re-distribution of open source software in a proprietary manner. This phenomenon raises significant concerns that can discourage industrial sectors from adopting them.¹⁷ It suffices that most applications of open source simulators may well be tailored toward research and development studies, and a few tailored to real industrial infrastructure applications. Other open source license types found in use include Apache License (AL), Berkeley Software Distribution (BSD), and Apache Software License (ASL), all having two simulators each. Consumer Credit Act (CCA) and Open Source Modelica Consortium Public License (OSMC-PL) are seen in a single simulator each.

One simulator – OpenModelica – supports multiple licenses: GPL v3 and OSMC-PL. OSMC-PL is more of a customized license, allowing for proprietary extensions to be licensed under different conditions to the core OpenModelica code. A multiple license phenomenon can occur when a simulator is covered by a license different from the operating system it supports. Most often, the platform license would have to be satisfied as well. As noted by Fogel,⁵⁰ a typical issue with customized licenses is that developers struggle to clearly determine if such customized licenses are indeed compatible with other popular open source licenses, such as GPL, to allow for free integration with associated license (GPL) applications. The defined open license type CD++ could not be determined from documentation, and the assumption is that the simulator might have a kind of open source license definition that is not easily available to developers. Again, this type of scenario is not supportive of the open source

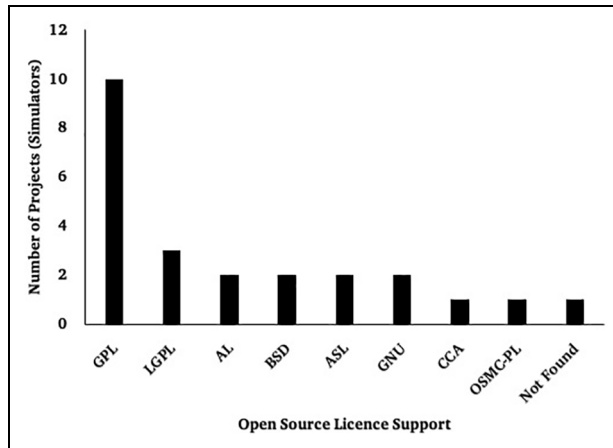


Figure 5. Operating system support.

principle, as interested developers should be able to obtain, without difficulty, relevant license details for open source projects for the purpose of driving improvements.¹⁷

4.2.4. Operating systems. Most (20, representing 95.23%) of the open source simulation tools had support for Linux operating system platforms. Windows was supported by 19 simulators, and Mac OS/X/iOS had nine simulators. Other platforms, such as Android, FreeBSD, and Solaris, had less support, as shown in Figure 5. Predictably, a relationship exists between the choice of programming language for open source simulators and the license distribution type. Linux is developed based on C and C++ programming languages,⁵¹ and is made available free for download on an open source license. Undoubtedly, a common language of development can foster easy and fast interaction between simulator and platform. Moreover, simulation systems often involve complex and multi-tasking functions. Linux seem to address multi-tasking well, and is a more flexible platform for wider infrastructure and application compatibility. It also provides a better capability for runtime error management compared to other platforms. From the developers' perspective, Linux has been judged above other platforms in providing greater convenience, capability, security, interface, and recovery.⁵² Developers view Linux to be friendlier⁵³ and not requiring updated hardware resources – it can run in, or interact with, older hardware environments without significant lags. It is a better lightweight system that is more robust in terms of crashes, and has better capabilities for security.⁵⁴ Above all, it is inexpensive to own and use, with free regular updates, features that come not without expense in other platforms, such as Windows and Mac operating systems.

4.2.5. User interface support. The debate on user-friendly usage preferences between GUIs and CLIs is still open in

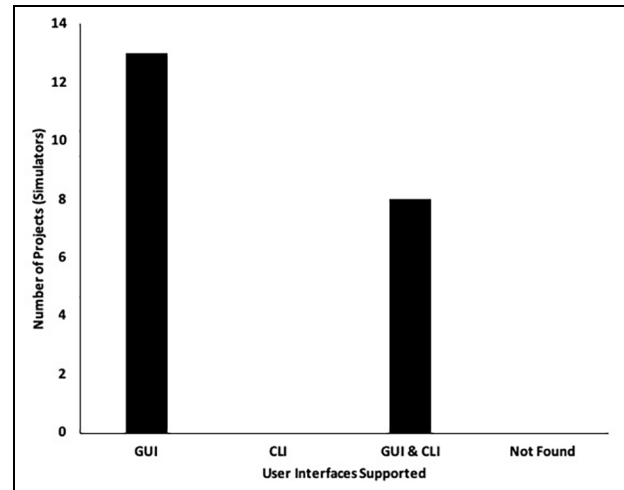


Figure 6. User interface support. GUI: graphical user interface; CLI: command-line interface.

the computing community. Expert developers seem to prefer, support, and adopt CLIs rather than GUIs in view of a faster execution and easier, shortened, and more consistent portability of commands,⁵⁵ with better understanding of deeper interactions amongst command sets.⁵⁶ This is perhaps influenced by the desire for complete control over application functionalities and behavior, an ability that is difficult to achieve to a similar degree for GUIs. This contrasts with the thoughts from non-expert users (novices) who seem to get more done better using GUIs,⁵⁵ which is also true for expert users – for speedy completion of more tasks, better accuracy, lesser frustration, and fatigue.^{57,58}

From Figure 6, more than half (13, representing 61.9%) of the simulators strictly had GUI support. More than a third of the simulators (eight, representing 38.10%) supported or combined a form of GUI and CLI. No simulators with sole CLI support were included due to its usability drawback, described earlier. Note that the simulators discarded for falling into this category were either developed or compatible with C++ and Python programming languages, which both have GUI features, yet were not included in the simulator design architectures. Perhaps the purpose and scope of simulator applications, mostly generic, may have influenced the choice of CLIs by designers.

It is important to clarify these design viewpoints so that the necessary trade-offs and balance between GUIs and CLIs can be appreciated when implementing user interfaces for open source simulators. For operational ICSs, GUIs are typically the norm, which is especially significant since the ICS community would not only comprise expert system programmers, but also operations users and analysts. This latter group may not necessarily have the expert skills for coding or interpreting commands needed

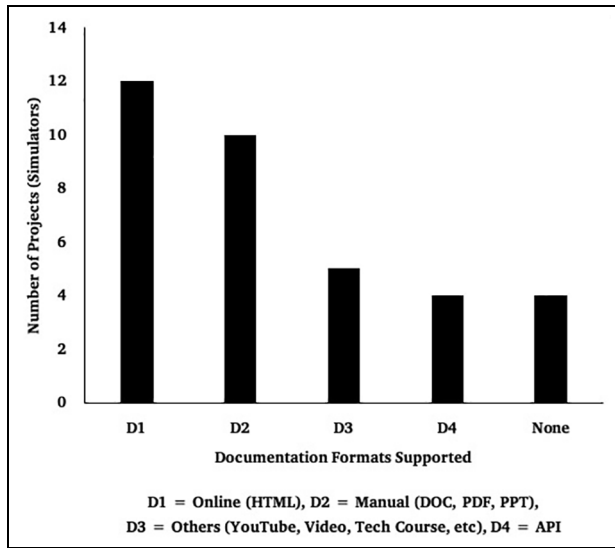


Figure 7. Documentation type support. API: application programming interface.

for a CLI. Thus, whilst expert system programmers may prefer CLI-based systems in simulation contexts, the need to consider operational experience to enhance the credibility of results would suggest that GUI support is also important. In addition, the complexities within the ICS domains call for simulators with more intuitive and user-friendly interfaces better suited to accommodate simpler understanding of the interplay amongst system components and their interactions. This can be better achieved with GUIs than with CLIs. However, regardless of the strengths of GUIs, it is worth emphasizing that the CLI cannot be totally eliminated as it constitutes the foundation and building block of every computing and application platform. CLIs also play a very significant role in the development of GUIs. However, adding GUIs introduces a significant capacity for shortcuts, such as using a click to activate actions that would ordinarily be achieved using multiple lines of text commands on a CLI. GUIs also bring the benefits of visualizing results and logs, simplifying usability, and improving understanding and interpretation. These are quite significant in the evolving domain of big data-driven industrial networks and Internet.

4.2.6. Documentation. From Figure 7, the results of evaluating the documentation modes of open source simulators showed that 12 (57.14%) of the open source simulation projects used online Hypertext Markup Language (HTML) formats for their documentation, 10 projects (47.62%) incorporated manuals in varied formats, for example, DOC, PDF, and PPT, for documenting their development and usability guides, five projects used other

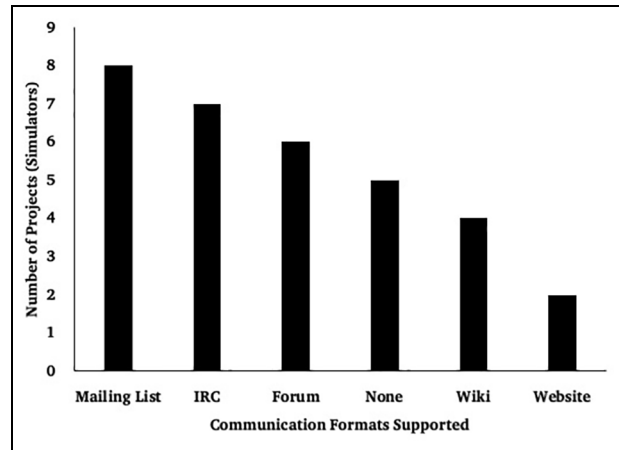


Figure 8. Communication types. IRC: Internet Relay Chat.

unconventional but interactive forms, such as videos and technical courses, while four projects used application programming interfaces (APIs). However, we were unable to identify any dedicated form of documentation for the other four open source projects (PowerDEVS, PySimulator, Cooja, and SCADASim). Arguably, good documentary support is a significant feature that can affect the effective use of open source simulators. Documentation that provides guides to installation, use, and maintenance of simulators can help free users from depending on developers to answer or resolve even the most minor issues or errors. This, in turn, lowers the support and cost pressure on the project developers.⁵⁹ The availability of helpful documentation in the form of user manuals and tutorials can greatly assist users in learning how to correctly use or apply the features of open source simulators and to load updates. This can significantly save time spent on troubleshooting and error management.

4.2.7. Communication. From Figure 8, for communications for open source projects, results showed that Mailing Lists topped the list of approaches employed, with eight projects. This is closely followed by Chat (Internet Relay Chat (IRC)) with seven projects, Forums with six projects, Wiki with four projects, and Websites with two projects. Again, it is surprising to find that some projects did not have any dedicated means for communications. Presumably, such projects may have very low community interests 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 uses of the simulator. We agree with earlier depositions⁵⁹ that the continuous availability and release of project updates, version improvements and modifications, support for functional issues, and maintenance information

are key services that should come bundled with open source simulation projects. Not having these can make an open source project unattractive to user communities.

4.2.8. Version support. Version support typically shows how the open source simulation projects are being managed in terms of successive improvements and product updates.⁶⁰ Analyzed results showed that 10 of the simulators used “Sub-version”; six projects used Git, two used mercurial support for updating their projects, while one simulator used “CVS” (see the Appendix). We were unable to clearly determine the existence of, and version support type, for five simulator projects.

4.2.9. Simulation modes. Analysis based on simulation mode showed that 10 open source simulation tools supported only discrete event simulation (DES) processing modes. Seven simulators cover both DES and CS (continuous simulation), for example, DWSim,⁶¹ OpenModelica,⁶² and Cooja⁶³ supported both DES and CS. Other uncommon simulation modes captured by simulators include TDS (trace-driven simulation) in one simulator, agent-based simulation (ABS) in one simulator, dynamic simulation (DS) in one simulator, and Discrete Event System Specification (DEVS) in two simulators (see Table 2).

In the industrial domain, discrete event processes also characterize 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 a current batch is completed.⁶⁴ This is the situation in most current non-continuous industrial processes, and differs from the emerging continuous time-step event simulations where processes are modeled continuously – no waits are 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, there remains a contrasting argument in favor of domain-specific DES environments, which are said to facilitate easier model development.⁶⁴ Thus, a trade-off looms and opens the debate between the depth of domain-specific application and the width of a generalized application area for industrial environment simulators.

4.2.10. Times cited. The results of analyzing all 21 open source simulators based on “Times Cited” (shown in Figure 9) reveal the popularity of these open source simulators amongst the academic and industrial communities of users. NS-2 (683 times cited), OMNET++ (626 times cited), and NS-3 (338 times cited) simulation-related works were the most cited amongst the set of simulators. Not in exactly the same ranking order, the same simulators, NS-2 (1384 articles), OMNET++ (549 articles), and NS-

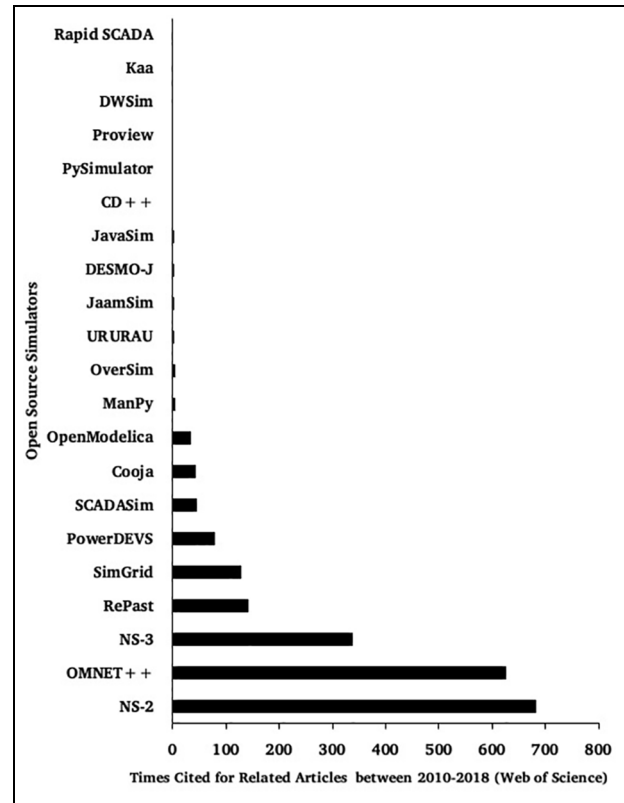


Figure 9. Ranking based on most cited articles.

3 (597 articles), recorded the greatest number of research-related article publications amongst all the simulators (see Table 2), also within the review parameters earlier described. These results suggest that these three simulators, NS-2, OMNET++, and NS-3, have the widest interest and utilization by the open source community. From Figure 9, based on the “Times Cited” measure, it appears that NS-2 was more widely utilized than the other two simulators. In the same utilization scale, successive open source simulators ranked based on the “Times Cited” criteria include OMNET++, NS-3, Repast, SimGrid, PowerDEVs, SCADASim, Cooja, OpenModelica, and Manpy, in that order.

5. Discussion

The findings presented are discussed relative to the outlined research objectives earlier presented in Section 1.

For the common open source simulation tools with industrial capabilities used in networks, ICS, and IIoT research and development projects, evidently, there are many open source simulators that handle the range of applications. This is a positive thing, as it means that developers and researchers have several options for the choice of open source simulator, depending on their

application area. Having simulators categorized for specific purpose-based application groups means that simulators within each class may replicate attributes and functions within their respective fields 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.

It is useful that a good number of the open source tools support generic contexts. This enables the potential for developing architectures and infrastructures with expandable features into multi-class domains. It is beneficial to have such infrastructures as they can enable easier and faster modeling and simulation of complex systems or functionalities. Essentially, a flexible and modular-capable open source simulator infrastructure presents a viable solution to achieving such simulation capacity.¹⁷ Open source simulators that claim to have generic architecture models should be able to present or demonstrate capabilities for such expansion and functionality features. These need to be clearly identified and classified.

Unlike the generic group, simulators within the communications and WSN domain (NS-2, NS-3, OMNET ++, and OverSim) have varied levels of expanded capabilities within their feature landscape. These simulators also enjoy huge interest from the scientific community, as evidenced from analyzed results. This is good for IIoT simulation for a couple of reasons, firstly, the realization that communications and sensor networks are crucial areas of interest and knowledge needs for developers. Secondly, communications and sensor networks also form part of the IIoT development trend presently experiencing great expansion. Thus, these simulators can provide a good understanding of the possible behaviors of similar sub-architectures within the larger IIoT. These also apply to the ICS/SCADA group of simulators. SCADASim appears to have a growing community of users besides its authors.¹⁶ In other domains, such as IoT/IIoT and general operations, with fewer external contributors, utilization appears to be limited and mostly originates from the authors of the respective open source simulators. Presumably, this is either connected to the growth trends in these areas or the open source simulators themselves. At least this is true for the IoT/IIoT domains, as developments in these domains are not as old as that of wireless/sensor networks.

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 might be considered more representative of deployed systems, but they do not purposefully address security. Unfortunately, these open source tools are not yet popular. Their proprietary counterparts, such as AWS, Google

Cloud IoT, ThingWorx, Microsoft Azure, Cisco IoT Cloud, Mindsphere, and Bosch IoT, have features that can allow trials and simulations and are seemingly attracting wider attention (and creating impact) currently. The lack of popularity for the use of open source IoT/IIoT simulators may be because the domains are currently still immature, and investigations and explorations on how effective and standardized these can be are still on-going. The concept of simulating IoT/IIoT is also an idea in formation and subject to testing, and a common architecture is yet to be achieved.⁶⁵ Only Cooja appears to have gained some attention from open source users, but it is expected that a steady growth in interest and utilization should happen in the future. This is also true for counterpart tools, such as Proview and KAA, which promise quite remarkable capabilities for simulating IIoT features.

Having a community of developers and users is crucial for open source simulators, but, it is not sufficient on its own to guarantee that the necessary fidelity is achieved by open source simulators. A community of developers without properly controlled engagements can adversely impact the open source mission. This is especially the case when user interests and contributors begin to withdraw their support due to negative and discouraging behaviors in the community.⁶⁶ Notwithstanding this, in the multitude of code/design contributors, there is great potential to achieve significant bug reduction and better systems structures and capabilities.¹⁷ It is usual to make simplifying assumptions in any simulations, but if those simplifications are not well articulated or understood, then the chances are that codes and design structures will be wrongfully or harmfully reused. This can be 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 work of no practical value.

In addition, since the industrial domain is chiefly characterized by a more complex combination of user-agents with varied expertise, it is pertinent to have open source simulators that can attract varied developers and users. This can be achieved by quality coding that is readable, and an ability to disseminate and effectively manage open source projects.¹⁷ Several of the reviewed open source simulators seem to lag in following valuable open source development philosophies that can drive success. Some of the essentials for achieving this are presented in a documented guideline.⁵⁰ Consequently, a critical community and appropriate editorial control are essential precursors to the development of credible open source simulation

platforms. At present, much still needs to be done in reaching this point for application domains like IIoT, where complex feedback systems with uncertainties are common.

On the *functional structures and characteristics (including purpose, operating system support, usage license, and degree of credibility) that encompass identified Open Source simulators*, some earlier studies tried to address this with pointers to some relevant characteristics. Most frequent are reviews of WSN simulators. Key functional structures and characteristics appearing common and related to other purpose-driven domains (e.g., ICS/SCADA, Operations, etc.) include license 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 specific open source simulator(s). However, these need to be carefully considered, selected, and well-described together with their relevant and common attributes. From a security perspective, other important characteristics that need consideration include the existence and number of vulnerabilities in open source simulators and, possibly, the availability of potential fixes/updates. This is especially crucial in this era of interconnectivity and federations of simulation systems because, as such, learning about simulators can guide appropriate decisions and choices based on available complementary security capacities. Vulnerabilities in open source tools (including simulators) have been well-acknowledged as issues of concern, and have been ranked above functionality and licensing characteristics.⁶⁷ Thus, including vulnerability checks as part of the prioritization criteria for open source simulators can help developers and users to timely respond to critical security issues, and to meet security objectives while building or executing the simulators.

In terms of *how widely the open source simulators are used*, we reason that understanding the level of open source usability or how widely the simulators are used by the open source community also brings to light the associated potential for future relevance and consistency development of the simulator. For example, for additional reasons, NS-2, OMNET ++, and NS-3 are still viewed as relevant aside from having common standard open source features,⁵⁰ because of how widely and consistently the simulators are currently used for simulation studies and research and test purposes. In particular, OMNET ++ and NS-3 simulators are being updated quite frequently following the works of contributors that span different areas of applications. NS-2 lags behind those two with its last update dating back to November 2011. Subsequent utilization of these open source simulators often builds upon

prior works, reports, and recommendations, especially concerning real-world applications. Logically, most open source simulators seem to originate from research institutes, and then strive to penetrate the academic and industrial environments.¹⁷ This is expected to translate to applications not restricted to educational domains but extended to real/industrial environments. Undoubtedly, clear indications of a widely interested and contributing community for open source simulators, especially from a publication perspective, suggests success stories on the research use of the simulators as well as their applications in real-world applications.

In terms of *the kind of research that has built on reviewed open source simulators (e.g., operational performance, optimizations, or security analysis)*, we find that the strengths of open source simulators vary according to the purposes for which they can be used. However, most purposes appear to tend toward performance analysis and optimization. For example, the strengths of open source simulators within the communication and sensor network group are generally characterized 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 observed 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 of users/developers and restricted capabilities in demonstrating aspects of the 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.

On determining if *any of these simulators been designed with capability for security analysis*, we find that very few (three) of the open source simulators provide positive answers to these questions. Only JavaSim,⁶⁸ SCADASim,⁶⁹ and KAA⁴⁵ simulators were found to have characterized inbuilt capabilities for some security modeling and analysis. In particular, SCADASim is said to be specifically developed for security-related analysis for ICS/SCADA systems. However, it is noted that some of the simulators that handle functional and operational performance, (e.g., those in the communications and WSNs group, such as NS-2, NS-3, OMNET ++) are typically used to simulate and measure quality of service (QoS) metrics – bandwidth, throughput, latency, jitter⁷⁰ – within certain predefined configurations. These metrics can be impacted by security-related issues. To incorporate and evaluate security functionalities or features, open source

simulators in this category typically require the creation or modification of packet formats.⁷¹

Security features are not normally automatically built into the simulators. For example, Hegde and Manvi⁷¹ 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 ad hoc and sensor networks are typically characterized by resource-constrained nodes relative to power, computational capacities, memory, and communications bandwidth,^{72,73} other aspects of security considered often include determining the impact of certain security implementations on the QoS parameters. An example includes the work of Aldosari et al.,⁷⁴ involving the development of an independent single security layer in NS-2 to manage the majority of security mechanisms distributed over other network layers on an IoT use case. In Karare et al.,⁷⁵ a collaborative approach is presented for improving QoSs in WSNs by using SAFEQ and the Watchdog algorithm. These examples show that service-oriented security features can be studied and analyzed within some contexts related to the 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.

In terms of *the aspects of security analysis and modeling that have been explored using these open source tools and the aspects that can be explored for future (further) analysis*, we note that the finding related to the former objective indicates that analyzing the impacts of encryption and decryption configurations on normal operations is a common aspect explored with performance-based simulators. Also, more security-tailored tools, such as JavaSim, SCADASim, and Kaa, explore still other security areas. For example, Queiroz et al.⁶⁹ presented and demonstrated the capability of SCADASim for testing and evaluating denial-of-service (DoS), man-in-the-middle, eavesdropping, and spoofing attacks. JavaSim has been used to simulate injection prevention for web applications.⁶⁸ KAA is a relatively new open source simulator for the IIoT, and although the authors claim that the simulator supports security capabilities such as authentication and encryption, there are no independent narratives on the effectiveness of this simulator to replicate the described security features.

It would be interesting to explore other types of attacks, such the black hole, the 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 properly address the broad range of ancillary security simulation objectives. It would seem that these requirements were not considered at the inception of the open source projects. The lack of security-related work in publications related to most of the open source simulators indicates that security has not been a prime issue or interest within the community of open source users and developers. Wider interests and emphases appear to focus on functionality, performance analysis, and optimization. A change is required from performance-only to security-inclusive modeling (architectures) if the on-going deployment of IIoT systems is not to result in a widespread distribution of individual vulnerabilities with exposures to those who would wish us cause harm.

In terms of *how policy and regulations can support the growth and development of open source simulators*, we observe that many of the open source simulators emerge from projects hosted by government-funded research institutes and academic institutions. Some of the projects yield tools made for in-house use or for specific sector applications, often for research and development purposes. If the use and utility of open source simulator software is to improve in the UK, especially for security-related IoT/ICS modeling, the UK government can make an important contribution by shaping the conditions under which decisions about open source simulation tools are used and, also, through leading by example. This translates into a form of influence that could be used strategically to encourage more widespread adoption of these tools as well as establishing a culture of sharing lessons learned with regard to the implementation and management of these tools.

The use of open source tools in government already has global traction.⁷⁶ In the UK, for example, the general use of open source tools, including in security contexts, research, and development, has seen some active promotion.⁷⁶ While most of the focus has been on other tools and applications, extending this thinking explicitly to open source simulators should have established resonance.⁷⁷ There are two compelling reasons why this makes sense for nation-states. Firstly, the uptake of open source tools generally (and simulators, specifically) can lower the cost of security, which is an important consideration for governments trying to cope with shrinking internal budgets as well as managing the private ownership of critical national infrastructure. Secondly, engaging the open source community on security challenges generates benefits beyond particular applications, as it gains strength through critical mass.

In terms of the cost advantages, many open source projects result in tools made for in-house use or for specific sector applications. The benefit of these implementations is not necessarily shared as widely as it could or should be. By encouraging the use of open source simulators, and the dissemination of implementation and modeling findings, costs can be reduced for other users in the same

sector or in allied sectors. Policy initiatives that incentivize the dissemination of implementation experiences and findings could be considered along the same lines as research “impact” in academic and research institutions within countries (e.g., UK) as a way of stimulating concerted efforts at sharing research finding with a broader range of user communities.

While the access to state-of-the-art tools by a wider community of critical infrastructure users will quickly offset the cost of owning and maintaining such tools in a commercial arrangement, there are also cost benefits associated with better security that apply more broadly – although they may not always be directly quantifiable. Taking the UK example, the growth of the digital economy and the stated goal of being the “safest place to live and do business online”⁷⁸ is a central element of the UK’s 21st century global identity. Cyber security, in this context, and its relevance for policy innovation, takes on a significance greater than simple project costs. At this level, broader, more cost-effective simulation of critical infrastructure systems extends to the level of national security and national interest. In this longer-term view, allowing modifications without license restrictions leads to immediately shared, tested, peer-reviewed innovation instead of waiting for the provision of solutions from a commercial provider with a viable business case. This leads to a second, related rationale for governments in encouraging the adoption of open source tools (including simulators). Doing so galvanizes a large community of collaborators as opposed to relying on a relatively small team of developers. This improves the quality of simulation and modeling results, defined by good testing and code reviews.⁷⁹ It also helps to reduce the operational risks of simulation while supporting common solutions to common problems and encouraging innovation. The pace of change and the demands for innovative and secure development of IoT systems means that, where possible, good practice and insights must be reused or replicated within and across sectors to avoid the unnecessary duplication of investment and effort.^{76,80}

Of course, applications and projects vary widely, and decisions about the suitability of open source options need to be carefully considered in each case. While some actors prefer commercial solutions because of the support contracts, our research finds that there are generally open source simulators available that provide the same efficiency and functionality but at a much lower cost. If the choice of whether to opt for a commercial or an open source simulator were to be based purely on the suitability for the project, it is most likely that open source simulators would be more widely used and the benefits of a larger community of users would compound due to shared resources. Encouraging this type of decision-making on modeling and simulations tool choices is one place where

government can have a positive influence by advancing the understanding that, where no significant overall cost and vulnerability difference exists between open source and commercial simulators, open source options should be adopted based on the additional advantage of flexibility.⁸⁰ There are several avenues through which governments can promote this. Using the UK use case again, the government is already leading by example in the adoption of open source tools and it can continue to do so with regard to IoT simulators.⁸⁰ This will drive the necessary capacity development into sectors that may currently favor commercial options, helping them to better appreciate the benefits of reusability. It will also demonstrate how to develop open source components that are cost-effective, more easily maintained and, consequently, more secure. Another area where government policy for open source simulators can be helpful relates to considering new business models for the commercialization of government-funded simulator research and development projects and innovative pathways for extracting financial benefits for research that relies upon open source simulators.

In addition to leading by example, there is also considerable influence attached to government procurement. That influence can be strategically directed to encourage greater uptake of open source simulators. For example, it will be useful to ensure that open source simulators are given appropriate consideration in government-funded (research) projects, tool selection, and adoption activities. In the context where government funding is supporting a project, the project leaders should be asked to justify the use of commercial simulation tools as opposed to open source options. Open source simulators should always be considered where they deliver the best value for money and utilization outputs, taking into account the supplementary advantages of flexibility and reusability. Evidence of these considerations could be included in funding contracts.⁸⁰ Where there is no government funding, it would be useful to promote the use of open source simulators as good practice through an awareness campaign focused on the benefits of using open source simulators for even a small part of any project. It is important to note here that the open source community of developers does not always collaborate as successfully as it could, in part because poor communication practices can alienate talented participants. Homogenization in technical communities is a long-recognized weakness and inter-personal and workplace dynamics are cited as a key factor in driving under-represented people out of the sector.⁸¹ There is scope for real improvement here in terms of retaining the most valuable contributors, which will be essential if the open source community is to deliver the kind of diversity necessary for the systems of the future – systems that will incorporate automated decision-making through the IoT, machine learning, and artificial intelligence. Government

could play a facilitating role in coordinating the production of some best practice guidelines or recommendations for effective communications and organizational efficiency that outline expectations of respectful and professional conduct.

Another area where government policy for open source simulators can be helpful relates to the commercialization of government-funded simulator research and development projects. Governments need to be strategic in their backing for open source project research without license restrictions. Essentially, a trade-off would be required between accessibility to state-of-the-art tools by a wider community of critical infrastructure users and the cost of owning and maintaining such tools on a commercial basis. It is not advised that policies should completely refuse support for projects under fully fledged open source licenses, such the GPL or LGPL, when in fact such support can sometimes be aimed at meeting certain service requirements and is ordinarily too expensive to obtain commercially. Policies should also favor governments' promotion of research and development, and the commercialization of research outputs with less strict open source constraints. Adequate awareness is also required concerning the need and value of innovative developments via the use of open source simulations.

6. Conclusion and future work

To engage the wider community of researchers, developers, and users, open source simulators provide an essential foundation since it is possible to conduct work using them readily and cheaply. Despite the long history around open source development for simulators and the huge research efforts undertaken using them, endeavors toward improving use and utility are not be taken lightly. The prospect of having a single open source simulator for the IIoT is relatively overwhelming or infeasible considering the level of complexity resulting from the close interdependence of disparate systems. Based on this study, *we draw attention to a summary of factors and desirable service characteristics to consider in relation to open source simulation tools for the ICS and IIoT*. These can also pass as potential measures for deciding (i) *on development criteria to adopt or (ii) on selecting appropriate open source tools to suit specific security simulation needs*.

To support compatibility with a wider range of simulation tools with respect to data types, formats, and systems, future developers and users of open source simulators may consider more common simulation attributes, for example, Linux platform-compatible simulators, to reduce the cost of ownership, use, and updates. This can also support wider infrastructure compatibility, multi-simulator interactions, robustness in withstanding crashes and errors, and efficient recovery and security. C++, C#, and Python compatible simulators will support faster execution and

performance, user friendliness, and easier learning and use. GPL and LGPL license distribution tools will reduce restrictions on use, modification, and re-distribution. This can also support both academic and industrial uses and driving improvements. GUI and CLI tool support will simplify usability and increase better understanding and interpretation via visualization. This can also ensure better control of functionality and enable multi-interface design capabilities. A diversity of users and support (experts and novices) is also allowed, as well as sustaining an active user community with continuous updates. DES and CS compatible simulators can achieve multiple-domain process application where needed. DES-only or CS-only compatible tools can achieve easy model development for specific single domain-process processes. Document manuals (.doc, .pdf), and online documentation formats will enable easy usability guides for open source simulators, reducing support pressures and total dependence on developers. IRCs, Mailing List, Websites, and Forums can achieve the timely communication of updates, and support timely support and maintenance.

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. For example, consider the case of cyber-attackers exploiting unpatched gateway component vulnerabilities that enable remote connectivity and network compromise of an electricity generating wind turbine farm using a RaspberryPi card with a cellular module. Supposing remote connectivity gives the attacker access to automation programmable logic controllers that allows them to interrupt power generation routed to feed to urban communities. The impacts can include physical, economic, and social dimensions if not addressed and mitigated. Thus, the existence of vulnerabilities and patches is also a key factor that must not be overlooked. Not having a policy related to open source places a system at a competitive disadvantage – positioned behind growing trends and developments driving innovations.

From both ICS and IIoT perspectives, it is obvious that there are open source simulation projects that are exploring the representation of these systems and their environments to a degree that becomes useful for research and application. Even though this is the case, the big challenge of configuring and managing these simulators for the desired (security-related) scenarios 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 the IIoT, be built with the active engagement of experts from interdisciplinary teams along with user contributions that are integrated under tight editorial control. However, 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.


Government has a role to play in incentivizing the take up of open source simulators and it can promote this through a number of mechanisms. Continuing to lead by example through internal projects and also through government-funded projects will help to stimulate an approach to considering open source first, with real cost and security benefits. While every project will need to be assessed for suitability for open source simulation, government funding requirements that ask for justification for using commercial solutions would allow for the necessary flexibility in choice while also building an awareness of the benefits of open source solutions. It may also be worthwhile to consider providing governance structures (controlling, monitoring, and supervision) and guidelines for running and maintaining open source projects they fund or support. This can be very useful if encouraged at the early stages of projects on new and growing trends, such as the IoT and IIoT; it can also provide a pathway toward improving the transparency and credibility of open source simulation outcomes.

Further work in this area will include investigating data models to represent ICS/IIoT systems enabling information integration, management, and simulation. We will further explore in-depth analysis and evaluation of the simulation capacities and fidelity of cloud-based open source IIoT simulators, in relation to security, since most of the tools in this class are not typically developed to address security in the first place. Investigations into simulation frameworks that can allow multi-core or distributed simulations to be configured and operated are also required. At present, there are gaps in the above areas and an exigency in needing to fill them to assure proper understanding, adoption, and effectiveness of open source simulations within ICS and IIoT domains.

Funding

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) (Grant No. EP/N02334X/1).

ORCID iD

Uchenna Daniel Ani  <https://orcid.org/0000-0001-6064-480X>

References

1. Gilchrist A. Introducing Industry 4.0. In: *Industry 4.0: The Industrial Internet of Things*. Bangken, Thailand: Apress, pp.195–215.
2. Lee RM, Assante MJ and Conway T. TLP: analysis of the cyber attack on the Ukrainian power grid. Washington DC, https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf (2016, accessed 13 December 2019).
3. Siemens. Securing industrial control systems. The challenge and common-sense. Germany, https://assets.new.siemens.com/siemens/assets/api/uuid:2b57b21f-c87f-4dec-8368-90333cedd18e/version:1535709643/whitepaper_securityen082018web.pdf (2018, accessed 6 November 2019).
4. Folkert L. Forensic analysis of industrial control systems. SANS Institute, <https://www.sans.org/reading-room/whitepapers/forensics/forensic-analysis-industrial-control-systems-36277> (2015, accessed 19 December 2019).
5. Stirland J, Jones K, Janicke H, et al. Developing cyber forensics for SCADA industrial control systems. In: *proceedings of the international conference on information security and cyber forensics (InfoSec2014)*, Kuala Lumpur, 8 October 2014, pp. 98–111. Kuala Lumpur: SDIWC.
6. Crahmaliuc R. Open-source or proprietary software - what is best for users? SIMSCALE blog website, <https://www.simscale.com/blog/2017/06/open-source-vs-proprietary-software/> (accessed 14 July 2019).
7. Umbraco. What is open source software? *Umbraco online dictionary*, <https://umbraco.com/about-us/umbraco-dictionary/open-source/> (2018, accessed 10 January 2020).
8. Bessen J, Evans DS, Lessig L, et al. Government policy toward open source software. In: Hahn RW (ed.). *Brookings institution book*. Washington, DC: Brookings Institution Press and AEI, 2002, p.124. <https://www.brookings.edu/book/government-policy-toward-open-source-software/>
9. O'Connor A, Ong KW, Sander T, et al. Government policies on open source. Mimeo, <https://courses.cs.washington.edu/courses/csep590/04au/clearedprojects/Ferlo.pdf> (2005, accessed 22 November 2019).
10. Lakhani KR and Von Hippel E. How open source software works: 'free' user-to-user assistance. *Res Pol* 2003; 32: 923–943.
11. Margan D and Candric S. The success of open source software: a review. In: *2015 38th international convention on information and communication technology, electronics and microelectronics, MIPRO 2015 - proceedings*, Opatija, Croatia, 25–29 May 2015, pp.1463–1468. IEEE. <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84946151994&partnerID=40&md5=0d2b83265a6271d713ceeb5936fec9b0>
12. Bajracharya S, Ossher J and Lopes C. Sourcerer: an infrastructure for large-scale collection and analysis of open-source code. *Sci Comput Program* 2014; 79: 241–259.
13. Kabir MH, Islam S, Javed H, et al. Detail comparison of network simulators. *Int J Sci Eng Res* 2014; 5: 203–218.
14. Kellner A, Behrends K and Hogrefe D. *Simulation environments for wireless sensor networks*. Göttingen, Germany:

- Georg-August-Universität Institute of Computer Science, 2010.
15. Egea-López E, Vales-Alonso J, Martínez-Sala AS, et al. Simulation tools for wireless sensor networks. In: Bruzzone AG (ed.) *Summer simulation multiconference – SPECTS'05*, 24–28 July 2005, pp.2–9. Philadelphia, PA: The Society for Modeling and Simulation International (SCS). <http://cite-seerx.ist.psu.edu/viewdoc/download?doi=10.1.1.379.4745&rep=rep1&type=pdf>
 16. Mathioudakis K, Frangiadakis N, Merentitis A, et al. Towards generic SCADA simulators: a survey of existing multi-purpose co-simulation platforms, best practices and use-cases. *Conf-Scoop.Org*, http://conf-scoop.org/ACE-2013/6_Kostas_ACE.pdf (2013, accessed 11 January 2020).
 17. Dagkakis G and Heavey C. A review of open source discrete event simulation software for operations research. *J Simul* 2016; 10: 193–206.
 18. Chhimwal P, Rai DS and Rawat D. Comparison between different wireless sensor simulation tools. *IIOSR J Electron Commun Eng* 2013; 5: 54–60.
 19. Almadhor A. A survey on generic SCADA simulators. *Int J Comput Appl* 2015; 128: 38–43.
 20. Grant MJ and Booth A. A typology of reviews: an analysis of 14 review types and associated methodologies. *Health Info Libr J* 2009; 26: 91–108.
 21. Webster J and Watson RT. Analyzing the past to prepare for the future: writing a literature review. *MIS Q* 2002; 26: xiii–xxiii.
 22. Tranfield D, Denyer D and Smart P. Towards a methodology for developing evidence-informed management knowledge by means of systematic review. *Br J Manag* 2003; 14: 207–222.
 23. Lu Y. Industry 4.0: a survey on technologies, applications and open research issues. *J Ind Inf Integr* 2017; 6: 1–10.
 24. Azadeh A, Shirkouhi SN and Rezaie K. A robust decision-making methodology for evaluation and selection of simulation software package. *Int J Adv Manuf Technol* 2010; 47: 381–393.
 25. Tewoldeberhan TW, Verbraeck A, Valentin E, et al. An evaluation and selection methodology for discrete event simulation software. In: Yücesan E, Chen C-H, Snowdon JL, et al. (eds) *proceedings of the 2002 winter simulation conference*, San Diego, CA, 2002, pp.67–75. USA: IEEE.
 26. Lindman J, Paajanen A and Rossi M. Choosing an open source software license in commercial context: a managerial perspective. In: *proceedings of the 36th EUROMICRO conference on software engineering and advanced applications (SEAA 2010)*, 2010, pp.237–244. Lille, France: IEEE.
 27. Fogel K. How to run a successful free software project - producing open source software. 2nd ed. Karl Fogel, <https://producingoss.com/en/producingoss.pdf> (2005, accessed 14 December 2019).
 28. Bajracharya S, Ossher J and Lopes C. Sourcerer: an infrastructure for large-scale collection and analysis of open-source code. *Sci Comput Program* 2014; 79: 241–259.
 29. Engelfriet A. Choosing an open source licence. *IEEE Softw* 2010; 27: 48–49.
 30. Grant MJ and Booth A. A typology of reviews: an analysis of 14 review types and associated methodologies. *Health Info Libr J* 2009; 26: 91–108.
 31. Liang G, Hou H, Hu Z, et al. Usage count: a new indicator to detect research fronts. *J Data Inf Sci* 2017; 2: 89–104.
 32. de Winter JCF, Zadpoor AA and Dodou D. The expansion of Google Scholar versus Web of Science: a longitudinal study. *Scientometrics* 2014; 98: 1547–1565.
 33. Kendall S. PubMed, Web of Science, or Google Scholar? A behind-the-scenes guide for life scientists. *Res Guid* 2017; 17–18.
 34. Abdelhabib B and Brahim B. JAPROSIM: a Java framework for process interaction discrete event simulation. *J Object Technol* 2008; 7: 103–119.
 35. Belattar B and Bourouis A. Yet another java based discrete-event simulation library. *J Softw* 2014; 9: 82–88.
 36. Titzer BL, Lee DK and Palsberg J. Avrora- scalable sensor network simulation with precise timing. In: *4th international symposium on information processing in sensor networks (IPSN 2005)*, Los Angeles, CA, 25–27 April 2005, paper no. 69, pp.477–482. Boise, ID: IEEE.
 37. Robertson AR and Ibbett RN. HASE: a flexible high performance architecture simulator. In: *twenty-seventh Hawaii international conference on system sciences*, 1994. HI: IEEE Xplore. Epub ahead of print 1994. DOI: 10.1109/HICSS.1994.323165.
 38. Carneiro TG de S, Andrade PR de, Câmara G, et al. An extensible toolbox for modeling nature-society interactions. *Environ Model Softw* 2013; 46: 104–117.
 39. Garrido JM. *Object oriented simulation: a modeling and programming perspective*. New York: Springer, 2009.
 40. Team Simpy. Overview — SimPy 3.0.10 documentation. *Simpy documentation*, <https://simpy.readthedocs.io/en/latest/> (2017, accessed 16 April 2018).
 41. Van Tendeloo Y and Vangheluwe H. An evaluation of DEVS simulation tools. *Simulation* 2017; 93: 103–121.
 42. McInnes A and Thorne B. ScipySim: towards distributed heterogeneous system simulation for the SciPy platform (work-in-progress). Theory of modeling & simulation: DEVS ..., <http://dl.acm.org/citation.cfm?id=2048487> (2011, accessed 17 December 2019).
 43. Levis P, Lee N, Welsh M, et al. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In: *proceedings of the 1st international conference on embedded networked sensor systems*, Los Angeles, CA, 5–7 November 2003, pp.126–137. New York: ACM.
 44. Mandator and SSAB Oxelösund. About ProviewR, <http://www.proview.se/v3/index.php/about-proview-leftmenu-27> (2018, accessed 16 April 2018).
 45. KaaIoT. Kaa IoT overview — IoT development product. *Kaa open-source Internet of Things platform*, <https://www.kaaproject.org/overview/> (2018, accessed 16 April 2018).
 46. Quinson M. SimGrid: a generic framework for large-scale distributed experiments. In: *proceedings of the tenth international conference on computer modeling and simulation*, Cambridge, UK, 1–3 April 2018, paper no. 23, pp.126–131. London, UK: IEEE.
 47. Prechelt L. An empirical comparison of seven programming languages. *Computer (Long Beach Calif)* 2000; 33: 23–29.

48. Nanz S and Furia CA. A comparative study of programming languages in rosetta code. In: Kellenberger P (ed.) *proceedings - international conference on software engineering*, Florence, Italy, 16–24 May 2015, pp.778–788. Los Alamitos, CA: IEEE.
49. Smith B. Object-oriented programming. In: Taylor T and Behr M (eds) *Advanced Action Script 3.0: design patterns*. New York: Apress, 2015, pp.1–23.
50. Fogel K. Participating as a business, non-profit, or government agency. How to run a successful free software project-producing open source software, <http://www.amazon.com/How-Successful-Free-Software-Project/dp/1441437711?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1441437711> (2005, accessed 19 August 2019).
51. Centioli C, Iannone F, Mazza G, et al. Open source real-time operating systems for plasma control at FTU. *IEEE Trans Nucl Sci* 2004; 51: 476–481.
52. Alhassan HA and Bach C. Operating system and decision making. In: *ASEE 2014 zone I conference*, Bridgeport, CT, 3–5 April 2014, paper no. 70. Connecticut: IEEE.
53. Srinivasa KG, Raddi HCS, Krishna MS, et al. MeghaOS: cloud based operating system and a framework for mobile application development. In: Abraham A, Agrawal D, Abraham S, et al. (eds) *2011 world congress on information and communication technologies*, Mumbai, India, 11–14 December 2012, pp.858–863. Mumbai, India: IEEE.
54. Jasiunas M, Chakraborty A and Kearney D. *A distributed operating system supporting strong mobility of reconfigurable computing applications in a swarm of unpiloted airborne vehicles*. Epub ahead of print 2009. DOI: 10.1109/FPT.2009.5377685.
55. Chen J-W and Zhang J. Comparing text-based and graphic user interfaces for novice and expert users. In: *AMIA annual symposium proceedings*, Chicago, IL, 10–14 November 2007, paper no. 20, pp.125–129. Maryland: PMC/PubMed.gov.
56. Olofsson R and Hultstrand S. *Git - CLI or GUI which is most widely used and why?* Karlskrona, Sweden: Blekinge Institute of Technology, 2015.
57. Rauterberg M. An empirical comparison of menu-selection ((CUI) and desktop (GUI) computer programs carried out by beginners and experts. *Behav Inf Technol* 1992; 11: 227–236.
58. Temple B and Sloane I. *The benefits of the graphical user interface: a report on new primary research*. Redmond: Wash, 1990.
59. Arisha A and Baradie MEL. On the selection of simulation software for manufacturing application. In: *nineteenth international manufacturing conference (IMC-19)*, Belfast, Northern Ireland, 28–30 August 2002, pp.495–507. Belfast: ARROW@DIT School, Dublin Institute of Technology.
60. De Alwis B and Sillito J. Why are software projects moving from centralized to decentralized version control systems? In: *2009 ICSE workshop on cooperative and human aspects on software engineering*, Vancouver, BC, Canada, 17 May 2009, pp.36–39. Vancouver, BC, Canada: IEEE.
61. Moudgalya KM. Home | DWSIM. Project website, <https://dwsim.fosee.in/> (2017, accessed 16 April 2018).
62. OSMC. Welcome to OpenModelica - introduction. *Open Source Modelica Consortium (OSMC)*, <https://openmodelica.org/> (2016, accessed 16 April 2018).
63. Mehmood T. 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 & routing protocols. <http://arxiv.org/abs/1712.08303>, 2017.
64. Chen S. *Comparison of batch versus continuous process in the pharmaceutical industry based on safety consideration*. Texas: Texas A&M University, 2017.
65. Lin S-W, Miller B, Durand J, et al. The Industrial Internet of Things volume G1: reference architecture. IIC:PUB:G1:V1.80:20170131, http://www.iiconsortium.org/IIC_PUB_G1_V1.80_2017-01-31.pdf<http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/intelligent-process-automation-the-engine-at-the-core-of-the-next-generation-operating-model><http://www.mckinsey.com/i> (2017, accessed 14 December 2019).
66. Geiger RS. *Summary analysis of the 2017 GitHub open source survey*. Epub ahead of print 2017. DOI: 10.17605/OSF.IO/ENRQ5.
67. WhiteSource. The state of open source vulnerabilities management. Industry research report. New York: WhiteSource, 2018, pp.1–15. <https://www.whitesourcesoftware.com/wp-content/uploads/2018/10/The-State-of-Open-Source-Vulnerabilities-Management-2018.pdf>
68. Larson D, Liu J and Zuo Y. Performance analysis of JavaScript injection detection techniques. In: *IEEE international conference on electro information technology*, Milwaukee, WI, 5–7 June 2014, pp.140–148. IEEE Computer Society.
69. Queiroz C, Abdun M and Tari Z. SCADASim—a framework for building SCADA simulations. *IEEE Trans Smart Grid* 2011; 2: 589–597.
70. Kumar N. Mobile ad hoc network: issue and challenges related to QoS and solutions. *Int J Eng Technol Sci Res* 2017; 4: 415–419.
71. Hegde N and Manvi SS. Simulation of wireless sensor network security model using NS2. *Int J Latest Trends Eng Technol* 2014; 4: 113–119.
72. Asif M, Khan S, Ahmad R, et al. Quality of service of routing protocols in wireless sensor networks: a review. *IEEE Access* 2017; 5: 1846–1871.
73. Dorri A and Kamel SR. Security challenges in mobile ad hoc networks: a survey. *Int J Comput Sci Eng Surv* 2015; 6: 15–29.
74. Aldosari HM, Snaesl V and Abraham A. A new security layer for improving the security of internet of things (IoT). *Int J Comput Inf Syst Ind Manag Appl* 2016; 8: 275–283.
75. Karare AR, Sonekar SV and Akanksha K. Improving the quality of services in wireless sensor network by improving the security. In: *international conference on industrial automation and computing*, Nagpur, 2014, pp.43–47. Nagpur, India: Jhulelal Institute of Technology. <https://pdfs.semanticscholar.org/a22f/0089845b36aa07cc3b60e8902745f07e0e31.pdf>

76. Lewis JA. Government open source policies. Government Survey Report, https://opensource.org/files/100416_Open_Source_Policies.pdf (2010, accessed 12 December 2019).
77. Gent T. *The government's role in open source*. Home office blog post, Home Office Digital, Data and Technology, 2016.
78. Department for Digital Culture Media & Sports (DCMS). *A safe and secure cyberspace - making the UK the safest place in the world to live and work online*. London, UK: Department for Digital, Culture, Media & Sport, 1 March 2017. <https://www.gov.uk/government/publications/uk-digital-strategy/5-a-safe-and-secure-cyberspace-making-the-uk-the-safest-place-in-the-world-to-live-and-work-online>
79. Shaw J. 6 Benefits of using open source software in government (industry perspective). Government Technology (gt) Magazine, <http://www.govtech.com/opinion/6-Benefits-of-Using-Open-Source-Software-in-Government.html> (2016, accessed 14 December 2019).
80. The Cabinet Office. Open source, open standards and re-use: Government action plan. Policy Paper: An open source strategy for government. London, UK: The Cabinet Office, 27 January 2010, pp.1–9. <https://www.gov.uk/government/publications/open-source-open-standards-and-re-use-government-action-plan> (accessed 11 December 2019).
81. Reed J and Acosta-Rubio J. *Innovation through inclusion: The multicultural cybersecurity workforce*. Santa Clara, CA: Frost & Sullivan in Collaboration with ICMCP, 2018, pp.1–10. <https://www.isc2.org/-/media/Files/Research/Innovation-Through-Inclusion-Report.ashx> (accessed 11 December 2019).

Appendix Characterization of open source simulators (part 2).

Simulators	Documentation	Communications	Version Control	Website	Type	Development status
OMNET ++ JaamSim NS-3 DESMO-J	PDF, online Manuals DOC, online, API, PDF PDF	Mailing list Forum Wiki, IRC None/not found	Subversion Git Mercurial Subversion	https://omnetpp.org/ http://jaamsim.com https://www.nsnam.org http://desmoj.sourceforge.net/download.html	Simulator Simulator Simulator Simulator	Active Active Active Active
PowerDEVS	None/not found	None/not found	Subversion	https://sourceforge.net/projects/powerdevs/files/?source=navbar	Simulator	Active
JavaSim	DOC, online, API, PDF	Mailing lists	None/not found	https://github.com/nmcl/javaSim/commits/master	Simulator	Active
CD ++	DOC, videos	None/not found	None/not found	http://cell-devs.sce.carleton.ca/mediawiki/index.php/Main_Page	Simulator	Active
RePast URJRAU	Online YouTube videos	Mailing lists None/not found	CVS Mercurial	https://repastr.github.io/download.html https://ururau.ucam-campos.br/download/	Simulator Simulator	Active Active
SimGrid OverSim	Online Online	Mailing list, IRC Mailing list	Subversion Git, subversion	http://simgrid.gforge.inria.fr http://www.oversim.org/wiki/	Simulator Simulator	Active Active
PySimulator	None/not found	None/not found	Git	https://github.com/PySimulator/OverSimDownload	Simulator	Active
Cooja	None/not found	IRC, mailing list, website, Wiki	Git	https://anrg.usc.edu/contiki/index.php/Cooja_Simulator	Simulator; emulator, real	Active
Proview	Online	Forum, Wiki	Subversion	http://proview.se/doc/en_us/qguide_f.html	Emulator, real	Active
DWSim	Online, API, YouTube	Forum, mailing lists	Subversion	http://dwsim.informs.com.br/wiki/index.php?title=Main_Page	Simulator	Active
OpenModelica SCADASim	Online, book, PPT tutorial None/not found	Forum, mailing lists IRC	Subversion Git	https://openmodelica.org https://github.com/caxqueiroz/scadasim	Simulator Simulator; emulator, real	Active Active
ManPy Kaa	PDF Online	IRC Forum, IRC, blog	Git Subversion	http://www.many-simulation.org https://www.kaaproject.org/industrial-automation/	Simulator Simulator; emulator, real	Active Active
Rapid SCADA	Online, YouTube, PDF	Forum, website	Subversion	https://rapidscada.org	Simulator; emulator, real	Active
NS-2	DOC, online, API, PDF	Wiki, IRC	Mercurial	http://nsnam.sourceforge.net/wiki/index.php/User_Information	Simulator	Active

Author biographies

Uchenna Daniel Ani is currently a Research Fellow with the PETRAS National Centre of Excellence for IoT Systems Cybersecurity, Department of Science, Technology, Engineering and Public Policy (STeAPP), University College London (UCL). He holds a PhD in ICS Cybersecurity. His research focuses on ICS and IoT cybersecurity modeling, simulations, and risk analysis, digital forensics, and data-driven cyber security with applications to manufacturing, transportation, and energy systems critical infrastructures.

Jeremy Watson is Professor of Engineering Systems and Director/Principal Investigator, PETRAS National Centre of Excellence for IoT Systems Cybersecurity, Department of Science, Technology, Engineering and Public Policy (STeAPP), University College London (UCL). He is a Chief Scientist and Engineer at BRE and Former President of the Institution of Engineering and technology (IET). He holds a PhD in Synthesized Speech Aids for the Vocally Handicapped. His research focuses on interactions in, and the design of, socio-technical critical infrastructure systems, cybersecurity (including cyber-physical) emerging technology identification, development and deployment, and strategic innovation processes.

Madeline Carr is a Professor of Global Politics and Cybersecurity, Department of Science, Technology,

Engineering and Public Policy (STeAPP), University College London (UCL). She is also the Director of the Research Institute for Sociotechnical Cyber Security (RISCS), and Co Investigator at PETRAS National Centre of Excellence for IoT Systems Cybersecurity. She leads the Digital Policy Lab at UCL, which supports policy making to adapt to the pace of change in society's integration of digital technologies.

AI Cook is the CEO of Critical Insights Limited UK, an information technology consultancy company. He has research interests in critical infrastructure security modeling and simulations.

Jason RC Nurse is an Associate Professor (Senior Lecturer) in Cyber Security at the School of Computing at the University of Kent, and a Visiting Academic at the University of Oxford. His research focuses on the interaction between users and aspects of cyber security, privacy, and trust across the broader spectrum of modern technologies in use today. His interests encompass topics such as security and privacy in the IoT, usable security and security awareness programs in organizations and for the public, technical and psychological aspects of cybercrime, identity security, and privacy risks in cyberspace. He holds a PhD in Computing from the University of Warwick.