Assessing the Quality of Low-Code and Model-driven Engineering Platforms for Engineering IoT Systems

Felicien Ihirwe, Davide Di Ruscio, Simone Gianfranceschi, Alfonso Pierantonio

Assessing the Quality of Low-Code and Model-driven Engineering Platforms for Engineering IoT Systems

Felicien Ihirwe[1,2,*], Davide Di Ruscio[1], Simone Gianfranceschi[2], and Alfonso Pierantonio[1]
[1]Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, L'Aquila, Italy
[2]Innovation Technology Services Lab, Intecs Solutions S.p.A, Pisa, Italy
[2]{felicien.ihirwe,simone.gianfranceschi}@intecs.it
[1]{davide.diruscio,alfonso.pierantonio}@univaq.it
*corresponding author

***Abstract*—Over the last few years, industry and academia have proposed several Low-Code and Model-driven Engineering (MDE) platforms to ease the engineering process of the Internet of things (IoT) systems. However, deciding whether such engineering platforms meet the minimum required software quality standards is not straightforward. Software quality can be defined as the degree to which a software system achieves its intended goal. Various software quality standards have been established to aid in the software quality assessment process; however, due to the nature of engineering IoT platforms, such models may not entirely suit the IoT domain. This paper presents a model for assessing the software quality of Low-Code and MDE platforms for engineering IoT platforms. The proposed software quality model is based on and extends the ISO/IEC 25010:2011 software product quality model standard. It is intended to assist IoT practitioners in assessing and establishing quality requirements for engineering IoT platforms. To determine the effectiveness of the proposed model, we used it to evaluate the quality of 17 IoT engineering platforms, and the results obtained are promising.**

Keywords*—*Software quality; Low-code Development platforms; Model-driven Engineering; Internet of Things

I. Introduction

The Internet of Things (IoT) systems offer enormous benefits in our daily lives by enabling seamless communication with our surroundings. Such systems demand many development skills, from handling tiny microcontrollers to more extensive and complex cloud-based systems. In the IoT domain, systems often include safety-critical tasks that, if mishandled, might have disastrous consequences and even cost human lives [1]. To guarantee robustness and safety during operation, it is critical to investigate the correctness and the quality of the platforms and the process by which systems are developed.

Over the last years, both academia and industry have proposed novel languages and tools to support the engineering of IoT systems. To this end, MDE and Low-Code paradigms are employed to conceive engineering platforms specific to the IoT domain. In the software engineering process, MDE promotes the use of models as first-class citizens in software development [2]. Its goal is to improve productivity and reduce time to market by allowing the development of systems using models defined with concepts that are much less linked to the underlying implementation technology and much closer to the problem domain [3]. Low-Code Development Platforms (LCDPs) are cloud-based software development platforms that leverage a Platform-as-a-Service paradigm to enable users with little or no programming skills to create fully functional apps with dynamic graphical user interfaces [4]–[6]. For the sake of readability, we use *IoT engineering platforms* when we need to refer Low-Code and MDE platforms indistinguishably.

Deciding whether an IoT engineering platform meets the required standards for adoption in terms of quality is not a straightforward process as it involves considering and exploring various sources of information. MDE and low-code development technologies frequently rely on rigorous verification and validation processes conducted under predefined arbitrary constraints to guarantee the quality of generated systems. However, in most cases, the quality of the employed engineering is not taken into account seriously or is overlooked [7], [8].

Practitioners typically rely on well-established standards and practices to improve confidence in whether a system or a product fits the wanted quality requirements. The ISO/IEC 25000 to ISO/IEC 25099 series of International Standards, titled *"Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQUARE)"*, aims to address issues concerning software quality requirements specification and evaluation [9]. The ISO/IEC 25010:2011 standard, in particular, [10], is intended to help people who are interested in developing or purchasing systems and software products to specify and evaluate their quality requirements.

This paper presents a quality model for assessing the quality of IoT engineering platforms based on the ISO/IEC 25010:2011 standard. In doing so, we enhanced the standard's product quality model by defining a product quality assessment model that is more suitable for the IoT domain. Initially, the ISO/IEC-25010:2011 standard, which replaced ISO/IEC-9126-1:2001 standard, defines two main sets of quality models, namely ***"quality in use model"***, and ***"product quality model"***. The former is related to the outcome of interaction when a product is used in a specific context. In contrast, the latter is related to software platforms' static and dynamic properties. These models are defined in terms of characteristics, with some of them further subdivided into sub-characteristics [10]. Both models are equally crucial for thoroughly assessing the quality of a given platform. However, the proposed model focuses on the *product quality model* since it emphasizes on platform's technical quality rather than usage quality which can be challenging to measure for platforms in their early

development stages.

To evaluate the effectiveness of the proposed model, we employ it to assess the quality of 17 IoT platforms selected from our previous studies [11], [12]. We present the methodology we used to choose such platforms, perform the quality assessment, and subsequently present and discuss the obtained results. We summarize this paper's contribution as follows:

- We propose a quality model based on the ISO/IEC 25010:2011 standard for evaluating the quality of IoT engineering platforms;
- We assess the quality of 17 IoT engineering platforms by relying on the proposed model;
- We present and discuss the findings as well as the limitation of the study in order to validate the effectiveness of the proposed model.

The remaining of the paper is structured as follows: The background of the study is presented in Section II. The proposed product quality model is presented in Section III. In Section IV we go through the selected primary studies and present the followed evaluation process. In Section V, we present the results from the conducted assessment by reflecting specific research questions. Section VI discusses the results as well as its limitations. In Section VII, we present the related work and Section VIII concludes the paper and discusses perspective future work.

II. Background

A. IoT Engineering Platforms

The main goal of MDE is to increase productivity and reduce time to market by enabling the development of systems using models defined with concepts that are much less tied to the underlying implementation technology, and much closer to the problem domain [3]. Same as MDE, low-code software development processes aim to improve software development processes by raising abstraction and hiding implementation-level details. Both approaches employ model-driven development (MDD) in their development stack; the important distinction seems to be how they enforce factors such as cloud-based deployment, target users, setup, and so on [13]. In addition to that, not all MDE approaches seek to reduce the amount of code required to develop software solutions, and not all low-code approaches are model-driven [13]. Although this is true, their difference in practice is not yet clear, and it is widely debated how much work done in MDE is directly transferable to LCDPs [14].

Nowadays, we witness a growing number of successful generic LCDPs on the market (e.g., Google App Maker and Microsoft Power Platform). Despite their success, LCDPs' development capabilities are still limited regarding how complex, intelligent, and sophisticated they may be [5]. In the IoT domain, only a few LCDPs are available, and they provide limited functionalities given the inherent complexity and heterogeneity of typical IoT systems. Among others, IoT-specific platforms such as Node-RED [15] and AtmosphericIoT [16] have demonstrated a significant push toward the development of fully-fledged multi-layer IoT platforms. However, their capacity to generate code and interact with low-level IoT devices is still limited. Unlike LCDPs, MDE presents a more significant number of platforms that can still generate low-level platform-specific code. Still, they often suffer from integration and interoperability issues because most of them are deployed and operated locally [11].

B. Software Quality Models

The discipline of Software Quality Engineering [17] is concerned with improving the approach to software quality. However, the various perspectives present throughout the software life cycle show what constitutes software quality is frequently contested. In this context, relying on software quality models to support the quality management of software systems is widely accepted [18]. A quality model is typically defined as a set of sub-characteristics and their interrelationships that serve as the foundation for specifying quality standards and evaluating quality [19]. Various standards have been defined in the literature to assist in the evaluation process. The ISO/IEC 9126:1991 standard aimed at defining a software quality paradigm and a set of guidelines for assessing the characteristics associated with it [20]. This standard was later revised by ISO/IEC 25010:2011, which included a model for software system quality with well-formed specifications for quality characteristics of software products [7].

The ISO/IEC 25010:2011 standards help design, develop and acquire systems and software products with the specification for evaluating their quality requirements [9]. The standards are made up of two quality models, each with its own set of characteristics, some of which are further subdivided into sub-characteristics. First, the *"quality in use model"* focuses on the outcome of interaction when a product is used under particular contexts [10]. This model is primarily intended to provide targets for promoting the development and verification efficiency, as well as to anticipate quality in use before delivery [9]. This model includes five characteristics, namely *effectiveness, efficiency, satisfaction, risk freedom, and context coverage* where some of them are further subdivided into nine sub-characteristics. The *"product quality model"* refers to the static and dynamic qualities of a software platform. This model primarily focuses on providing assessment ground for people supplying software products and those acquirers who wish to get more involved in the process technically [9]. The model is divided into eight characteristics, namely *functional appropriateness, performance efficiency, compatibility, reliability, usability, security, maintainability, and portability* which are further elaborated into 31 sub-characteristics.

In the past, the ISO/IEC 25010:2011 standard has been adopted to assess not only the product quality of IoT systems [21]–[23] but also in domains such as Big data [24], Machine Learning [25], Software Product Lines (SPL) [26], Customer Relationship Management (CRM) systems [27] and mobile apps [28], just to mention a few. In MDE domain, approaches such as [29] relied on it for assessing the performance of MDE quality studies, while [7], [8] adopted it in order to assess

the quality of Domain Specific Languages (DSLs) whileand design architectures quality [30] adopted it to assess the quality of design architectures. Although ISO/IEC 25010:2011 cannot be considered a one-size-fits-all solution, it can be a beneficial approach for evaluating the quality of IoT engineering platforms. A detailed description of the mentioned quality characteristics is given in the next section.

III. The product quality model

In this section, we discuss the *product quality model* of the ISO/IEC 25010:2011 standard to enable the quality evaluation of IoT engineering platforms. To this end, by referring to Figure 1, in the following, we discuss the common model characteristics concerning the peculiarities of the IoT domain.

A. Functional suitability

This factor evaluates how well a platform meets specified and implied objectives when used in specific contexts. In the IoT engineering context, its corresponding sub-characteristics can be defined as follows.

Functional completeness is the extent to which a given platform supports the design of all the layers that comprise an IoT ecosystem. For instance, the platform's capacity to represent the complete IoT ecosystem from the edge, fog, to the cloud layers while seemingly handling all of the communication heterogeneity that might occur. Concerning *Functional Correctness*, this can be defined as the extent to which the platform applies specific correctness methodologies during the development phases, including but not limited to correct by construction, model-checking, model validation, and rule-based modeling. Finally, the *Functional appropriateness* refers to the amount to which the given functionalities facilitate the completion of defined activities and objectives. IoT engineering platforms have diverse functionalities; in this case, we can propose to analyze how the specified correctness methodology permits achieving the goal.

B. Performance efficiency

This characteristic determines how the platforms consume resources under specific conditions. This characteristic is mainly related to run-time attributes of the underlying system infrastructure, which in some situations cannot be predicted accurately. In the context of IoT engineering platforms, the corresponding sub-characteristics can be defined as follows: *Time behavior* refers to how well a platform function meets its criteria in terms of response and processing times, as well as throughput rates. The platform's responsiveness can primarily indicate this, and it is worth noting that it can only be evaluated on platforms that are accessible.

Resource utilization measures how much and what kind of resources are used to suit the platform's needs. Although this can somewhat be predicted when a platform runs on well-known underlying infrastructures, the actual utilization can only be evaluated accurately during the platform's usage. Finally, the *Capacity* is defined to assess the platform's ability to model and coordinate a large and complex system that spans multiple IoT sub-domains.

C. Compatibility

This characteristic implies the platform's capacity to interchange data with other platforms. The two sub-characteristics are defined as follows: *Co-existence* refers to the degree to which a platform continues to operate efficiently while sharing a shared environment and resources with other platforms without hammering the other platforms. *Interoperability* is referred to the degree to which a platform can exchange information with other platforms while in use. In our context, this can be assessed in terms of the platform's capability to support the data exchange during the development process by consuming or sending information to/from external services via dedicated means such as REST APIs.

D. Reliability

This characteristic reflects how a platform can complete a set of tasks in a given amount of time. Its sub-characteristics can be defined as follows: *Maturity* reflects how well the platform supports all of the basic functionalities of a typical engineering platform, such as design, code generation, and deployment. *Availability* is concerned with the extent to which a platform is operational when needed. *Fault tolerance* is the degree to which a platform performs as expected despite the presence of hardware or software faults. In our case, this can be evaluated as the platform's capability to support advanced mechanisms, including but not limited to self-adaption and self-healing. Finally, the *Recoverability* reflects the extent to which a platform, in case of interruptions or failures, can recover data directly affected and re-establish the desired state of the system. To that end, the platform should be able to handle the mechanisms including but not limited to self-recovery or self-redeployment, etc.

E. Usability

This characteristic reflects how well specific users can use a platform to achieve particular goals with efficiency, effectiveness, and satisfaction. The sub-characteristics are as follows: *Appropriateness recognizability* is defined as the degree to which users can determine whether a platform is appropriate for their needs. This can be promoted in the IoT engineering world by the degree to which the platform is customized concerning the underlying environment. *Learnability* can be defined as the extent to which the platform aids developers in learning how to use it, e.g. with context-based modeling support, on-the-fly suggestions, and so on.

Operability represents the extent to which a platform has attributes that make the entire development process more accessible, for instance, using functionalities like auto-completion for textual languages, guide-through mechanisms, multi-view modeling, palette show/hide, and palette element search. *User error protection* reflects the extent to which the platform provides some protection to avoid errors, such as static analysis or on-the-fly error handling. *User interface aesthetics* is the extent to which the platform provides pleasant and satisfying user interfaces. Finally, *Accessibility* can be considered as to

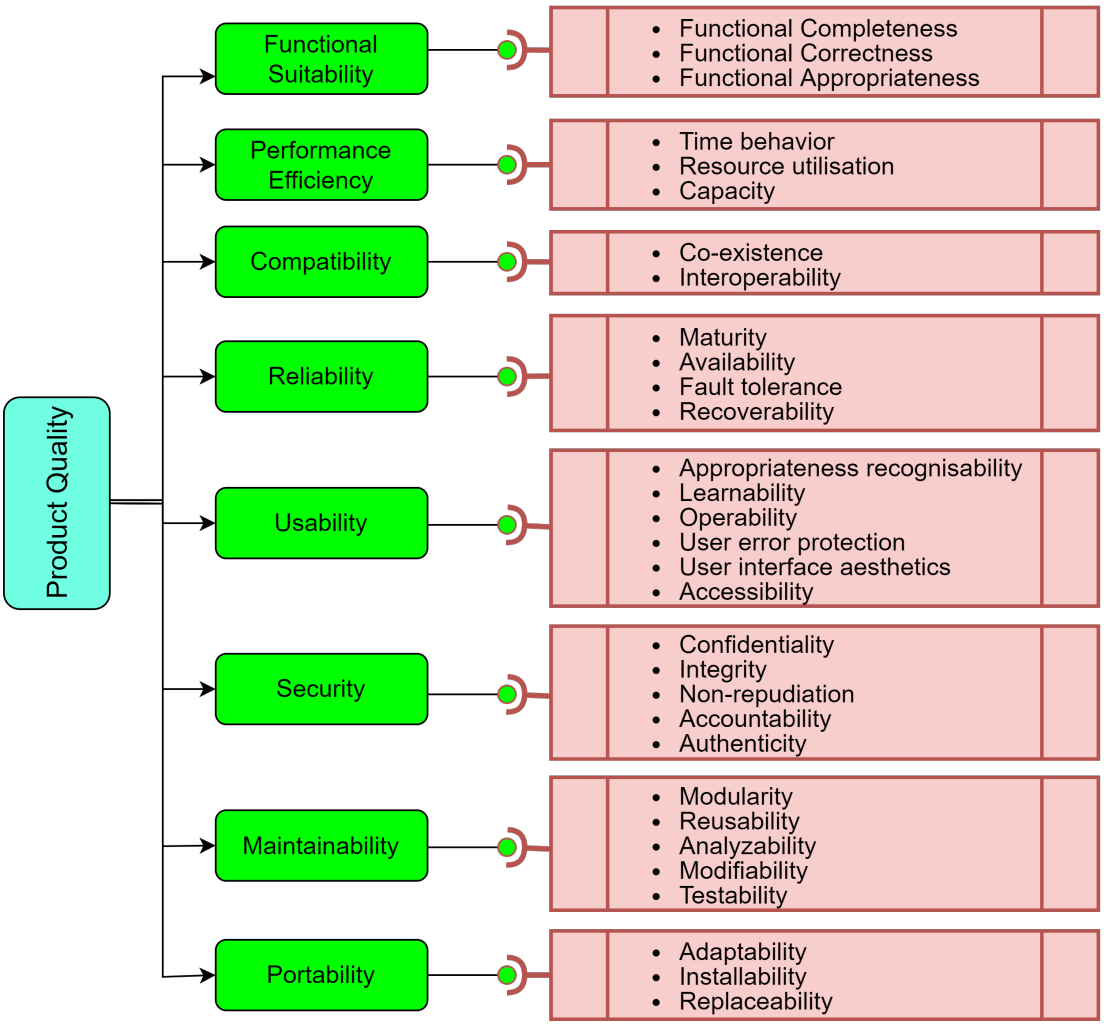


Figure 1: Software Product Quality Model

whether the platform is easily reachable in case of needs, either being locally or online.

F. Security

This characteristic refers to how successfully a platform protects information and data so that users, services, and external systems have appropriate access to data according to specific authorization levels. Its six sub-characteristics have been renamed as follows: *Confidentiality* refers to the extent to which a platform assures that data is exclusively available to those who have been granted access. *Integrity* can be defined as the extent to which an IoT platform prohibits unwanted access, modification of the platform, or data.

Non-repudiation can be measured by the extent to which the platform enables methods for recording all actions conducted during development and proves that they have been performed so that they cannot be contradicted later. *Accountability* can be measured as the extent to which a platform enables tracing attributes like versioning, historical action retrieval, and so on. Finally, *Authenticity* can be described as the ability of the platform to support different types of authentication mechanisms while accessing platform resources.

G. Maintainability

This characteristic refers to the degree of a platform to be improved, repaired, or adapted to changes. Its corresponding sub-characteristics are defined below: *Modularity* can be measured as the extent to which a platform is decoupled into discrete sub-systems that can be modified independently of other parts. *Reusability* reflects the extent to which a platform or a component of a platform can be used in more than one system. While evaluating this, we can focus on the platform's ability to be decomposed into small reusable sub-systems.

Analyzability assesses the efficacy and efficiency with which it is feasible to determine the impact of modifying parts of the platform by either removing it or injecting failures into it. In the IoT engineering context, whether the platform provides means supporting the analysis of the system under development and/or the platform itself is also looked at. In terms of *Modifiability*, the assessment can be performed on the

extent to which a platform can be successfully and efficiently modified without introducing flaws or deteriorating the quality of existing products. Finally, *Testability* can be assessed in terms of the capability of the platform to provide testing supports for its components or the system under development.

H. Portability

This characteristic refers to how fast and successfully a platform can be moved from or to different hardware and software operational environments. Its corresponding sub-characteristics are defined below: *Adaptability* refers to the extent to which a platform can be effectively and efficiently adapted to different environments. *Installability* reflects the degree of effectiveness and efficiency with which a platform can be successfully installed and/or uninstalled in a given environment. *Replaceability* measures the extent to which a platform can be updated, replaced, and redeployed in the same environment and still performs as expected.

IV. Quality assessment of IoT engineering platforms

This section shows the quality assessment process followed to analyze 17 IoT engineering platforms using the product quality model discussed in the previous section. The platforms of interest have been identified as presented in Section IV-A. The research questions that we answered through the performed evaluation are presented in Section IV-B. The quality assessment process that has been followed is presented in Section IV-C.

A. Selection of the evaluated IoT engineering platforms

In [11], we examined 16 different platforms to gain a better understanding of the current state of the art in supporting the development of IoT systems, with a focus on languages and tools available in the MDE field and emerging LCDPs. In [12], we examined 22 IoT modeling environments, assessing their strengths and weaknesses in terms of cloud-based modeling capacity, accessibility, openness, and artifact generation. Combining the two data sources, we started with a total of 38 approaches, which have been filtered by considering the following exclusion criteria:

- Approaches that were published before 2012 have been discarded;
- Duplicated approaches have been removed;
- Approaches that do not permit developing fully functional IoT applications have not been considered.
- Approaches that depend on already-included platforms have been filtered out;
- Approaches that are generic and that do not explicitly target the IoT domain have not been considered in this study.

By applying such criteria, 15 out of the initial 38 IoT development platforms were identified. In addition, we have added two more approaches that we believe are very promising and were published after our first study, bringing the total to 17. Figure 2 depicts the detailed paper selection procedure that we used. To better understand the selected basic studies, we have categorized them quantitatively based on publishing year, publisher type, article type, and the distribution among MDE and LCDPs approaches. The result is shown in Figure 3a, 3b, 3c and 3d respectively.

B. Research questions

The performed assessment aimed at answering the following research questions:

- ***RQ1****: To what extent do the considered IoT engineering platforms meet the characteristics of the proposed quality model?*
- ***RQ2****: What are the most and the least addressed quality sub-characteristics by the considered IoT engineering platforms?*

C. Assessment process

The quality assessment process of the considered IoT engineering platform has been done iteratively by going over all of the reference papers. We established a set of questions for each sub-characteristic that must be addressed to confirm the platform's competence with respect to what is included in the model described in Section III. By doing this, we made the evaluation process result easier and more reflective of the proposed model. Following that, we read the entire document and responded to the questions. Each question can be answered with "Yes" or "No". For instance, the following questions have been formulated to help in assessing the platform's *Functional Suitability*:

- Does the platform support the design and development of IoT systems?
- Does it support all layers (Edge, Fog, Cloud)?
- Does it mention any support for dealing with different communication protocols?

Consequently, given the paper under analysis, if the corresponding presented platform succeeds on at least 50% of the questions, it is marked as supporting the quality characteristic of interest. This procedure has been developed and implemented for all 31 sub-characteristics. The defined questions have been created purely to facilitate the review process and are entirely consistent with what is provided in the model. Unfortunately, due to space constraints, we could not present the extended table of questionnaire used to assess all of the quality characteristics. In this regards, a full set of evaluation questionnaire used in the evaluation has been published and can be accessed from an online database available at [31].

As mentioned in the previous section, some of the characteristics heavily depend on the dynamic properties of the underlying system infrastructures as well as during their usage. For instance, verifying sub-characteristics related to performance efficiency and usability might be difficult in general. Thus, during the performed analysis we relayed on information provided in the reference papers when available. We considered such kind of characteristics unsupported if the corresponding articles do not mention any mechanism addressing them.

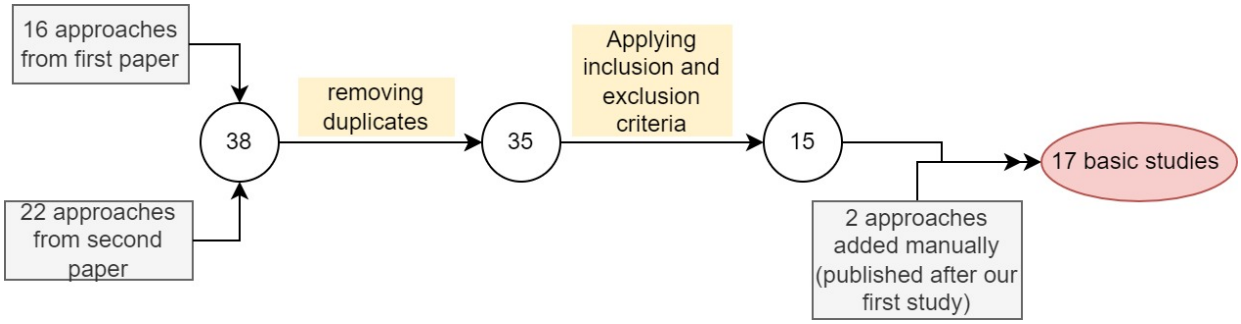
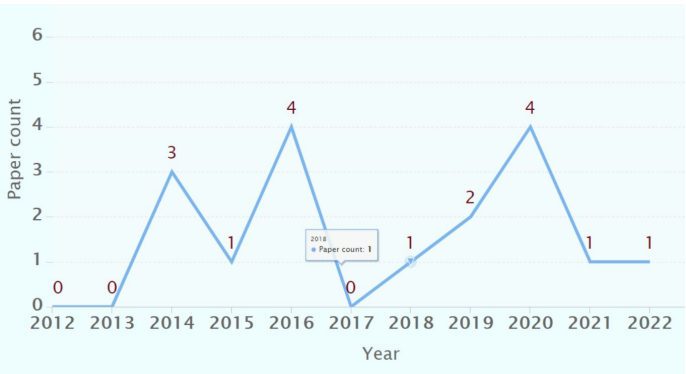
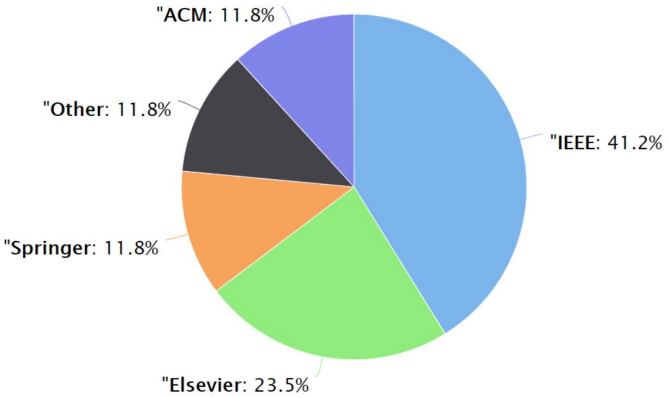


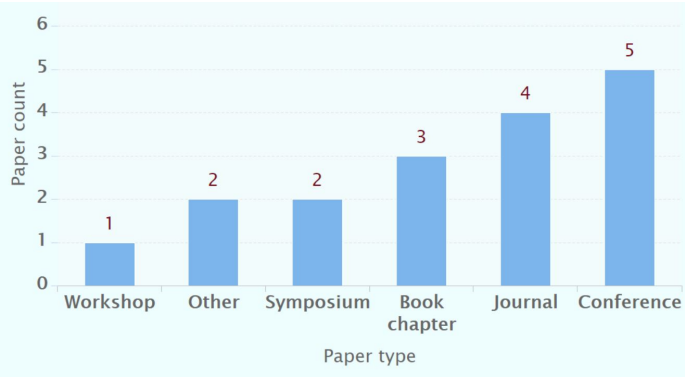
Figure 2: Primary studies selection process



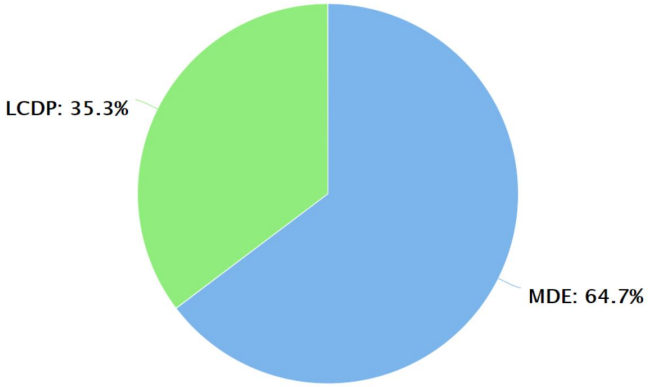
(a) Publications by year



(b) Publications by publisher



(c) Publications by paper type



(d) Publications by category

Figure 3: Overview of the selected basic studies

V. Assessment Results

This section presents the comprehensive findings of our assessment by answering the research questions. Table I summarizes the results of the study. The MDE and LCDP platforms supporting each quality characteristic of the considered product model are shown.

1) Quality characteristics support of IoT engineering platforms (RQ1): According to the proposed model, eight characteristics with their associated sub-characteristics were evaluated on the approaches considered in this work. This section elaborates on the overall quality performance of such development platforms by comparing LCDPs and MDE platforms.

As shown in Table I, MDE approaches support an average of 12 of the 31 possible sub-characteristics, whereas LCDPs support an average of 18. Furthermore, different sub-characteristics are not supported at all. Figure 4 shows the overall characteristic-level performance of the analyzed platforms. Figure 4a and Figure 4b depict the overall performance of MDE platforms and LCDPs against the eight main characteristics, respectively. To this end, an aggregated performance sum from all sub-characteristics was produced for each characteristic.

TABLE I: Assessment overview

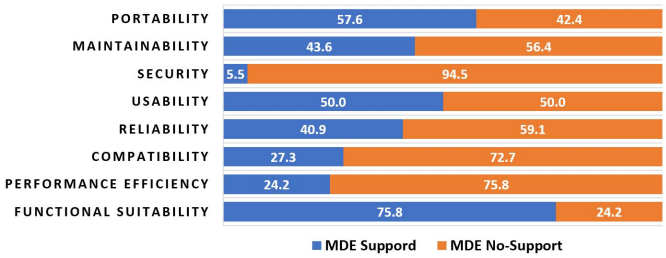
Characteristic	Sub-characteristic	MDE platforms	MDE(%)	LCDP platforms	LCDP(%)
Functional suitability			**75.8**		**83.3**
	Func. Completeness	[32] [33] [34] [35] [36] [37] [38] [39] [40]	81.82	[41] [15] [16] [42] [43]	83.3
	Func. Correctness	[44] [33] [32] [35] [36] [45] [38] [39] [40]	81.82	[15] [16] [42] [43] [46]	83.3
	Func. Appropriateness	[44] [34] [32] [36] [38] [39] [40]	63.6	[15] [16] [42] [41] [46]	83.3
Performance efficiency			**24.2**		**38.9**
	Time-behavior	Unsupported	0	[15] [16]	33.3
	Resource Utilization	[32]	9.09	[15] [16]	33.3
	Capacity	[33] [32] [36] [37] [38] [39] [40]	63.3	[15] [16] [41]	50
Compatibility			**27.3**		**58.3**
	Co-existence	Unsupported	0	[15] [16] [42]	50
	Interoperability	[32] [35] [36] [45] [38] [39]	54.5	[15] [16] [42] [43]	66.6
Reliability			**40.9**		**41.7**
	Maturity	[44] [34] [32] [35] [36] [45] [38] [39] [40]	81.82	[15] [16] [42] [41] [43]	83.3
	Availability	[36] [40]	18.18	[15] [16]	33.3
	Fault tolerance	[44] [33] [38] [39] [40]	45.45	[15]	16.6
	Recoverability	[32] [40]	18.18	[15] [16]	33.33
Usability			**50**		**66.7**
	Appropriateness	[33] [34] [32] [36] [40]	45.45	[15] [42] [41]	50
	Learnability	[38]	9.09	[15] [16] [46] [43] [39]	83.3
	Operability	[44] [32] [36] [45] [37] [38] [39] [40]	72.72	[15] [16] [42] [41] [46]	83.3
	User error protection	[36] [37] [38] [39] [40]	45.45	[15] [16] [42] [46]	66.6
	User interface	[44] [33] [34] [32] [35] [36] [45] [37] [38] [39] [40]	100	[15] [16] [42] [41] [46] [43]	100
	Accessibility	[36] [40]	18.18	[15] [16]	33.3
Security			**5.5**		**36.7**
	Confidentiality	[32]	9.09	[16] [42]	33.3
	Integrity	[32]	9.09	[16] [42] [43]	50
	Non-repudiation	Unsupported	0	[16]	16.6
	Accountability	Unsupported	0	[15] [16]	33.3
	Authenticity	[32]	9.09	[16] [42] [43]	50
Maintainability			**43.6**		**30**
	Modularity	[34] [32] [35] [36] [39] [40]	54.54	[15] [16] [41] [43]	66.6
	Reusability	[44] [39] [33] [32] [36] [38] [40]	63.3	[15]	16.6
	Analyzability	[33] [32] [38]	27.3	[15]	16.6
	Modifiability	[33] [34] [32] [36] [38] [39]	54.5	[15] [43]	33.3
	Testability	[39] [40]	18.18	[15]	16.6
Portability			**57.6**		**66.7**
	Adaptability	[32] [35] [38] [40]	36.36	[15] [16] [42]	50
	Installability	[44] [33] [34] [32] [35] [36] [45] [37] [38] [39] [40]	100	[15] [16] [42] [41] [46] [43]	100
	Replaceability	[32] [36] [39] [40]	36.36	[15] [16] [43]	50
	Overall	*135 out of 341 possible*	**39.6**	*95 out of 186 possible*	**51.1**
	Standard deviation		30.6		25.5

Both categories (MDE and LCDPs) perform well enough in quality characteristics related to *Functional suitability*, *Portability*, and *Usability*, with general supporting rates of 75.8%, 57.8%, and 50% for MDE and 83.3%, 66.7%, and 66.7% for LCDPs, respectively. It is important to note that the listed supporting/non-supporting rates reflect the aggregated characteristic performance calculations from its corresponding sub-characteristics. Although this provides us with an overall picture of characteristic performance, it does not allow us to make a final judgment on whether certain sub-characteristics are better supported than others. For example, in terms of *Usability* aspects, all 11 MDE platforms satisfy the *User interface* quality characteristic, although only two of them satisfy the accessibility sub-characteristic.

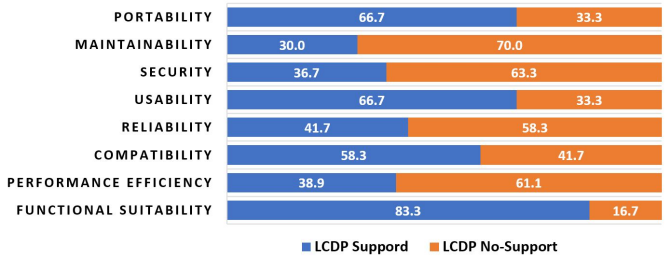
MDE platforms have limited security and performance efficiency support, with overall support rates of 5.5% and 24.2%, respectively. For instance, in MDE, from the studies considered, only IoTML [32] promotes security by enhancing authentication, confidentiality, and integrity mechanisms while accessing the platform. On the other hand, LCDPs fall short concerning security and maintainability, with overall support rates of 36.7% and 30%, respectively. Each of the security aspects is implemented at least once by LCDPs, with the AtmosphereIoT [16] platform supporting all. Consequently, as shown in Table I, MDE accounts for 135 points out of 341 possible support, representing approximately 39.6%, whereas LCDPs account for 95 points out of 186 possible, which would be about 51.1%.

Answer to RQ1: Overall, MDE quality is around 39.6%, while LCDPs account for 51.1%, resulting in an overall quality of the selected engineering platforms being approximately 45.5%.

2) Quality sub-characteristics support of IoT engineering platforms (RQ2): In the previous section, we focused on overall quality characteristic support; in this section, we focus on individual sub-characteristic by highlighting the most and least addressed ones. Even though the average supporting rate of studied quality sub-characteristics for both MDE and LCDPs will be equal to the average supporting rate for the



(a) MDE platforms



(b) LCDPs

Figure 4: Quality characteristics support

quality characteristics presented above (Sec. V-1: MDE:39.6%, LCDP:51.1%), they differ in their deviation from the mean of individual supports. According to the standard deviation indicated in Table I, MDE platforms have an average supporting standard deviation of 30.6, while LCDPs have a standard deviation of roughly 25.5, which is less than of MDE platforms. Such figures suggest that LCDPs are more likely to consistently touch all of the model's individual quality sub-characteristics than MDE approaches.

As shown in Table I, IoT MDE and LCDPs cover all quality sub-characteristics such as *user interface* and *installability*. This is expected, given that the fundamental principle of MDE and LCDP technologies centers around enabling an easy-to-use and executable environment for developing applications with less effort, which cannot be accomplished without providing a user interface.

Figure 5 depicts an overall performance of sub-characteristics among IoT MDE, and LCDPs engineering platforms. As indicated, none of the chosen MDE approaches address quality sub-characteristics such as non-repudiation, time-behavior, accountability, and co-existence. LCDPs, on the other hand, fall short on fault-tolerance, non-repudiation, reusability, analyzability, and testability, with a consistent minimal support rate of 16.7%, which implies a generic rate of at least 1 out of 6 LCDPs supports at least one of the sub-characteristics. On the other hand, Figure 6 depicts the average quality performance for both MDE and LCDP platforms. As can be seen, sub-characteristics such as user interface, installability, and functional completeness are the most well supported by both categories, while testability, time-behavior, accountability, and non-repudiation are the least supported.

Answer to RQ2: The top three quality sub-characteristics addressed by both technologies are *user interface*, *installability*, and *maturity*. On the other hand, selected MDE approaches were unable to address quality sub-characteristics such as non-repudiation, accountability, co-existence, and time-behavior. In contrast, LCDPs fall short of addressing fault tolerance, non-repudiation, and reusability with a consistent rate of 1 out of 6 LCDPs.

VI. Discussion

In this section, we discuss the proposed model's suitability and its limitation with respect to general software quality assessment of software systems.

A. Model suitability

The proposed model aims to assist practitioners who have to design and develop IoT systems by exploiting low-code or MDE platforms. According to the results presented in Section V, security-related characteristics are the least addressed. This is particularly pertinent given how IoT security concerns are dynamic and unstructured, leading to uncertainty among software developers in terms of concepts and terms [47]. MDE approaches are most affected since most conventional MDE platforms are deployed locally and used offline, making incorporating any form of security capabilities less required (e.g., authentication). We can argue that the dominant Eclipse Development Environment[1], which hosts a lot of classical MDE-based platforms, has a significant impact on this problem. On the other hand, although LCDPs neither excel in such security-related aspects, it is critical and rational to be integrated since such platforms are deployed in cloud-based environments, which are more likely to be attacked by unwanted intruders.

Furthermore, besides security, the results show that LCDPs lack quality criteria for general maintainability. For instance, according to the Table I, the reusability of LCDP is shown to be less supported among others. This is generally true and can mainly be since more LCDPs are tailor-made, and most of their application developments and deployments are bound to a particular technology [13], making them difficult to modify and reuse elsewhere. In our study, only Node-RED [15] showcased means for supporting the usability of aspects of its components through the Node-RED modules that can be composed, built, deployed and reused separately from one instance to another.

Due to the tight coupling between functions and their sub-functions found in different LCDPs and some MDE platforms, analysability becomes very hard to achieve. Our proposed model defines software analysability as the means to assess the efficacy and efficiency with which it is feasible to determine the impact of modifying parts of the platform by either removing it or injecting failures into it. According to the obtained results, only 27% of MDE platforms support such features, while for LCDPs, only one out of 6 support them. This

[1]https://www.eclipse.org/

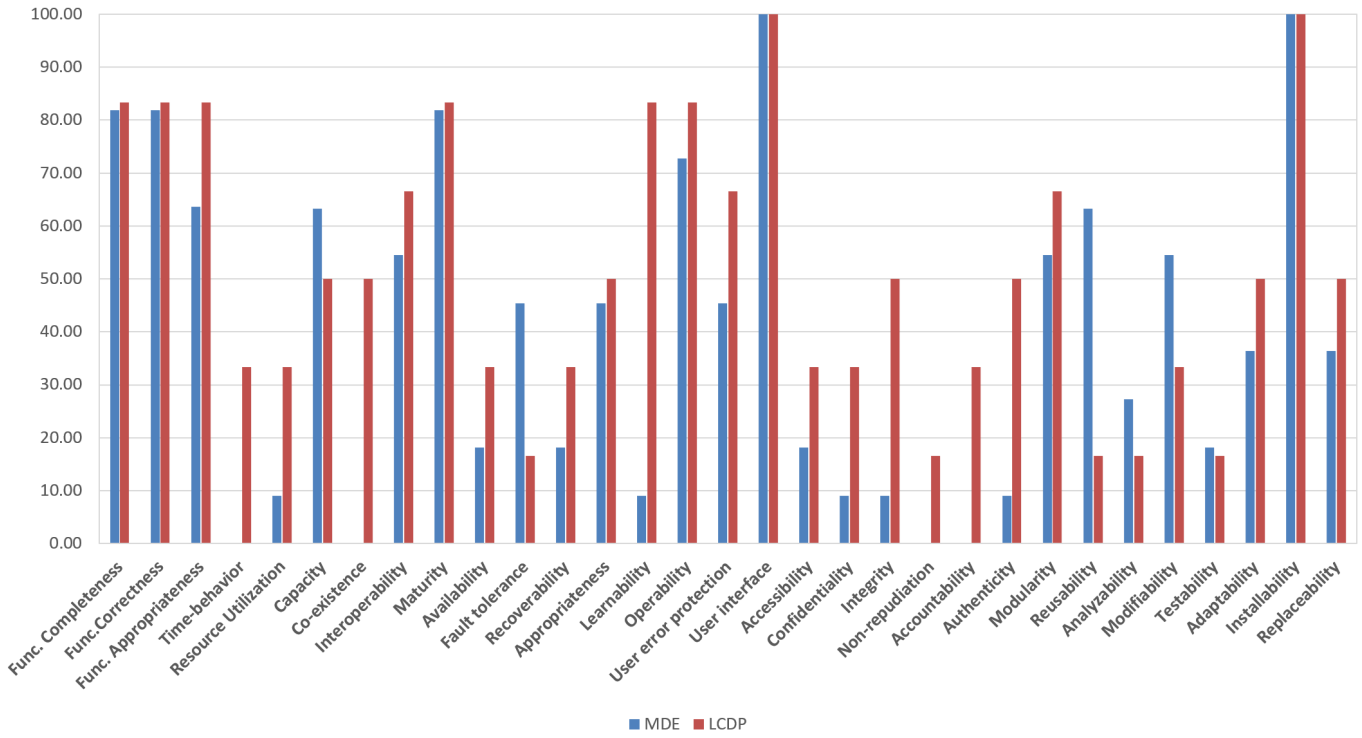


Figure 5: Quality sub-characteristics performances

quality characteristic is generally considered in the software design phase through different model-based system analyses. Still, in the actual implementation of the system, such quality is often ignored [48]. Concerning software *testability*, as indicated above, only 2 out of 11 of the analysed platforms showed means for supporting it. The testing can either be done at the platform sub-functions level or at the system they develop. In general, developers tend to disregard this aspect. For instance, in the LCDP domain, low-code testing is still in its early stages, with no formal structure to the domain's ideas, concepts, and hypotheses which undoubtedly contributes to such lacking [6].

Another interesting fact is that the overall performance of IoT engineering platforms (MDEs and LCDPs combined), regardless of characteristics and sub-characteristics, is about 45.5%, in which MDE accounts for 39.6%, whereas LCDPs have 51.1%. Although we acknowledge that these measures cannot be regarded as a definitive measure of the IoT engineering platforms' quality questions, we believe they can point researchers in the right direction regarding the present state of the art in IoT engineering quality evaluation support. Consequently, we can finally draw the line of how the proposed model satisfies the quality evaluation procedure as "promising"; it unveiled several concepts that reflect what is already available in the IoT engineering domain.

B. Limitations

In this paper, we have extended the ISO/IEC 25010:2011 standard model to propose a software product model for assessing the quality of IoT engineering platforms. To evaluate the effectiveness of the proposed model, we employed it to determine the quality of 17 IoT platforms selected from our previous studies [11], [12]. In terms of limitations, we can state the followings:

- Although the presented model contributes positively to the software product quality assessment of IoT engineering platforms, it only performs very well when the evaluation is done at the platform's technical implementation level. However, the proposed model performed poorly regarding run-time software product quality assessment, such as quality linked to performance efficiency. This is primarily due to the implementation nature of the LCDP and MDE platforms. We believe that assessing such quality could be heavily influenced by the environment in which such software is deployed.
- The evaluation methods employed in this study and the reported results are critical because they are exclusively based on what was identified in the selected tool's papers. However, in some instances, the published content of the paper may not correctly reflect the full capabilities of the platform under consideration. Furthermore, software platforms evolve, and new development is regularly contributed to the platforms. Therefore, we believe that integrating the results found in the papers and the actual inputs from the tool vendors can significantly increase the legitimacy of the findings. We intend to address this in future research.

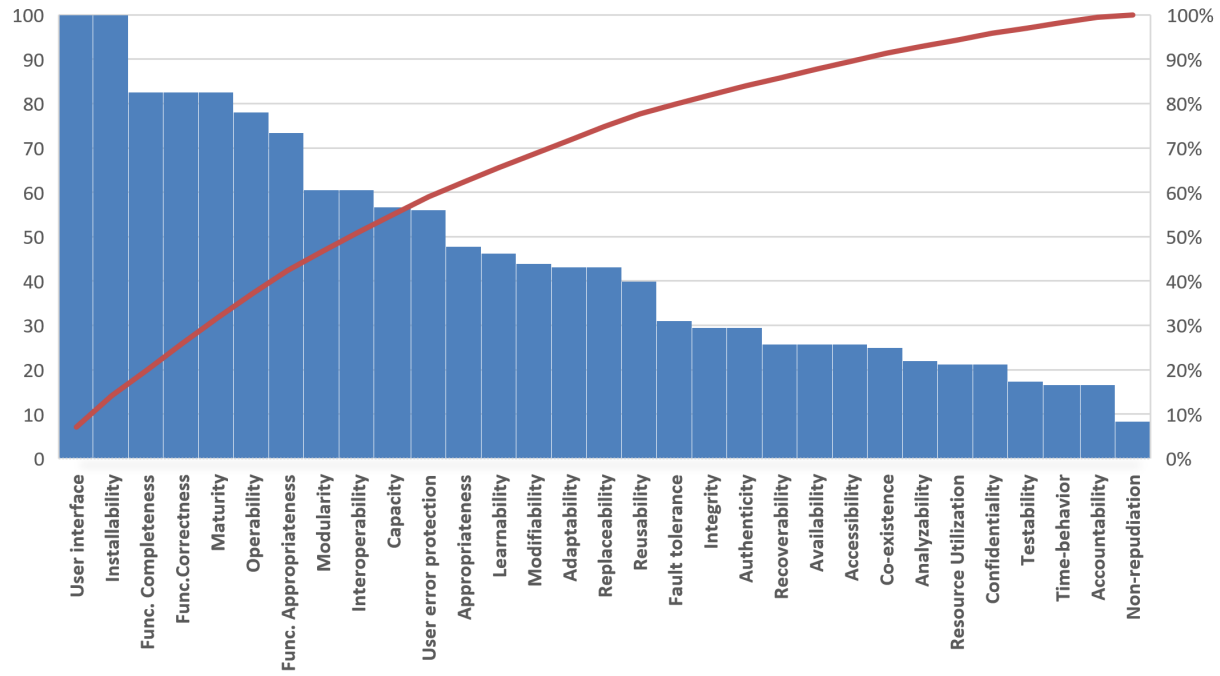


Figure 6: Average quality sub-characteristics performances

VII. Related Work

ISO/IEC 25010:2011 standard has been heavily used to assess the quality of complex software and systems. To name a few, [21] relied on it to assess the quality of online health awareness systems, [22], [49] for IoT brokers, and [23] for IoT implementations. Although this quality model's scope is intended general for software and computer systems, it can also be applied to assess larger systems and services [10]. For instance, in [28], the model has been adopted in order to evaluate the quality of mobile applications, as well as in [24], [25] for Machine Learning and Big data systems.

There have been several approaches in the modeling domain for quality measurements. For example, in [7], a Framework for Qualitative Assessment of DSLs (FQAD) was presented. The FQAD framework is based on ISO/IEC 25010:2011 when determining the evaluation's perspective, the assessment's goal, and selecting relevant quality characteristics to guide the assessment process. In addition to that, the authors in [8] relied on the standard to assess the quality assurance of DSLs. In contrast, [30], the standard was employed to evaluate the quality of design architectures.

In [26], the authors used this standard to perform a mapping study on the quality assessment of software product lines. The standard was used in [50] to assess the security quality of mobile cloud computing-based technologies. Finally, it was used in [27] to evaluate the quality aspects of customer relationship management (CRM) systems. A large number of extensions and suggestions on the standard have been proposed. To name a few, authors in [9] proposed the extension to meet the lifetime service-oriented quality aspect of software systems. In [24], the authors conducted a study to assess the product quality of Big data systems concerning non-functional requirements, while in [51], the authors extended the model to Semantic Web exploration tools.

Finally, the authors in [52] presents a highly comprehensive effort to assist the quality evaluation of MDD platforms, as well as a mapping from the proposed model to the ISO/IEC 25010:2011 standard. In their paper, a multi-criteria decision-making (MCDM) model for MDD platforms is used to help in choosing an optimal quality sufficient platform for their requirements. Nevertheless, the approach as well as the results presented are too generic, whereas our approach focuses primarily on IoT specific platforms. Furthermore, the approach, in our opinion, focuses mostly on LCDPs, while other conventional MDE platforms, such as those based on the Eclipse Development Environment, are not taken into account at all.

Even though all of the approaches relied on the standard used in this paper, none specifically target IoT engineering platforms. Based on the previous discussions, we can conclude that this study is novel and unique compared to other studies. While there are many quality assessments of DSLs available, only a small number of them refer to an established standard in the evaluation [7], and we could not find any that addresses the IoT modeling domain in particular. As a result, we are confident that our study is the premier to use a well-established quality standard to evaluate the product quality of IoT engineering platforms.

VIII. Conclusion and Future work

In this paper, we presented an extension of the ISO/IEC 25010:2011 product quality model for assessing the quality aspects of IoT engineering platforms. We evaluated the software product quality of 17 IoT engineering platforms using the proposed model. The findings revealed that the overall performance of IoT engineering platforms is roughly 45.5%, with LCDPs doing slightly better than MDE platforms. Furthermore, security and maintainability aspects are found to be less addressed, whereas functional appropriateness, portability, and usability were found to be the most addressed. In the future, we plan to evaluate the quality in the use of the IoT engineering platform by extending the *quality in use model* of the ISO/IEC 25010:2011 standard in which we will be able to accommodate other quality aspects beyond the software product quality model.

References

[1] F. Ciccozzi, I. Crnkovic, D. Di Ruscio, I. Malavolta, P. Pelliccione, and R. Spalazzese, "Model-driven engineering for mission-critical iot systems," *IEEE Software*, vol. 34, no. 1, pp. 46–53, 2017. [Online]. Available: https://doi.org/10.1109/MS.2017.1

[2] F. Ihirwe, D. D. Ruscio, S. Mazzini, and A. Pierantonio, "Towards a modeling and analysis environment for industrial iot systems," *STAF Workshops*, vol. abs/2105.14136, 2021. [Online]. Available: http://ceur-ws.org/Vol-2999/messpaper1.pdf

[3] D. Di Ruscio, R. Eramo, and A. Pierantonio, *Model Transformations*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 91–136. [Online]. Available: https://doi.org/10.1007/978-3-642-30982-3_4

[4] A. Sahay, A. Indamutsa, D. Di Ruscio, and A. Pierantonio, "Supporting the understanding and comparison of low-code development platforms," in *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2020, pp. 171–178. [Online]. Available: https://doi.org/10.1109/SEAA51224.2020.00036

[5] M. Tisi, J.-M. Mottu, D. S. Kolovos, J. de Lara, E. M. Guerra, D. Di Ruscio, A. Pierantonio, and M. Wimmer, "Lowcomote: Training the Next Generation of Experts in Scalable Low-Code Engineering Platforms," in *STAF 2019*, ser. CEUR Workshop Proceedings (CEUR-WS.org), Eindhoven, Netherlands, Jul. 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02363416

[6] F. Khorram, J.-M. Mottu, and G. Sunyé, "Challenges & opportunities in low-code testing," in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, ser. MODELS '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3417990.3420204

[7] G. Kahraman and S. Bilgen, "A framework for qualitative assessment of domain-specific languages," *Softw. Syst. Model.*, vol. 14, no. 4, p. 1505–1526, oct 2015. [Online]. Available: https://doi.org/10.1007/s10270-013-0387-8

[8] C. Moreira, J. Cobos-Q, W. V. Solis, C. Sánchez-Zhunio, and I. P. C. Orellana, "Evaluating the usability in domain-specific languages," in *Information Systems Development: Crossing Boundaries between Development and Operations (DevOps) in Information Systems (ISD2021 Proceedings), Valencia, Spain, September 8-10, 2021*, 2021. [Online]. Available: https://aisel.aisnet.org/isd2014/proceedings2021/hci/3

[9] J. Estdale and E. Georgiadou, "Applying the iso/iec 25010 quality models to software product," in *Systems, Software and Services Process Improvement*, X. Larrucea, I. Santamaria, R. V. O'Connor, and R. Messnarz, Eds. Cham: Springer International Publishing, 2018, pp. 492–503. [Online]. Available: https://doi.org/10.1007/978-3-319-97925-0_42

[10] I. ISO, "ISO/IEC 25010:2011, Systems and software engineering - Systems and Software Quality Requirements and Evaluation (SQuaRE) - System and software quality models," *ISO*, vol. 34, p. 2910, 2011, last accessed March 2022. [Online]. Available: https://www.iso.org/standard/35733.html

[11] F. Ihirwe, D. Di Ruscio, S. Mazzini, P. Pierini, and A. Pierantonio, "Low-code engineering for internet of things: A state of research," in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, ser. MODELS '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3417990.3420208

[12] F. Ihirwe, A. Indamutsa, D. D. Ruscio, S. Mazzini, and A. Pierantonio, "Cloud-based modeling in iot domain: a survey, open challenges and opportunities," in *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 2021, pp. 73–82. [Online]. Available: https://doi.org/10.1109/MODELS-C53483.2021.00018

[13] D. Di Ruscio, D. Kolovos, J. de Lara, A. Pierantonio, M. Tisi, and M. Wimmer, "Low-code development and model-driven engineering: Two sides of the same coin?" *Software and Systems Modeling*, vol. 21, pp. 437–446, 2022. [Online]. Available: https://doi.org/10.1007/s10270-021-00970-2

[14] J. Cabot, "Positioning of the low-code movement within the field of model-driven engineering," in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, ser. MODELS '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3417990.3420210

[15] Node-RED, "Node-RED: Low-code development platform," 2022, accessed March 2022. [Online]. Available: https://nodered.org/

[16] AtmosphereIoT, "Fast time to first data," 2022, last accessed March 2022. [Online]. Available: https://atmosphereiot.com/

[17] S. H. Kan, *Metrics and Models in Software Quality Engineering*, 2nd ed. USA: Addison-Wesley Longman Publishing Co., Inc., 2002. [Online]. Available: https://dl.acm.org/doi/10.5555/559784

[18] M. Ortega and et al, "Construction of a systemic quality model for evaluating a software product," *Softw. Qual. J.*, vol. 11, no. 3, 2003. [Online]. Available: https://doi.org/10.1023/A:1025166710988

[19] O. Gordieiev, V. Kharchenko, N. Fominykh, and V. Sklyar, "Evolution of software quality models in context of the standard iso 25010," in *Proceedings of the Ninth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX. June 30 – July 4, 2014, Brunów, Poland*, Cham, 2014, pp. 223–232. [Online]. Available: https://doi.org/10.1007/978-3-319-07013-1_21

[20] M.-A. Côté, W. Suryn, and E. Georgiadou, "In search for a widely applicable and accepted software quality model for software quality engineering," *Software Quality Journal*, vol. 15, no. 4, p. 401–416, dec 2007. [Online]. Available: https://doi.org/10.1007/s11219-007-9029-0

[21] A. Hussain and E. Mkpojiogu, "An application of the iso/iec 25010 standard in the quality-in-use assessment of an online health awareness system," *Jurnal Teknologi*, vol. 77, 11 2015. [Online]. Available: https://doi.org/10.11113/jt.v77.6107

[22] E. Bertrand-Martinez, P. Dias Feio, V. d. Brito Nascimento, F. Kon, and A. Abelém, "Classification and evaluation of iot brokers: A methodology," *International Journal of Network Management*, vol. 31, no. 3, may 2021. [Online]. Available: https://doi.org/10.1002/nem.2115

[23] J. J. Tambotoh, S. M. Isa, F. L. Gaol, B. Soewito, and H. L. H. S. Warnars, "Software quality model for internet of things governance," in *2016 International Conference on Data and Software Engineering (ICoDSE)*, 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICODSE.2016.7936138

[24] M. S. Rahman and H. Reza, "Systematic mapping study of non-functional requirements in big data system," in *2020 IEEE International Conference on Electro Information Technology (EIT)*, 2020, pp. 025–031. [Online]. Available: http://dx.doi.org/10.1109/EIT48999.2020.9208288

[25] J. Siebert, L. Jöckel, J. Heidrich, K. Nakamichi, K. Ohashi, I. Namba, R. Yamamoto, and M. Aoyama, "Towards guidelines for assessing qualities of machine learning systems," in *13th International Conference on Quality of Information and Communications Technology - , QUATIC 2020, Faro, Portugal, September 9-11, 2020*, vol. 1266, 2020, pp. 17–31. [Online]. Available: https://doi.org/10.1007/978-3-030-58793-2_2

[26] A. M. Luana, A. J. Paulo, P. F. André, and C. Heitor, *IET Software*, vol. 14, pp. 572–581(9), December 2020. [Online]. Available: https://doi.org/10.1049/iet-sen.2020.0037

[27] J. Bernardes Boarim and A. R. Cavalcanti da Rocha, "CRM systems quality evaluation," in *XX Brazilian Symposium on Software Quality*, ser. SBQS '21. New York, NY, USA: Association for Computing

Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3493244.3493273

[28] J. Koepp, M. V. Baron, P. R. Hernandes Martins, C. Brandenburg, and et al, "The quality of mobile apps used for the identification of pressure ulcers in adults: Systematic survey and review of apps in app stores," *JMIR Mhealth Uhealth*, vol. 8, no. 6, p. e14266, Jun 2020. [Online]. Available: https://doi.org/10.2196/14266

[29] M. Goulão, V. Amaral, and M. Mernik, "Quality in model-driven engineering: A tertiary study," vol. 24, no. 3, p. 601–633, sep 2016. [Online]. Available: https://doi.org/10.1007/s11219-016-9324-8

[30] M. Darwish and E. Shehab, "Framework for engineering design systems architectures evaluation and selection: Case study," *Procedia CIRP*, vol. 60, pp. 128–132, 2017, complex Systems Engineering and Development Proceedings of the 27th CIRP Design Conference Cranfield University, UK 10th – 12th May 2017. [Online]. Available: https://doi.org/10.1016/j.procir.2017.01.058

[31] F. Ihirwe, D. D. Ruscio, S. Gianfranceschi, and A. Pierantonio, "Software product quality evaluation questionnaire for IoT LCDP&MDE," Jun. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6631200

[32] D. Conzon, M. R. A. Rashid, X. Tao, A. Soriano, R. Nicholson, and E. Ferrera, "BRAIN-IoT: Model-based framework for dependable sensing and actuation in intelligent decentralized iot systems," in *2019 4th International Conference on Computing, Communications and Security (ICCCS)*, 2019, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/CCCS.2019.8888136

[33] B. Costa, P. F. Pires, F. C. Delicato, W. Li, and A. Y. Zomaya, "Design and analysis of iot applications: A model-driven approach," in *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, DASC/PiCom/DataCom/CyberSciTech*, 2016, pp. 392–399. [Online]. Available: http://dx.doi.org/10.1109/DASC-PICom-DataCom-CyberSciTec.2016.81

[34] K. Thramboulidis and F. Christoulakis, "UML4IoT—A UML-based approach to exploit iot in cyber-physical manufacturing systems," *Computers in Industry*, vol. 82, pp. 259–272, 2016. [Online]. Available: https://doi.org/10.1016/j.compind.2016.05.010

[35] F. Pramudianto, C. A. Kamienski, E. Souto, F. Borelli, L. L. Gomes, D. Sadok, and M. Jarke, "Iot link: An internet of things prototyping toolkit," in *2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing*, 2014, pp. 1–9. [Online]. Available: http://dx.doi.org/10.1109/UIC-ATC-ScalCom.2014.95

[36] N. Harrand, F. Fleurey, B. Morin, and K. E. Husa, "Thingml: A language and code generation framework for heterogeneous targets," ser. MODELS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 125–135. [Online]. Available: https://doi.org/10.1145/2976767.2976812

[37] D. Soukaras, P. Patel, H. Song, and S. Chaudhary, "IoTSuite: A toolsuite for prototyping internet of things applications," *The 4th Workshop on on Computing and Networking for Internet of Things (ComNet-IoT 2015)*, 2020.

[38] W. Rafique, X. Zhao, S. Yu, I. Yaqoob, M. Imran, and W. Dou, "An application development framework for internet-of-things service orchestration," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4543–4556, 2020. [Online]. Available: http://dx.doi.org/10.1109/JIOT.2020.2971013

[39] J. A. Barriga, P. J. Clemente, E. Sosa-Sanchez, and A. E. Prieto, "Simulateiot: Domain specific language to design, code generation and execute iot simulation environments," *IEEE Access*, vol. 9, pp. 92 531–92 552, 2021. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2021.3092528

[40] J. C. Kirchhof, B. Rumpe, D. Schmalzing, and A. Wortmann, "Montithings: Model-driven development and deployment of reliable iot applications," *Journal of Systems and Software*, vol. 183, p. 111087, 2022. [Online]. Available: https://doi.org/10.1016/j.jss.2021.111087

[41] C. González García, B. C. Pelayo G-Bustelo, J. Pascual Espada, and G. Cueva-Fernandez, "Midgar: Generation of heterogeneous objects interconnecting applications. a domain specific language proposal for internet of things scenarios," *Comput. Netw.*, vol. 64, p. 143–158, may 2014. [Online]. Available: https://doi.org/10.1016/j.comnet.2014.02.010

[42] G. Cueva-Fernandez, J. P. Espada, V. García-Díaz, C. G. García, and N. Garcia-Fernandez, "Vitruvius: An expert system for vehicle sensor tracking and managing application generation," *Journal of Network and Computer Applications*, vol. 42, pp. 178–188, 2014. [Online]. Available: https://doi.org/10.1016/j.jnca.2014.02.013

[43] B. El Khalyly, M. Banane, A. Erraissi, and A. Belangour, "Interoevery: Microservice based interoperable system," in *2020 International Conference on Decision Aid Sciences and Application (DASA)*, 2020, pp. 320–325. [Online]. Available: http://dx.doi.org/10.1109/DASA51403.2020.9317159

[44] F. Ciccozzi and R. Spalazzese, "Mde4iot: Supporting the internet of things with model-driven engineering," in *Intelligent Distributed Computing X*. Cham: Springer International Publishing, 2017, pp. 67–76. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-48829-5_7

[45] T. Nepomuceno, T. Carneiro, P. H. Maia, M. Adnan, T. Nepomuceno, and A. Martin, "Autoiot: A framework based on user-driven mde for generating iot applications," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, ser. SAC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 719–728. [Online]. Available: https://doi.org/10.1145/3341105.3373873

[46] Y. Valsamakis and A. Savidis, *Personal Applications in the Internet of Things Through Visual End-User Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, pp. 809–821. [Online]. Available: https://doi.org/10.1007/978-3-662-49275-8_71

[47] B. A. Mozzaquatro, R. Jardim-Goncalves, and C. Agostinho, "Towards a reference ontology for security in the internet of things," in *2015 IEEE International Workshop on Measurements & Networking (M&N)*, 2015, pp. 1–6. [Online]. Available: https://doi.org/10.1109/IWMN.2015.7322984

[48] E. Bouwers, J. P. Correia, A. v. Deursen, and J. Visser, "Quantifying the analyzability of software architectures," in *2011 Ninth Working IEEE/IFIP Conference on Software Architecture*, 2011, pp. 83–92. [Online]. Available: https://doi.org/10.1109/WICSA.2011.20

[49] O.-A. Schipor, R.-D. Vatavu, and J. Vanderdonckt, "Euphoria: A scalable, event-driven architecture for designing interactions across heterogeneous devices in smart environments," *Information and Software Technology*, vol. 109, pp. 43–59, 2019. [Online]. Available: https://doi.org/10.1016/j.infsof.2019.01.006

[50] D. X. Jara Juárez and P. Cedillo, "Security of mobile cloud computing: A systematic mapping study," in *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*, 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ETCM.2017.8247486

[51] J. L. González, R. García, J. M. Brunetti, R. Gil, and J. M. Gimeno, "Swet-qum: A quality in use extension model for semantic web exploration tools," in *Proceedings of the 13th International Conference on Interacción Persona-Ordenador*, ser. INTERACCION '12. New York, NY, USA: Association for Computing Machinery, 2012. [Online]. Available: https://doi.org/10.1145/2379636.2379651

[52] S. Farshidi, S. Jansen, and S. Fortuin, "Model-driven development platform selection: Four industry case studies," *Softw. Syst. Model.*, vol. 20, no. 5, p. 1525–1551, oct 2021. [Online]. Available: https://doi.org/10.1007/s10270-020-00855-w