This article has been accepted for publication in IEEE Software. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MS.2024.3479880

Test-Driven Development Tool for IoT Systems

Gleiston Guerrero-Ulloa*†, Duval Carvajal-Suárez*, Paulo Novais‡ Miguel J. Hornos†, Carlos Rodríguez-Domínguez†
*Faculty of Engineering Sciences, State Technical University of Quevedo, Quevedo, Ecuador
gguerrero@uteq.edu.ec, duval.carvajal2017@uteq.edu.ec
†Software Engineering Dept., Research Centre for Information and Communication Technologies (CITIC-UGR), Aynadamar Campus, University of Granada, 18071 Granada, Spain
gleiston@correo.ugr.es, mhornos@ugr.es,carlos.rodriguez@ugr.es
‡Department of Informatics, Algoritmi Center, University of Minho, 4710-057 Braga, Portugal
pjon@di.uminho.pt

Abstract—TDDT4IoTS is a comprehensive tool designed to streamline the development of IoT systems by integrating both device design and software application development within a test-driven framework. Its test-driven development approach enhances the reliability and efficiency of IoT systems by ensuring early error detection and facilitating the integration of diverse technologies. By addressing critical challenges such as interoperability and standardization, TDDT4IoTS empowers developers to overcome key obstacles in IoT adoption, ensuring scalable, robust, and feature-rich applications. The tool's strong usability, validated through a System Usability Scale (SUS) evaluation, along with future enhancements like Artificial Intelligence (AI)-driven automation for keyword marking and expanded platform support, underscores its significant contribution to accelerating IoT innovation and development.

Index Terms—Internet of Things (IoT), CASE tool, test-driven development, extended use case, symbol language, model generation, software generation, IoT development, usability evaluation, software modeling

1 Introduction

Recognizing the transformative potential of the Internet of Things (IoT) across various domains, it is worth noting that several challenges, such as a lack of standardization, system interoperability, and the high computational complexity of AI, hinder its adoption [1].

Rapid technological progress demands accelerated IoT System (IoTS) development to stay aligned with current trends. Efficient IoTS development in sectors like healthcare, security, and energy can lead to significant improvements [2]. Therefore, increased investment and commitment to developing robust IoT solutions are essential for effective digital transformation.

IoTS development has advanced with the establishment of best practices and tools. Methodologies such as INTER-Meth, Servus, and TDDM4IoTS have been developed [1]with the latter aligning with system and software lifecycles [3] and providing a systematic framework for IoTS development.

This work was supported by the grant PID2022-139297OB- I00 funded by MICIU/AEI/10.13039/501100011033 and by ERDF/EU.

However, no tool currently provides comprehensive support for both device and software development for IoTSs. Tools like Tinkercad Circuits and Arduino IDE enhance IoT device modeling, testing, and verification [4], but they lack alerts for the misuse of component input/outputs, which is particularly important for novice developers and can lead to device malfunctions.

Additionally, Savio et al. [5] analyzed 13 open-source software tools for modeling systems using Unified Modeling Language (UML) diagrams, including ArgoUML, StarUML, UMLet, DiaUML, BOUML, Violet, UML Designer, Modelio, NClass, PlantUML, Umbrello, Open ModelSphere, and Papyrus. Their study concluded that StarUML and UML Designer are the most effective tools for creating modeling diagrams. However, these tools require developers to create each diagram separately, which risks a lack of logical consistency between different diagrams.

To overcome these limitations, we introduce TDDT4IoTS (Test-Driven Development Tool for IoTSs). This tool enables the creation of behavioral models, such as Use Case Diagrams (UCDs) and Sequence Diagrams (SDs), as well as structural models like Class Diagrams (CDs), which are automatically generated from requirements gathered through Detailed Use Cases (DUCs). Requirement acquisition is critical in software development [1], [3], [6], [7], [8], [9], [10]. Furthermore, TDDT4IoTS facilitates the generation of unit tests from DUCs.

2 Related Work

This section reviews various tools that can accelerate the development process of software applications, including modeling and code generation (CodGen), as well as tools for IoT hardware design and configuration.

Node-RED's graphical interface and visual programming facilitate IoTS development by enabling prototyping and application development through visual data flows and services. However, Node-RED is limited to hardware management and does not support the development of software applications for user interaction or data monitoring [11].

ReqMIoT [12] is an integrated modeling environment designed to facilitate the elicitation, analysis, and specification of IoTSs. It provides a domain-specific textual Use Case (UC) modeling environment and automated mapping support for generating UCDs, partial domain models, and summary tables. However, its scope is limited to the requirements analysis and specification stage of the IoTS development process.

In contrast, RM2DM [13] is a tool for generating object-oriented design models for enterprise information systems, using validated requirements models as input. By focusing on the transition from requirements to the design phase, RM2DM enables the generation of comprehensive CDs and SDs, capturing both the static structure and dynamic behavior of the system. The main advantage of RM2DM is its ability to automate the design process; however, it is not specifically focused on IoTSs.

Various software modeling (SoftMod) tools (e.g., ArgoUML, StarUML, UMLet, etc.) allow for the creation of UML diagrams and CodGen. Some other similar tools (e.g., Violet, UML Designer, etc.) lack the latter functionality. The need to create separate diagrams in these tools leads to inefficiency when dealing with changes in system requirements [1], [6].

In terms of CD modeling, approaches have been explored for generating UCDs and CDs from natural language text using machine learning or heuristic rules [6]. However, challenges arise due to different semantic meanings of the same text fragment, which hinders precise machine understanding. Complex infrastructure requirements also limit widespread adoption. Dawood & Sahraoui [7] provide a review of CD generation using natural language processing, highlight merits and demerits of 13 different approaches.

TDDT4IoTS stands out as a tool for IoTS development by integrating a systematic approach based on Test-Driven Development (TDD), ensuring robustness and reliability through early error detection. This tool not only addresses key issues of interoperability and communication but also facilitates the automation of behavior modeling and structural diagrams, something that previous tools like Node-RED or ReqMIoT do not achieve comprehensively. The primary contribution of TDDT4IoTS lies in its ability to automatically generate unit tests from DUCs, setting it apart from the aforementioned tools, which primarily focus on graphical design or requirements specification.

Unlike existing tools that tend to fragment the development process between hardware configuration and application development, TDDT4IoTS provides a comprehensive solution for both aspects, supporting both experienced and novice developers. This innovation adds significant value, particularly in educational contexts or for novice developers, surpassing the limitations of tools like UMLet and StarUML, which do not offer support for the integration of physical components with software. Furthermore, TDDT4IoTS innovates by offering customizable component configurations and data exchange methods, addressing the challenges faced by other tools in terms of component management and communication in IoTSs.

3 TDDT4IoTS

TDDT4IoTS is a web-based tool built using a layered architecture, designed with microservices and cloud computing for scalability and interoperability across various platforms [14]. Optimized database and application code, along with efficient client-side resource usage, reduce server workload.

This tool supports project planning, progress monitoring, and IoTS development, facilitating both the modeling and generation of software for user interaction with IoT devices (IoTDs). It also enables IoTD design and generates configuration code [8], [9], [10].

Its source code and user manual are available on GitHub at https://github.com/dcarvajals/TddT4IoTs.

3.1 User Management and Authentication

User authentication and access control are essential components of TDDT4IoTS. Users are identified by their email addresses, which serve as unique usernames. Passwords are encrypted and securely stored in PostgreSQL. Forgotten passwords are reset via an emailed codechr [2].

Project management features enable team members to share projects, facilitating collaboration and controlling access. Invitations are accessible through the user's dashboard or, for non-users, via email, ensuring streamlined access to shared resources.

3.2 TDDT4IoTS Layered Architecture

The architecture of the TDDT4IoTS application is showed in Figure 1. It follows a four-layer client/server architecture: the front-end layer (client), the middleware layer, the business layer or server, and the data management layer.

TDDT4IoTS is an open-source and free tool under the MIT License, which permits use, modification, and distribution. It employs various open-source technologies, including GoJS for data visualization, which offers free usage for open-source projects under certain conditions. For more details on GoJS licensing, please refer to the documentation provided by Northwoods Software.

3.2.1 *Front-End Layer*

The technologies used in the Front-End layer not only facilitate the creation of user-friendly interfaces but also accessibility across various devices. This layer leverages HTML5, CSS3, and JavaScript to enhance interactivity and functionality in web pages. Additional, frameworks and libraries such as AngularJS, jsUML2, Bootstrap, jQuery, Store2, SweetAlert2, Toastr, and GoJS streamline development by offering reusable components, UML modeling tools, responsive styles, DOM manipulation, notifications, and data visualization.

3.2.2 *Middleware Layer*

Security is managed using JSON Web Token (JWT), which facilitates secure access to web services. Communication between client and server is handled through RESTful web services in JSON format. TDDT4IoTS allows developers to collaborate on the design of the same device, with real-time updates synchronized via WebSockets, which are also used

This article has been accepted for publication in IEEE Software. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MS.2024.3479880

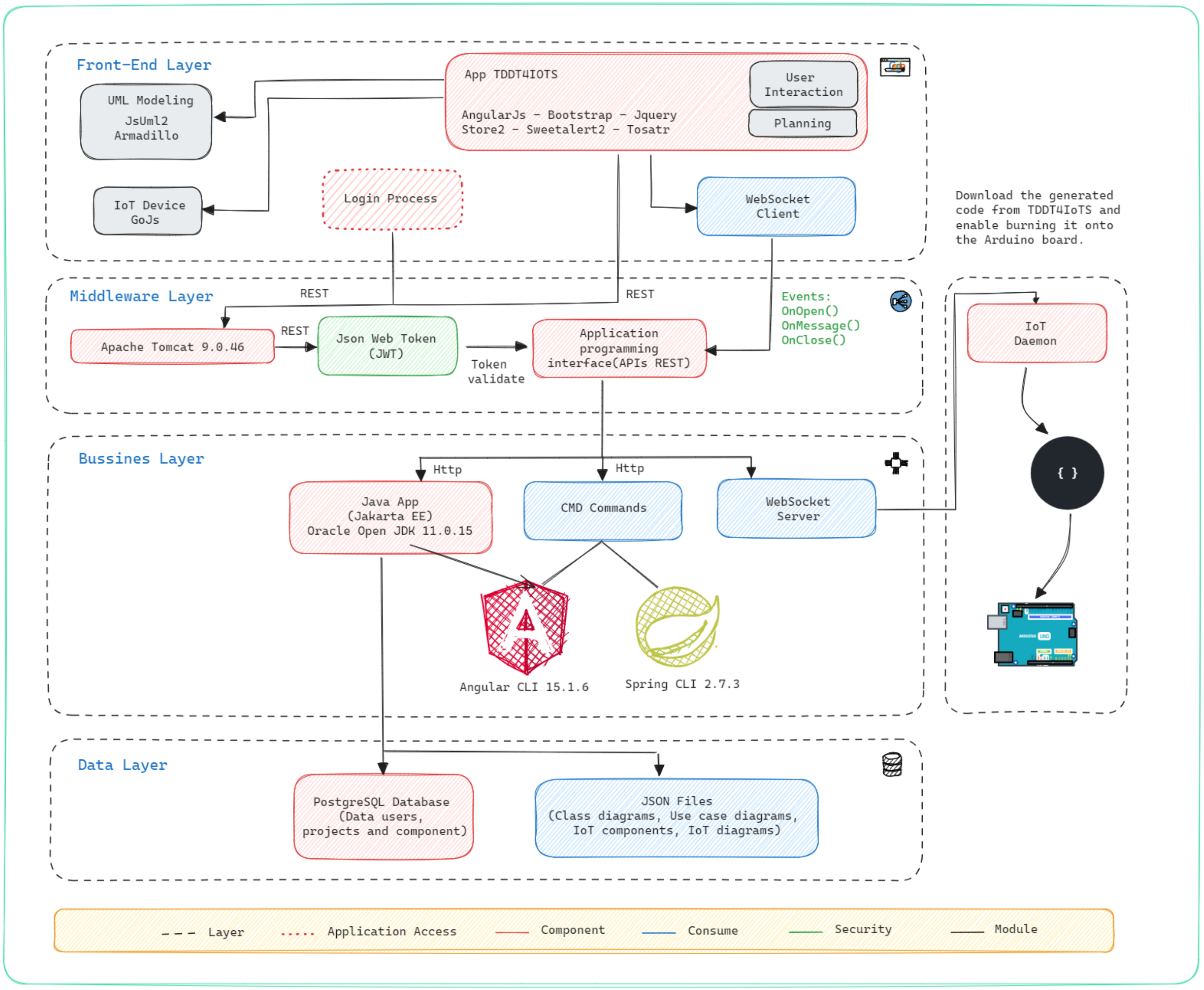

Fig. 1. Architecture of the TDDT4IoTS application, structured as a four-layer client/server architecture: the client (front-end) uses HTML5, CSS3, and JavaScript; the middleware is managed by Apache Tomcat and RESTful web services in JSON format; the server contains business logic developed in Java with JDK 17 and Maven, and data access is handled through the Java Persistence API; and the data management layer utilizes PostgreSQL and JSON files. Additionally, various JavaScript frameworks and libraries, such as AngularJS, jsUML2, Bootstrap, jQuery, Store2, SweetAlert2, Toastr, and GoJS, enhance interactivity, styling, and data visualization in the user interface.

to upload code to the Arduino board. Apache Tomcat is used to deploy and run the web application.

Arduino, being the most popular platform for developing IoTSs, was prioritized in this initial version of TDDT4IoTS [8], [9], [10]. As a result, the tool is equipped to work with Arduino boards and compatible devices.

3.2.3 Business Layer

The business logic of the TDDT4IoTS application is developed in Java using JDK 17. Maven is utilized as a project management tool to organize the project and streamline its lifecycle, including building, compiling, and managing dependencies. Additionally, Java classes are used in conjunction with Angular CLI and Spring CLI to generate projects.

3.2.4 Data Layer

The Java Persistence API is used to access and manipulate data in relational databases, with JDBC facilitating communication between the business and persistence layers.

PostgreSQL is employed to manage user and project data, while component and diagram data are stored in JSON files. Together, these tools and technologies help build efficient, visually appealing, and feature-rich web applications. They provide a well-organized project structure, simplify dependency management, and enable seamless client-server interaction. The application is designed modularly, with distinct modules for planning, modeling, and IoTD management.

3.3 Modular Design of TDDT4IoTS

The tool is modularly designed and consists of three main modules, explained in the following subsections. The latter two modules are dedicated to SoftMod and IoTD development, respectively.

3.3.1 Planning Module

In this module, project members and their roles are defined. Project deliverables are outlined, each contributing to 100% of the project's total scope. Deliverables are broken down into activities, each assigned a percentage of the overall deliverable. Tasks are then assigned to developers, including the project facilitator, with task percentages summing to 100% of the corresponding activity. Progress tracking is conducted to ensure the successful completion of each deliverable.

3.3.2 UML Modeling Module

IoTS development is divided into two modules. In the UML modeling module, the process begins with creating a UCD, representing either high-level or detailed requirements. Each UC is documented using a Symbol Language (SymLan) to highlight key elements for generating the corresponding CD. The more detailed the UCs, the more effective the tool becomes.

Once the UCs are detailed, the tool automatically generates the CD, followed by the creation of unit tests by specifying evaluation values for the IoTS. The project is then generated in Java for the back-end (using PostgreSQL, SQL Server, or MySQL for the database) and Angular for the front-end, following the Maven structure with Spring Boot. The generated code includes entity classes, data access classes, and RESTful web services or Spring Boot controllers.

TDDT4IoTS also allows CDs to be created from scratch, with required elements stored in JSON files, ensuring compatibility and easy integration with other tools and platforms.

3.3.3 IoT Module

The IoT module allows developers to add new components to their physical devices, specifying the necessary code, including configuration libraries. Developers can define pin types (analog or digital) and functionality (input/output). TDDT4IoTS offers a wizard interface for configuring these components, including dual-function pins and setting parameters for edge processing or cloud transmission. For data transmission components like WiFi or Bluetooth, developers can set parameters for data exchange via HTTP, web services, or custom protocols like WebSockets. Simple data entries enable input for specifications like URLs and credentials. Protocols such as ZigBee, WirelessHART, and Message Queuing Telemetry Transport (MQTT), among others, are also supported [14]. The IoT daemon uploads control code to boards, interacts with the hardware, and is downloaded locally to the developer's computer.

4 Case Studies

Several studies presented at prominent conferences and published in scientific journals have referenced the TDDT4IoTS tool for system development, including IdeAir [8], Ñawi [9], and P4L [10]. TDDT4IoTS has been used for designing and configuring IoTDs and for developing the front-end (Angular) and back-end (Spring Boot with Java) of user applications.

As a case study for this paper, we present the P4L system, version 2.0, which provides information on plant status and environmental pollution levels. The development results using our tool are shown in Figure 2, illustrating key user interface (UI) screenshots:

(a) Displays the UI for creating and viewing UCs, actors, and their relationships.
(b) Shows the list of the SymLan symbols and their meanings for use in DUCs.
(c) Shows a DUC, where SymLan is used to specify details such as the description, preconditions, postconditions, and action sequences.
(d) Displays a CD automatically generated from annotated UCs, depending on the detail and correct use of SymLan.
(e) Allows developers to specify input data for methods to generate the tests required before refactoring, a common task in TDD.
(f) Shows part of the back-end project structure edited in NetBeans 18. Developers can generate and download Java code using Spring Boot for the back-end and Angular for the front-end, packaged in a zip file with a test package, compatible with popular IDEs such as NetBeans, Eclipse, and IntelliJ IDEA.

In projects developed using TDDT4IoTS, an average of 95% of the generated code is utilized, with discarded code generally pertaining to non-entity classes in UCs that do not require persistence management.

Another notable advantage of TDDT4IoTS is its support for the design and CodGen of IoTDs. Figure 3 displays several UIs related to IoTD development.

TDDT4IoTS allows developers to add new electronic components to their IoTDs. Adding a new device involves uploading the component's vectorized image (see figure 3(a)), which will be used by the graphical UI for representation or preview.

Developers must specify the names and types of the component's input and output ports, along with the signal types they handle (see figure 3(b)).

An administrator must approve the component's inclusion for use by the entire TDDT4IoTS community.

For communication components, developers can specify the communication protocol (e.g., WiFi, Bluetooth, etc.) and the data exchange protocol (e.g., REST, WebSockets, etc.) to generate code that reduces the time and effort required to write the complete program controlling the device, including classes, libraries, and functions. Figure 3(c) shows version 2.0 of P4L, illustrating the use of these new components in IoTD development.

5 Evaluation of TDDT4IoTS

The TDDT4IoTS tool was evaluated using the 10 questions from the System Usability Scale (SUS) questionnaire by Brooke [15], correlating the results with developers' experiences. Additionally, the experience of developers who have extensively used TDDT4IoTS was also evaluated.

This article has been accepted for publication in IEEE Software. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MS.2024.3479880

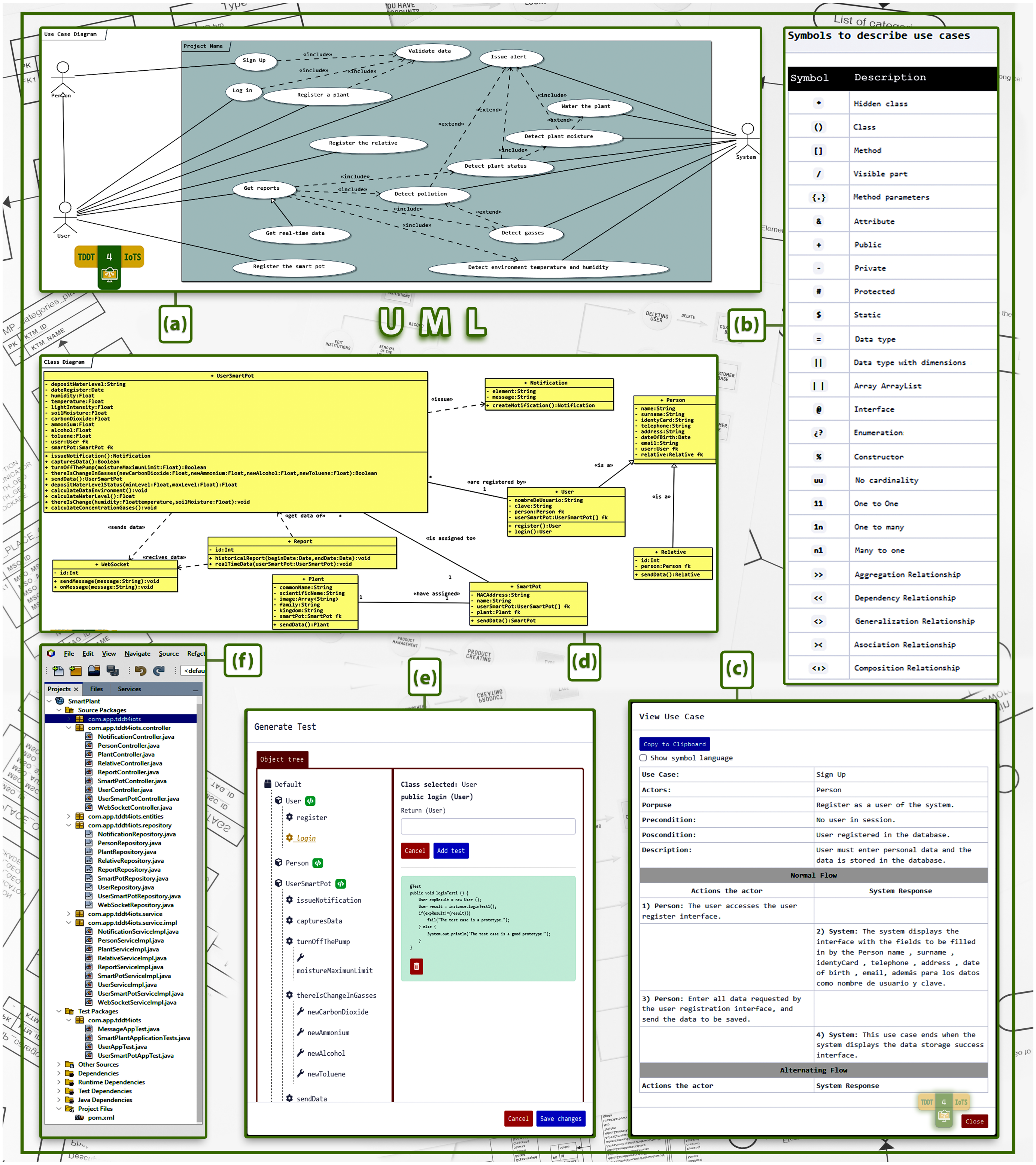

Fig. 2. Screenshots of various UIs from the SoftMod module in TDDT4IoTS, illustrating the basic workflow for modeling IoTS software: (a) creating a UC, (b) consulting SymLan symbols and their meanings, (c) DUC, (d) reviewing the CD, (e) generating unit tests, and (f) completing the code after downloading it.

5.1 Usability Evaluation

Table 1 presents the statistics for each SUS question, reflecting developers' perceptions of the usability of TDDT4IoTS. It includes central tendency measures, analysis of variance (ANOVA) data, and the correlation between 'Experience in using the tool' and each question (1-10), with values rounded to two to four decimal places.

The SUS questionnaire provides valuable insights into users' perceptions of TDDT4IoTS. The survey targeted first- and second-year software engineering students to evaluate

This article has been accepted for publication in IEEE Software. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MS.2024.3479880

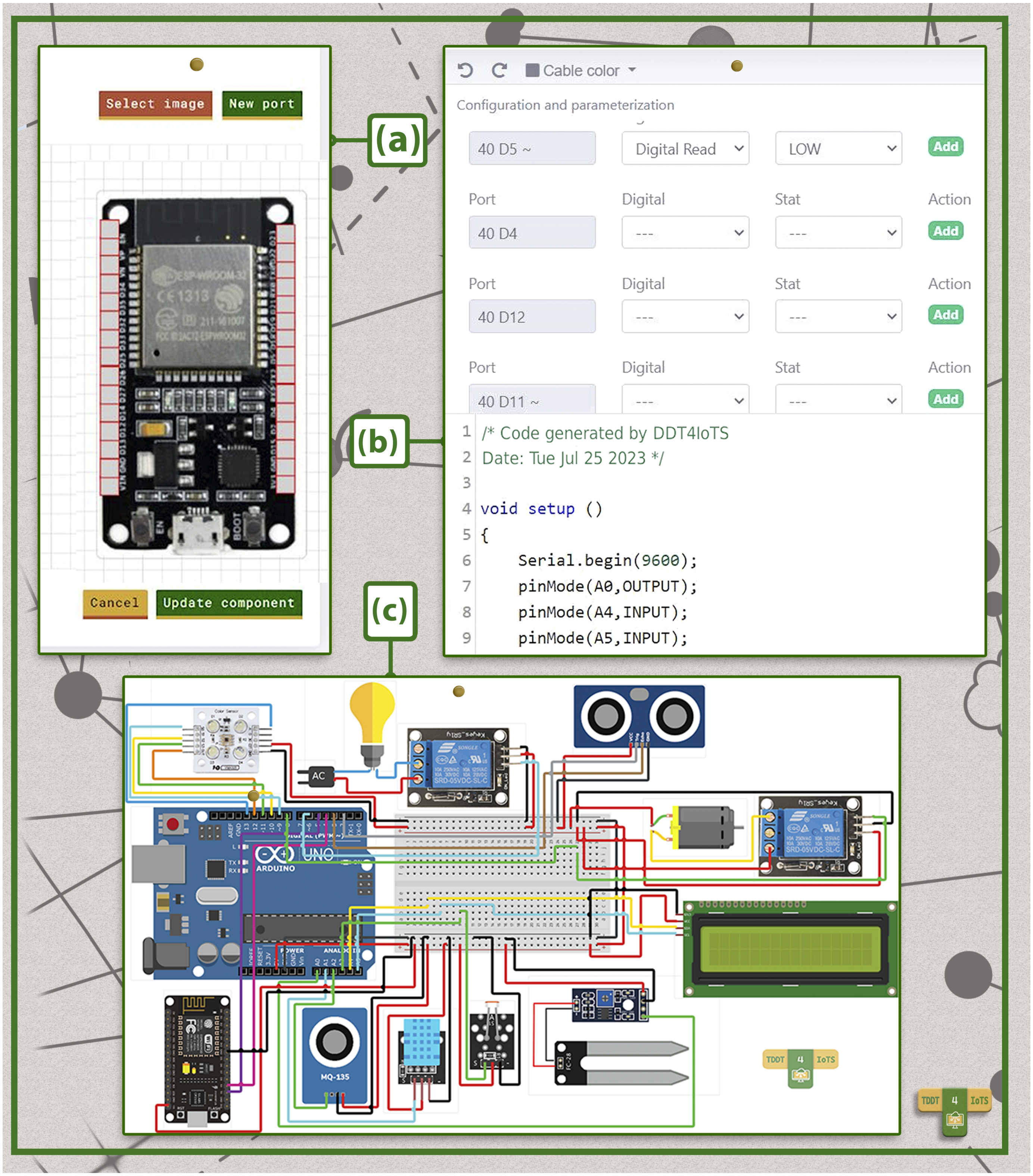

Fig. 3. Screenshots of the IoTD development module UIs in TDDT4IoTS, illustrating the workflow: (a) adding a new device if it does not already exist, (b) specifying its characteristics, and (c) interconnecting hardware components, and generating Arduino code.

the tool's intuitiveness and ease of use, minimizing potential biases from experienced users. The overall SUS score was 72.26%, indicating good usability, though there were significant variations in perceived complexity and the need for technical assistance.

These issues are linked to participants' limited experience and knowledge of object-oriented design. The high standard deviation in responses indicates variability in user perceptions, with some finding the tool intuitive while others encountered challenges.

TABLE 1
Central Tendency Measures and ANOVA and Correlation Analysis by Experience and Question

Measure	Experience	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Mean	First time	4.04	3.09	3.91	3.57	3.87	3.39	3.78	3.65	3.91	3.13
	Less than 3 times	3.64	3.18	3.55	3.09	4.09	3.82	3.09	3.55	3.64	2.82
	3-5 times	4.22	3.56	3.78	3.56	4.33	4.00	4.11	3.78	4.11	2.67
	More than 5 times	4.13	3.63	3.63	3.13	4.13	3.38	3.25	3.25	3.75	3.38
	General	**4.00**	**3.27**	**3.76**	**3.39**	**4.04**	**3.59**	**3.61**	**3.59**	**3.86**	**3.02**
Median	First time	4.00	3.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	3.00
	Less than 3 times	4.00	4.00	4.00	3.00	4.00	4.00	3.00	4.00	4.00	3.00
	3-5 times	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	3.00
	More than 5 times	4.00	4.00	4.00	3.00	4.00	3.50	3.00	3.00	4.00	3.50
	General	**4.00**	**4.00**	**4.00**	**4.00**	**4.00**	**4.00**	**4.00**	**4.00**	**4.00**	**3.00**
Mode	First time	4.00	2.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	2.00
	Less than 3 times	4.00	4.00	4.00	4.00	4.00	4.00	3.00	4.00	4.00	3.00
	3-5 times	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	3.00
	More than 5 times	4.00	4.00	4.00	4.00	4.00	3.00	3.00	3.00	4.00	4.00
	General	**4.00**	**4.00**	**4.00**	**4.00**	**4.00**	**4.00**	**4.00**	**4.00**	**4.00**	**4.00**
Std Dev	First time	0.767	0.996	0.793	1.161	0.757	0.891	0.736	0.935	0.733	1.014
	Less than 3 times	0.505	1.079	0.522	1.044	0.539	0.603	0.701	0.688	0.505	1.25
	3-5 times	0.667	0.726	0.833	0.726	0.500	0.500	0.601	0.833	0.601	1.118
	More than 5 times	0.641	0.518	0.744	0.835	0.354	0.744	0.707	0.707	0.463	1.061
	General	**0.693**	**0.918**	**0.737**	**1.021**	**0.631**	**0.779**	**0.777**	**0.829**	**0.633**	**1.086**
ANOVA: Experience /Q_{1-10}	Sum of Squares	2.067	2.597	1.192	2.498	1.528	3.363	6.945	1.353	1.278	2.860
	Degrees of Freedom (df)	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
	F-statistic	1.4768	1.0285	0.7190	0.7882	1.3021	0.9523	4.6882	0.6422	1.0679	0.7985
	P-value associated with the F-statistic (PR>F)	0.2329	0.3885	0.5456	0.5065	0.2848	0.1341	0.0060	0.5917	0.3717	0.5009
Correlations	Experience/Q_{1-10}	0.0766	0.2398	-0.1327	-0.1175	0.2220	0.1095	-0.1188	-0.1105	-0.0203	-0.0006

Survey results indicate that usability issues are more pronounced among novice users, reflecting their initial difficulties with the tool. ANOVA analysis revealed notable differences in perceived inconsistency, highlighting the need to improve the tool's coherence for novice users. Overall, the tool was well received, but improvements in complexity and technical assistance are needed to enhance usability and educational effectiveness.

5.2 Evaluation of Developer Experience

Since its development and through its latest improvements, TDDT4IoTS has been used by all surveyed developers, including the authors of IdeAir [8], Ñawi [9], P4L [10], and other IoTs developed with this tool. A total of 21 developers were surveyed, all of whom were students or former students of Systems Engineering and Software Engineering, with experience ranging from 1 to 9 years.

The survey included questions about years of experience in IoTS development, agreement on the importance of UCDs for modeling system behavior, the role of CDs as crucial structural diagrams for system modeling (SysMod), and the appropriateness of using the SymLan language in DUCs for automatic model generation.

All developers (100%) agreed that UCDs are essential for modeling system behavior and that CDs are vital for structural modeling. Regarding the appropriateness of SymLan, 96% of respondents agreed, while 4% were neutral. Additional questions explored the interest in automating SD creation in SysMod, the importance of state diagrams (StDs) in SysMod, and overall interest in TDDT4IoTS.

With the exception of the question regarding StDs, all other questions received 100% responses indicating 'Strongly important' or 'Very important'. The StD question saw 9.5% of respondents express moderate importance opinion, while the rest rated it as 'Strongly important' or 'Very important'.

6 Conclusions and Future Work

In conclusion, TDDT4IoTS is a crucial tool for developing high-quality IoTSs, addressing common challenges and filling significant gaps in the field. Its TDD-based approach ensures the creation of robust and reliable systems through early error detection and requirement traceability, enabling rapid and efficient development. The architecture leverages a range of technologies to deliver feature-rich web applications and sophisticated IoT solutions.

Case studies demonstrate that TDDT4IoTS enhances developer productivity. By automating the modeling and CodGen processes for device interaction and configuration, developers can focus on core functionalities. This addresses interoperability and communication challenges, ensuring that IoT systems meet the desired requirements and improve customer satisfaction.

The SUS evaluation scored 72.26%, indicating good usability. Future enhancements may incorporate AI techniques for automatic keyword marking in generating CDs and the creation of SDs and StDs, which would add significant value.

Currently, TDDT4IoTS supports web application development, IoTD design, and the configuration of Arduino boards and variants like ESP8266. Future work will expand support to additional platforms, including MediaTek LinkIt Smart 7688 Duo and the ROHM IoT Kit.

Security is presently managed at the discretion of developers. Future versions will implement enhanced authentication services, such as multi-factor authentication and digital certificates, to strengthen security.

We plan a comprehensive evaluation of TDDT4IoTS, focusing on cost savings and fault detection improvements, based on usage data and user feedback. Additionally, we will compare TDDT4IoTS with other IoT tools to better contextualize its advantages and limitations.

ACKNOWLEDGEMENTS

This work was supported by the grant PID2022-139297OB-I00, funded by MICIU/AEI/10.13039/501100011033 and by ERDF/EU.

REFERENCES

[1] G. Guerrero-Ulloa, C. Rodríguez-Domínguez, and M. J. Hornos, "Agile methodologies applied to the development of Internet of Things (IoT)-based systems: A review," *Sensors*, vol. 23, pp. 790–824, 1 2023. [Online]. Available: https://doi.org/10.3390/s23020790

[2] R. Sharma and N. Sharma, "Attacks on resource-constrained iot devices and security solutions," *International Journal of Software Science and Computational Intelligence*, vol. 14, pp. 1–21, 10 2022. [Online]. Available: https://doi.org/10.4018/ijssci.310943

[3] G. Guerrero-Ulloa, M. J. Hornos, and C. Rodríguez-Domínguez, "TDDM4IoTS: A test-driven development methodology for Internet of Things (IoT)-based systems," *Communications in Computer and Information Science*, vol. 1193 CCIS, pp. 41–55, 12 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-42517-3_4

[4] N. Y. Pérez Suárez and S. H. Ruiz Obando, "Characterization of electronic circuit simulation software as alternatives for use in higher education," Ph.D. dissertation, University of Santander, Medellin, Colombia, 2020. [Online]. Available: https://repositorio.udes.edu.co/server/api/core/bitstreams/79e61716-264b-4d64-8a04-87a0575ab657/content

[5] E. Sávio, S. Freire, G. C. Oliveira, M. Eurizene, S. Gomes, and M. E. D. Sousa, "Analysis of open-source case tools for supporting software modeling process with UML," in *ACM International Conference Proceeding Series*, A. Malucelli and S. Reinehr, Eds. Association for Computing Machinery, 10 2018, pp. 51–60. [Online]. Available: https://doi.org/10.1145/3275245.3275251

[6] M. Ramzan, G. S. Sadiqi, M. S. Bashir, S. Raza, and A. Batool, "A rule-based approach for automatic generation of class diagram from functional requirements using natural language processing and machine learning," *Journal of Computing & Biomedical Informatics*, vol. 7, pp. 1–16, 2024. [Online]. Available: https://doi.org/10.56979/702/2024

[7] O. S. Dawood and A.-E.-K. Sahraoui, "From requirements engineering to UML using natural language processing - survey study," *European Journal of Engineering and Technology Research*, vol. 2, no. 1, pp. 44–50, 2017. [Online]. Available: https://laas.hal.science/hal-01703317/document

[8] G. Guerrero-Ulloa, A. Andrango-Catota, M. Abad-Alay, M. J. Hornos, and C. Rodríguez-Domínguez, "Development and assessment of an indoor air quality control IoT-based system," *Electronics*, vol. 12, p. 608, 1 2023. [Online]. Available: https://doi.org/10.3390/electronics12030608

[9] G. Guerrero-Ulloa, A. Fernández-Loor, F. Moreira, P. Novais, C. Rodríguez-Domínguez, and M. J. Hornos, "Validation of a development methodology and tool for IoT-based systems through a case study for visually impaired people," *Internet of Things*, p. 100900, 2023. [Online]. Available: https://doi.org/10.1016/j.iot.2023.100900

[10] G. Guerrero-Ulloa, A. Méndez-García, V. Torres-Lindao, V. Zamora-Mecías, C. Rodríguez-Domínguez, and M. Hornos, "Internet of Things (IoT)-based indoor plant care system," *Journal of Ambient Intelligence and Smart Environments*, vol. 15, pp. 47–62, 3 2023. [Online]. Available: https://doi.org/10.3233/ais-220483

[11] J. P. Dias, B. Lima, J. P. Faria, A. Restivo, and H. S. Ferreira, *Visual Self-healing Modelling for Reliable Internet-of-Things Systems*. Springer, 2020, pp. 357–370. [Online]. Available: https://doi.org/10.1007/978-3-030-50426-7_27

[12] P. Boutot and S. Mustafiz, "ReqMIoT: An integrated requirements modelling environment for IoT systems," in *5th International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT 2023)*, 2023, pp. 38–45. [Online]. Available: https://doi.org/10.1109/serp4iot59158.2023.00012

[13] Z. Tian, Y. Yang, and S. Cheng, "RM2DM: A tool for automatic generation of OO design models from requirements models," in *45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion 2023)*, 2023, pp. 36–40. [Online]. Available: http://doi.org/10.1109/icse-companion58688.2023.00020

[14] N. Monios, N. Peladarinos, V. Cheimaras, P. Papageorgas, and D. D. Piromalis, "A thorough review and comparison of commercial and open-source IoT platforms for smart city applications," *Electronics*, vol. 13, 2024. [Online]. Available: https://doi.org/10.3390/electronics13081465

[15] J. Brooke, *Usability Evaluation In Industry*, 1st ed., P. W. Jordan, B. Thomas, I. L. McClelland, and B. Weerdmeester, Eds. CRC Press, 6 1996.

Gleiston Guerrero-Ulloa is an Associate Professor at the State Technical University of Quevedo, Quevedo, 120501, Ecuador. His research interests encompass the Internet of Things, Software Engineering, and Human Interaction. Prof. Guerrero-Ulloa earned his Master's Degree in Software Development and his Ph.D. in Information and Communication Technologies from the University of Granada. He has recently started his career as a reviewer for scientific articles in various high-impact international journals. You can reach him at gguerrero@uteq.edu.ec or gleistonguerrero@gmail.com.

Duval Carvajal-Suárez is a Systems Engineer and developer at Viamatica, located in Guayaquil, Ecuador. His research interests include software architecture, IoTSs, and server configuration. Mr. Carvajal-Suárez graduated from the State Technical University of Quevedo and is currently pursuing a Master's Degree in Software Development at the University of Granada. He is a co-founder of CleanCode, a company based in Guayaquil, Ecuador. He has begun his research career, with his first paper published in the Journal of Technology and Science Education. He can be contacted at duvalcarvajal2000@gmail.com or duval.carvajal2017@uteq.edu.ec.

Paulo Novais is a Full Professor of Computer Science at the Department of Informatics, the School of Engineering, the University of Minho (Portugal) and a researcher at the ALGORITMI Centre in which he is the leader of the research group ISLab - Synthetic Intelligence lab, and the coordinator of the research line Computer Science and Technology (CST). He is the director of the PhD Program in Informatics and co-founder and Deputy Director of the Master in Law and Informatics at the University of Minho. His interest, in the last years, was absorbed by the different, yet closely related, concepts of Ambient Intelligence/Ambient Assisted Living, Conflict Resolution, behavioral Analysis, Intelligent Tutors and the incorporation of AI methods and techniques in these fields. He can be contacted at pjon@di.uminho.pt

Miguel J. Hornos is currently an Associate Professor at the University of Granada at Granada, 18071, Spain, where he previously received his PhD in Computing. His main research interests include Internet of Things and Intelligent Environments, Multi-Criteria Decision-Making, and Software Reliability. Prof. Hornos is an Editorial Board member of the Journal of Reliable Intelligent Environments, published by Springer, where he has also acted as Guest Editor of three Special Issues. He is currently serving as General Workshops co-chair for the 20th Intl. Conf. on Intelligent Environments (IE 2024). He is also serving as Guest Editor of several Special Issues in three different prestigious and JCR-indexed international journals. Contact him at mhornos@ugr.es.

Carlos Rodríguez-Domínguez is Associate Professor at the University of Granada, Granada, 18071, Spain. His research interests include Ubiquitous Computing, Collaborative Systems and OLAP systems. Prof. Rodríguez-Domínguez received his PhD in Computing from the University of Granada. He is co-founder of Everyware Technologies, spin-off company of the University of Granada, Granada, 18010, Spain. He has also been Guest Editor of the Journal of Reliable Intelligent Environments, published by Springer, as well as reviewer in several JCR-indexed international journals and guest editor in one of them. Contact him at carlosrodriguez@ugr.es.