More Accurate Cost Estimation for Internet of Things Projects by Adaptation of Use Case Points Methodology

Radek Silhavy, Miroslav Bures, Melchizedek Alipio, *Member, IEEE*, and Petr Silhavy

***Abstract*—This article adapts the use case points (UCPs) method to estimate the size and development effort (DE) required for the Internet of Things systems. Despite the extensive use of UCP in software engineering, it has yet to be adapted for IoT systems, which is essential for project management and resource planning. Our proposed adaptation, UCP for IoT, is based on a four-layer IoT architecture and tailors the standard software UCP to the specifications of IoT systems. It was validated using a case study of three IoT systems, demonstrating its applicability and effectiveness in estimating the DE required for IoT projects. However, the results also highlight the need for further improvements, particularly given the absence of historical data sets for IoT projects. Our future work will focus on gathering such data sets and further refining the proposed model.**

***Index Terms*—Development effort (DE) estimation, Internet of Things, size estimation methods, use case model, use case points (UCPs).**

I. Introduction

THIS article presents the benefits of adopting the use case points (UCPs) method for software or system size estimation in IoT systems. The motivation for this work stems from the observation that while significant research has been on various aspects of IoT design and modeling, discussions of size and cost estimations of IoT systems are significantly underresearched.

Recently, researchers have broached the topic of designing IoT-specific modeling in security or other design areas [1], [2]. However, only a few studies have focused on behavioral modeling in functional design [3], [4]. These studies allow the design of high-level functional models of IoT systems.

For example, Martins and Domingos [5] discussed adaptation business process modeling (BPMN) for modeling IoT systems' behavior and later using it to generate a platform neural code. Batool and Niazi [6] proposed a methodology for modeling complex scenarios in the IoT domain using a combination of complex networks and agent-based modeling. Suri [7] discussed allocating IoT resources in configurable business process models, which act as an extension of BPMN. Korkan et al. [8] extended the Thing Description standard proposal into sequential behavioral modeling. da Silva Fonseca et al. [9] proposed behavioral modeling based on Petri nets, while Song et al. [10] discussed a formal method for behavioral modeling of Smart IoT Systems using a combination of process algebra and lattice structures. Finally, Moghaddan et al. [11] discussed the Internet of Behaviors (IoB) concept, which focuses on connecting the digital world with human behavior to create intelligent connected systems that are human-driven in design, development, and adaptation.

While these studies have made significant contributions to IoT modeling, none of them discusses behavioral modeling as a baseline for the development effort (DE) estimation nor discusses a use case model as a functional description for IoT systems. The system size, DE, or cost estimation differs from behavioral modeling.

Our contribution is motivated by the growing potential for model-driven development (MDD), which brings visual languages such as the unified modeling language (UML) or system modeling language (SysML) to the IoT community. The use case modeling is well established in modern software and system engineering; therefore, the size estimation method, UCPs, can be introduced and used in the IoT domain. This article introduces the UCPs method to estimate software or system size and DE in IoT systems. By leveraging the well-established use case modeling approach in software and system engineering and the potential of MDD in the IoT domain, we provide a practical and effective solution for estimating the size of IoT systems, which can aid in project planning, resource allocation, and cost estimation.

This article discusses use case modeling for IoT systems and furt on the IoT-specific system size and DE method based on the UCPs approach. The following two research questions were investigated.

Manuscript received 23 May 2023; accepted 29 May 2023. Date of publication 31 May 2023; date of current version 24 October 2023. The work of Radek Silhavy and Petr Silhavy was supported by the Faculty of Applied Informatics, Tomas Bata University in Zlín under Grant RVO/FAI/2021/002. The work of Melchizedek Alipio was supported by the Department of Science and Technology-Science Education Institute (DOST-SEI) of the Republic of the Philippines through the Engineering Research and Advance Technology (ERDT) Postdoctoral Grant and the Office of the Provost of the De La Salle University Manila, Philippines. *(Corresponding author: Radek Silhavy.)*

Radek Silhavy and Petr Silhavy are with the Faculty of Applied Informatics, Tomas Bata University in Zlín, 760 05 Zlín, Czech Republic (e-mail: rsilhavy@utb.cz).

Miroslav Bures is with the System Testing Intelligent Laboratory, Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, 166 36 Prague, Czech Republic

Melchizedek Alipio is with the System Testing Intelligent Laboratory, Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, 166 36 Prague, Czech Republic, and also with the Department of Electronics and Computer Engineering, De La Salle University Manila, Manila, Philippines (e-mail: melchizedek.alipio@dlsu.edu.ph).

Digital Object Identifier 10.1109/JIOT.2023.3281614

RQ1: What are the use case model design rules of IoT systems?

RQ2: How should the software UCP methodology be adapted for IoT systems size/DE estimation to reflect the specifics of these systems?

This study contributes to the size and effort estimation methods related to the development of IoT systems. It introduces the adaptation of the UCP method, which is mainly studied and used in software engineering for IoT systems [12]. This UCP adaptation describes how a four-layer IoT architecture can be mapped to a size estimation model and tailors the standard software UCP to the specifications of IoT systems. The proposal was verified using a case study of a smart hydroponic system.

A. Research Highlights and Contribution

Although UCP is well-known, frequently adopted, and widely studied in software engineering, it has not yet been adapted for IoT systems. However, it is valuable as a size and DE estimator in IoT systems project management. It enables IoT project planning and can be used for updates and new function cost evaluations. The main highlights can be summarized as follows.

1) This study introduces the adaptation of the UCP method for estimating the size and effort required for developing IoT systems.
2) The UCP adaptation is based on a four-layer IoT architecture, and it tailors the standard software UCP to the specifications of IoT systems.
3) The proposal was validated using a case study of three IoT systems, demonstrating the applicability of the UCP method for estimating the DE required for IoT projects.
4) The study presents rules and recommendations for creating use case models for IoT systems and suggests that use case modeling can be adapted to fit the scope of IoT.
5) The study highlights the importance of system size estimation for project management and resource planning in IoT systems and suggests that existing effort estimation methods for software systems can be adapted for specific IoT environments.

The remainder of this article is organized as follows. Section II discusses related work. In Section III, the IoT specifications are discussed. Section IV discusses UCP adaptation, which discusses use case modeling and UCPs within the scope of IoT. Section V presents case studies. The results, discussion, and future directions are discussed in Section VI. Threats to validity are presented in Section VII. Finally, Section VIII concludes this article.

II. Related Works

Karner proposed the UCP method in 1993 [12], which is often proposed based on an analogy between projects [13], [14]. Generally, in cases where the focus is not on the scope of the proposed software system but on its implementation time, the DE must be calculated [15], which is the ratio between the UCP and the number of person-hours (PHs) required for its implementation. For large-scale applications, both the project and a corresponding IoT adaption [16] are created based on the assumption that, in the first iteration, all actors are considered moderately complex, whereas all use cases are considered complex. Ochodek et al. [17] recommended omitting actors entirely as model attributes. He also proposed another modification called scenario decomposition [17], which entails dividing scenarios into smaller ones with fewer steps. The necessity for UCP implementation is the significance of the UCP components [18], which was previously studied.

Several modifications have been proposed to improve the UCP method, including the use of extension associations to improve calculation accuracy [19] and the calculation of subcomponents using the Extended UCP and Modified UCP methods [15], [20], [21]. Other modifications focus on scenario decomposition and the identification of transactions and events within scenarios [22], [23], [24]. The internal structure of scenarios is also considered a key factor in the accuracy of UCP estimation [20].

Alternative approaches to modifying UCP methods include using Bayesian networks and fuzzy sets to create a probability estimation model [15] and adapting UCP for large-scale projects [16]. Some authors recommend to propose using linear regression models [25], [26] or neural network models [27], [28] for size estimation.

Overall, the ongoing efforts to modify and extend UCP methods reflect the importance of accurately estimating software system size in software engineering. The significance of UCP components, the methods used to calculate subcomponents, and the effects of scenario processing and transaction identification continue to be the subject of research and development in this field.

In the broader context of IoT project cost estimation, a contribution has been made by Evdokimov et al. [29]. Their work primarily estimates total costs and identifies cost-influencing factors in IoT projects. They contend that to implement and leverage IoT effectively in software engineering, it is crucial to resolve these cost estimation issues. Their study examines the aspects of IoT technology that impact costs, with the ultimate goal of providing clients with accurate project cost estimates before completion. Interestingly, Evdokimov et al. concluded that the program evaluation and review technique (PERT) offers a distinct advantage in accurately estimating IoT project costs. This finding aligns with us and highlighting the importance of considering diverse cost estimation techniques in IoT project management planning.

IoT system development methodologies contain effort estimation activities as their natural part. In this sense, the UCP DE estimation method is independent on particular development methodology. The ability to formulate a UML use case model is the only prerequisite for adopting UCP for an IoT project. In IoT system development, there are methodologies in which adoption of the UCP method may be complicated due to the lack of a requirements-gathering phase. To give a few examples, Ciccozzi and Spalazzese [30] introduced the MDE4IoT methodology, in which the requirements engineering phase is omitted. Similarly, the methodology presented by Khaleel et al. [31] also lacks a documented functional

gathering phase. On the contrary, several other methodologies assume that functional requirements are already known in advance, which enables the possibility of UCP adoption. Lekidis et al. [32] presented a model-driven approach based on a priority component network. Brambilla et al. [33] introduced MDD for IoT, with a focus on the user interface. Ito et al. [34] demonstrated the MDD approach with an example and show the effectiveness of use case modeling. The UML compliance of development methodologies has been studied by Guerro-Ulloa et al. [35], resulting in a proposal for a methodology that heavily focuses on requirements gathering and analysis. Harbouche et al. [36] presented a model-driven methodology with a documented requirements derivation process. Usländer and Batz [37] discussed an agile methodology for IoT that incorporates requirements and use cases.

As observed, MDD is a prerequisite for implementing UML/SysML in IoT development. Furthermore, it is also essential to utilize the UCP method in IoT projects. The MDD approach is usually understood as automation of development [38]. Pramudianto et al. [39] discussed an MDD in IoT software parts. Similarly, Conzon et al. [40] discussed MDD as automation. Brambilla et al. [33] introduced MDD for an IoT graphical interface design. However, when using an MDD, the model is typically not visualized. UML is used to model software solutions [41], [42] or systems in the UML profile, which are called SysML [43]. UML is the industry standard for visual modeling in MDE; it offers graphical description tools and diagrams illustrating various system properties. UML has several well-defined extension artifacts, such as stereotypes, constraints, values, and tags. These primary extension techniques enable designers to construct customized models for specific areas such as modeling system security issues or profiles representing IoT systems. The OMG systems modeling language (OMG SysML) [43] is a UML profile that enables system modeling. It is an extended subset of UML that facilitates the definition, analysis, design, verification, and validation of systems comprising hardware, software, data, persons, processes, and facilities. It also enables model and data transfer using XML metadata interchange (XMI). SysML is a visual modeling language that offers semantics and notation; it is not a technique or technology. SysML has also been extended to represent IoT systems. IoT systems with basic blocks such as sensors, actuators, and communication systems are software-intensive. Therefore, UML can effectively represent IoT systems because its graphical approach supports the interpretability of designed systems. Consequently, UML was proposed as an option for IoT systems modeling by [2] when a new profile for IoT system design and wrapper generator was presented.

III. IoT Development Specifics

IoT systems typically differ from ordinary software and systems in terms of complexity and heterogeneity, which naturally affects the accuracy of effort estimation methods. Software parts in IoT systems may differ from public software systems for a variety of reasons. Individual reasons, as listed below, are based on Fahmideh's review, which discusses software engineering perspectives for IoT [44].

1) Several programming languages can be used to implement the software components of an IoT system. A mix of OOP-based back-end languages with low-level languages used to implement firmware in devices can be present in the system [45], [46], [47], [48].
2) A wider spectrum of communication protocols can be employed in IoT systems. In addition, the employment of proprietary protocols is much more probable than that in the case of a purely software system. Furthermore, the level of standardization in the field of IoT is currently lower than that in software systems [47], [49].
3) In IoT systems, a mixture of different languages and programming styles combined with proprietary communication protocols makes the integration of individual system parts more challenging and prone to defects [47], [49].
4) In an IoT project, electronics and software specialists must collaborate. This heterogeneity, combined with insufficient knowledge of the technologies used, may prolong the actual implementation time [50], [51].
5) The nature of IoT systems generally implies a greater complexity and level of interconnection of their parts, which might impact the estimation process [52].
6) In some projects, IoT hardware is being developed concurrently with software. Thus, prolonging individual deadlines might impact the overall schedule more extensively than in a purely software-oriented project [53], [54].

Not every reason might impact every IoT project. However, all of these can significantly influence the accuracy of estimations, especially if the estimation method is developed and balanced for public software projects.

IV. UCP_{IoT}: UCP Adaptation for IoT Systems

For the UC model and UCP methodology, we consider IoT systems as systems comprising the following four layers.

1) The sensing layer contains sensors, actuators, or other devices for gathering, emitting, or processing data.
2) The network layer comprises network connectors and data gathering, including analog-to-digital (A/D) converters, data filters, and processing services.
3) The data processing layer is software-oriented, where data are preprocessed, and some fundamental analysis is performed. This layer is also a connector for clouds, business applications, and other systems that consume data.
4) The application layer comprises data management data; it is the layer of end-user application in all problem domains.

IoT use case modeling covers the sensing, network, and data-processing layers. The data-processing layer is not included into the use case modeling. In the UC model, actors trigger an activity, which can be considered as a conversation with a system. In the case of IoT systems, all members of the sensing, network, and data processing layers must be included as actors. Furthermore, in a typical UC model, actors may represent multiple entities. For modeling, we consider all actors

as external to the modeled system. Use cases represent high-level (textual) descriptions of the IoT functions. Each use case is represented by a primary scenario, and, whenever applicable, by several alternate methods. For the IoT, as a use case, all functions of the proposed systems need to be captured, and the behavior of the system includes the behavior of the actors. Both actors and use cases should follow the UML/SysML naming convention. Typically, actors are named using singular nouns in uppercase. Furthermore, the name of the sensor or actuator can be used; in any case, all actor names must be unique. In contrast, use cases are in verb form to emphasize an activity or process.

A. Use Case Model Design Approach for IoT

Use case diagrams (UC) are vital for describing IoT systems at a high level in terms of system goals and are identical for UML and SysML. System behavior is elaborated using system actors and described in detailed scenarios. A sample model is shown in (Fig. 1).

The UC model is defined by a diagram frame that represents the border of the system. In the diagram frame, a system boundary is used to illustrate the edge of the modeled system or package if the system decomposition is used. One way to determine this is to find actors. Subsequently, use cases were identified when searching for actors. The preparation of the UC was based on the following.

1) A business process model, user goals.
2) Documentation of requirements or their model.
3) Domain model, user needs.

The diagram construction approaches can be summarized as follows.

1) *Actor-based approach [55]:* The System Actors are identified. For instance, who uses the system and why; whether the system is in a relationship with another system, who performs the installation etc. Actors can also be understood as time in situations where an activity occurs at a particular time.
2) *Use Case Determination From Business Processes:* Use case can be understood as a link-up to the business process.

The use case models for IoT contain the following elements.

1) System boundary is the formal capture of the enclosure of the application before the surrounding environment. In IoT, sensing, network, and data processing layers functions should be modeled as part of described systems.
2) Primary actors are entities related to the sensing or network layers, which represent all sensors, actuators, data gathering, emitting, or signal processing.
3) Secondary actors are entities related to the data processing and application layers or connectors to external systems (cloud, external software).
4) Use cases are a textual description of the activity and interaction between the actor and the system. The primary actor is identified for each use case, and primary and alternative scenarios are created. Scenarios that describe the interaction between the actor and the system are mandatory.

Primary actors are placed outside the system boundary on the left side, and secondary actors are placed on the right side. The use cases are placed inside the system boundary. The use case model is not a process diagram. Therefore, the use case process lacks support for queueing. In IoT use-case models, the following relationships are recommended.

1) Association—the relationship between the actor and the use case.
2) Include—the relationship between two use cases; one needs another.
3) Extend—the relationship between two use cases; one needs another under specific conditions.
4) Generalization—relationship that can be used between two actors or two use cases.

The association is only used to emphasize the connection between actors and use cases (system functionality). These relationships are essential for the modularity principle. For instance, a use case, which is included in another use case, necessitates the completion of a parent use case. Typically, the sensor use case can be included in the data processing use case. Extend indicates a similar situation, but with a condition that must be fulfilled. If a condition is true, the use case behavior is included in the parent use case. Similarly, a generalization is applied to actors when standard functions are available for more than one actor. Generalization is applied to use cases when a group of similar use cases exists, which are identical in implementation despite slight differences.

B. Summary of UC Model Creation

The following steps can be used for IoT systems. In the formulation of these steps, a SysML approach is considered, which is the same as the basis for MDD. The necessary steps are as follows.

1) Definition of system boundary—based on IoT system goals.
2) Identify primary actors (sensing layer and network layer).
3) Identify secondary actors (data processing and application layer).
4) Identify use cases for primary and secondary actors.
5) Elaborate relationships—include and extend for use cases.
6) Implement generalization for actors and use cases.

Applying these steps is essential for creating a good use case model that can be used in MDD and for developing effort estimation methods. Moreover, the identification of actors should correlate with the IoT system structure, respecting the four-layer approach. However, these actors have different complexities. Therefore, these steps should be carefully carried out based on technical complexity, number of actors, or technology (high- versus low-level programming language). In IoT, the problem domain use cases differ from those in "classical" software. In software, practitioners prefer high-level descriptions of use-case scenarios. IoT prefers the low-level approach,

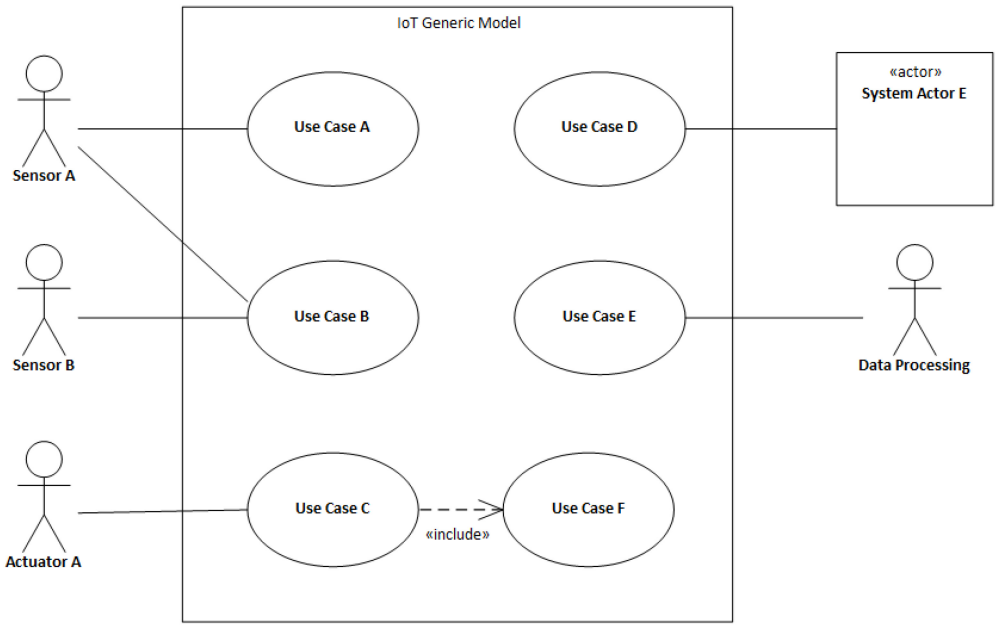

Fig. 1. Use case model for generic IoT system.

TABLE I
USE CASE SCENARIO SAMPLE

Title: Read pH Level		
ID: UC-04		
Characteristics:		
This use case reads the data from the pH sensor and sends it to the processing unit		
Primary actor:		
pH Sensor		
Secondary actors:		
None		
Entry conditions:		
Analog signal from pH Sensor		
Output conditions:		
Processed analogue pH value		
Main scenario:		
Step	Actor/System	Description
1	System	The system assigns a variable pin connection to the pH sensor
2	pH Sensor	The sensor collects the pH level from the hydroponics farm
3	System	The system reads the analogue value from the pH sensor connected to the analogue pin every 30 minutes
4	System	The system verifies the analogue value of the pH level
5	System	The system store the analogue value of the pH sensor to a variable
6	System	The system sends the analogue value to the ADC for conversion
7	System	The system establishes a connection with the cloud database server
8		The use case ends
Alternative scenarios:		
UC-04 – pH sensor reading invalid		

TABLE II
ACTOR COMPLEXITY CATEGORIES

Actor complexity (AC)	Weighting factor (*WFa*)
Simple actor	1
Average actor	2
Complex actor	3

which enables of scenarios to be described in great detail. However, the same writing style must be used with the same level of detail in each use case. Mixing high- and low-level scenarios can lead to issues. An example of the use case scenario is presented in Table I. Full use case models scenarios are available as auxiliary material.[1]

C. Process of Use Case Points Estimation for IoT

The proposed size and DE estimation were adapted from a software engineering size estimation called UCPs [12]. Actors and use cases are connected in the UML [42], [56] use-case models. The UCP technique assumes that the number of actors and use cases can be utilized to establish the size of the suggested software system.

The UCP methodology is based on eliciting four components, which create a size estimation that can be used to estimate DE in PHs. Later, it can even be used to estimate costs if they are driven by PHs or by the number of UCPs. The two main components are processed using the UC model.

1) *Actors* are use used to form unadjusted actors weights (UAWs).
2) *Use cases* are used to create unadjusted use case weights (UUCWs).

The other two components are designed as part of UCP methodology.

1) Technical complexity factors (TCFs).
2) Environmental complexity factors (ECFs).

UCP value, which represents the size in the final stage and is calculated using

$$\text{UCP} = (\text{UAW} + \text{UUCW}) \times \text{TCF} \times \text{ECF}. \tag{1}$$

In UCP, actors and use cases are categorized into classes based on the complexity of the implementation. In this study, the implementation complexity categories (AC) of three actors were identified (Table II).

The UAW value is determined using (2) as a weighted sum of the actors' complexity groups

$$\text{UAW} = \sum \text{AC} \times \textit{WFa}. \tag{2}$$

Simple actors represent sensors, actuators, application interfaces, or Web services. Average actors are actors with straightforward interfaces (data gathering and emitting) or actors that require more complex processing (e.g., A/D converters) or communication and require more implementation

[1]https://dx.doi.org/10.21227/q8py-jm64

TABLE III
DISTRIBUTION OF THE COMPLEXITY OF IMPLEMENTING USE CASES

Use case complexity (UCC)	Steps count	Weighting factor (*WFb*)
Simple UC	(0;5)	5
Average	5;9	10
Complex	(9; ∞)	15

TABLE IV
TECHNICAL FACTORS FOR UCP_{IoT}

Factor	Weighting Factor (*WFc*)	Factor influence (*SIa*)
T1	2	0;5
T2	2	0;5
T3	1	0;5
T4	1	0;5
T5	1	0;5
T6	0.5	0;5
T7	0.5	0;5
T8	2	0;5
T9	1	0;5
T10	1	0;5
T11	1	0;5
T12	1	0;5

effort. The third group, complex actors, represents data processing layer members or actors using a graphical user interface for their activities [12].

A similar approach is used to determine UUCW. The number of steps in the use case scenarios is used to distinguish the complexity (Table III). The number of steps is calculated based on the primary and alternate scenarios. Finally, the value of UUCW is determined using (3).

However, the step-by-step approach ignores the interrelationships of the use cases (generalization, includes, and extends). The resulting size estimation (UUCW) is determined from (3), where the UCC complexity group of the implementation of the use case (Table IV) and *WFb* are the weighting factors for the selected group

$$\text{UUCW} = \sum \text{UCC} \times WFb. \tag{3}$$

After determining UAW and UUCW, the essential characteristics of the proposed IoT system, development team, and environment must be determined. For this purpose, two correction mechanisms are proposed in the UCP methodology.

The first correction mechanism is TCF, which enables us to determine the influence of the essential technical attributes on the system (system characteristics). The influence factor value determines how essential an item is to IoT. The TCF correction is performed according to (4). *WFc* is the weighting factor, and *SIa* is the influence of the factor on the IoT project. In other words, the TCF correction mechanism ensures that the weight of each factor and its influence on the project is determined. The TCF are defined according to the original design of the UCP method [12], where the further examination was presented by Nhung et al. [57]

$$\text{TCF} = 0.6 + \left(0.01 \times \sum_{T1}^{T12} WFc \times SIa\right). \tag{4}$$

TABLE V
ENVIRONMENTAL FACTORS FOR UCP_{IoT}

Factor	Weighting factor (*WFd*)	Significance (*SIb*)
E1	1.5	0; 5
E2	0.5	0;5
E3	1	0;5
E4	0.5	0;5
E5	1	0;5
E6	2	0;5
E7	-1	0;5
E8	2	0;5

A list of technical factors is adopted as follows.

- *T1 (Decentralization):* It specifies the complexity of the architecture in general. A higher significance value indicates a need for a more complex solution architecture (for example, multitier or decentralized architecture).
- *T2 (Responsibility):* It specifies how vital the system response speed is. It is also related to the expected load on the system. A higher significance value indicates that a faster system response or higher load is required.
- *T3 (Efficiency):* It specifies whether the goal is to increase the overall efficiency or to merely achieve a specific functionality. If a higher significance value is chosen, the goal is higher efficiency in running the system.
- *T4 (Processing Complexity):* It specifies whether the system internally processes complex data. A higher value means that the complexity of algorithms and internal data processing is significant, i.e., a higher range of the system.
- *T5 (Reusability):* If higher code reuse and redemption are expected, then the significance value of the parameter can be set to a lower value.
- *T6 (User Experiences):* The significance value of this parameter can be set low if users can be assumed to be highly competent at using the system.
- *T7 (Usability):* If the goal is to achieve higher usability, the significance of this parameter must be set higher.
- *T8 (Development Complexity):* It determines whether multiple platforms are required during development. If so, then the significance is higher.
- *T9 (Maintenance):* If the goal is to develop a system that is easy to maintain and expand, the significance of this parameter must be set to a higher value.
- *T10 (Security):* It determines whether an existing security design can be used. A higher significance value represents a need for better security.
- *T11 (Third-Party Solutions):* If the finished components of suppliers can be directly integrated into the system, then the significance of this parameter is set to a smaller value.
- *T12 (Deployment):* If the system is complex and requires advanced deployment, significance is set to a higher value.

Environmental factors are used to adjust the resulting number of UCPs describing the capabilities of the development team and the development environment. Eight environmental factors are considered, each of which had a weight attributed to it (Table V). Environmental factors can be characterized based on the following parameters.

- *E1* *(Methodology and Project):* It describes the experience of the development team in the problem domain and development methodology. The significance value of the parameter represents the effect of knowledge or inexperience.
- *E2* *(Experience With the Problem/Application Domain):* If the development team has no experience, then the significance is set to 0.
- *E3* *(Experience With Development Environments):* A higher significance value indicates that the team has more experience with used development environments.
- *E4* *(Analyst):* It indicates the experience of the analyst in the team, who is expected to process functional and nonfunctional requirements.
- *E5* *(Motivation):* A higher significance value of this parameter indicates that the team is highly motivated to complete project tasks.
- *E6* *(Stability of Requirements):* Frequent changes in requirements lead to prolonged development. A higher significance value indicates that frequent changes are expected. the significance of this parameter is best set according to previous experience with a particular customer.
- *E7* *(External Staff):* If the involvement of external workers is expected, then a nonzero significance value indicates an extension in development time.
- *E8* *(Difficulty in the Programming Language):* It indicates the complexity of the language and its combination; a higher significance value is chosen for more complex languages. The settings depend on the specific development team and their previous experience.

The value of the environmental factors of the ECF is determined using the relation (5), where *WFd* is the weighting factor, and *SIb* is the influence of the factor on the project being solved. The constants given in the relation are taken from the original design of the UCP method [12]

$$\text{ECF} = 1.4 + \left(-0,03 \times \sum_{E8}^{E1} WFd \times SIb\right). \tag{5}$$

Both *SIa* values (significance of technical factors) and *SIb* (significance of environmental factors) are determined in the interval 0–5, where 0 indicates that no given factor has an effect, 3 indicates an average level of influence, and 5 indicates a significant influence on the scope of the IoT. The resulting value of UCP_{IoT} describes the size of the proposed IoT system based on the UC model. The IoT system size in UCP points can be used to estimate the DE or cost. It can also be used for project development and resource planning.

When the number of UCPs is estimated, DE in PHs can be determined. Transforming the estimated size in the UCP into PHs is based on the productivity factor (PF). The PF is categorized as fair (20), low (28), and very low (36) by Schneider and Winters [58]. These parameters were further discussed and analyzed by Azzeh and Nassif [59].

PF is based on environmental factor count (*ECFc*). Table VI summarizes count intervals for *ECFc* (6). If E1–E6 are less than 3, then the counted is increased by 1; if E7 and E8 are higher than 3, the count is again increased by 1. When the ECF count is less than or equal to 2, fair productivity is used (20). If the ECF count is between 3 and 4, low productivity (28) is used. Finally, when the ECF count is 5 or more, very low productivity is used (36)

TABLE VI
ECF COMPLEXITY COUNT FOR PF

ECFc	PF
<= 2	20
<3;4>	28
>= 5	36

$$ECFc = \sum_{E1}^{E8}\left(\frac{1 \text{ if}(E1; E6) < 2 \text{ or}(E1; E6) > 3}{0 \text{ if}(E1; E6) > 2\text{or}(E1; E6) < 3}\right). \tag{6}$$

The DE is calculated using the PF, which is determined based on *ECFc* value using the following formula:

$$\text{DE} = \text{UCP}_{\text{IoT}} \times \text{PF}_{\text{ECFc}}. \tag{7}$$

The DE is obtained in PHs; if an estimation in person-days (PDs) is needed, then PHs can be divided by 8

$$\text{DE}_{\text{PD}} = \frac{DE}{8}. \tag{8}$$

The difference between UCP_{IoT} and the established UCP software is discussed in the following section.

D. Difference of the UCP_{IoT} to Standard Software UCP

The effort-determination process is specifically designed for IoT systems in this UCP_{IoT} adaptation. Compared with the established classical software-based UCP, the following changes were made.

1) The actors were divided into primary and secondary groups, enabling better control over the estimation process and for situations requiring calibration. Certain IoT systems may contain only primary actors; secondary actors are not mandatory. Standard UCP methods assign only two complexity levels to actors: application interface or human. Primary and secondary actors are also known to form a use-case scenario when the secondary actors are involved in use-case processing.
2) The estimation was based on an adapted use case modeling approach, which describes all items related to the four-layer IoT architecture as actors. This approach helps to create a use case model in a specific way, which is the origin and beneficial for IoT systems, similar to the MDD design approach for IoT. The four-layer approach for use case modeling helps create a more consistent and descriptive use case model, which enables us to develop better UCPs components, such as actors and use case scenarios.
3) Use cases were counted by the number of steps; independently for both actor groups. Typically, use case scenarios are all counted together to complex groups. If they are counted separately as primary and secondary actors, then the UCP method can be more easily tuned. Further investigation shows that each group adds a different complexity level to the system.

4) Use cases were described for actor/system interaction and data/signal processing. In typical software engineering projects, only use cases describing the interaction between actors and systems are used. In the proposed UCP_{IoT} adaptation, actors can also be data processors. The interaction description is, therefore, more detailed. Consequently, the UCP method is more accurate, despite being a basic adaption without any tuning.
5) *TCF* and *ECF* were partly modified for IoT specifics. Both attributes were adopted in a majority of the system components, as they are used in the UCP method. All the attributes were evaluated for IoT projects, although only a few of them were rephrased.
6) The calculation formula was redesigned for typical four-layer IoT architecture related to two groups of actors. Estimation formulas were updated to adopt the concept of primary and secondary actors and use cases. This modification is beneficial for future method tuning and further adaptation.
7) All factors and complexity weights were assumed to be coded numerically. In the UCP method, calculations are generally performed by combining numeric and scale variables. Conversely, in our method, all the variables are assumed to be weighted coefficients.

These changes make UCP_{IoT} an original contribution with significantly different characteristics when compared to the established UCP as described in [12].

The UCP_{IoT} method implements the following revised formula:

$$UCP_{IoT} = \left(UAW_p + UAW_S + UUCW_p + UUCW_s\right) P \times TCF \times ECF. \quad (9)$$

V. Case Study

A. Smart Farming

The UCP calculation method is illustrated below for a smart agriculture project, which was originally designed by Alipio et al. [60]. The project was implementing the IoT infrastructure at Yoki's Farm in Tagaytay City, Philippines and was conducted between January 2016 and March 2017, with four team members participating in the development of hardware and software parts.

The hardware includes a hydroponic farm built with a sensor network to monitor and control the system. The software comprises data analytics and a cloud server for data storage and predictive analyses. The Web interface serves as a graphical interface for any user to remotely access the farm. The complete process of the system is shown in the IoT-based hydroponics system, designed to create a closed feedback loop that monitors and controls the farm based on the parameters required by a specific variety of plants.

The project is detailed in [60], and the full use case model, including scenarios, is available as auxiliary material.[1]

The estimated system is described by the use case model, which displays a functional design (Fig. 2). The system comprises several actors and use cases.

TABLE VII
Primary Actor Size Contribution for Smart Hydroponics Farm System

Complexity	Weight	No. of Actors	Total
Simple	1	5	5
Average	2	1	2
Complex	3	-	0
UAW_P	-	-	7

TABLE VIII
Secondary Actor Size Contribution for Smart Hydroponics Farm System

Complexity	Weight	No. of Actors	Total
Simple	1	-	0
Average	2	-	0
Complex	3	1	3
UAW_S	-	-	3

TABLE IX
Primary Use Case Size Contribution for Smart Hydroponics Farm System

Complexity	Weight	No. of UC	Total
Simple	5	-	0
Average	10	3	30
Complex	15	6	90
$UUCW_P$	-	-	120

The actors were identified as follows.

1) Primary actors—pH Sensor, Humidity Sensor, Light Sensor, Temperature Sensor, EC Sensor, ADC, Motor Pump, Light Bulb, and Humidifier.
2) Secondary actors—Farmer.

Use cases were identified as follows.

1) Primary UC—Reads pH Level, Reads Relative Humidity Level, Reads Electrical Conductivity Level, Convert A/D Signal, Control Light Bulb, and Control Humidifier.
2) Secondary UC—Login, Monitor Hydroponics Farm, Control Hydroponics Farm, and Generate Bayesian Inference Model.

As shown, the UAW value is based on UAW_P (Table VII) and UAW_S (Table VIII). *UAW* represents ten points of size in total. UUCW consists of $UUCW_P$ (Table IX) and $UUCW_S$ (Table X). *UUCW* represents a total of 180 points. The factors *TCF* and *ECF* (Table XI) were calculated as follows:

$$TCF = 0.6 + (0.01 \times 47) \quad (10)$$

$$ECF = 1.4 + (-0.03 \times 22.50). \quad (11)$$

The total number of UCPs (size points) was estimated using the following formula:

$$UCP_{IoT} = (7 + 3 + 120 + 60) \times 1.017 \times 0.725 \quad (12)$$

$$UCP_{IoT} = 140.010. \quad (13)$$

UCP_{IoT} (13) calculated using (9) illustrates the relative size of the system. It can be used to compare various system sizes and as the basis for estimating DE.

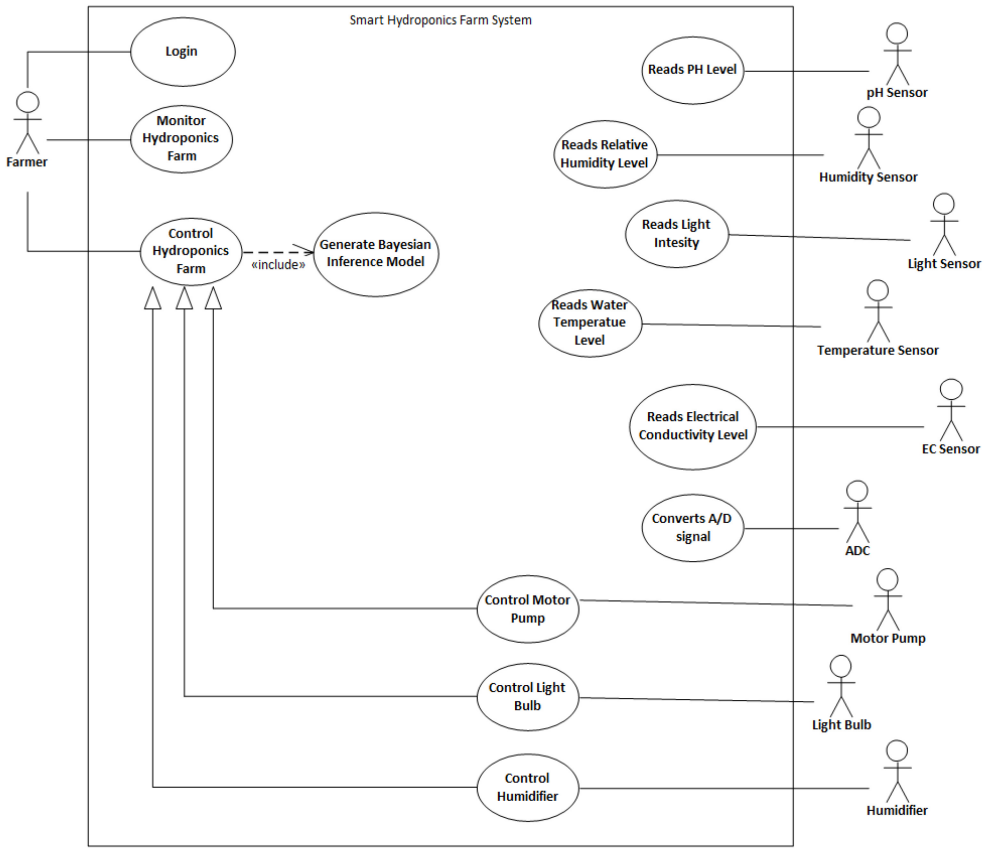

Fig. 2. Use case model of smart hydroponics farm.

TABLE X
SECONDARY USE CASE SIZE CONTRIBUTION FOR SMART HYDROPONICS FARM SYSTEMS

Complexity	Weight	No. of UC	Total
Simple	5	-	0
Average	10	-	0
Complex	15	4	60
$UUCW_S$	-	-	60

The next step was to estimate the DE. In the case study, *ECFc* (6) was equal to 5, which indicates very low productivity

$$DE = 140.010 \times 36 = 5040.360PH \tag{14}$$

$$DE_{PD} = \frac{5040.360}{8} = 630PD. \tag{15}$$

The estimated DE in PHs was calculated to be 5040.360, which is 630 PDs.

The known (recorded) developmental effort (DE) was 5560 PHs (695 PDs), which was recorded during the development of Alipio et al. [60]. The DE recorded is detailed in Table XII.

B. Agrinex—Smart Irrigation System

The UCP calculation method is illustrated below for an Agrinex project, which was originally designed by Tiglao et al. [61]. Agrinex is an agricultural WSAN that uses multiple sensors to gather soil data and employs drip irrigation for actuation. It utilizes a dynamic mesh network for flexible node integration and a Web application for remote control. The sensor and actuator node (SAN) combines sensor and actuator functions, saving power and enabling localized decision making. The system includes microcontrollers, transceivers,

TABLE XI
TCF FACTORS AND *ECF* FACTORS FOR SMART FARMING IOT SYSTEM

Factor	WFc	SIa	Factor	WFb	SIb
T1	2	3	E1	1.5	2
T2	2	4	E2	0.5	2
T3	1	4	E3	1	1
T4	1	1	E4	0.5	1
T5	1	1	E5	1	4
T6	0.5	3	E6	2	4
T7	0.5	5	E7	-1	3
T8	2	5	E8	2	4
T9	1	3	-	-	-
T10	1	4	-	-	-
T11	1	2	-	-	-
T12	1	4	-	-	-

TABLE XII
RECORDED DE IN PDS

Component	Developers	Days	PD
Login Page	2	29	58
Monitoring Farm	3	31	92
Control Farm (and ML modelling)	3	32	96
Sensor set-up and configuration	3	28	84
Actuator set-up and configuration	3	30	90
ADC set-up and configuration	1	5	5
Total person-days	-	-	695

TABLE XIII
PRIMARY ACTOR SIZE CONTRIBUTION FOR AGRINEX SYSTEM

Complexity	Weight	No. of Actors	Total
Simple	1	4	4
Average	2	1	2
Complex	3	-	0
UAW_P	-	-	6

TABLE XIV
SECONDARY ACTOR SIZE CONTRIBUTION FOR AGRINEX SYSTEM

Complexity	Weight	No. of Actors	Total
Simple	1	-	0
Average	2	-	0
Complex	3	1	3
UAW_S	-	-	3

TABLE XV
PRIMARY USE CASE SIZE CONTRIBUTION FOR AGRINEX SYSTEM

Complexity	Weight	No. of UC	Total
Simple	5	-	0
Average	10	1	10
Complex	15	4	60
$UUCW_P$	-	-	70

TABLE XVI
SECONDARY USE CASE SIZE CONTRIBUTION FOR AGRINEX SYSTEM

Complexity	Weight	No. of UC	Total
Simple	5	-	0
Average	10	-	0
Complex	15	3	45
$UUCW_S$	-	-	45

and sensors, with a servo motor and valve for irrigation. Overall, Agrinex optimizes agricultural monitoring and control for efficient water usage and improved crop yield. Full use case model, including scenarios, is available as auxiliary material.[1] The estimated system is described by the use case model, which displays a functional design (Fig. 3).

The actors were identified as follows.

1) Primary actors—Humidity Sensor, Temperature Sensor, Moisture Sensor, Servo Motor, and ADC.
2) Secondary actors—User.

Use cases were identified as follows.

1) Primary UC—Reads Humidity Level, Reads Temperature Level, Reads Soil Moisture Level, Control Servo Motor, and Convert analog-to-digital signals.
2) Secondary UC—User Login, Monitor Irrigation System, and Threshold Control.

The UAW value is based on UAW$_P$ (Table XIII) and UAW$_S$ (Table XIV). *UAW* represents nine points of size in total. UUCW consists of UUCW$_P$ (Table XV) and UUCW$_S$ (Table XVI). *UUCW* represents a total of 115 points. The factors *TCF* and *ECF* (Table XVII) were calculated as follows:

$$\text{TCF} = 0.6 + (0.01 \times 46) \tag{16}$$

$$\text{ECF} = 1.4 + (-0.03 \times 22.50). \tag{17}$$

TABLE XVII
TCF FACTORS AND *ECF* FACTORS FOR AGRINEX

Factor	WFc	SIa	Factor	WFb	SIb
T1	2	2	E1	1.5	2
T2	2	5	E2	0.5	2
T3	1	4	E3	1	1
T4	1	2	E4	0.5	1
T5	1	1	E5	1	4
T6	0.5	3	E6	2	4
T7	0.5	5	E7	-1	3
T8	2	4	E8	2	4
T9	1	3	-	-	-
T10	1	4	-	-	-
T11	1	2	-	-	-
T12	1	4	-	-	-

TABLE XVIII
RECORDED DE IN PDS FOR AGRITEX SYSTEM

Component	Developers	Days	PD
Login Page	2	30	60
Monitoring	3	30	90
Sensor set-up and configuration	3	30	90
Actuator set-up and configuration	3	30	90
ADC set-up and configuration	1	5	5
Mesh network - WSAN	2	45	90
Total person-days	-	-	485

The total number of UCPs (size points) was estimated using the following formula:

$$\text{UCP}_{\text{IoT}} = (6 + 3 + 70 + 45) \times 1.06 \times 0.725 \tag{18}$$

$$\text{UCP}_{\text{IoT}} = 91.428. \tag{19}$$

The next step was to estimate the DE. In the case study, *ECFc* (6) was equal to 5, which indicates very low productivity

$$DE = 91.428 \times 36 = 3291.408PH \tag{20}$$

$$DE_{PD} = \frac{3291.408}{8} = 411PD. \tag{21}$$

The estimated DE in PHs was calculated to be 3291.408, which is 411 PDs.

The known (recorded) developmental effort (DE) was 3880 PHs (485 PDs), which was recorded during the development of the project. The DE recorded is detailed in Table XVIII.

C. Water Quality Monitoring

In the third case study, we are using a system originally designed by Alipio [62]. The system is composed of three main components: 1) hardware; 2) software; and 3) a Web interface. The hardware includes sensor devices and a microcontroller for collecting water quality data. The software incorporates data analytics, machine learning, and a cloud server for storage and predictive analysis. The Web interface allows remote access to the system, while mobile technology sends SMS messages to inform residents about water source conditions. The sensor node devices consist of sensors, a microcontroller, and wireless modules. ZigBee is used for wireless transmission to a local server. The system is deployed in rural areas, specifically in the CALABARZON region of the

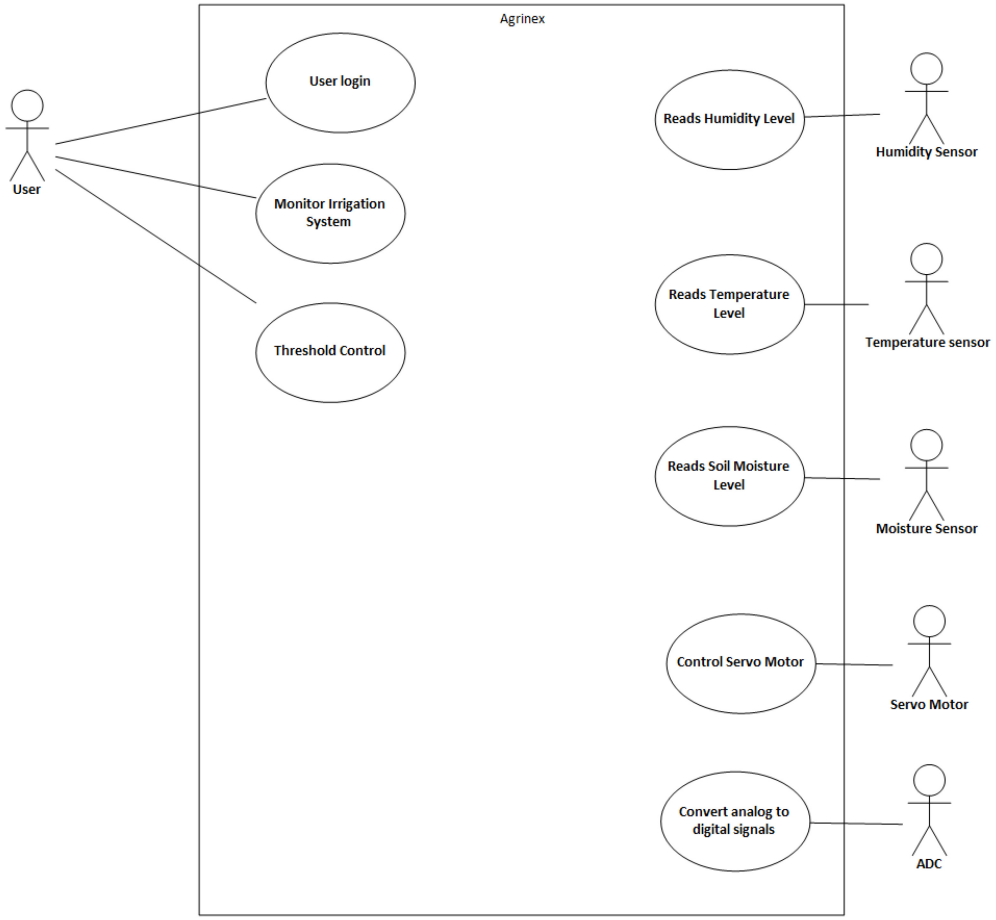

Fig. 3. Use case model of Agrinex.

Philippines, to monitor water quality. The estimated system is described by the use case model, which displays a functional design (Fig. 4). Use case model, including scenarios, is available as auxiliary material.[1]

The actors were identified as follows.

1) Primary actors—pH Sensor, TDS Sensor, Turbidity Sensor, and Temperature Sensor.
2) Secondary actors—User, Time, and SMS Module.

Use cases were identified as follows.

1) Primary UC—Reads pH Level, Reads TDS Level, Reads Turbidity Intensity, and Reads Temperature Level.
2) Secondary UC—Login, Monitoring systems parameters, and Generate Predictive Model.

The *UAW* value is based on UAW_P (Table XIX) and UAW_S (Table XX). *UAW* represents 13 points of size in total. *UUCW* consists of UUCW_P (Table XXI) and $UUCW_S$ (Table XXII). *UUCW* represents a total of 135 points. The factors *TCF* and *ECF* (Table XXIII) were calculated as follows:

$$\text{TCF} = 0.6 + (0.01 \times 30) \tag{22}$$

$$\text{ECF} = 1.4 + (-0.03 \times 21.50). \tag{23}$$

The total number of UCPs (size points) was estimated using the following formula:

$$\text{UCP}_{\text{IoT}} = (7 + 6 + 90 + 45) \times 0.90 \times 0.755 \tag{24}$$

$$\text{UCP}_{\text{IoT}} = 102.604. \tag{25}$$

TABLE XIX
PRIMARY ACTOR SIZE CONTRIBUTION FOR WATER QUALITY MONITORING SYSTEM

Complexity	Weight	No. of Actors	Total
Simple	1	3	3
Average	2	2	4
Complex	3	-	0
UAW_P	-	-	7

The next step was to estimate the DE. In the case study, *ECFc* (6) was equal to 4, which indicates low productivity

$$DE = 102.604 \times 28 = 2872.912PH \tag{26}$$

$$DE_{PD} = \frac{2872.912}{8} = 359PD. \tag{27}$$

The estimated DE in PHs was calculated to be 2872.912, which is 359 PDs.

The known (recorded) development effort (DE) was 4120 PHs (515 PDs), which was recorded during the system design and development. The DE recorded is detailed in Table XXIV.

VI. RESULTS, DISCUSSION, AND FUTURE RESEARCH DIRECTION

The proposed estimation methodology has been applied to three diverse projects, demonstrating its potential in predicting

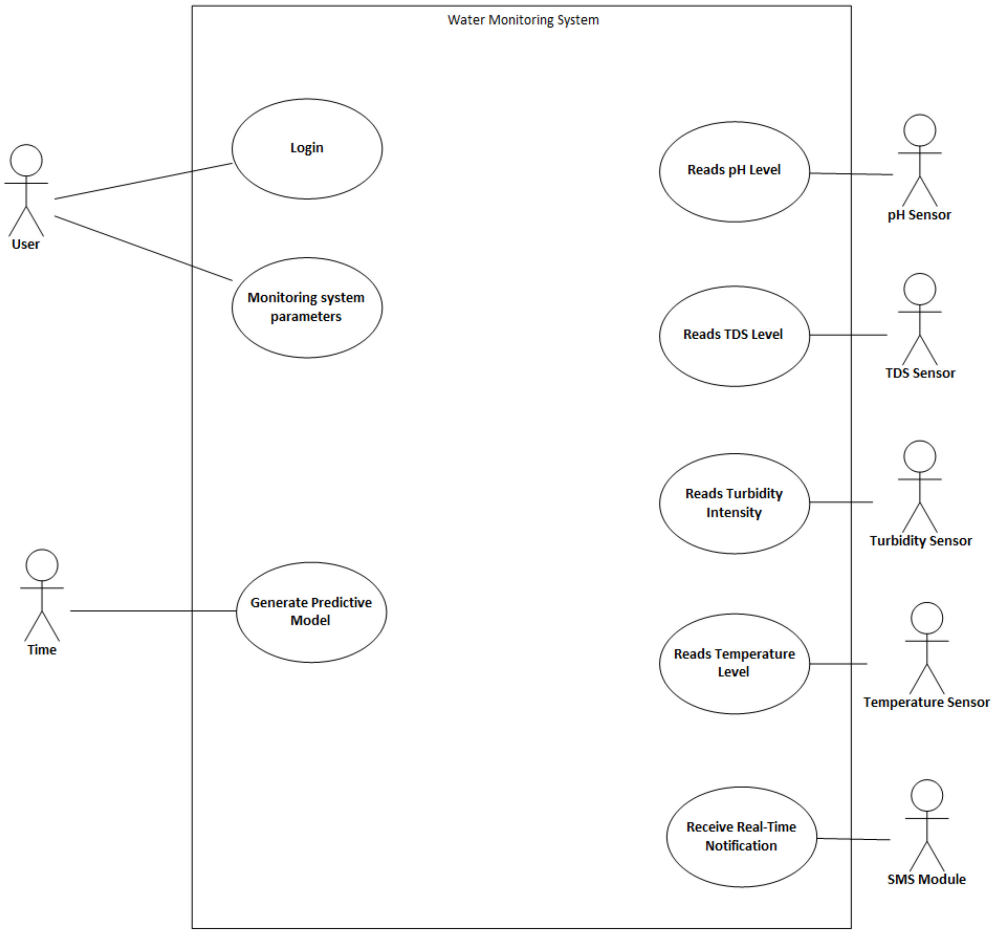

Fig. 4. Use case model of water quality monitoring.

TABLE XX
SECONDARY ACTOR SIZE CONTRIBUTION FOR WATER QUALITY MONITORING SYSTEM

Complexity	Weight	No. of Actors	Total
Simple	1	1	1
Average	2	1	2
Complex	3	1	3
UAW_S	-	-	6

TABLE XXI
PRIMARY USE CASE SIZE CONTRIBUTION FOR WATER QUALITY MONITORING SYSTEM

Complexity	Weight	No. of UC	Total
Simple	5	-	0
Average	10	0	0
Complex	15	6	90
$UUCW_P$	-	-	90

developmental effort (DE) with varying degrees of accuracy. Results for the smart farming show that known (recorded) developmental effort (DE) was 5560 PHs, which was recorded during the development. The predicted DE is 5040.360, showing an absolute error value of 519.64 PHs, which amounts to a relative error percentage of 9.35% (28).

For the Agritex project, the known (recorded) developmental effort (DE) was 3880 PHs, and the predicted PD is 3291.408, resulting in an error of 588.592 PHs (15.2%).

TABLE XXII
SECONDARY USE CASE SIZE CONTRIBUTION FOR WATER QUALITY MONITORING SYSTEM

Complexity	Weight	No. of UC	Total
Simple	5	-	0
Average	10	-	0
Complex	15	3	45
$UUCW_S$	-	-	45

TABLE XXIII
TCF FACTORS AND *ECF* FACTORS FOR WATER QUALITY MONITORING SYSTEM

Factor	WFc	SIa	Factor	WFb	SIb
T1	2	2	E1	1.5	2
T2	2	1	E2	0.5	5
T3	1	2	E3	1	5
T4	1	4	E4	0.5	2
T5	1	3	E5	1	4
T6	0.5	5	E6	2	2
T7	0.5	1	E7	-1	0
T8	2	1	E8	2	1
T9	1	2	-	-	-
T10	1	4	-	-	-
T11	1	2	-	-	-
T12	1	2	-	-	-

Finally, for a Water Quality monitoring project, the known (recorded) developmental effort (DE) was 4120 PHs and the estimated DE in PHs was calculated to be 2872.912. It shows

TABLE XXIV
RECORDED DE IN PDS FOR WATER QUALITY MONITORING SYSTEM

Component	Developers	Days	PD
Login Page	2	30	60
Monitoring	2	30	60
Network and communication	2	30	60
Sensor set-up and configuration	3	30	90
ADS set-up and configuration	2	10	20
Mesh network configuration	3	30	90
Deployment and data collection	3	25	75
ML Modeling and Evaluation	2	30	60
Total person-days	-	-	515

an error of 1247.088 PHs (30.3%)

$$RE = \frac{(y - \hat{y})}{y} \times 100[\text{in } \%] \tag{28}$$

where y is the known DE and $\hat{y}$ is the predicted DE.

As shown, the predicted *DE* using UCP_{IoT} is comparable to the known *DE*. Typically, the UCP method is relatively less accurate because *RE* depends on *PF*. Previous research related to software engineering DE has yielded predictions that are significantly less accurate. The promising results are limited by the lack of empirical evidence and historical data sets specifically tailored for IoT systems, which should be used for further testing. The UCP method, commonly used in software engineering, relies on historical data for accurate results, which are not readily accessible for IoT projects. Consequently, the absence of project-specific data hampers the estimation process. In the presented case studies, the UCP-IoT approach yielded relatively accurate predictions, but the mean percentage error highlighted the need for further improvement. However, to enhance the precision of estimation models for IoT projects, there is a pressing need for empirical evidence derived from a wider range of projects. Such empirical evidence would help refine and enhance the estimation methodologies, ensuring more reliable predictions for the developmental effort required in IoT endeavors.

The case study shows that the proposed UCP_{IoT} approach can be used for IoT DE estimation. However, identifying the actors in IoT systems remains challenging because it typically involves more hardware than software systems.

Although the UCP method is widely adopted in software engineering, it requires historical data sets to provide accurate results. However, such data sets, i.e., those describing system DE or system sizing, are not available for IoT systems.

The objective of this study is to promote discussions in the IoT community about applying UML/SysML and functional modeling based on use case models. In functional modeling, entities can be based on a four-layer architecture and mapped as actors in the use-case model. In addition, high-level and low-level use-case scenarios must be further investigated to obtain a functional description of IoT. For instance, a low-level scenario may be better to establish a detailed description of communication protocols or software-hardware interaction.

In this study, use case model associations are used as a common interpretation of IoT system interactions. However, in systems that are heavily dependent on hardware and have limited software interactions, the relationships between elements, such as actors, should be emphasized to better reflect the structure of IoT systems.

The UCP_{IoT} in this study was proposed using standard constants and formulas. Although this standard adaptation provided accurate predictions for a specific case study, a research data set that contains project descriptions must be built for further adjustments. In addition, the estimation formula can be modified further using various prediction algorithms (machine learning or statistical learning).

Further research on cost or DE drivers for IoT systems is essential to obtain more accurate estimations and improve IoT project planning and management. Furthermore, a solution must be provided for transforming the IoT system size in UCP to PHs. Typically, PF is used for this conversion. However, it depends on many factors, such as the programming language, problem domain, and application domain; it is, therefore, difficult to determine. Thus, PF must be set correctly to effectively apply an estimation approach, which is challenging. This adaptation of the UCP method for IoT systems demonstrates its practical applicability, as confirmed by the presented case study.

VII. THREATS TO VALIDITY

The proposed UCP_{IoT} adaptation was verified using an individual case study. However to enhance its credibility, data must be gathered from more IoT projects.

This method was adopted from UCP, which is based on a software-specific functional point approach. The proposed UCP adaptation was correctly designed and verified in the case study; however, future studies should focus on obtaining more relevant information from various case studies. Considering the heterogeneity of IoT systems, new factors may emerge depending on the type of IoT system. Therefore, the methods for IoT system size estimation should be sufficiently flexible to integrate new influencing factors that may arise in the future. Furthermore, *TCF* and *ECF* factors should be tested to ensure if they are appropriate for the specific systems.

UCP_{IoT} method itself can be discussed further. Several studies have been conducted on map size and effort estimation methods in software engineering, including UCP and UCP modification. Additionally, the inspiration of potential further modification can be seen in studies on software DE estimation and related topics, including comparative analysis of soft computing techniques [63], MDD of user interfaces for IoT systems [33], systematic mapping study on the employment of neural networks [64], middleware for Internet distribution in the context of cloud computing and the Internet of Things [38], and in review of ensemble effort estimation [65]. These studies may provide additional context and related work for the article's discussion of effort estimation using use cases.

Because the original UCP was inspired by the functional points method [66], it also includes several design issues discussed by Ouwerkerk and Abran [67]. UCP design flaws are primarily based on scale transformation when the values of

the UCP components are calculated [68]. These design issues require immediate attention; however, more relevant case studies or IoT project repositories must be gathered before they can be solved.

VIII. Conclusion

In conclusion, this article addresses the adaptation of the UCP methodology for estimating the size and effort required in developing IoT systems. The study demonstrates the potential of UCP as a valuable size and DE estimator in IoT project management.

This article presents a case study involving three diverse IoT projects, showcasing the applicability of the UCP method in estimating the DE for IoT projects. The results indicate that the predicted developmental effort using the UCP-IoT approach is comparable to the known effort, despite the limitations imposed by the lack of empirical evidence and historical data sets specifically tailored for IoT systems.

Additionally, the study provides rules and recommendations for creating use case models for IoT systems. It emphasizes the inclusion of actors from the sensing, network, and data processing layers in the use case models and highlights the importance of system size estimation for project management and resource planning in IoT systems. This article suggests that existing effort estimation methods for software systems can be adapted to address the specific characteristics of IoT environments.

However, this study also suggests the need for further research to improve the accuracy of the proposed UCP adaptation. It suggests exploring UCP factors in more detail, gathering research data sets comprising IoT project descriptions, and fine-tuning the proposed model to enhance its accuracy. The authors intend to focus on these areas in future studies.

In response to RQ1 (What are the use case model design rules of IoT systems?), this article provides guidelines for designing use case models for IoT systems. It suggests including actors from the sensing, network, and data processing layers in the models and following the UML/SysML naming convention for actors and use cases. It emphasizes the high-level and low-level use case scenarios to obtain a functional description of IoT systems.

Regarding RQ2 (How should the software UCP methodology be adapted for IoT systems size/DE estimation to reflect the specifics of these systems?), this article proposes an adaptation of the UCP methodology for IoT systems. It considers the four-layer architecture of IoT systems and tailors the UCP to accommodate the unique characteristics of IoT projects. The study demonstrates the practical applicability of the proposed adaptation through the case study results.

Overall, this research contributes to the field by providing insights into the estimation of size and DE in IoT systems using the UCP methodology. It highlights the importance of further research, empirical evidence, and tailored data sets for improving the accuracy of estimation models for IoT projects. The guidelines and adaptations proposed in this study pave the way for more effective project planning and resource management in the rapidly evolving field of IoT.

In future work, we intend to further explore the UCP factors to improve the accuracy of UCP_{IoT}. For this purpose, our future studies will focus on gathering research data sets comprising IoT project descriptions, which are essential for empirical studies, UCP_{IoT} model tuning, and model accuracy evaluation.

References

[1] P. J. Escamilla-Ambrosio, D. A. Robles-Ramirez, T. Tryfonas, A. Rodriguez-Mota, G. Gallegos-Garcia, and M. Salinas-Rosales, "IoTsecM: A UML/SysML extension for Internet of Things security modeling," *IEEE Access*, vol. 9, pp. 154112–154135, 2021.

[2] K. Thramboulidis and F. Christoulakis, "UML4IoT—A UML-based approach to exploit IoT in cyber-physical manufacturing systems," *Comput. Ind.*, vol. 82, pp. 259–272, Oct. 2016.

[3] J. Muangprathub, N. Boonnam, S. Kajornkasirat, N. Lekbangpong, A. Wanichsombat, and P. Nillaor, "IoT and agriculture data analysis for smart farm," *Comput. Electron. Agr.*, vol. 156, pp. 467–474, Jan. 2019.

[4] M. Geller and A. A. de Moura Meneses, "Modelling IoT systems with UML: A case study for monitoring and predicting power consumption," *Amer. J. Eng. Appl. Sci.*, vol. 14, no. 1, pp. 81–93, 2021.

[5] F. Martins and D. Domingos, "Modelling IoT behaviour within BPMN business processes," *Procedia Comput. Sci.*, vol. 121, pp. 1014–1022, Dec. 2017.

[6] K. Batool and M. A. Niazi, "Modeling the Internet of Things: A hybrid modeling approach using complex networks and agent-based models," *Complex Adapt. Syst. Model.*, vol. 5, pp. 1–19, Mar. 2017.

[7] K. Suri, "Modeling the Internet of Things in configurable process models," Ph.D. dissertation, Sciences et Technologies de l'Information et de la Commun., Université Paris-Saclay, Gif-sur-Yvette, France, 2019.

[8] E. Korkan, S. Kaebisch, M. Kovatsch, and S. Steinhorst, "Sequential behavioral modeling for scalable IoT devices and systems," in *Proc. Forum Specification Design Languages (FDL)*, 2018, pp. 5–16.

[9] J. P. da Silva Fonseca, A. R. de Sousa, and J. J.-P. Z. de Souza Tavares, "Modeling and controlling IoT-based devices' behavior with high-level Petri nets," *Procedia Comput. Sci.*, vol. 217, pp. 1462–1469, Jan. 2023.

[10] J. Song, D. Karagiannis, and M. Lee, "Modeling method to abstract collective Behavior of smart IoT systems in CPS," *Sensors*, vol. 22, no. 13, p. 5057, 2022.

[11] M. T. Moghaddam, H. Muccini, J. Dugdale, and M. B. Kjægaard, "Designing Internet of Behaviors Systems," in *Proc. IEEE 19th Int. Conf. Softw. Archit. (ICSA)*, 2022, pp. 124–134.

[12] G. Karner, "Metrics for objectory," Diploma thesis, Univ. Linkoping, Linköping, Sweden, Dec. 1993.

[13] A. Idri, F. A. Amazal, and A. Abran, "Analogy-based software development effort estimation: A systematic mapping and review," *Inf. Softw. Technol.*, vol. 58, pp. 206–230, Feb. 2015.

[14] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," *Inf. Softw. Technol.*, vol. 54, no. 8, pp. 820–827, 2012.

[15] F. Wang, X. Yang, X. Zhu, and L. Chen, "Extended use case points method for software cost estimation," in *Proc. Int. Conf. Comput. Intell. Softw. Eng.*, 2009, pp. 1–5.

[16] P. Mohagheghi, B. Anda, and R. Conradi, "Effort estimation of use cases for incremental large-scale software development," in *Proc. 27th Int. Conf. Softw. Eng.*, 2005, pp. 303–311.

[17] M. Ochodek, J. Nawrocki, and K. Kwarciak, "Simplifying effort estimation based on use case points," *Inf. Softw. Technol.*, vol. 53, no. 3, pp. 200–213, 2011.

[18] T. Urbanek, Z. Prokopova, and R. Silhavy, "On the value of parameters of use case points method," in *Artificial Intelligence Perspectives and Applications* (Advances in Intelligent Systems and Computing). Cham, Switzerland: Springer Int., 2015, pp. 309–319.

[19] S. Azevedo, R. J. Machado, A. Bragança, and H. Ribeiro, "On the refinement of use case models with variability support," *Innov. Syst. Softw. Eng.*, vol. 8, no. 1, pp. 51–64, 2011.

[20] M. R. Braz and S. R. Vergilio, "Software effort estimation based on use cases," in *Proc. 30th Annu. Int. Comput. Softw. Appl. Conf.*, vol. 1, 2006, pp. 221–228.

[21] S. Diev, "Use cases modeling and software estimation," *ACM SIGSOFT Softw. Eng. Notes*, vol. 31, no. 6, p. 1, 2006.

[22] G. Robiolo, C. Badano, and R. Orosco, "Transactions and paths: Two use case based metrics which improve the early effort estimation," in *Proc. 3rd Int. Symp. Empir. Softw. Eng. Meas.*, 2009, pp. 422–425.

[23] M. Ochodek, B. Alchimowicz, J. Jurkiewicz, and J. Nawrocki, "Improving the reliability of transaction identification in use cases," *Inf. Softw. Technol.*, vol. 53, no. 8, pp. 885–897, 2011.

[24] J. Jurkiewicz, J. Nawrocki, M. Ochodek, and T. Glowacki, "HAZOP-based identification of events in use cases an empirical study," *Empir. Softw. Eng.*, vol. 20, no. 1, pp. 82–109, 2015.

[25] R. Silhavy, P. Silhavy, and Z. Prokopova, "Algorithmic optimisation method for improving use case points estimation," *PLOS One*, vol. 10, no. 11, 2015, Art. no. e0141887.

[26] R. Silhavy, P. Silhavy, and Z. Prokopova, "Applied least square regression in use case estimation precision tuning," in *Software Engineering in Intelligent Systems* (Advances in Intelligent Systems and Computing). Cham, Switzerland: Springer Int., 2015, pp. 11–17.

[27] A. B. Nassif, L. F. Capretz, and D. Ho, "Estimating software effort based on use case point model using Sugeno fuzzy inference system," in *Proc. 23rd IEEE Int. Conf. Tools Artif. Intell.*, 2011, pp. 393–398.

[28] A. B. Nassif, L. F. Capretz, and D. Ho, "Estimating software effort using an ANN model based on use case points," in *Proc. 11th Int. Conf. Mach. Learn. Appl. (ICMLA)*, vol. 2, 2012, pp. 42–47.

[29] I. V. Evdokimov, A. R. J. Alalwan, R. Y. Tsarev, T. N. Yamskikh, O. A. Tsareva, and A. N. Pupkov, "A cost estimation approach for IoT projects," in *Proc. J. Phys. Conf. Series*, 2019, Art. no. 42083.

[30] F. Ciccozzi and R. Spalazzese, "Mde4IoT: Supporting the Internet of Things with model-driven engineering," in *Proc. 10th Int. Symp. Intell. Distrib. Comput.*, Paris, France, 2017, pp. 67–76.

[31] H. Khaleel et al., "Heterogeneous applications, tools, and methodologies in the car manufacturing industry through an IoT approach," *IEEE Syst. J.*, vol. 11, no. 3, pp. 1412–1423, Sep. 2017.

[32] A. Lekidis, E. Stachtiari, P. Katsaros, M. Bozga, and C. K. Georgiadis, "Model-based design of IoT systems with the BIP component framework," *Softw. Pract. Exp.*, vol. 48, no. 6, pp. 1167–1194, 2018.

[33] M. Brambilla, E. Umuhoza, and R. Acerbis, "Model-driven development of user interfaces for IoT systems via domain-specific components and patterns," *J. Internet Services Appl.*, vol. 8, p. 14, Sep. 2017.

[34] A. Ito et al., "A study of design model for IoT system," in *Proc. 16th Int. Conf. Comput. Aided Syst. Theory*, 2018, pp. 126–133.

[35] G. Guerrero-Ulloa, M. J. Hornos, and C. Rodríguez-Domínguez, "TDDM4IoTS: A test-driven development methodology for Internet of Things (IoT)-based systems," in *Proc. 1st Int. Conf. Appl. Technol.*, 2020, pp. 41–55.

[36] A. Harbouche, N. Djedi, M. Erradi, J. Ben-Othman, and A. Kobbane, "Model driven flexible design of a wireless body sensor network for health monitoring," *Comput. Netw.*, vol. 129, pp. 548–571, Dec. 2017.

[37] T. Usländer and T. Batz, "Agile service engineering in the Industrial Internet of Things," *Future Internet*, vol. 10, no. 10, p. 100, 2018.

[38] G. Blair, D. Schmidt, and C. Taconet, "Middleware for Internet distribution in the context of cloud computing and the Internet of Things editorial introduction," *Ann. Telecommun.*, vol. 71, nos. 3–4, pp. 87–92, 2016.

[39] F. Pramudianto et al., "IoT Link: An Internet of Things prototyping toolkit," in *Proc. IEEE 11th Int. Conf. Ubiquitous Intell. Comput. IEEE 11th Int. Conf. Auton. Trusted Comput. IEEE 14th Int. Conf. Scalable Comput. Commun. Assoc. Workshops*, 2014, pp. 1–9.

[40] D. Conzon, P. Brizzi, P. Kasinathan, C. Pastrone, F. Pramudianto, and P. Cultrona, "Industrial application development exploiting IoT vision and model driven programming," in *Proc. 18th Int. Conf. Intell. Next Gener. Netw.*, 2015, pp. 168–175.

[41] J. Arlow, "Use cases, UML visual modelling and the trivialisation of business requirements," *Requirements Eng.*, vol. 3, no. 2, pp. 150–152, 1998.

[42] M. J. Chonoles, *OCUP 2 Certification Guide: Preparing for the OMG Certified UML 2.5 Professional 2 Foundation Exam*. Cambridge, MA, USA: Elsevier/Morgan Kaufmann, 2018.

[43] D. Dori, *SysML: Foundations and Diagrams*. New York, NY, USA: Springer, 2016, pp. 135–156. [Online]. Available: https://doi.org/10.1007/978-1-4939-3295-5_12

[44] M. Fahmideh, A. Ahmad, A. Behnaz, J. Grundy, and W. Susilo, "Software engineering for Internet of Things: The practitioners' perspective," *IEEE Trans. Softw. Eng.*, vol. 48, no. 8, pp. 2857–2878, Aug. 2022.

[45] G. Fortino, A. Guerrieri, W. Russo, and C. Savaglio, "Towards a development methodology for smart object-oriented IoT systems: A metamodel approach," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2015, pp. 1297–1302.

[46] S. Maleki, C. Fu, A. Banotra, and Z. Zong, "Understanding the impact of object oriented programming and design patterns on energy efficiency," in *Proc. Eighth Int. Green Sustain. Comput. Conf. (IGSC)*, 2017, pp. 1–6.

[47] D. Alulema, J. Criado, L. Iribarne, A. J. Fernández-García, and R. Ayala, "SI4IoT: A methodology based on models and services for the integration of IoT systems," *Future Gener. Comput. Syst.*, vol. 143, pp. 132–151, Jun. 2023.

[48] M. Artikov, J. Meier, and A. Winter, "Towards integrated IoT-languages," in *Proc. Int. Conf. Inf. Sci. Commun. Technol. (ICISCT)*, 2019, pp. 1–5.

[49] K. Liu et al., "On manually reverse engineering communication protocols of Linux-based IoT systems," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6815–6827, Apr. 2021.

[50] D. Balsamo, M. Magno, K. Kubara, B. Lazarescu, and G. V. Merrett, "Energy harvesting meets IoT: Fuelling adoption of transient computing in embedded systems," in *Proc. IEEE 5th World Forum Internet Things (WF-IoT)*, 2019, pp. 413–417.

[51] A. Burger, C. Cichiwskyj, S. Schmeißer, and G. Schiele, "The elastic Internet of Things—A platform for self-integrating and self-adaptive IoT-systems with support for embedded adaptive hardware," *Future Gener. Comput. Syst.*, vol. 113, pp. 607–619, Dec. 2020.

[52] M. Bures, M. Klima, V. Rechtberger, B. S. Ahmed, H. Hindy, and X. Bellekens, "Review of specific features and challenges in the current Internet of Things systems impacting their security and reliability," in *Trends and Applications in Information Systems and Technologies*. Cham, Switzerland: Springer, 2021, pp. 546–556.

[53] F. Pauls, S. Haas, S. Köpsell, M. Roitzsch, N. Asmussen, and G. Fettweis, "On trustworthy scalable hardware/software platform design," in *Proc. Smart Syst. Integr. (SSI)*, 2022, pp. 1–6.

[54] N. A. Kulatunga and D. J. Sampaio, "IoT platforms and hardware integrations for industry 4.0 applications," in *Proc. 14th Conf. Ind. Inf. Syst. (ICIIS)*, 2019, pp. 209–214.

[55] J. Arlow and I. Neustadt, *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*. Hoboken, NJ, USA: Pearson Educ., 2005.

[56] R. Y. Lee, *Object-Oriented Software Engineering With UML: A Hands-on Approach*. Hauppauge, NY, USA: Nova Sci. Publ., Inc., 2018.

[57] H. Le Thi Kim Nhung, H. T. Hoc, and V. Van Hai, "An evaluation of technical and environmental complexity factors for improving use case points estimation," in *Proc. 4th Comput. Methods Syst. Softw. Softw. Eng. Perspect. Intell. Syst.*, vol. 1, 2020, pp. 757–768.

[58] G. Schneider and J. P. Winters, *Applying Use Cases: A Practical Guide*. Hoboken, NJ, USA: Pearson Educ., 2001.

[59] M. Azzeh and A. B. Nassif, "Analyzing the relationship between project productivity and environment factors in the use case points method," *J. Softw. Evol. Process*, vol. 29, no. 9, 2017, Art. no. e1882.

[60] M. I. Alipio, A. E. M. D. Cruz, J. D. A. Doria, and R. M. S. Fruto, "On the design of nutrient film technique hydroponics farm for smart agriculture," *Eng. Agr. Environ. Food*, vol. 12, no. 3, pp. 315–324, 2019.

[61] N. M. Tiglao, M. Alipio, J. V. Balanay, E. Saldivar, and J. L. Tiston, "Agrinex: A low-cost wireless mesh-based smart irrigation system," *Measurement*, vol. 161, Sep. 2020, Art. no. 107874.

[62] M. I. Alipio, "Data-driven IoT-based water quality monitoring and potability classification system in rural areas," in *Proc. Int. Conf. Inf. Commun. Technol. Convergence (ICTC)* 2020, pp. 634–639.

[63] M. Azzeh, A. B. Nassif, and S. Banitaan, "Comparative analysis of soft computing techniques for predicting software effort based use case points," *IET Softw.*, vol. 12, no. 1, pp. 19–29, 2018.

[64] R. A. Dos Santos, D. Vieira, A. Bravo, L. Suzuki, and F. Qudah, "A systematic mapping study on the employment of neural networks on software engineering projects: Where to go next?" *J. Softw. Evol. Process*, vol. 34, no. 3, 2022, Art. no. e2402.

[65] A. Idri, M. Hosni, and A. Abran, "Systematic literature review of ensemble effort estimation," *J. Syst. Softw.*, vol. 118, pp. 151–175, Aug. 2016.

[66] *Software and Systems Engineering—Software Measurement—IFPUG Functional Size Measurement Method*, ISO/IEC 20926:2009, 2009.

[67] J. Ouwerkerk and A. Abran, "An evaluation of the design of use case points (UCP)," in *Proc. Int. Conf. Softw. Process Product Meas. MENSURA*, 2006, pp. 83–97.

[68] A. Abran, *Software Metrics and Software Metrology*. Hoboken, NJ, USA: Wiley, 2010.

Radek Silhavy received the Ph.D. degree in engineering informatics from the Faculty of Applied Informatics, Tomas Bata University in Zlín, Zlín, Czech Republic, in 2009.

He is an Associate Professor and a Senior Researcher with the Faculty of Applied Informatics, Tomas Bata University in Zlín. He is an Associate Professor of System Engineering and Informatics with a demonstrated history of working in research, higher education, project management, and software analysis. His research interests are predictive analytics for software engineering, empirical methods in software engineering, or prediction models focused on cost, size, and effort estimations in system/software engineering.

Dr. Silhavy is also involved in academic publishing as the editor-in-chief, editor, or reviewer.

Miroslav Bures received the Ph.D. degree in computer science from the Czech Technical University in Prague, Prague, Czech Republic, in 2006.

He leads the System Testing Intelligent Laboratory, Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague. In 2010, he was appointed with Czech Technical University in Prague, where he is currently an Associate Professor of Computer Science. He has led several projects in the field of test automation for software and Internet of Things systems, covering the topics of automated generation of test scenarios and automated execution of tests. His research interests include quality assurance and reliability methods, model-based testing, path-based testing, combinatorial interaction testing, test automation for software, Internet of Things, and mission-critical systems.

Melchizedek Alipio (Member, IEEE) received the Ph.D. degree in electrical and electronics engineering from the University of the Philippines Diliman, Quezon City, Philippines, in 2018.

He is a Postdoctoral Researcher with the System Testing Intelligent Laboratory, Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic. He is also with the Department of Electronics and Computer Engineering, De La Salle University, Manila, Philippines, in 2019. His research interests include wireless sensor networks, Internet of Things, intelligent systems, and applied artificial intelligence and machine learning.

Dr. Alipio received the Best Paper Awards at the 2017 IEEE Global Conference in Consumer Electronics, the 2022 37th International Technical Conference on Circuits/Systems, Computers, and Communications, and the 2023 IEEE World AI IoT Congress.

Petr Silhavy received the Ph.D. degree in engineering informatics from the Faculty of Applied Informatics, Tomas Bata University in Zlín, Zlín, Czech Republic, in 2009.

He is an Associate Professor with the Faculty of Applied Informatics, Tomas Bata University in Zlín. He is a Senior Research and an Associate Professor of System Engineering and Informatics with a demonstrated history of working in research and higher education. He has expertise as a CTO and a Software Developer in database programming, database design, data management, and data science. His research interests are prediction and empirical methods for software engineering.