Journal of Reliable Intelligent Environments (2024) 10:215–244
https://doi.org/10.1007/s40860-024-00229-9

REVIEW

Development methodologies for IoT-based systems: challenges and research directions

Miguel J. Hornos[1] · **Mario Quinde**[2]

Received: 5 November 2023 / Accepted: 8 July 2024 / Published online: 9 August 2024

Abstract
The spread of IoT-based systems presents several potential benefits to society but still has crucial challenges in different research areas. From the software development point of view, an established methodology for IoT-based systems development is still yet to be found despite the considerable research efforts that are being made in the area. This article presents a literature review of the existing methodologies for IoT-based systems development, highlighting their benefits and limitations. The article also describes and analyses the existing critical challenges in finding a methodology addressing the complex nature of IoT-based systems. This analysis leads to present the open research directions in developing IoT-based systems, which are pathways to drive the research efforts towards addressing the key issues in the area with the aim of finding a methodology that is simple for developers but that ensures high-quality IoT-based systems.

Keywords Internet of Things (IoT) · Development methodologies · Software engineering process · System life cycle stages · Methodological approaches · Support tools

1 Introduction

The Internet of Things (IoT) aims to connect everyday objects (*things*) to the Internet, leveraging a suite of technologies that enable these objects to form a network and autonomously interact with one another to achieve common goals and intelligently respond to changes in the environment where they are deployed [1–5]. Therefore, an IoT system comprises numerous heterogeneous devices, which generate vast amounts of data and events. Consequently, the IoT paradigm must integrate, process, and respond to this massive influx of events in real-time [6].

According to Cisco [7], an estimated 500 billion devices will be connected to the Internet by 2030. These projections underscore the overwhelming potential impact of this technology. In fact, the rise of IoT has revolutionized our interaction with the surrounding world. From smart home devices to connected urban infrastructures, IoT promises to transform various sectors, including transportation, healthcare, agriculture, and energy, to name a few [8–15].

This positive impact of IoT extends beyond the social realm to the economic sphere. Thus, research from the McKinsey Global Institute [16] estimates that IoT could have a global economic impact ranging from $5.5 trillion to $12.6 trillion by 2030.

These figures highlight the immense socioeconomic impact and the current and future significance of IoT-based systems. However, developing this type of systems presents unique challenges requiring specialized methodological approaches. Consequently, emphasis must be placed on providing suitable development methodologies and tools for these systems, ensuring their quality and reliability.

Historically, software development methodologies have evolved to address the industry's changing needs and adapt to new technological paradigms. From the Waterfall methodology [17] to agile approaches [18, 19], the development community has consistently sought more efficient and effective ways to carry out projects, considering even team size and application domain [20].

✉ Miguel J. Hornos
mhornos@ugr.es

Mario Quinde
mario.quinde@udep.edu.pe

[1] Software Engineering Department, Research Centre for Information and Communication Technologies (CITIC-UGR), University of Granada, Granada, Spain

[2] Department of Industrial and Systems Engineering, Universidad de Piura, Piura, Peru

Unlike traditional information system development, which focuses solely on software, IoT system development must account for the configuration and implementation of all hardware devices that should be deployed in the environment to be controlled. The autonomous intercommunication between these devices and decision-making without human intervention are fundamental requirements in these systems, which also differentiates them from traditional information systems. The interaction between things and people through well-defined interfaces should also be considered [2, 21, 22]. In addition, their ubiquitous, wirelessly connected, and heterogeneous nature introduces additional concerns related to security, privacy, and interoperability. In some sense, all these quality properties are entangled [23]. Thus, development methodologies must address these specifics to ensure the delivery of robust and reliable IoT solutions. Consequently, it seems logical to assume that methodologies and tools for traditional system development may not be best suited for IoT system development [24, 25] and may need adaptation, in the best of cases, for use in this new paradigm.

Developing IoT systems is complex, as it entails managing interactions among multiple devices and communication systems, in addition to connections to Cloud, Fog, and/or Edge Computing systems, as well as the limited computational capabilities of many of the hardware devices that make up these systems. Moreover, these devices are heterogeneous in several respects. In fact, they vary in their nature, with different types of devices (sensors, actuators, processors, etc.) integrated into everyday objects. They also differ in how they interact, using various communication protocols and programming languages to dictate their behaviour. Furthermore, they serve different purposes, such as monitoring, communicating, controlling, and performing tasks, among others. Regarding software, many IoT systems are developed with intelligent agents [26–28]. These agents are programs capable of identifying environmental changes through the analysis of data captured by sensors and responding appropriately by acting on their environment through actuators. Typically, their ability to act proactively and "intelligently" stems from advanced Artificial Intelligence (AI) algorithms running on Cloud/Fog/Edge Computing systems. These systems not only host the agents but also process the information captured by the sensors. This vast diversity implies that the development of substantial IoT systems must be tackled by interdisciplinary teams [29, 30].

The rising adoption of IoT across various industries [31–33] and the consequent demand for more sophisticated and tailored solutions is clear. In this context, and given the nature and challenges of IoT systems, this article pretends to lay a solid foundation on current development methodologies and their applications in the IoT context, with the aim to assist researchers and professionals in selecting and adapting the most suitable methodologies for their specific projects. The presented challenges and open research directions in the area aid researchers and professionals in being more ready to address the issues they may face in their projects. It also directs research efforts towards problems whose solutions may positively impact the spread of IoT-based systems and their benefits to society.

The remainder of this article is organised as follows: Sect. 2 provides a comprehensive overview of the current methodologies used in developing IoT-based systems, pinpointing their strengths, limitations, and the scenarios in which it is convenient to use them. The analysis of these methodologies leads to highlighting the current challenges in IoT-based systems development, which are explained and criticised in Sect. 3, considering the complex and multifaceted nature of IoT-based systems. Section 4 proposes open research avenues in developing IoT-based systems based on the previous analysis of the current methodologies and challenges in the area. Finally, Sect. 5 presents the main conclusions of this article.

2 Methodologies for IoT-based system development

2.1 Overview of existing methodologies

2.1.1 Development methodologies for traditional information systems

Among the methodologies designed for the development of more traditional software systems, we can mention the **Waterfall methodology** [17]. Recognized as the earliest known software development methodology, the Waterfall approach addresses the development of large-scale information systems (ISs) holistically. It mandates that development teams follow a sequential set of steps, refraining from moving forward until the preceding phase is fully completed. This methodology offers limited flexibility for unforeseen changes, a significant drawback given that end-users often are uncertain about their requirements at the start of the development process.

Unlike the Waterfall methodology, the **Spiral methodology** [34] views software development phases as iterative tasks rather than a linear sequence. This approach, with its spiral repetition of tasks, enables early detection of unfeasible system developments, allowing teams to halt projects before investing significant resources [35].

As previously mentioned, during the analysis phase of a software system, end-users or clients typically lack a clear understanding of all the functionalities and quality

properties the system should possess. The **Prototyping methodology** [36] emerged to formalize the presentation of iterative product versions to end-users for evaluation. Given its proven utility, prototyping has been integrated into other methodologies, especially agile ones.

Among the most popular development methodologies are **agile methodologies** and those following a **model-based approach**. Given their importance, we will delve into them separately in the upcoming subsections: 2.1.2 for agile methodologies and 2.1.3 for model-based approaches.

While methodologies designed for the development of conventional ISs have been used to develop IoT systems [24], their application to these systems presents notable limitations. Specifically, these methodologies do not address specific aspects of IoT systems, such as the design and deployment of hardware devices (such as sensors, actuators, processors, etc.) in the target environment or the heterogeneity of the components and technologies used in them. They also overlook other crucial aspects, like the incorporation of AI techniques to empower the system with decision-making capabilities and context-awareness, enabling it to react appropriately to any event happening on the environment controlled by the IoT system [37].

Nevertheless, several of these methodologies have undergone adaptations to cater to IoT system development, as will be elaborated upon in Sect. 2.3.

2.1.2 Agile methodologies

Agile methodologies have revolutionized the software development world. They emerged in response to challenges associated with the development of large systems, where factors such as budgetary, technological, and resource constraints threatened the successful completion of projects [38, 39]. These methodologies have proven effective in enhancing project success rates and in reducing the budgetary consumption of those that are abandoned [40].

At the heart of agile methodologies is their modular approach. Instead of tackling a system as a whole, it is divided into deliverables or modules. These modules allow the client to verify and utilize parts of the system before it is fully completed. This approach is grounded in the 4 values and 12 principles of the Agile Manifesto [41–43]. The goal is to deploy fully functional products within a short timeframe, typically 4 to 6 weeks.

However, like any methodology, there are advantages and challenges. Distinctive features of agile methodologies include small teams, deliverables negotiated with the client, and the flexibility to introduce changes at any time [42, 43]. Yet, these very features can pose challenges and difficulties in certain contexts, such as in the development of large enterprise software projects where time is of the essence [44]. The client's dependency in determining the priority of deliverables can also be a hindrance to the productivity of the development team [45].

Despite these challenges, agile methodologies have found applications beyond traditional software development. They have been used in the development of IoT systems, combining techniques from different methodologies, such as Scrum with eXtreme Programming (XP) [46] and Scrum with Rapid Prototyping (RP) [47, 48].

A key feature of agile methodologies is their adaptability. Unlike traditional methodologies like Waterfall or Spiral, agile methodologies allow for changes at any stage of development. This flexibility is crucial, as all requirements are seldom known at the onset of a project [49]. Tools like user stories, defined in the international standard ISO/IEC/IEEE 26515 [50], help capture requirements in a manner understandable to both developers and clients. However, developers' interpretation of these stories can be challenging [51].

Proper requirement gathering is vital for the success of any project. In the context of IoT systems, Non-Functional Requirements (NFRs) are of paramount importance. While agile methodologies are powerful, they may face challenges when dealing with NFRs, especially if solely relying on tools like user stories [52]. Therefore, it is essential for developers and users to collaborate to ensure the final system meets user expectations and needs [53].

In summary, agile methodologies offer a flexible and modular approach to software development. Although they pose challenges, their adaptability and emphasis on collaboration between developers and clients make them ideal for a wide range of projects, and they could even be applied to the development of IoT systems.

2.1.3 Model-based approaches

IoT-based systems pose unique challenges for developers due to the heterogeneity of their components and the technologies involved. This heterogeneity is evident in the variety of implementations and operational features provided by different manufacturers, often resulting in a diverse software and communication platform. Moreover, the need to deploy shared functionalities across multiple distinct devices amplifies the complexity of development.

In this context, development methodologies following model-based approaches emerge as a promising solution to address these challenges. Indeed, these methodologies were introduced with the intent of focusing on the functional design of a system, primarily operating at an abstraction level that is independent of any specific platform. Their core goal is to achieve comprehensive system implementations while minimizing the amount of platform-dependent code that developers must write.

Below, we present the primary model-based approaches and describe their essential features:

- **Model-based engineering (MBE)**: In MBE, which is the most general approach, models play a significant role in the development process, but they are not necessarily the key artifacts. These models capture everything programmers need to understand to write code in a target programming language. Automatic code generation is not a central aspect of MBE, allowing greater flexibility in how models are used in the development process. An example of an environment using MBE is the Open Model Based Engineering Environment [54].
- **Model-driven engineering (MDE)**: Unlike MBE, models are fundamental in the MDE development process. From the defined models, it is expected that at least part of the code or even other models will be automatically generated [55]. These transformations can be model-to-text (M2T) or model-to-model (M2M). Furthermore, in MDE, models and transformations can be defined in any modelling language and address different levels of abstraction.
- **Model-driven development (MDD)**: This approach, which can be considered a specialization or restriction of the previous one, focuses exclusively on the development process. Models are the primary artifact, allowing the use of any modelling language. Most of the implementation is automatically generated from the models, facilitating the management of heterogeneity in IoT system development [56].
- **Model-driven architecture (MDA)**: This is a more specific and standardized vision of MDD, proposed by the Object Management Group [57]. Unlike MDD, MDA specifies that UML (Unified Modelling Language) should be the primary modelling language and that any transformation should be specified using the QVT (Query/View/Transform) language. Additionally, MDA establishes a specific order for transformations, from Computation Independent Models (CIMs) to Platform Specific Models (PSMs), passing through Platform Independent Models (PIMs). The main goal of MDA is to focus on *modelling* software solutions, setting aside the technical specifics of implementations.

In conclusion, modelling has established itself as an essential tool in the development of IoT systems, providing a means to manage the inherent complexity and heterogeneity of these systems. From MBE to MDA, these approaches offer different levels of abstraction and automation, allowing developers to choose the one that best fits their needs and the specific context of their project. The relationship between these four approaches is graphically represented in Fig. 1. As can be seen, MDA is the most specific concept and can be considered a subset of MDD, which in turn is a subset of MDE, with MBE being the superset that encompasses all of them. Therefore, all model-driven processes are model-based, but not vice versa.

2.2 Life cycle stages of software systems

A methodology can be defined as a systematic way of doing things in a particular discipline [58]. Software Engineering is the discipline responsible for providing an appropriate methodology for the development of all types of computer systems [59]. Determining the steps that a methodology should specify for computer system development is challenging without considering the specific type of system to be developed or the approach to be used by the methodology to achieve it. In our study, we will focus on the development of IoT systems, although we will consider methodologies that follow any approach.

To unify the different nomenclatures found in the literature, we will consider that methodologies for IoT system development are organized into different phases or stages, which are subdivided into various activities or tasks. Fortino et al. [37], who also addressed the analysis of existing methodologies for IoT system development, to harmonize the terminology encountered during their review process, referred to the standard ISO/IEC/IEEE 24765 [60], the System Engineering Body of Knowledge (SEBoK) [61], and the Project Management Body of Knowledge (PMBoK) [62, 63]. In addition to these documents, we have consulted the work written by Farncombe [64], as well as the software engineering activities outlined by Pressman and Maxim [65] and Bourque and Fairley [59]. All these sources have aided in clarifying the names of the IoT system life cycle phases or stages that form the foundation for this work, along with their associated activities or tasks.

Furthermore, the following international standards have been consulted: ISO/IEC/IEEE 12207 [66], which addresses software life cycle processes; ISO/IEC/IEEE 15288 [67], which tackles system life cycle processes; ISO/IEC/IEEE 24748-1 [68], which provides unified guidance on the life cycle management of systems and software; ISO/IEC/IEEE 24748-2 [69], which offers specific guidelines on the management of system life cycle processes; and ISO/IEC/IEEE 24748-3 [70], which does the same for software life cycle process management. Methodologies are free to specify how to execute the stages and technical processes proposed in the ISO/IEC/IEEE standards, their order of execution, and even select which processes will be considered and which will not. The application of methodologies based on these international standards to the development of IoT systems

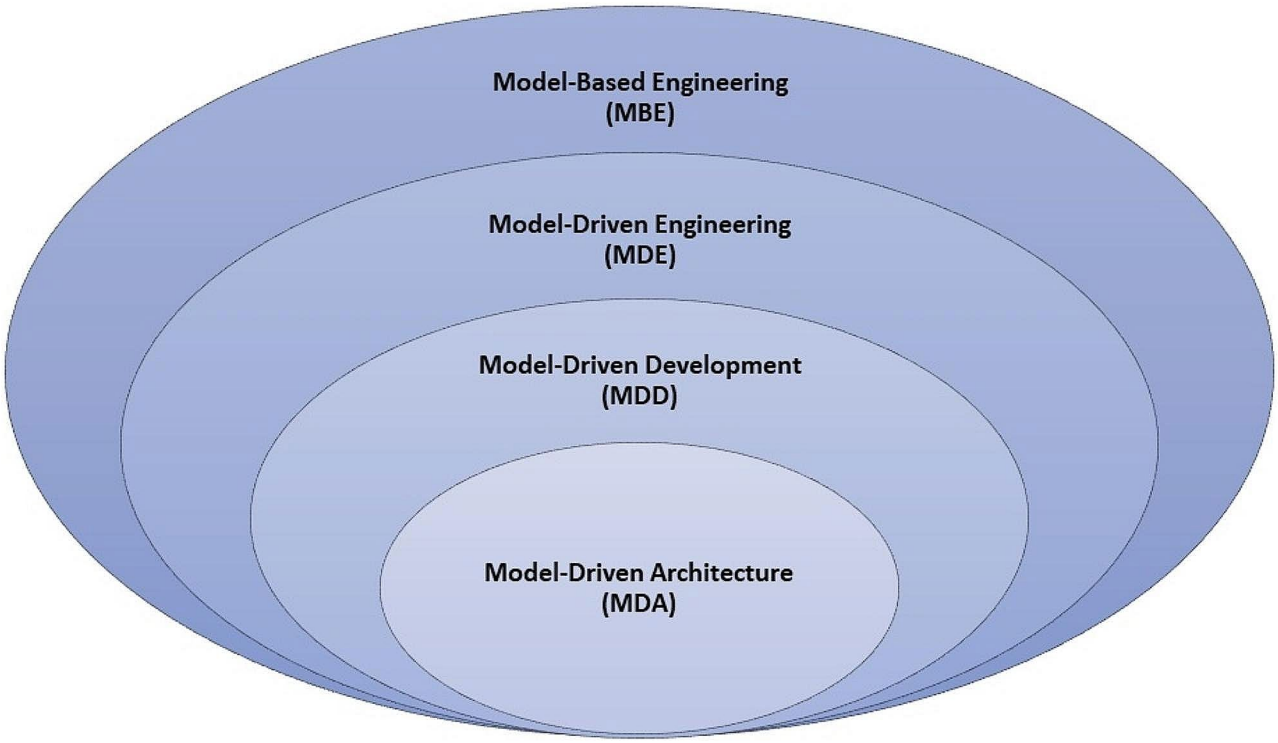

Fig. 1 Relationship between the model-based approaches

contributes to the delivery of a quality product on time and within the established budget [71].

After reviewing the aforementioned references, as well as many others, we observed that there are various classifications regarding the phases and activities involved in developing such systems, depending on the source. Consequently, we have decided to adopt the classification that delineates the following four major stages or phases, along with their corresponding activities (indicated in parentheses), for our study: Conception (planning and requirements elicitation), Modelling (analysis, design, model verification, and simulation), Construction (coding or implementation, integration, testing, and deployment), and Post-Construction (management or operation, maintenance, and system dismantling or decommissioning). These stages and their activities are illustrated in Fig. 2. The core stages of Modelling and Construction are considered to represent the actual *development* of the system.

Pressman and Maxim [65] recognize the need to consider maintenance as soon as the development model is chosen. However, this aspect is not clearly defined in any of the analysed methodologies. Moreover, certain studies propose activities that may be necessary depending on the goals of the specific methodology, such as model verification and refinement, as well as simulation [72–75].

We have considered methodologies that explicitly address one or more stages or phases of IoT system development, though they may not provide all the activities for each phase. For instance, if a methodology proposes *requirements analysis* but does not provide evidence of *requirements elicitation*, we have deemed that it does not fully comply with the Conception stage. Similarly, the first stage of the methodology presented by Rashid et al. [72] is *functional modelling* (i.e., *design*), which implicitly assumes prior activities such as *planning*, *requirements elicitation*, and *analysis*. Even though these activities are not explicitly mentioned, it is understood that they must have been conducted beforehand. Furthermore, only activities associated with system *design* are proposed by Coronato and De Pietro [76] and McGrath et al. [77], without addressing the necessary preliminary *analysis*. Therefore, we consider that these methodologies do not fully cover the Modelling stage.

Although some development methodologies [21, 73, 78, 79] clearly establish the phases and/or activities defined within them, others present these phases and/or activities in a very abstract manner. For instance, we identified five proposals [80–84] that only present a single activity in the Construction phase, termed *Development*. In contrast, most of the works we reviewed distinguish multiple activities within that "phase." As indicated above, we believe that

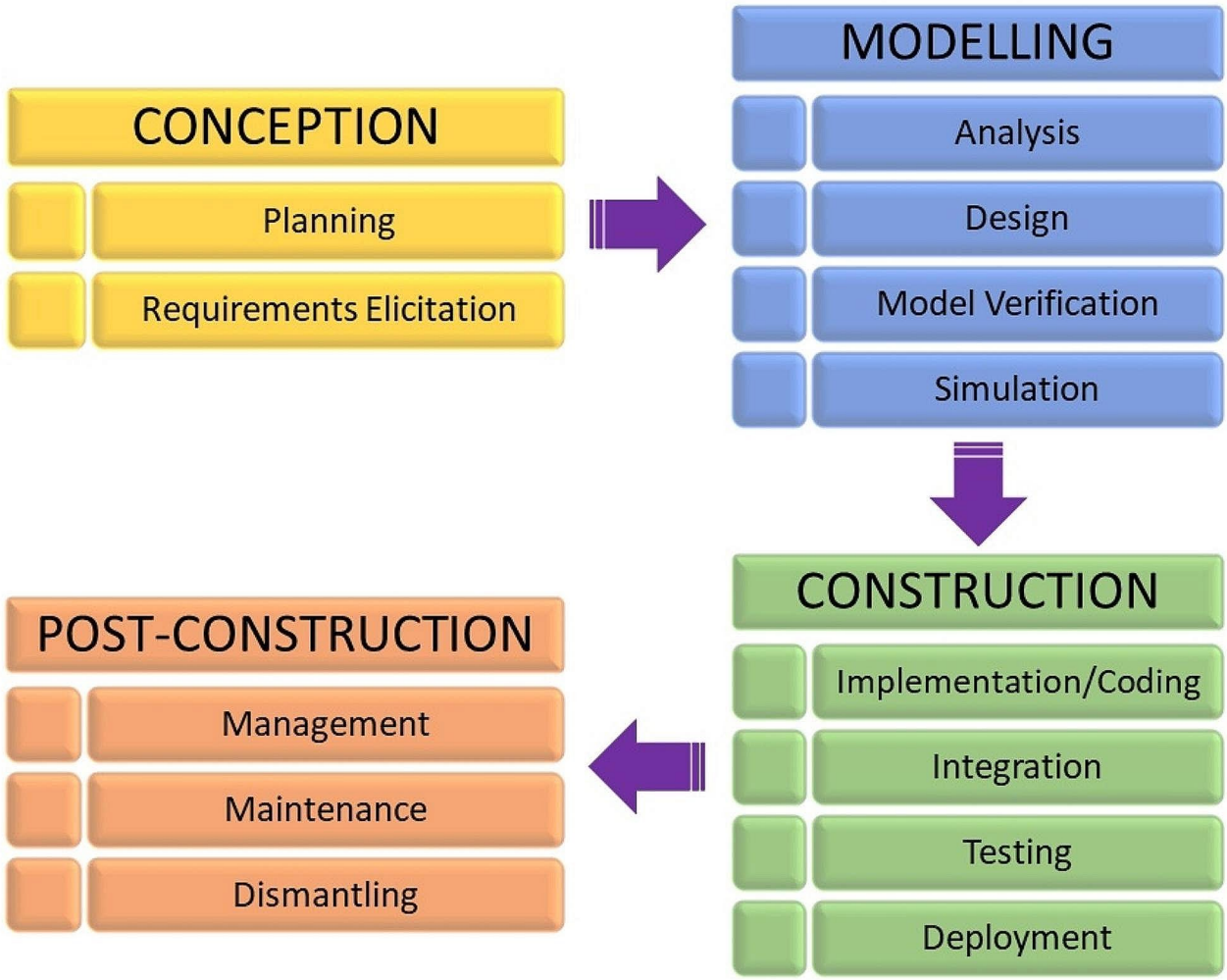

Fig. 2 Classification of stages and activities considered in our study

development could encompass all activities included in both the Modelling and Construction stages (refer to Fig. 2).

In the methodologies identified during our literature review, *design* emerged as the most frequently mentioned activity across the articles analysed. This is followed by activities related to software *coding or implementation*, as well as *analysis* tasks. Notably less mentioned are the activities corresponding to the Post-Construction stage, i.e., system *management*, *maintenance*, and *dismantling or decommissioning* after its useful life has ended. It is particularly noteworthy that system *dismantling or decommissioning* is addressed by only one methodology [85], and system *management* is mentioned in just a couple of works [86, 87]. Additionally, certain activities are unique to specific methodologies. For instance, the methodology presented by Guerrero-Ulloa et al. [21] uniquely recommends conducting an *operational feasibility study* to assess whether the system can continue functioning after installation.

Fahmideh et al. [88] present comprehensive software engineering guidelines for IoT system development, which may be beneficial for researchers formulating a development methodology for such systems. In their approach, they present 27 tasks that span from the system ideation to its installation. These tasks are categorized into three phases: *analysis*, *design*, and *implementation*. However, they overlook the Post-Construction phase, including crucial activities like system *management* and *maintenance*, which we believe are fundamental to the system life cycle.

2.3 Analysing key methodologies for IoT system development

In this section, we explore various methodologies that have been proposed to address the inherent challenges in the development of IoT systems. These methodologies can be grouped according to their approach and distinctive

features, allowing for a more structured comparison and a better understanding of their applications and limitations.

2.3.1 Methodologies based on (meta)modelling

Within these methodologies, there is a strong emphasis on the modelling and construction stages of IoT systems, as it should be. A common approach is the use of model transformations, which allows moving from high-level specifications to concrete implementations.

Lekidis et al. [89] propose a design flow that integrates MDE and Service-Oriented Architecture (SOA), focused on the Contiki platform. This approach is notable for its support in the verification and validation of requirements, ensuring compliance with Functional Requirements (FRs) and NFRs. On the other hand, MDE4IoT [90] focuses exclusively on modelling and generating the final product, without addressing activities such as planning or maintenance. It uses Domain-Specific Modelling Languages (DSMLs) for the transformation of models into executable artifacts. Harbouche et al. [91] present an MDE methodology that follows a top-down paradigm, with a strong emphasis on automation to derive system designs from global requirements, using specific metamodels for each level of abstraction.

The ROOD (Resource-Oriented and Ontology-driven Development) methodology [92] combines MDA with MDE-based tools, offering a dual approach that considers both the behaviour of resources (sensors and actuators) and the intelligent object. It is distinguished by its structure in three main stages, where the first constructs the MDA's CIM, the second the PIM, and the last, the PSM, and its emphasis on verifying the consistency of the model generated at each stage.

The methodology proposed by Sosa-Reyna et al. [56, 93, 94] is based on MDD/MDE and covers a complete development process, from requirements analysis to the generation of technological solutions, using UML and Business Process Model and Notation (BPMN). However, it does not address the development of user interfaces.

Brambilla et al. [95] present an MDD-based approach specifically designed to develop mobile applications for IoT systems. It is based on Interaction Flow Modelling Language (IFML), an extension of UML that allows for visually modelling user interactions with the system and representing the system's behaviour in response to such interaction, among other things. The authors have developed IoT-focused design patterns for the most common use cases, covering user interaction and data synchronization. Although the methodology is innovative in its modelling approach, based on design patterns, its applicability is restricted to certain areas and types of IoT systems.

IDeA [96] is a methodology for the development of IoT systems that is based on Model-Based Systems Engineering (MBSE) and provides high-level abstractions using metamodeling to address heterogeneity in IoT. Although it aligns with the ISO/IEC/IEEE 15288 standard [67], it does not detail all its phases nor provides specific guidelines for its application. Meanwhile, the Smart Environment Metamodel (SEM) framework [97], which also relies on metamodels, focuses on the functions and data of the IoT system to be developed, concentrating on requirements analysis, design, and implementation.

For the Construction stage, many of the methodologies analysed incorporate technologies capable of automatically generating code in various languages. Some of them [89–91, 98] generate code in C, C++, or some variant of this language, as they are the most popular languages in computer boards (controllers) used in the development of IoT systems. Other methodologies [90–92, 99] generate code in Java, and we have even found a methodology [100] that generates code for the Node.js environment.

To facilitate a better comparison of the different methodological approaches presented in this subsection, Table 1 shows the main characteristics of each of the proposals.

2.3.2 Service-oriented methodologies

AMG (Abstract, Model and Generate) [22] is a methodology based on SOA for the development of IoT software that follows a bottom-up approach, starting with concrete models to derive abstract services. It is structured in three fundamental phases: definition of abstractions, modelling, and code generation. The process begins with service descriptions to obtain graphical representations and subsequently, the source code.

As indicated in the previous section, the methodological proposal by Lekidis et al. [89] is not only based on MDE but also on SOA, meaning that this approach supports not only modelling but also the implementation and deployment of IoT systems. It uses the Behaviour, Interaction, Priority (BIP) component framework for designing web service applications, supporting the modular design and reuse of model artifacts. In addition, it applies the principles of separation of concerns in a component-based design process, facilitating the clear delineation of different aspects of the system for better manageability. The methodology proposed by Sosa-Reyna et al. [56, 93, 94] also appears in both sections. It combines MDD and MDE, using an SOA approach in the third stage to obtain a more refined model (PSM) from the PIM generated in the first two stages. Finally, in the fourth stage, the PSM is transformed into code.

SERVUS [49] is a methodology for developing Industrial IoT (IIoT) systems that focuses on solving interoperability

Table 1 Main characteristics of the methodological approaches based on (meta)modelling

Proposal name / Authors [Ref.]	Main characteristics of the proposal
Lekidis et al. [89]	• Proposes a component-based design flow. • Integrates MDE and SOA. • Supports verification and validation of FRs and NFRs. • Generates code automatically.
MDE4IoT / Ciccozzi & Spalazzese [90]	• Based on MDE. • Uses DSMLs to transform models into executable artifacts. • Enables runtime self-adaptation. • Reuses design artifacts.
Harbouche et al. [91]	• Follows a top-down paradigm based on MDE. • Transforms models into Promela programs for system analysis and formal verification. • Uses specific metamodels for each level of abstraction.
ROOD / Corredor et al. [92]	• Combines MDA with MDE-based tools. • Structured in 3 stages, with CIM, PIM, and PSM constructed respectively. • Semantically verifies the consistency of the models. • Supported by open tools within the Eclipse project.
Sosa-Reyna et al. [56, 93, 94]	• Based on MDD/MDE. • Uses UML and BPMN to define and model business processes and services. • Automatically transforms conceptual models (CIM, PIM, PSM) into executable code. • Formally verifies consistency among models and their correctness regarding FRs and NFRs.
Brambilla et al. [95]	• MDD-based approach to create user interfaces for IoT systems. • Uses reusable design patterns for user interaction and data synchronization. • Visually models user-system interactions using IFML, an extension of UML. • Automatically generates user interfaces from IFML models. • Only applicable to specific areas and certain types of IoT systems.
IDeA / Costa et al. [96]	• Based on MBSE and focused on the design phase. • Uses the SysML4IoT profile to model, SysML2NuSMV to translate from model to text, and the NuSMV model checker. • Provides a methodology to verify Quality of Service (QoS) properties.
SEM / Cicirelli et al. [97]	• Relies on metamodels focusing on two perspectives: functional and data. • The functional perspective focuses on the services provided by the IoT system. • The data perspective describes and analyses data from sensors and actuators. • Uses semantic technologies to verify the consistency of the models.
Ataide et al. [98]	• Uses Input-Output Place-Transition (IOPT) Petri nets modeling tools to automatically generate executable code for Arduino boards. • Involves creating Petri net models, splitting them into sub-models for each time domain, and generating C code files for deployment on Arduino boards.
Chauhan et al. [100]	• Proposes a specific development framework for Cyber-Physical Systems (CPSs). • Based on microservices architecture, autonomic computing, and MDD paradigms. • Addresses the necessary security and protective measures for CPSs. • Uses cloud computing and provides self-configuring services and resources.

issues through an SOA architecture aligned with the Industrial Internet Reference Architecture (IIRA) [33]. It covers the elicitation and analysis of requirements, as well as analysis and design activities, using user stories and use cases. However, it does not provide details on how to address the Post-Construction stage.

Table 2 summarizes the main characteristics of each of the service-oriented methodological proposals discussed in this subsection, facilitating a better comparison among them.

2.3.3 Agent-oriented methodologies

In this section, we identify several proposals that share the common characteristic of modelling IoT systems as sets of autonomous agents interacting with each other.

ELDAMeth [99] is an iterative simulation-based methodology for Distributed Agent Systems (DASs), which uses the ELDA (Event-driven Lightweight Distilled statecharts-based Agents) agent model and facilitates rapid prototyping through visual programming and automatic code generation for DASs within the JADE (Java Agent DEvelopment) framework. It encompasses low-level design, simulation-based validation, and implementation.

The methodology proposed by Pico-Valencia et al. [101] is based on the principles of both the Agile Manifesto

Table 2 Main characteristics of the service-oriented methodologies reviewed

Proposal name / Authors [Ref.]	Main characteristics of the proposal
AMG / Sulistyo [22]	• Follows a bottom-up approach based on SOA. • Structured in three phases: definition of abstractions, modeling, and code generation. • Starts with service descriptions to obtain graphical representations and source code. • Automates the development process to reduce cost and development time.
Lekidis et al. [89]	• Based on MDE and SOA. • Applies separation of concerns in a component-based design process. • Uses the BIP framework and tools for state-space exploration to verify properties. • Ensures deployed code is consistent with the validated model.
Sosa-Reyna et al. [56, 93, 94]	• Combines MDD/MDE with SOA, ensuring seamless component interaction. • Transforms conceptual models (CIM, PIM, PSM) into executable code. • Uses formal verification to ensure model consistency and correctness for FRs and NFRs. • Includes tools for model transformation and code generation.
SERVUS / Usländer & Batz [49]	• Analysis and design methodology for IIoT systems that bridges the gap between requirements analysis and system design. • Solves interoperability issues with an SOA architecture aligned with IIRA. • Maps FRs and NFRs to the capabilities and interaction patterns of IIoT service platforms. • Introduces a web-based system to support the methodology.

[41–43], promoting iterative development and continuous feedback, and the Linked Open Data (LOD) approach, focusing on ensuring interoperability and data integration across different systems. It models IoT systems as a network of autonomous agents, referred to as Linked Open Agents (LOAs), which communicate and cooperate to achieve the desired functionalities. It employs MDD techniques to automatically generate code and system models, facilitating rapid prototyping and deployment of IoT applications. Implementation and integration are addressed at a macroscopic level, where LOAs are coordinated within a network. The authors of this methodology also mention tools for the analysis and design stages but omit details on how to carry out certain activities, such as planning, requirements elicitation, coding, and deployment, for example.

Fortino et al. [102] also propose an agent-oriented approach based on metamodels for the development of Smart Objects (SOs). The analysis stage involves modelling the SOs using a specific metamodel, while the design focuses on modelling functional components and their interactions. The implementation uses a specialized metamodel for the JADE platform, known as JACOSO [103].

Table 3 Main characteristics of the agent-oriented methodologies analysed

Proposal name / Authors [Ref.]	Main characteristics of the proposal
ELDAMeth / Fortino & Russo [99]	• Iterative simulation-based methodology for DASs. • Facilitates rapid prototyping and automatic code generation. • Uses the ELDA agent model and the JADE framework. • Includes dynamic validation methods during the design phase.
Pico-Valencia et al. [101]	• Based on both Agile and LOD principles. • Models an IoT systems as a network of LOAs. • Involves the collection of global requirements and detailed design associated with each LOA. • Uses MDD techniques to automatically generate code and system models.
Fortino et al. [102]	• Agent-oriented approach based on metamodels for SOs. • Follows an iterative development process that enhances flexibility and adaptability to changing requirements. • Supports the design of functional components and their interactions. • Uses JACOSO, a specialized metamodel for the JADE platform.

Table 3 provides a summary of the main characteristics of the agent-oriented methodologies discussed in this subsection, allowing for an enhanced comparison among them.

2.3.4 Methodologies based on other approaches

This section will focus on presenting an overview of development methodologies for IoT-based systems grounded in various approaches, highlighting their main characteristics.

TDDM4IoTS (Test-Driven Development Methodology for IoT-based Systems) [21] is an iterative methodology that integrates concepts from MDE and TDD (Test-Driven Development) [104], as it focuses on the creation and exploitation of models from which system code snippets are generated, as well as the tests that the generated code must pass. It also adheres to the values and principles of the Agile Manifesto [41–43]. TDDM4IoTS aims to address both the business logic of the system to be developed and the user-system interaction, as well as the configuration and deployment of hardware (sensors, actuators, processors, etc.) and the programming of single-board computers (SBCs) such as Arduino and Raspberry Pi. It consists of 11 stages or activities, detailing resources and tools for each, and closely aligns with ISO/IEC/IEEE standards [50, 66–70]. To facilitate the use of this methodology, its authors have developed a supporting tool called TDDT4IoTS (Test-Driven Development Tool for IoT-based Systems) [105]. The methodology-tool tandem has been validated through its application to case studies in various domains, in which IoT systems

have been developed for: indoor air quality control [106], assisting visually impaired people with outdoor movements [107], and caring for indoor ornamental plants [108], among others.

INTER-Meth [109] is an iterative adaptation of the waterfall methodology that divides the problem into manageable subproblems to facilitate development and ensure its success. It defines six sequential development stages and allows iterations to improve adaptability to new requirements. It lacks specific guidelines and activities for hardware deployment, which is an essential part of IoT systems.

RASPSS (Rehabilitation Assistive Smart Product-Service System) [110] is another iterative methodology, aimed at the design and implementation of smart health services for special groups, such as individuals with disabilities or those requiring rehabilitation, focusing on the construction of intelligent IoT devices applying AI techniques. It also emphasizes the importance of user interaction.

The methodology proposed by Patel and Cassou [111] focuses on the roles played by the development team to address technological heterogeneity in IoT. It covers analysis, design, construction, and testing, so that experts in each of these activities carry them out.

The approach by Gogineni et al. [112] follows the V-Model XT, emphasizing the verification and validation of requirements and functionalities. It considers requirement elicitation and design, as well as integration and testing, but does not address Post-Construction stage activities. MEDISTAM-RT [113] is another methodology that uses the V model, following a verification approach based on timed traces semantics and UML-RT models to ensure the fulfilment of non-functional requirements (NFRs) such as timeliness and safety in the context of emergency treatment services, which are critical and time-constrained. This methodology emphasizes the need for formal approaches in the specification and verification of systems to ensure compliance with NFRs and proposes the combination of semi-formal and formal notations for system design and analysis, which are the development activities it focuses on. It does not address the Construction or Post-Construction stages.

MIRIE (Methodology for Improving the Reliability of Intelligent Environments) [74] aims to guide developers in modelling reliable Intelligent Environments (IEs) using existing software engineering tools and techniques [114] and capturing the essential behaviour of IE components through a series of models that are successively refined, starting with very basic ones to which details are progressively added. Creating reliable IEs is essential to increase user trust in such environments [115]. This methodology proposes using the Spin model checker [116] as a support tool, with simulation and formal verification capabilities, specifically focusing on the Modelling stage, although it includes an *informal modelling* activity in which all stakeholders participate, where the requirements and functionalities that the system must have are collected. This methodology also does not address the Construction or Post-Construction stages.

User-centric methodologies to develop IoT-based systems under the umbrella term of IEs have been studied. Augusto et al. [117] define IEs as *"environments in which the actions of numerous networked controllers are orchestrated by self-programming pre-emptive processes in such a way as to create an interactive holistic functionality that enhances occupants' experiences"*. IEs are built based on three main concepts that are different but also related to each other. These concepts are pervasive/ubiquitous computing, which studies the distribution of computational services emphasising the devices (including their networking and processing) and the human-computer interaction components; smart environments, which are environments equipped with sensing devices; and ambient intelligence, which refers to the intelligent software installed in the environments.

The concept of System-of-Systems (SoS) can be used to explain the complexity that influences the technical and management aspects of IEs [118]. This complexity has made researchers aware of the need to develop new methodologies aiming to guide the creation of IEs. The User-Centred Intelligent Environments Development Process (U-CIEDP) [119] is an example of a research work aiming to address this gap.

Research on providing tools supporting more specific activities within the development process of IEs has also been made. An ethical FRamework for Intelligent ENvironment Development (eFRIEND) [120] proposes methodologies guiding requirements elicitation for IEs. The Smart Environments Architecture (SEArch) [121] and the Context-Aware Systems Architecture (CASA) [122] aim to guide the architectural design of IEs. Other works explore verification in IEs [74, 113, 123]. Finally, the Context-Aware Test Suit (CATS) Design [124] and the COntext-Aware systems Testing and Validation (COATI) [125] are examples of methodologies for IEs testing and validation.

The U-CIEDP methodology [119] puts users at the centre of the IE development process to ensure that the services to be delivered closely match users' expectations. The methodology considers the software, hardware, networks and interfaces building up an IE as components that have to be designed and put together, acknowledging the human factor as the most influential one in the creation of this technology. The foundation of the U-CIEDP recipe for success is the idea that IE technology is deployed in the real world where it influences people's daily activities, committing to some actions that users may like or not but that, in some cases, cannot be reverted. This foundation has made the U-CIEDP to be recognised as a tool allowing the creation

of technology in a morally and ethically responsible way [126].

The U-CIEDP has been used to guide the co-creation of IE-related technology for people with special needs. In fact, it has been employed in the PersOnalised Smart Environments to increase Inclusion of people with DOwn's syNdrome (POSEIDON) project [119], the development of the Ambient Assisted Empowered Living (AnAbEL) system [127], which aims at enhancing autonomy and coaching for people with dementia or cognitive decline, and the creation and validation of the Approach to Develop context-Aware solutions for Personalised asthma managemenT (ADAPT) [128].

Research works on important issues strongly associated with IEs have also been motivated by the use U-CIEDP in the development of IEs. Some studies explore the use of argumentation to manage users' preferences in Ambient Intelligence [129, 130]. Ali et al. [131] investigate improving the adaptation process of a new smart home user. Although these research works do not follow the U-CIEDP as explicitly as those ones explained above, they have been motivated by the lessons learnt from applying the U-CIEDP.

The U-CIEDP has also inspired research on requirements elicitation [120], architecture design [121, 122], and testing and validation [125, 132] in IEs. These research works are responses aiming to address the key challenges that have been identified within the three main loops of the U-CIEDP when it was used to guide the development of IEs. These loops are the IEs' initial scoping, IEs' main development, and IEs' installation.

Table 4 highlights and summarizes the key features of the various methodological approaches presented in this subsection, facilitating a comprehensive comparison.

2.4 Conclusions from the methodology review conducted

The development process for such systems should cover each of the stages and activities of the system/software lifecycle outlined in the ISO/IEC/IEEE standards [50, 66–70], from planning and requirements elicitation to system decommissioning. However, most of the methodologies analysed focus on about half of these stages and activities (graphically represented in Fig. 2), with almost all of them concentrating on the modelling and construction stages of IoT systems, as is logical. In fact, very few of the methodologies examined in the previous section appropriately address either the planning or maintenance activities, for example, and none address the system dismantling or decommissioning activity, as shown in Table 5, where these symbols are used with the following meanings: ✓ "Proper compliance", ~ "Incomplete or inadequate compliance", and × "Not addressed or mentioned".

Planning is an activity often overlooked in IoT development methodologies. Only TDDM4IoTS [21] explicitly includes it, while the methodologies proposed by Gogineni et al. [112] and Fortino et al. [133] only show indications of considering it. The inclusion of planning can be a key differentiator in the success of IoT development methodologies. It should address aspects such as the analysis of existing technology to develop the project, the analysis of the environment where the system will be deployed, and the project's (technical, economical and operational) feasibility analysis.

Requirements elicitation and analysis are fundamental in the development of IoT systems. While authors like Gogineni et al. [112], Wang et al. [110], and Fortino et al. [102] integrate requirements analysis within the system analysis, other researchers, such as Guerrero-Ulloa et al. [21], Usländer and Batz [49], and Sosa-Reyna et al. [56, 93, 94], clearly distinguish these activities and emphasize the importance of precise requirements collection from the start of the project. This clarity in defining requirements is essential to avoid delays in IoT development. Nevertheless, there are authors, like Brambilla et al. [95] and Fortino and Russo [99], who assume that these activities are already resolved and, therefore, omit to mention applicable analysis methods or tools. Moreover, some methodologies do not refer to the necessary requirements at all, as is the case with MDE4IoT [90].

Regarding ***design***, methodologies are primarily based on different types of models and metamodels. Indeed, methodologies that follow model-based approaches, and fundamentally those based on MDE, MDD, and MDA, are among the most used for the development of IoT systems [24]. The success of these methodologies could be due to their help in solving the problem of the great technological heterogeneity and hardware components existing in the development of these systems.

As for the ***generation of software code***, methodologies such as AMG [22] and SEM [97] focus on modelling and generating software for IoT devices but often overlook the development of applications for user-system interaction. Although some authors, like Fortino et al. [102], Pico-Valencia et al. [101], and Wang et al. [110], recognize the importance of user-centred design, they seem to focus exclusively on obtaining the hardware component of the IoT system and the software for its configuration, with no evidence that they address the construction of applications for the end-user. Therefore, these types of methodologies have an opportunity to improve usability and user experience when developing quality IoT systems if they included this last aspect.

Table 4 Main characteristics of the methodologies based on different approaches studied

Proposal name / Authors [Ref.]	Main characteristics of the proposal
TDDM4IoTS / Guerrero-Ulloa et al. [21]	• Integrates MDE and TDD principles, focusing on generating system code and tests from models. • Adheres to Agile values for flexibility and iterative development. • Addresses business logic, user-system interaction, hardware configuration and deployment, and SBC programming. • Supported by the TDDT4IoTS tool.
INTER-Meth / Fortino et al. [109]	• Iterative adaptation of the waterfall methodology for flexibility and adaptability, allowing iterations for new requirements and refinements. • Divides the problem into subproblems for easier development. • Uses a layered approach to manage the integration of heterogeneous IoT systems, focusing on interoperability across layers. • Designed to ensure seamless interaction and communication among different IoT platforms.
RASPSS / Wang et al. [110]	• Focuses on the design and implementation of smart health services using AI techniques. • Emphasizes user interaction and personalized services for rehabilitation assistive devices. • Incorporates iterative development for continuous improvement and adaptation. • Integrates manufacturing and service systems for special groups.
Patel & Cassou [111]	• Role-based methodology addressing technological heterogeneity in IoT, ensuring expertise in each phase by assigning specific roles to team members. • Simplifies development by separating concerns and integrating code generation techniques. • Utilizes task-mapping and linking techniques for deployment. • Produces device-specific programming frameworks to manage scale and heterogeneity complexity.
Gogineni et al. [112]	• Proposes a systematic product development methodology for customizable IoT devices, following the V-Model XT. • Integrates design and manufacturing processes to streamline product development. • Emphasizes user-centric approach and customization according to user needs. • Includes validation and testing phases for ensuring functionality, reliability and performance.
MEDISTAM-RT / Benghazi et al. [113]	• Uses the V model, focusing on verification based on timed traces semantics and UML-RT models. • Combines semi-formal and formal notations for system design and analysis. • Ensures the fulfillment of certain NFRs, such as timeliness and safety. • Provides a formal representation of time-dependent behaviors.
MIRIE / Augusto & Hornos [74]	• Guides developers in modeling reliable IEs using software engineering tools. • Uses the Spin model checker for simulation and formal verification. • Focuses on the Modelling stage, with stakeholder participation in informal modeling activities. • Captures essential IE component behavior through progressively refined models.
U-CIEDP / Augusto et al. [119]	• User-centered methodology ensuring services match users' expectations. • Involves co-creation with users and other relevant stakeholders, emphasizing ethical and responsible technology development. • Follows an iterative development process for continuous feedback and refinement.
eFRIEND / Jones et al. [120]	• Proposes a comprehensive ethical framework for developing IEs. • Integrates ethical considerations practically into the development process. • Focuses on providing context-sensitive services in living and working environments. • Aims to create ethically sound systems that enhance user experience and protect rights and privacy.
SEArch / Augusto et al. [121]	• Developed in a bottom-up fashion driven by practical needs. • Provides a versatile infrastructure adaptable to various scenarios and use cases, applicable to a wide range of smart environment projects. • Integrates multiple methods and tools for development and implementation. • Offers a robust and adaptable architecture for smart environments, tailored to specific project needs.
CASA / Augusto et al. [122]	• Focuses on real-time data collection and modelling for context-aware systems. • Provides methodologies for identifying valuable contexts and activating services. • Emphasizes integration of diverse data sources for enhanced context-awareness.
CATS Design / Rodrigues et al. [124]	• Focuses on testing context-aware software systems. • Proposes a specific approach for designing functional test cases. • Includes empirical evaluation through proof of concept and observational study.
COATI / Augusto et al. [125]	• Emphasizes the importance of 'context' in developing and validating IEs. • Proposes modern testing techniques tailored for context-aware systems. • Aims to provide practical methods for testing in real-world application. • Seeks to enhance system reliability through rigorous testing and validation.

The ***Post-Construction stage***, which includes system management and maintenance activities (see Fig. 2), is essential, although it is often underestimated in the literature on IoT system development methodologies. Authors like Guerrero-Ulloa et al. [21] are an exception, as they present maintenance as an integral part of their methodology. The Post-Construction phase is critical, as it ensures that the software remains relevant and effective over time, adapting

Table 5 Compliance of the main methodological approaches analysed with the system/software lifecycle stages and activities outlined in ISO/IEC/IEEE standards

	Conception		Modelling				Construction				Post-Construction		
Proposal name / Authors [Ref.]	Planning	Requirements elicitation	Analysis	Design	Model verification	Simulation	Coding or implementation	Integration	Testing	Deployment	Management or operation	Maintenance	System dismantling or decommissioning
Lekidis et al. [89]	×	~	✓	✓	✓	~	✓	✓	~	~	×	×	×
MDE4IoT / Ciccozzi & Spalazzese [90]	×	×	✓	✓	~	×	✓	×	×	~	×	×	×
Harbouche et al. [91]	×	~	✓	✓	✓	✓	✓	✓	✓	~	×	×	×
ROOD / Corredor et al. [92]	×	~	✓	✓	✓	×	✓	✓	×	✓	×	×	×
Sosa-Reyna et al. [56, 93, 94]	×	✓	✓	✓	✓	×	✓	~	×	✓	×	×	×
Brambilla et al. [95]	×	×	×	✓	×	×	✓	×	×	✓	×	×	×
IDeA / Costa et al. [96]	×	~	✓	✓	✓	×	×	×	×	~	×	×	×
SEM / Cicirelli et al. [97]	×	~	✓	✓	✓	×	✓	×	×	~	×	×	×
AMG / Sulistyo [22]	×	×	~	✓	×	×	✓	×	×	×	×	×	×
SERVUS / Usländer & Batz [49]	~	✓	✓	✓	×	×	✓	×	✓	×	×	×	×
ELDA-Meth / Fortino & Russo [99]	×	×	×	✓	~	✓	✓	✓	✓	~	~	×	×
Pico-Valencia et al. [101]	~	~	~	✓	~	×	~	✓	~	~	×	×	×
Fortino et al. [102]	×	×	~	✓	×	~	✓	×	×	×	×	×	×

Table 5 (continued)

	Conception		Modelling				Construction				Post-Construction		
Proposal name / Authors [Ref.]	Planning	Requirements elicitation	Analysis	Design	Model verification	Simulation	Coding or implementation	Integration	Testing	Deployment	Management or operation	Maintenance	System dismantling or decommissioning
TDD-M4IoTS / Guerrero-Ulloa et al. [21]	✓	✓	✓	✓	×	×	✓	✓	✓	✓	~	✓	×
INTER-Meth / Fortino et al. [109]	~	~	✓	✓	~	✓	✓	✓	✓	✓	~	✓	×
RASPSS / Wang et al. [110]	×	~	~	✓	×	×	~	×	×	~	~	~	×
Patel & Cassou [111]	×	~	~	~	×	~	✓	~	×	✓	✓	✓	×
Gogineni et al. [112]	~	✓	✓	✓	✓	×	~	✓	✓	×	~	×	×
MEDIS-TAM-RT / Benghazi et al. [113]	×	✓	✓	✓	✓	×	×	×	×	×	× × × × × ×	×	×
MIRIE / Augusto & Hornos [74]	x	✓	✓	✓	✓	✓	×	×	×	×	×	×	×
U-CIEDP / Augusto et al. [119]	~	✓	✓	✓	✓	×	✓	×	✓	✓	×	×	×
E-FRIEND / Jones et al. [120]	✓	✓	✓	✓	×	×	×	×	×	×	×	×	×
SEArch / Augusto et al. [121]	~	✓	✓	✓	~	×	✓	✓	×	×	×	×	×
CASA / Augusto et al. [122]	~	✓	✓	✓	~	×	✓	✓	✓	×	×	×	×

Table 5 (continued)

	Conception		Modelling			Construction			Post-Construction		
Proposal name / Authors [Ref.]	Planning	Requirements elicitation	Analysis	Design	Model verification	Simulation	Coding or implementation	Integration	Testing	Deployment	Management or operation	Maintenance	System dismantling or decommissioning
CATS Design / Rodrigues et al. [124]	~	✓	✓	✓	~	×	✓	×	✓	×	×	×	×
COATI / Augusto et al. [125]	~	✓	✓	✓	~	×	✓	✓	✓	×	×	×	×

to changing needs and the constantly evolving technological environment.

In summary, a methodology with a balanced approach that not only encompasses the design and implementation of the system to be developed but also includes planning, requirements elicitation, user-centred development, and a robust Post-Construction phase is vital for the successful development of IoT systems. This comprehensive approach ensures that systems are not only delivered on time and meet initial expectations but can also evolve and remain effective in the future, having a long lifespan.

Another aspect that most of the methodologies reviewed do not address or do not explicitly mention is the activities related to the ***collection***, ***specification and fulfilment of NFRs*** [134, 135]. Remember that for the developed system to be of quality, it must not only satisfy the functional requirements but also the non-functional ones, so this is another important aspect to consider in an IoT system development methodology.

On the other hand, the uncertain nature of requirements in the early stages of IoT system development demands a flexible and adaptive approach from developers. The creation of ***early prototypes*** becomes an essential tool for clarifying expectations and refining requirements, taking advantage of methodologies that allow rapid iteration and development. In this sense, approaches such as Rapid Prototyping (RP) [81, 136] and agile methodologies, such as Scrum and XP, promote short feedback cycles and close collaboration with end-users. These agile practices, rooted in the four values and twelve principles of the Agile Manifesto [41–43], emphasize the importance of adaptability, continuous delivery, and the ability to respond to changes in customer requirements.

In this context, ***model-driven approaches***, such as MDE, MDD, and MDA, complement the previous approaches and stand out for their ability to accelerate software generation and thus have early versions of solutions. Therefore, it is imperative that researchers in the field of software engineering focus on integrating the strengths of model-driven approaches and agile methodologies to formulate a ***holistic methodology*** for IoT system development. Such a methodology should incorporate rapid prototyping and the flexibility to adapt to emerging requirements, not only functional but also non-functional, ensuring that the developed IoT systems are not only functional but are aligned with the end-user needs at all times.

Our article presents a comprehensive perspective of development methodologies for IoT systems. It summarises the advancements in the research area and describes the main existing challenges, providing an explanation of each challenge from different points of view. The article also proposes open research avenues to guide research efforts

towards the creation of development methodologies for IoT systems. Other articles reporting literature reviews on the subject are more specific and focused, for instance, on an explicit type of development [137, 138], agile methodologies [24], requirements engineering [139], interoperability [37], and a specific case study [140]. We do not claim our work is better than these, but we provide an extensive perspective that will give a broader picture of the subject.

Finally, after examining the findings from previous state-of-the-art review works on IoT system development methodologies [37, 138, 140] and conducting an in-depth literature review, we can confidently state that, currently, there is not a universally adopted methodology for IoT system development. As a result, there appears to be a pressing need for continued research in this area, aiming to provide developers of these systems with a methodology that simplifies their tasks and ensures the quality of the developed system.

3 Challenges in IoT-based system development

In the various subsections of this section, we explore several key challenges encountered by developers of IoT-based systems. These are challenges that the research community is expected to address in the forthcoming years. Figure 3 illustrates these challenges in the development of IoT-based systems.

3.1 Heterogeneity and interoperability

In IoT systems, the heterogeneity of components and systems poses a significant challenge regarding interoperability, demanding standards and protocols that allow for smooth communication between disparate devices and platforms [133, 141].

The concept of systems of systems [118] is often used to characterise IoT-based systems. This characterisation illustrates IoT-based systems' complexity, making their design, implementation, and maintenance more challenging. As in other types of systems, their different components must work together to ensure the appropriate delivery of the expected services. However, this integration is more challenging for IoT-based systems because they comprise more complex subsystems using heterogeneous technologies that must work together [117].

In this section, we are going to analyse the heterogeneity and interoperability challenges from two perspectives: the hardware and the software integration issues. The first one considers the heterogeneity of the devices used to build the equipment for IoT-based systems. The second issue relates

Fig. 3 Main challenges in IoT-based system development

to the different technologies used to develop the software for managing the hardware in IoT-based systems.

3.1.1 Hardware integration issues

IoT devices can be sensors, actuators, gadgets, appliances, or machines that are programmed to deliver services and transmit data [142]. These devices originate from various manufacturers, each with their own set of priorities for constructing their products. Consequently, IoT devices exhibit heterogeneity, stemming from the unique designs of manufacturers aimed at fulfilling specific use cases they have predetermined. For instance, this heterogeneity is evidenced in the existing all-in-one or separate devices that can measure similar indicators. Another example illustrating this heterogeneity is the devices' existing different processing capabilities, which allow them to provide from raw data to high-level contexts.

The hardware heterogeneity also considers the need to automate the configuration of the expected billions of IoT devices connected to the Internet, which cannot be done manually [143]. The success of IoT-based systems strongly depends on the orchestrated performance of the IoT devices capturing and delivering data that will be processed to provide the expected services. Hence, IoT devices must be configured and reconfigured quickly to deliver low-latency services when it is required (e.g., IoT systems for healthcare or self-driving cars).

The appropriate selection of reliable IoT devices is another critical process for building IoT-based systems. The lower cost of the materials used to build IoT devices and the open knowledge approach used in the area have permitted the democratisation of IoT technology, bringing more suppliers to the market. As explained above, this variety of manufacturers provides IoT devices with different characteristics that impact their performance and reliability. Choosing the most suitable IoT devices in the market or the appropriate sensing-as-a-services model to build IoT-based systems is critical to delivering highly precise services when required.

3.1.2 Software integration issues

Data analytics is also crucial to delivering the IoT systems' expected services, and it requires adequate preparation of the data sources collected by the different nodes that make up the systems. The variety of hardware and its higher processing capabilities permit the generation of data at higher rates and in different formats (structured, semi-structured, and unstructured), which makes the data heterogeneity issue another important challenge to tackle in the development of IoT systems. The delivery of services in real-time implies a data preparation stage that must also be done in real-time, which is difficult considering the volume of data to process, its variety, and the velocity at which the data is generated. Pre-processing techniques for systems that can be categorised as big data are still challenging to implement when real-time data preparation is required. Technologies implementing the concept of a data lake are envisioned to contribute to addressing this challenge.

The processing requirements of several AI techniques are another issue to consider in IoT systems development. AI already benefits society through predictive and prescriptive analytics [144]. Implementing these techniques can be demanding regarding the processing capabilities of the devices in which they operate. This gap between the processing demand of AI algorithms and the devices' processing capabilities is closing, but it is still a strong restriction to consider in developing IoT systems that implement architectures distributing the processing tasks among the different nodes, e.g., using fog and edge computing [145].

AI also brings the issue of integrating algorithms obtained from techniques that follow black-box approaches for training (e.g., neural networks). Although the use and integration of these algorithms through functions is possible and relatively simple, the issue highlighted in this research work refers to integrating the different logic behind the algorithms, which can be difficult (or even impossible) to explain. Knowing why the algorithms predict or prescribe the results is vital in the logic integration that is sometimes required to develop IoT systems complying with explainability requirements brought by ethical and legal contexts [146]. Integrating diverse algorithms is critical to achieving the system of systems concept illustrating an IoT system. If one of these algorithms lacks explainability, the IoT system will also lack this property.

The challenges of integrating diverse algorithms appear due to using systems from different providers that are required to develop the expected IoT system. The algorithms developed and used by these providers are usually an important component of their competitive advantage, not allowing them to open and share their logic with their users to protect their businesses. So, the question of how to know if the services of other providers are explainable or not arises. Organisations accrediting that the logic behind these services complies with the explainability requirements are a feasible option. However, different standards to comply with exist (e.g., country-specific regulations). This fact makes it more challenging to develop systems that may be collecting a wide variety of data and running diverse processing tasks.

A middleware is critical software to achieve the required integration of the different IoT system components. They are needed to connect applications that were not designed

to connect among themselves. Given that many devices and software from different providers exist and work interconnected in an IoT reality, the probability of them not being designed to work interconnectedly is very high. In this context, the middleware is considered the glue putting the different applications together to provide the IoT systems' expected services. However, although there are highly reliable middlewares, most have been designed for complex contexts (e.g., industrial middlewares), making them expensive to acquire, install, and maintain. This type of middleware may be too excessive for simpler contexts (e.g., smart homes) in which less complex scenarios occur, such as the one proposed by Palade et al. [147]. There is a gap in designing and bringing to the market middlewares fitting simpler integration needs and at a reasonable cost for what can be considered as lighter scenarios.

IoT systems use architectures associated with the concepts of fog and edge computing in which the processing tasks are less centralised and distributed among the different nodes that make up the network (fog computing) and even pushed towards the edge device in which the data sources are obtained (edge computing) [145]. These computing frameworks use the current nodes' higher processing capabilities better. However, they still require Internet availability to allow communication among the nodes —which are expected to be many— and create the contexts needed to deliver the expected features and services. Higher Internet availability and speed are required to integrate IoT systems. 5G networks are expected to comply with these requirements, but their cost and availability vary among the countries where it is deployed. This issue is essential to address for achieving a more democratised dissemination of IoT systems.

3.2 Scalability, adaptability and integration with emerging technologies

In the realm of IoT, *scalability* emerges as a pivotal aspect, not merely in terms of handling an increasing influx of devices but also in adapting to evolving needs. The exponential growth in the number of connected devices and the sheer volume of data generated present both opportunities and challenges. As the IoT ecosystem expands, it must seamlessly accommodate the addition of new devices, manage the vast data streams they produce, and ensure consistent performance across the board [135].

The inevitable wear and tear of hardware components, coupled with the rapid pace of technological advancements, necessitates the periodic replacement of outdated or malfunctioning parts. This cycle of obsolescence and renewal is accelerated by the emergence of more powerful and capable technologies, which promise enhanced functionalities and improved efficiencies. However, the integration of these new technologies into existing IoT systems is not without its complexities.

The *potential evolution of technology*, as well as the software it supports, underscores a critical gap in current methodologies and tools designed for IoT development. While various methodological proposals, frameworks, platforms, and tools exist to support the development of IoT systems [24, 37], they often fall short in addressing the long-term evolution of software and the system's consequent adaptation over time. Therefore, one of the principal challenges in the development of IoT systems lies in achieving a high degree of *reusability*, *adaptability*, and, most importantly, *extensibility* of an existing IoT system to adapt to new emerging needs.

To confront the inherent heterogeneity and evolution within IoT environments, developers must focus on creating systems that are not only interoperable and scalable but also capable of evolving alongside emerging technologies. This necessitates development methodologies with a long-term vision in IoT system design, which anticipate potential future developments and incorporate flexibility at their core, while also paying attention to the necessary *maintenance* of the system once developed. By doing so, IoT systems can remain relevant and functional, providing sustained value in a rapidly changing technological landscape.

3.3 Security and privacy

Technology development is a cornerstone in human history due to the impact, sometimes disruptive, on society. This impact can be in the form of benefits and can also negatively affect people's lives if misused with a lack of ethics. Security and privacy are global issues for developing solutions based on TICs. In the case of IoT-based systems, these issues are more challenging due to, again, the nature of these systems, which are expected to be deployed and seamlessly integrated with humans in diverse scenarios of their lives. Hence, humans may not even be aware of their interaction with IoT-based systems.

User-centric software development methodologies are needed to comply with users' preferences regarding their IoT-based systems' security and privacy settings, besides the legal context ruling the scenarios and places in which these systems are expected to operate. Privacy and security can also be addressed as part of the preferences management capabilities of IoT systems, which are critical for their adequate development [129]. Security and privacy preferences are dynamic and strongly influence the context-aware reasoning components of IoT-based systems, as well as the features they can deliver.

Some AI techniques require data-intensive processing to deliver highly accurate predictions. Involving private data as input in training these algorithms may increase the probability of obtaining better results. However, users could not be willing to share their private data. This issue has brought the need to investigate training methodologies that can include private data infringing on users' privacy. An example of addressing this issue is Federated Learning [148], which has already been applied to several use cases like the ones presented by Ogbuabor et al. [149] and Sánchez et al. [150]. Advanced AI techniques are also being used to protect data, such as deep learning [151].

The more complex and distributed processing architectures used to implement IoT-based systems (e.g., fog and edge computing) consider distributed and interconnected nodes with diverse processing tasks. That is, nodes are vulnerable to security threats and must be correctly configured and integrated to protect the IoT-based system. This issue becomes more challenging because these nodes can be technologically heterogeneous and may implement black-box processing tasks to protect their logical competitive advantage, which makes their security implementation more complex.

Although there are studies that have addressed these issues, such as the application of the Privacy-by-Design (PbD) framework to IoT applications [152], the monitoring of security attacks in real-time for IoT systems through DevSecOps [153], the implementation of access control models [154] or the application of blockchain for secure data handling [155], there are still pending security and privacy challenges to be resolved for IoT system developers [156–158], which will need to be addressed by the research community in this area.

3.4 User-centric approaches

As in any other system, requirements elicitation is critical to increasing the success probability of an IoT system development process. However, the diverse contexts in which IoT systems are expected to be implemented make the requirements definition process more challenging. IoT systems are expected to support people in their daily activities that include diverse situations, such as those that usually happen at home (smart homes), transportation (smart vehicles), work (smart offices, smart classrooms), healthcare self-management (mHealth applications and devices), and so on [117]. This variety of situations in which IoT systems will be used requires user-centric approaches to guarantee an adequate design and implementation targeting users' needs in different situations [119].

Users' preferences and needs are dynamic and not only depend on the situations an IoT system supports but also on less obvious factors defined by subjective perceptions of users (e.g., their mood), which also change over time. The users' dynamic and context-specific requirements demand adaptable IoT systems balancing their proactive and reactive features based on customisation that gives control to the users over the system. Some users may prefer an IoT system that is more intrusive in their job tasks but less intrusive at home. Other ones would like to avoid proactive (and even reactive) features altering their established and regular routines. Preferences management is still an open research theme to address that should strongly influence IoT development processes [129]. Identifying the next personalised Point-of-Interest (POI) [159] is a typical problem where the dynamic nature of users' needs and evolving preferences require a strong user-centric approach to deliver appropriate services.

Cognitive overload is another essential issue to avoid when developing IoT systems. The high availability of context-related data due to environments enriched with IoT devices would lead to delivering more information to users than they require. Cognitive overload refers to overloading users with data they are not interested in, which has a negative impact on the system's usability. This issue must be avoided, considering users' preferences to aid IoT systems in differentiating what information is relevant and not. The use of information supply chain approaches is key to balancing information supply and demand [160].

Environments enriched with IoT devices present several potential benefits to support peoples' daily activities. However, the idea of interacting in a Cyber-Physical Space brings challenges regarding human-computer interaction (HCI). IoT systems should be designed to support users' tasks, guaranteeing a natural integration with users. HCI grand challenges [161] must be tackled to permit IoT systems to deliver their expected benefits to society. Meaningful research to provide users with IoT systems that are easy to install in their daily environments, customise, and interact with is key to expanding the reach of these systems.

The user-centric challenges outlined previously are considered from the perspective of the end-users of IoT systems. However, user-centric challenges can also be studied from the IoT systems developers' point of view, who need support to address the different challenges explained in this section. Developers need automated tools that allow them to track IoT systems' designs, as well as their services implementation, testing, validation, and maintenance, to comply with the complex requirements of IoT systems [132]. Automating these tools is crucial to accelerate the IoT systems development process and comply with their dynamic and context-specific requirements adequately. AI can aid in automating the requirements mapping throughout the process development stages (conception, modelling, construction,

and post-construction). For instance, natural language processing techniques can support the automatic generation of software development products (e.g., UML diagrams) based on the requirements elicitation narrative.

3.5 Reliability or dependability

Reliability or dependability in IoT-based systems is an increasing concern due to their expanding application in critical areas such as health, automotive, and urban infrastructure. Thus, for example, human and animal digital health platforms [162], which are related to the Internet of Medical Things, are clear examples of critical applications with high dependability and safety requirements. The reliability or dependability of IoT systems refers to their ability to function without failure under expected conditions for a specified period. In this context, rigorous design, verification, and validation become essential to ensure that IoT systems meet the dependability and safety requirements demanded by these critical applications. To address these challenges, specific methodologies and tools have been developed. For instance, formal modelling and system specification allow for the verification of critical reliability and safety properties before implementation. Thus, the use of certain tools in methodologies specific to these types of systems, such as UPPAAL [163], Spin [74], or ProB [164], facilitates the formal verification of IoT systems, enabling designers to identify and correct potential errors in the early stages of development.

Furthermore, the current literature suggests a range of edge computing simulators that support the analysis of quality characteristics relevant to IoT system design. Ashouri et al. [165] highlight that while many simulators focus on qualities such as temporal behaviour and resource utilization, there is a need for further research to adequately support IoT architects. This indicates a gap in support for a broader range of qualities, underscoring the importance of developing more comprehensive tools. In the context of IoT-based electronic health systems, Prabha and Chatterjee [166] propose a hybrid consensus mechanism incorporating algorithms for creation, validation, fork handling, Merkle tree construction, and reward/punishment modules, demonstrating a robust approach to ensuring system security and reliability.

Moreover, to quantify the dependability of IoT systems, several metrics are utilized [167], including Mean Time To Failure (MTTF), Mean Time Between Failures (MTBF), failure rate, availability, and reliability. These metrics offer a quantitative basis to evaluate the robustness and readiness of IoT systems for deployment in critical applications. For example, MTTF and MTBF provide insights into the expected operational lifespan and maintenance needs of the systems, respectively, while reliability gives the probability of a system performing its required functions under specified conditions over a certain period. In the healthcare context, Tang and Xie [168] propose an availability model for an IoT system, offering performance metrics such as probabilities of full service, degraded service, and system unavailability. This not only enhances the understanding of the operational reliability of IoT systems but also facilitates the identification of areas for reliability improvement.

Xing [169] provides a comprehensive overview of the current state and future outlook of reliability within the IoT domain. Addressing reliability across the layered IoT architecture, the paper systematically synthesizes and reviews existing literature on reliability models and solutions across four layers: perception, communication, support, and application. It highlights that research on IoT reliability is still nascent, with much room for exploration in terms of under-explored behaviours and the evolving complexity and dynamics of IoT systems.

Consequently, advancements in specific methodologies and tools, together with the development of quantitative reliability metrics, are paving the way for enhancing the dependability of IoT systems. Techniques for software reliability measurement are examples of advancements in this area. A clear example of such techniques is presented by Yuen [170], who proposes the Fuzzy Cognitive Network Process as an alternative for software reliability and quality measurement. However, continuous research is essential to close existing gaps and address emerging challenges in this dynamic field.

4 Open research avenues

From our perspective, the research directions we outline below represent some of the open issues that remain unresolved concerning development methodologies for IoT systems. Our research community will need to dedicate efforts in the coming years to provide solutions to these issues, among others that will arise in this constantly evolving technological context.

Figure 4 summarizes the open research avenues to be presented in this section, each of which will be explained in its corresponding subsection. Strong relationships exist between the challenges outlined previously and these open research directions. Detailed explanations of each research direction will include how they address one or more of the issues described as challenges in the preceding section of this article.

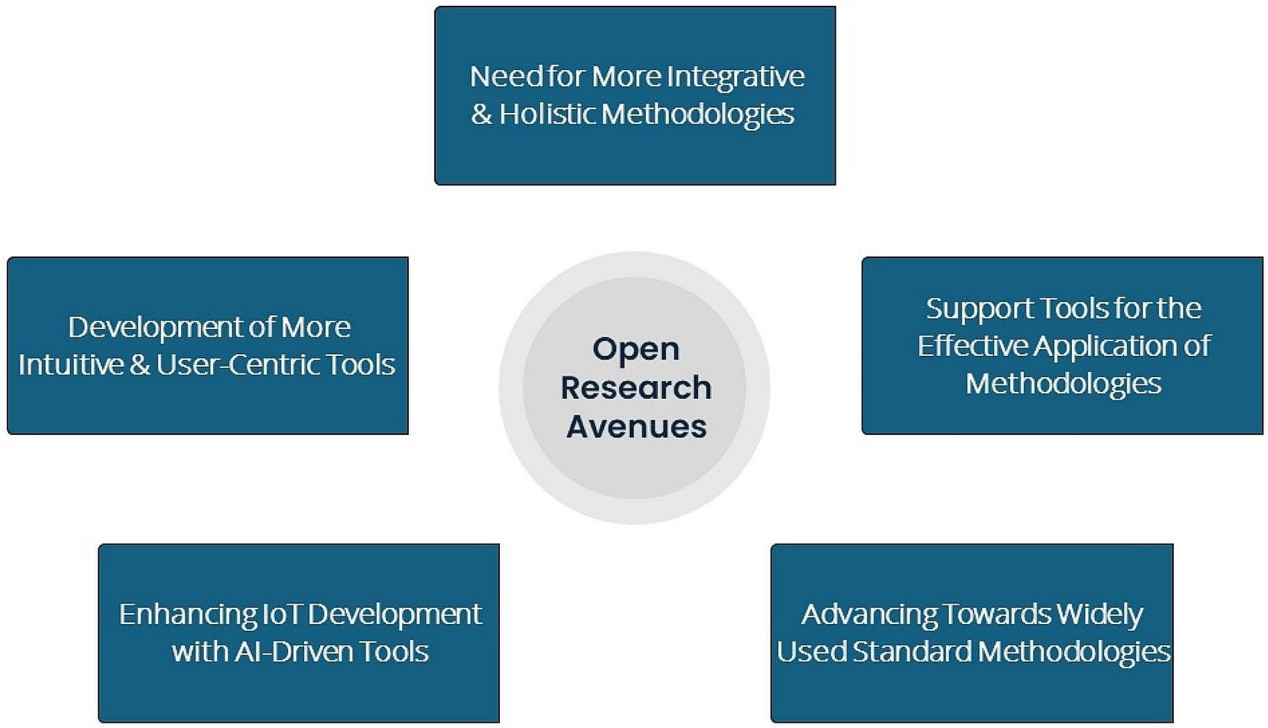

Fig. 4 Open research avenues in IoT-based system development

4.1 Need for more integrative and holistic methodologies

The literature review conducted on the predominant methodologies in IoT system development reveals a diversity of approaches and variability in the phases and activities considered by each. Some methodologies specialise in specific stages of the development lifecycle, such as design and construction, while others attempt to cover a broader spectrum. However, the absence of a methodology that comprehensively and exhaustively encompasses the entire lifecycle of IoT system development according to international standards is evident.

In this context, the need to forge more integrative and holistic methodologies becomes apparent. These methodologies must be designed to be inherently flexible and capable of addressing the specific needs and challenges associated with IoT system development. They must provide a robust framework that integrates the best practices from diverse software development paradigms, forming a versatile methodological structure adaptable to the dynamic and multifaceted nature of IoT systems.

An ideal methodology would integrate concepts from established software development paradigms, such as MDE and TDD. These approaches focus on the creation and use of models for generating code fragments and the tests that this code must pass to meet specified requirements, thus ensuring the quality and functionality of the system from the early stages of development.

Moreover, this methodology could also incorporate principles of agile development, such as prioritizing functional software over comprehensive documentation, the ability to respond to unforeseen changes, and continuous collaboration between clients/users and developers. This agile approach would facilitate adaptation to emerging needs and foster effective communication, crucial elements in the development of IoT systems.

A methodology with these characteristics should address not only the business logic and user-system interaction but also the configuration and deployment of the necessary hardware, such as sensors, actuators, and processors, as well as the configuration and programming of single-board computers, such as Arduino and Raspberry Pi. The methodology must be robust enough to handle the complexity inherent in these components and flexible enough to adapt to their evolution.

In conclusion, the development of IoT systems requires a methodology as dynamic and multifaceted as the systems it aims to create. Only through an integrative and holistic approach can we overcome current and future challenges, ensuring that the developed IoT systems are not only

functional and efficient but also sustainable and capable of evolving alongside emerging technologies and user needs.

4.2 Support tools for the effective application of methodologies

An effective methodology for the development of IoT systems must not only be robust and flexible but also accessible and applicable through appropriate support tools. These tools are essential to facilitate the practical application of the methodology and must be capable of providing comprehensive assistance throughout all phases of development, from conception to post-construction.

Customization is a key aspect of IoT systems development, and an ideal support tool should allow developers to adjust the various aspects of the system to meet the specific needs of the project. This is achieved through an intuitive interface that guides the developer from the initial system specification, which could take the form of extended use case descriptions, user stories, or any other alternative specification.

Using structured templates, developers could input these specifications into the tool, which would then automatically generate a conceptual model of the system to be developed and the associated tests that the generated software must pass. This approach aligns with the TDD philosophy, ensuring that each piece of generated code meets the established requirements from the outset.

The automatically generated conceptual model could serve as a starting point for developers to refine and produce a more detailed model, corresponding to the solution domain. The support tool could use this refined model for the automatic generation of code, which would then be validated by the previously established tests, ensuring the quality and coherence of the system under development.

Thus, the mentioned tool would not only provide a solid initial software structure but also offer the flexibility for developers to make adjustments and improvements as necessary, enriching and perfecting the software until the final product is achieved.

Furthermore, a comprehensive IoT system development tool should include modules dedicated to hardware design. These modules would allow developers to design and configure IoT devices in a graphical and intuitive manner, connecting electronic components and configuring such devices efficiently.

In summary, a good support tool for an IoT system development methodology is essential and must be a fundamental pillar to facilitate its application, providing developers with a robust and versatile platform. This platform should not only facilitate the design and configuration of IoT devices but also support the development of front-end and back-end applications, ensuring that the development process is as innovative and dynamic as the systems intended to be created with it.

4.3 Advancing towards widely used standard methodologies

In the current landscape of IoT system development, there is a concerning trend: most development methodologies are exclusively used by their creators. This situation hinders knowledge transfer and the standardisation of practices in a field as dynamic and expansive as IoT. Therefore, the emergence of standard methodologies that can be widely adopted is imperative, thereby facilitating common methods, tools, and practices among developers.

These methodologies should be designed with an intuitive and accessible approach, equipped with support tools that guide users' step by step in their practical application. Simplicity and clarity must be fundamental pillars so that even novice developers and software engineering students can confidently apply them during their academic projects and professional internships [171].

It is essential that these methodologies provide structured guidance to software engineers, supporting them throughout the development process. They should include options for automatic or semi-automatic generation of system components, which not only ensures the quality and correctness of the code from the initial phases but also optimises development times, a critical factor in a rapidly evolving market.

For widespread adoption, it is crucial that the methodology is properly documented and supported by training and support materials. This is particularly relevant for developers interested in applying it, especially for those who are novices, as it will help to reduce the learning curve of the methodology and its support tools, allowing them to understand and apply the methodology more effectively in less time. In addition to the creation of detailed training materials, the inclusion of clear guidelines integrated into the support tools is fundamental to democratizing the development of IoT systems and maximizing the efficiency and potential of these methodologies.

In summary, the IoT community faces an urgent need to consolidate standard methodologies that are widely accepted and utilized. Only through a collaborative effort to make these methodologies better known and applied can we move towards a future where the development of IoT systems is a more efficient, coherent, and universally understood process.

4.4 Enhancing IoT development with AI-driven tools

The integration of AI techniques in the development of IoT systems is a burgeoning field that promises to revolutionize the way these systems are designed and implemented. The support tool for a methodology can be significantly enhanced by infusing it with greater intelligence and new capabilities, thereby improving both the quantity and quality of the generated code and the efficiency of the IoT system development process.

Advanced AI techniques can be employed to analyse textual information pertaining to the specification of an IoT system. Developers typically begin with system descriptions articulated through extended use cases (EUCs), user stories, or other forms of system specification. An intelligent analysis of these system descriptions, introduced into the tool via EUCs, for instance, can automatically identify key elements, such as potential classes, attributes, relationships, etc., which are crucial for the automatic generation of a conceptual class diagram. This initial diagram can then be refined to produce a more comprehensive class diagram, corresponding to the domain of the solution design.

Recent solutions for generating conceptual class diagrams from requirements described in natural language include the works of Nasiri et al. [172], and Omer and Eltyeb [173], among others. While these solutions can generate class diagrams with attributes, methods, and relationships, they do not address the generation of software code. Therefore, they represent a step in the right direction, but more work is needed in this regard to provide further assistance to developers.

Incorporating AI techniques allows for the extraction of information from the system specification that can also help to include design patterns in the design class diagram. Design patterns [174] are essential for promoting good software engineering practices and acquiring quality attributes in the developed systems.

The incorporation of these new AI-driven features will endow the tool with greater intelligence and capabilities that can significantly contribute to improving the application of the corresponding methodology. Moreover, it will enhance the efficiency of the IoT system development process, facilitating design and implementation, and reducing the time required to do so. By automating at least part of the developers' workload, the tool can mitigate potential human errors, ease their workload, and result in higher quality IoT systems.

Furthermore, with the advent of Artificial Intelligence of Things (AIoT), the processing and analytical capabilities of interconnected devices have been significantly enhanced. However, the complexity and scale of AIoT-based systems present challenges for traditional machine learning and deep learning approaches. Neuro-symbolic approaches [175] can help overcome these challenges, ensuring the development of reliable and effective AIoT-based systems.

4.5 Development of more intuitive and user-centred tools

Users can be studied from the perspectives of IoT-based systems' end-users and the IoT-based systems developers. This subsection includes both types of users. In both cases, user-centric approaches must be used, considering the volatile requirements to address and all the features and software development products associated with them.

IoT-based systems' end-users have different IT backgrounds that define their preferences regarding their interaction with IoT-based systems. Other less objective factors also influence how they want to integrate themselves with IoT-based systems. All these affect their system acceptance, adoption, and appropriation and constrain the success of the development process.

IoT-based systems may use wearable and ambient devices to collect data about their end-users and their environment. The availability of data makes it possible to provide users with several notifications regarding situations they are not interested. Achieving an appropriate balance between information supply and demand is important to make IoT-based systems more intuitive. The automation of the system information supply chain and the definition of the use cases the system is expected to support can aid in avoiding information overload and improve usability.

From the developers' perception, the Conception stage is key to defining the system requirements expected to address users' needs and preferences regarding the features and their interaction with the system. Automated tools for requirements elicitation must be investigated because they are important supporting tools for identifying and mapping end-users' needs. These requirements define the Modelling, Construction and Post-Construction stages, which also need automated tools to map their tasks and software development products with the requirements defined in the Conception stage.

Supporting tools for developers can use more advanced automation techniques, including AI, to generate software development products faster. These techniques applied in the software development areas are being investigated, but more research effort on it must be made. Automating these key tasks will ease the software development process and increase the probability of complying with the functional and (especially) non-functional requirements to guarantee the users' more natural integration with IoT-based systems.

Developing usability assessment techniques and tools to assess IoT-based systems is another open challenge for developers. IoT-based systems require different validation tools that can efficiently obtain end-users' perceptions about the more complex scenarios in which these systems operate. Usability is key to achieving IoT-based systems that are easy to integrate with users, but its assessment is usually challenging because it needs to gather end-users' subjective opinions about the system. This issue is more relevant when it comes to assess usability in situations that involve diverse aspects of users' daily activities and considering that the outcomes of IoT-based systems usability assessment shall be automatically associated with the different products of the software development process.

5 Conclusions

IoT-based systems are already benefiting society across a broad range of sectors, including home automation, industry, transportation, healthcare, and agriculture, among many others. Their socioeconomic impact has been growing at an unstoppable pace for years. They are a hot topic to study from different perspectives because of their imminent and growing influence on people's lives and the envisioned potential benefits they are expected to deliver.

However, the development of IoT systems is complex, as it requires consideration of a multitude of aspects: heterogeneity, interoperability, and deployment of various physical devices (sensors, actuators, smartwatches, boards, servers, etc.) and software components that make up the system, communication protocols to be used, where to process data (at the edge, fog, or cloud), how to configure and program the different hardware devices, which programming languages and software technologies to use, among a long list of issues. Therefore, there is a need to address the challenges that all these issues pose to developers when creating such a system.

The literature review conducted highlights the importance of searching for a robust and more established methodology for IoT-based systems development. This methodology must address the unique and tangled characteristics of IoT-based systems and become a simple tool for developers to ensure the system's quality.

Despite the considerable efforts and advancements that are being made, existing challenges are still many and significant in this research area. The article evidences that more research is needed to address the complex nature of IoT-based systems that, along with the diverse scenarios in which they are expected to operate, stress the influence of IoT-based system development challenges. Hence, the described challenges are all important, although they may overlap due to some transversal issues developers should consider throughout the development process they choose to follow.

The article also presents the open research avenues in the development of IoT-based systems. They are proposed as pathways to guide the research efforts in the area. They aim to ensure that the key gaps in developing these systems are better understood and systematically closed to allow the proliferation of high-quality IoT-based systems. The development processes of these systems must evolve towards including tools facilitating developers' tasks, ensuring high-quality development standards, and reducing the time spent on repetitive tasks not adding value to the process. Automation is vital to achieving this vision, but it must be based on the previous identification of the critical tasks to automate. This automation will allow developers to focus on the more crucial tasks that generate more value for IoT-based systems' end-users.

Author contributions M.J.H. conceived the article and outlined its structure. Each author wrote different parts of the manuscript. Both authors have reviewed it.

Funding This research was supported by the grant PID2019-109644RB-I00, funded by MCIN/AEI/10.13039/501100011033, i.e., the Spanish Ministry of Science and Innovation (State Research Agency), and by the grant PID2022-139297OB-I00, funded by MICIU/AEI/10.13039/501100011033 and by ERDF/EU, i.e., the Ministry of Science, Innovation and Universities (State Research Agency) and the European Regional Development Fund (ERDF), a way of making Europe.

Data availability Not applicable.

Declarations

Ethical approval Not applicable.

Competing interests The authors declare no competing interests.

References

1. Andrade RMC, Aragão BR, Oliveira P, Maia MEF, Viana W, Nogueira TP (2021) Multifaceted infrastructure for self-adaptive IoT systems. Inform Softw Technol 132:106505. https://doi.org/10.1016/j.infsof.2020.106505
2. Ng ICL, Wakenshaw SYL (2017) The Internet-of-Things: review and research directions. Int J Res Mark 34(1):3–21. https://doi.org/10.1016/j.ijresmar.2016.11.003
3. Madakam S, Ramaswamy R, Tripathi S (2015) Internet of Things (IoT): a literature review. J Comput Commun 3(5):164–173. https://doi.org/10.4236/jcc.2015.35021
4. Stankovic JA (2014) Research directions for the Internet of Things. IEEE Internet Things J 1(1):3–9. https://doi.org/10.1109/jiot.2014.2312291
5. Singh D, Tripathi G, Jara AJ (2014) A survey of Internet-of-Things: Future vision, architecture, challenges and services. Proceedings

of the 2014 IEEE World Forum on Internet of Things (WF-IoT 2014), 287–292. https://doi.org/10.1109/wf-iot.2014.6803174
6. Almeida RB, Junes VRC, Machado R, Da Rosa DYL, Donato LM, Yamin A, Pernas AM (2019) A distributed event-driven architectural model based on situational awareness applied on internet of things. Inform Softw Technol 111:144–158. https://doi.org/10.1016/j.infsof.2019.04.001
7. Cisco (2016) Internet of Things at a glance. https://www.audentia-gestion.fr/cisco/pdf/at-a-glance-c45-731471.pdf (Accessed: 14 October 2023)
8. Mocrii D, Chen Y, Musilek P (2018) IoT-based smart homes: a review of system architecture, software, communications, privacy and security. Internet Things 1–2:81–98. https://doi.org/10.1016/j.iot.2018.08.009
9. Syed A, Sierra-Sosa D, Kumar A, Elmaghraby A (2021) IoT in smart cities: a survey of technologies, practices and challenges. Smart Cities 4(2):429–475. https://doi.org/10.3390/smartcities4020024
10. Baker S, Wang X, Atkinson I (2017) Internet of Things for smart healthcare: technologies, challenges, and opportunities. IEEE Access 5:26521–26544. https://doi.org/10.1109/access.2017.2775180
11. Guerrero-Ulloa G, Rodríguez-Domínguez C, Hornos MJ (2018) IoT-Based system to help care for dependent elderly. In Communications in Computer and Information Science (Vol. 895, pp. 41–55). https://doi.org/10.1007/978-3-030-05532-5_4
12. Guerrero-Ulloa G, Hornos MJ, Rodríguez-Domínguez C, Fernández-Coello MM (2020) IoT-Based Smart Medicine Dispenser to Control and Supervise Medication Intake. In Ambient Intelligence and Smart Environments (AISE) book series (Vol. 28, pp. 39–48). https://doi.org/10.3233/aise200021
13. Quy VK, Nguyen V, Van Anh D, Quý NM, Ban NT, Lanza S, Randazzo G, Muzirafuti A (2022) IoT-enabled smart agriculture: architecture, applications, and challenges. Appl Sci 12(7):3396. https://doi.org/10.3390/app12073396
14. Abir SMAA, Anwar A, Choi J, Kayes ASM (2021) IoT-Enabled smart energy grid: applications and challenges. IEEE Access 9:50961–50981. https://doi.org/10.1109/access.2021.3067331
15. Ahmad T, Zhang D (2021) Using the internet of things in smart energy systems and networks. Sustainable Cities Soc 68:102783. https://doi.org/10.1016/j.scs.2021.102783
16. Chui M, Collins M, Patel M (2021) IoT value set to accelerate through 2030: Where and how to capture it. McKinsey, Company. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/iot-value-set-to-accelerate-through-2030-where-and-how-to-capture-it (Accessed: 14 October 2023)
17. Royce W W. (1987) Managing the development of large software systems: concepts and techniques. Int Conf Softw Eng 328–338. https://doi.org/10.5555/41765.41801
18. Abrahamsson P (2017) Agile software development methods: review and analysis. arXiv org. https://doi.org/10.48550/arXiv.1709.08439
19. Anwer F, Aftab S, Waheed U, Muhammad S S. (2017) Agile software development models TDD, FDD, DSDM, and Crystal methods: a survey. Int J Multidisciplinary Sci Eng 8(2):1–10. https://www.ijmse.org/Volume8/Issue2/paper1.pdf (Accessed: 15 October 2023)
20. Keshta N, Morgan Y (2017) Comparison between traditional plan-based and agile software processes according to team size & project domain (a systematic literature review). 8th IEEE Annual Inform Technol Electron Mob Communication Conf (IEMCON 2017) 567–575. https://doi.org/10.1109/IEMCON.2017.8117128
21. Guerrero-Ulloa G, Hornos MJ, Rodríguez-Domínguez C (2020) TDDM4IoTS: A Test-Driven Development Methodology for Internet of Things (IoT)-based Systems. In Communications in Computer and Information Science (Vol. 1193, pp. 41–55). https://doi.org/10.1007/978-3-030-42517-3_4
22. Sulistyo S (2013) Software development methods in the Internet of Things. In Lecture Notes in Computer Science (Vol. 7804, pp. 50–59). https://doi.org/10.1007/978-3-642-36818-9_6
23. Rodríguez-Domínguez C, Santokhee A, Hornos MJ (2022) Intelligent environments with entangled quality properties. J Reliable Intell Environ 8(3):223–226. https://doi.org/10.1007/s40860-022-00182-5
24. Guerrero-Ulloa G, Rodríguez-Domínguez C, Hornos MJ (2023) Agile methodologies applied to the development of Internet of Things (IoT)-based systems: a review. Sensors 23(2):790. https://doi.org/10.3390/s23020790
25. Nakagawa H, Ogata S, Aoki Y, Kobayashi K (2020) A model transformation approach to constructing agent-oriented design models for CPS/IoT systems. Proceedings of the ACM Symposium on Applied Computing (SAC'20), 815–822. https://doi.org/10.1145/3341105.3374033
26. Marafie Z, Lin K, Wang D, Lyu H, Liu Y, Meng Y, Ma J (2021) AutoCoach: an intelligent driver behavior feedback agent with personality-based driver models. Electronics 10(11):1361. https://doi.org/10.3390/electronics10111361
27. Yang H, Xie X (2020) An actor-critic deep reinforcement learning approach for transmission scheduling in cognitive Internet of Things systems. IEEE Syst J 14(1):51–60. https://doi.org/10.1109/jsyst.2019.2891520
28. Kaminski NJ, Murphy MH, Marchetti N (2016) Agent-based modeling of an IoT network. 2016 International Symposium on Systems Engineering (ISSE 2016), 1–7. https://doi.org/10.1109/syseng.2016.7753151
29. Esteves Maria R, Rodrigues Junior LA, Guarino de Vasconcelos LE, Pinto M, Tsoucamoto AF, Silva PTA, Lastori HN, da Cunha A, Vieira Dias A (2015) L. A. Applying Scrum in an interdisciplinary project using Big Data, Internet of Things, and credit cards. Proceedings of the 12th International Conference on Information Technology: New Generations (ITNG 2015), pp. 67–72. https://doi.org/10.1109/itng.2015.17
30. Morais dos Santos MV, Barbosa da Silva PD, Otero L, Wisnieski AG, Gonçalves RTS, Maria GE, Vieira Dias R, Marques LA, da Cunha A (2016) Applying Scrum in an interdisciplinary project for fraud detection in credit card transactions. Adv Intell Syst Comput 448:461–471. https://doi.org/10.1007/978-3-319-32467-8_41
31. Khaleel H, Conzon D, Kasinathan P, Brizzi P, Pastrone C, Pramudianto F, Eisenhauer M, Cultrona P, Rusiná F, Lukáč G, Paralič M (2017) Heterogeneous applications, tools, and methodologies in the car manufacturing industry through an IoT approach. IEEE Syst J 11(3):1412–1423. https://doi.org/10.1109/jsyst.2015.2469681
32. da Costa CM, Baltus P (2021) Design methodology for industrial Internet-of-Things wireless systems. IEEE Sens J 21(4):5529–5542. https://doi.org/10.1109/jsen.2020.3031659
33. Industry IoT Consortium (2023) The Industrial Internet Reference Architecture. https://www.iiconsortium.org/IIRA/ (Accessed: 15 October 2023)
34. Boehm B (1988) A spiral model of software development and enhancement. IEEE Comput 21(5):61–72. https://doi.org/10.1109/2.59
35. Hijazi H, Khdour T, Alarabeyyat A (2012) A review of risk management in different software development methodologies. Int J Comput Appl 45(7):8–12
36. Lantz KE (1986) The Prototyping Methodology. Prentice-Hall: Saddle River, NJ, USA
37. Fortino G, Savaglio C, Spezzano G, Zhou M (2021) Internet of Things as system of systems: a review of methodologies, frameworks, platforms, and tools. IEEE Trans Syst Man Cybernetics 51(1):223–236. https://doi.org/10.1109/tsmc.2020.3042898

38. de Oliveira RP, Massoni T, de Araújo NM, Sarmento CF, dos Santos FS (2021) Ants doing legwork: Investigating motivators for software development career abandonment. In Proceedings of the XXXV Brazilian Symposium on Software Engineering (SBES'21) (pp. 353–362). https://doi.org/10.1145/3474624.3474644
39. Matsubara PGF, Steinmacher I, Gadelha B, Conte T (2021) Buying time in software development: how estimates become commitments? In Proceedings of the IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2021) (pp. 61–70). https://doi.org/10.1109/chase52884.2021.00015
40. Ravaglia CC, Méxas MP, Dias AC, Silveira Batista D, Da Silva Nunes HMC, K (2021) Management of software development projects in Brazil using agile methods. Indep J Manage Prod 12(5):1357–1374. https://doi.org/10.14807/ijmp.v12i5.1353
41. Beck K, Beedle M, van Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R et al (2001) Manifesto for Agile Software Development. http://agilemanifesto.org/ (Accessed: 28 October 2023)
42. Fowler M, Highsmith J (2001) The Agile Manifesto. Softw Dev 9(8):28–35
43. Hazzan O, Dubinsky Y (2014) The Agile Manifesto. In SpringerBriefs in Computer Science (pp. 9–14). https://doi.org/10.1007/978-3-319-10157-6_3
44. Thesing T, Feldmann C, Burchardt M (2021) Agile versus Waterfall project management: decision model for selecting the appropriate approach to a project. Procedia Comput Sci 181:746–756. https://doi.org/10.1016/j.procs.2021.01.227
45. Younus AM, Younis H (2021) Conceptual framework of agile project management, affecting project performance, key: requirements and challenges. Int J Innovative Res Eng Manage 8(4):10–14. https://doi.org/10.21276/ijirem.2021.8.4.3
46. Nugra H, Abad A, Fuertes W, Galárraga F, Aules H, Villacís C, Toulkeridis T (2016) A low-cost IoT application for the urban traffic of vehicles, based on wireless sensors using GSM technology. In Proceedings of the IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT 2016) (pp. 161–169). https://doi.org/10.1109/ds-rt.2016.24
47. Peterson B, Vogel B (2018) Prototyping the Internet of Things with Web technologies: is it easy? In Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops 2018) (pp. 518–522). https://doi.org/10.1109/percomw.2018.8480268
48. Terán PG, Plua RK (2018) Home automation application for the monitoring and control of an electric water heater using AWS technology. In Proceedings of the IEEE 38th Central America and Panama Convention (CONCAPAN 2018) (pp. 1–6). https://doi.org/10.1109/concapan.2018.8596474
49. Usländer T, Batz T (2018) Agile service engineering in the Industrial Internet of Things. Future Internet 10(10):100. https://doi.org/10.3390/fi10100100
50. ISO/IEC/IEEE (2018) 26515–2018 - ISO/IEC/IEEE International Standard - Systems and software engineering — Developing information for users in an agile environment. In ISO/IEC/IEEE 26515:2018(E) (pp. 1–32). https://doi.org/10.1109/ieeestd.2018.8584455
51. Dalpiaz F, van der Schalk I, Brinkkemper S, Aydemir FB, Lucassen G (2019) Detecting terminological ambiguity in user stories: Tool and experimentation. Inform Softw Technol 110:3–16. https://doi.org/10.1016/j.infsof.2018.12.007
52. Pecchia C, Trincardi M, Di Bello P (2016) Expressing, managing, and validating user stories: Experiences from the market. In Advances in Intelligent Systems and Computing (Vol. 422, pp. 103–111). https://doi.org/10.1007/978-3-319-27896-4_9
53. Muntés-Mulero V, Ripollés O, Gupta S, Dominiak J, Willeke ER, Matthews P, Somosköi B (2019) Agile risk management for multi-cloud software development. IET Software 13(3):172–181. https://doi.org/10.1049/iet-sen.2018.5295
54. OpenMBEE (2023) Open Model Based Engineering Environment. https://www.openmbee.org/ (Accessed: 27 October 2023)
55. Whittle J, Hutchinson J, Rouncefield M (2014) The state of practice in model-driven engineering. IEEE Softw 31(3):79–85. https://doi.org/10.1109/ms.2013.65
56. Sosa-Reyna CM, Tello-Leal E, Lara-Alabazares D (2018) Methodology for the model-driven development of service oriented IoT applications. J Syst Architect 90:15–22. https://doi.org/10.1016/j.sysarc.2018.08.008
57. OMG (2023) Model Driven Architecture (MDA). https://www.omg.org/mda/ (Accessed: 27 October 2023)
58. Uzunov AV, Falkner K, Fernández EB (2015) A comprehensive pattern-oriented approach to engineering security methodologies. Inform Softw Technol 57:217–247. https://doi.org/10.1016/j.infsof.2014.09.001
59. Bourque P, Fairley RE (2014) Guide to the Software Engineering Body of Knowledge (SWeBOK®): Version 3.0. IEEE Computer Society
60. ISO/IEC/IEEE (2017) 24765–2017 - ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary. In ISO/IEC/IEEE 24765:2017(E) (pp. 1-541). https://doi.org/10.1109/ieeestd.2017.8016712
61. Cloutier RJ (2023) Guide to the Systems Engineering Body of Knowledge (SEBoK). https://sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_(SEBoK) (Accessed: 30 October 2023)
62. Project Management Institute (2013) Software Extension to the PMBOK® Guide – Fifth Edition. Project Management Institute
63. Project Management Institute (2021) A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Seventh edition and the Standard for Project Management. Project Management Institute
64. Farncombe A (2004) Project stories: combining life-cycle process models. Scenarios, stories, use cases: through the Systems Development Life-Cycle. Wiley, pp 299–324
65. Pressman RS, Maxim BR (2019) Software Engineering: a practitioner's approach - Eighth Edition. McGraw-Hill Education
66. ISO/IEC/IEEE (2017) 12207–2017 - ISO/IEC/IEEE International Standard - Systems and software engineering -- Software life cycle processes. In ISO/IEC/IEEE 12207:2017(E) First edition 2017-11 (pp. 1-157). https://doi.org/10.1109/ieeestd.2017.8100771
67. ISO/IEC/IEEE (2023) 15288–2023 - ISO/IEC/IEEE International Standard - Systems and software engineering–System life cycle processes. In ISO/IEC/IEEE 15288:2023(E) (pp. 1-128). https://doi.org/10.1109/ieeestd.2023.10123367
68. ISO/IEC/IEEE (2018) 24748-1-2018 - ISO/IEC/IEEE International Standard - Systems and software engineering - Life cycle management - Part 1: Guidelines for life cycle management. In ISO/IEC/IEEE 24748-1:2018(E) (pp. 1–82). https://doi.org/10.1109/ieeestd.2018.8526560
69. ISO/IEC/IEEE (2018) 24748-2-2018 - ISO/IEC/IEEE International Standard - Systems and Software Engineering– Life Cycle Management– Part 2: Guidelines for the Application of ISO/IEC/IEEE 15288 (System Life Cycle Processes). In ISO/IEC/IEEE 24748-2:2018(E) (pp. 1–90). https://doi.org/10.1109/ieeestd.2018.8764712
70. ISO/IEC/IEEE (2020) 24748-3-2020 - ISO/IEC/IEEE International Standard - Systems and software engineering–Life cycle management–Part 3: Guidelines for the application of ISO/IEC/IEEE 12207 (software life cycle processes). In ISO/IEC/IEEE 24748-3:2020(E) (pp. 1–76). https://doi.org/10.1109/ieeestd.2020.9238526

71. Laporte CY, Vargas EP (2014) The development of international standards to facilitate process improvements for very small entities. In Software Design and Development: Concepts, Methodologies, Tools, and Applications (pp. 1335–1361). IGI Global. https://doi.org/10.4018/978-1-4666-4301-7.ch065
72. Rashid N, Quirós G, Faruque MAA (2019) A survivability-aware cyber-physical systems design methodology. In 2019 IEEE 15th IEEE International Conference on Automation Science and Engineering (CASE 2019) (pp. 848–853). https://doi.org/10.1109/coase.2019.8843113
73. Henke C, Michael J, Lankeit C, Trächtler A (2017) A Holistic Approach for Virtual Commissioning of Intelligent Systems: Model-Based Systems Engineering for the Development of a Turn-Milling Center. In 2017 Annual IEEE International Systems Conference (SysCon) (pp. 1–6), https://doi.org/10.1109/SYSCON.2017.7934735
74. Augusto JC, Hornos MJ (2013) Software simulation and verification to increase the reliability of Intelligent Environments. Adv Eng Softw 58:18–34. https://doi.org/10.1016/j.advengsoft.2012.12.004
75. Orfanus D, Heimfarth T, Janácik P (2009) An approach for systematic design of emergent self-organization in wireless sensor networks. In Computation World: Future Computing, Service Computation, Adaptive, Content, Cognitive, Patterns (ComputationWorld 2009) (pp. 92–98). https://doi.org/10.1109/computationworld.2009.87
76. Coronato A, De Pietro G (2010) Formal design of ambient intelligence applications. IEEE Comput 43(12):60–68. https://doi.org/10.1109/mc.2010.335
77. McGrath W, Etemadi M, Roy S, Hartmann B (2015) Fabryq: Using phones as gateways to prototype Internet of Things applications using Web scripting. In Proceedings of the 2015 ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2015) (pp. 164–173). https://doi.org/10.1145/2774225.2774835
78. Broenink JF, Vos PJD, Lu Z, Bezemer MM (2016) A co-design approach for embedded control software of Cyber-Physical Systems. In 11th Systems of Systems Engineering Conference, (SoSE 2016) (pp. 1–5). https://doi.org/10.1109/SYSOSE.2016.7542927
79. Jensen JC, Chang DH, Lee EA (2011) A model-based design methodology for Cyber-Physical Systems. In 7th International Wireless Communications and Mobile Computing Conference (IWCMC 2011) (pp. 1666–1671). https://doi.org/10.1109/IWCMC.2011.5982785
80. Desjardins A, Viny JE, Key C, Johnston N (2019) Alternative avenues for IoT: designing with non-stereotypical homes. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI'19) (pp. 1–13). https://doi.org/10.1145/3290605.3300581
81. Gianni F, Mora S, Divitini M (2019) RAPIoT toolkit: Rapid prototyping of collaborative Internet of Things applications. Future Generation Comput Syst 95:867–879. https://doi.org/10.1016/j.future.2018.02.030
82. Wiberg M (2018) Addressing IoT: Towards material-centered interaction design. In Lecture Notes in Computer Science (Vol. 10901, pp. 198–207). https://doi.org/10.1007/978-3-319-91238-7_17
83. Mezghani E, Expósito E, Drira K (2017) A model-driven methodology for the design of autonomic and cognitive IoT-based systems: application to healthcare. IEEE Trans Emerg Top Comput Intell 1(3):224–234. https://doi.org/10.1109/tetci.2017.2699218
84. Berkenbrock GR, Medeiros Berkenbrock D, Alves C (2015) O. C. The need of software development process for wireless sensor networks with cooperative nodes. In 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM 2015) (pp. 930–933). https://doi.org/10.1109/inm.2015.7140412
85. Lutze R (2020) Digital Twin Based Software Design in eHealth - A New Development Approach for Health/Medical Software Products. In Proceedings of the 2020 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC 2020) (pp. 1–9). https://doi.org/10.1109/ICE/ITMC49519.2020.9198546
86. Pal A, Mukherjee A, Balamuralidhar P (2015) Model-Driven Development for Internet of Things: towards easing the concerns of application developers. In Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (Vol. 150, pp. 339–346). https://doi.org/10.1007/978-3-319-19656-5_46
87. Meshkova E, Riihijärvi J, Oldewurtel F, Jardak C, Mähönen P (2008) Service-oriented design methodology for wireless sensor networks: a view through case studies. In 2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 2008) (pp. 146–153). https://doi.org/10.1109/sutc.2008.43
88. Fahmideh M, Abbasi AA, Behnaz A, Grundy J, Susilo W (2022) Software Engineering for Internet of Things: the practitioners' perspective. IEEE Trans Software Eng 48(8):2857–2878. https://doi.org/10.1109/tse.2021.3070692
89. Lekidis A, Stachtiari E, Katsaros P, Bozga M, Georgiadis CK (2018) Model-based design of IoT systems with the BIP component framework. Software: Pract Experience 48(6):1167–1194. https://doi.org/10.1002/spe.2568
90. Ciccozzi F, Spalazzese R (2016) MDE4IoT: Supporting the Internet of Things with Model-Driven Engineering. In Studies in Computational Intelligence (Vol. 678, pp. 67–76). https://doi.org/10.1007/978-3-319-48829-5_7
91. Harbouche A, Djedi N, Erradi M, Ben-Othman J, Kobbane A (2017) Model driven flexible design of a wireless body sensor network for health monitoring. Comput Netw 129:548–571. https://doi.org/10.1016/j.comnet.2017.06.014
92. Corredor I, Bernardos AM, Iglesias J, Casar JR (2012) Model-driven methodology for rapid deployment of smart spaces based on resource-oriented architectures. Sensors 12(7):9286–9335. https://doi.org/10.3390/s120709286
93. Sosa-Reyna CM, Tello-Leal E, Lara-Alabazares D (2018) An approach based on Model-Driven Development for IoT applications. In Proceedings of the IEEE International Congress on Internet of Things (ICIOT 2018) (pp. 134–139). https://doi.org/10.1109/iciot.2018.00026
94. Sosa-Reyna CM, Tello-Leal E, Lara-Alabazares D, Mata-Torres JA, Lopez-Garza E (2018) A methodology based on Model-Driven Engineering for IoT application development. In Twelfth International Conference on Digital Society and eGovernments (ICDS 2018) (pp. 36–41)
95. Brambilla M, Umuhoza E, Acerbis R (2017) Model-driven development of user interfaces for IoT systems via domain-specific components and patterns. J Internet Serv Appl 8:14. https://doi.org/10.1186/s13174-017-0064-1
96. Costa B, Pires PF, Delicato FC (2016) Modeling IoT Applications with SysML4IoT. In Proceedings of the 42nd Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2016) (pp. 157–164). https://doi.org/10.1109/seaa.2016.19
97. Cicirelli F, Fortino G, Guerrieri A, Spezzano G, Vinci A (2017) Metamodeling of smart environments: from design to implementation. Adv Eng Inform 33:274–284. https://doi.org/10.1016/j.aei.2016.11.005
98. Ataide A, Barros JP, Brito IS, Gomes L (2017) Towards automatic code generation for distributed cyber-physical systems: A first prototype for Arduino boards. In Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2017) (pp. 1–4). https://doi.org/10.1109/etfa.2017.8247737

99. Fortino G, Russo W (2012) ELDAMeth: an agent-oriented methodology for simulation-based prototyping of distributed agent systems. Inform Softw Technol 54(6):608–624. https://doi.org/10.1016/j.infsof.2011.08.006
100. Chauhan S, Patel P, Delicato FC, Chaudhary S (2016) A development framework for programming cyber-physical systems. In Proceedings of the 2nd International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS 2016) (pp. 47–53). https://doi.org/10.1145/2897035.2897039
101. Pico-Valencia, P., Holgado-Terriza, JA., & Paderewski, P. (2019) A systematic method for building Internet of Agents applications based on the Linked Open Data approach. Future Generation Computer Systems 94:250-271, https://doi.org/10.1016/j.future.2018.11.042.
102. Fortino G, Guerrieri A, Russo W, Savaglio C (2015) Towards a development methodology for smart object-oriented IoT systems: a metamodel approach. In Proceedings of the 2015 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2015) (pp. 1297–1302). https://doi.org/10.1109/smc.2015.231
103. Fortino G (2016) Agents meet the IoT: toward ecosystems of networked smart objects. IEEE Syst Man Cybernetics Magazine 2(2):43–47. https://doi.org/10.1109/msmc.2016.2557483
104. Astels D (2003) Test-driven development: a practical guide. Prentice Hall
105. Guerrero-Ulloa G, Carvajal-Suarez D, Pachay-Espinoza A, Brito-Casanova G, Hornos MJ, Rodríguez-Domínguez C (2023) TDDT4IoTS: Test-Driven Development Tool for IoT-based Systems. https://aplicaciones.uteq.edu.ec/tddt4iots/ (Accessed: 16 February 2024)
106. Guerrero-Ulloa G, Andrango-Catota A, Abad-Alay M, Hornos MJ, Rodríguez-Domínguez C (2023) Development and assessment of an indoor air quality control IoT-based system. Electronics 12(3):608. https://doi.org/10.3390/electronics12030608
107. Guerrero-Ulloa G, Fernández-Loor A, Moreira F, Nováis P, Rodríguez-Domínguez C, Hornos MJ (2023) Validation of a development methodology and tool for IoT-based systems through a case study for visually impaired people. Internet Things 23:100900. https://doi.org/10.1016/j.iot.2023.100900
108. Guerrero-Ulloa G, Méndez-García A, Torres-Lindao V, Zamora-Mecías V, Rodríguez-Domínguez C, Hornos MJ (2023) Internet of Things (IoT)-based indoor plant care system. J Ambient Intell Smart Environ 15(1):47–62. https://doi.org/10.3233/ais-220483
109. Fortino G, Gravina R, Russo W, Savaglio C, Wasielewska K, Ganzha M, Paprzycki M, Pawłowski W, Szmeja P, Tkaczyk R (2021) INTER-Meth: a methodological approach for the integration of heterogeneous IoT systems. In Interoperability of Heterogeneous IoT Platforms: A Layered Approach (pp. 195–230). Springer. https://doi.org/10.1007/978-3-030-82446-4_7
110. Wang Z, Cui L, Guo W, Zhao L, Yuan X, Gu X, Tang W, Bu L, Huang W (2022) A design method for an intelligent manufacturing and service system for rehabilitation assistive devices and special groups. Adv Eng Inform 51:101504. https://doi.org/10.1016/j.aei.2021.101504
111. Patel P, Cassou D (2015) Enabling high-level application development for the Internet of Things. J Syst Softw 103:62–84. https://doi.org/10.1016/j.jss.2015.01.027
112. Gogineni S, Riedelsheimer T, Stark R (2019) Systematic product development methodology for customizable IoT devices. Procedia CIRP 84:393–399. https://doi.org/10.1016/j.procir.2019.04.287
113. Benghazi K, Hurtado MV, Hornos MJ, Rodríguez ML, Rodríguez-Domínguez C, Pelegrina AB, Rodríguez-Fórtiz MJ (2012) Enabling correct design and formal analysis of Ambient Assisted Living systems. J Syst Softw 85(3):498–510. https://doi.org/10.1016/j.jss.2011.05.022
114. Hornos MJ (2017) Application of Software Engineering techniques to improve the reliability of Intelligent Environments. J Reliable Intell Environ 3(1):1–3. https://doi.org/10.1007/s40860-017-0043-0
115. Hornos MJ, Rodríguez-Domínguez C (2018) Increasing user confidence in intelligent environments. J Reliable Intell Environ 4(2):71–73. https://doi.org/10.1007/s40860-018-0063-4
116. Holzmann GJ (2003) The Spin model checker: primer and reference manual. Addison-Wesley Professional
117. Augusto JC, Callaghan V, Cook DJ, Kameas A, Satoh I (2013) Intelligent Environments: a manifesto. Human-centric Comput Inform Sci 3(1):12. https://doi.org/10.1186/2192-1962-3-12
118. Nielsen CB, Larsen PG, Fitzgerald J, Woodcock J, Peleška J (2015) Systems of systems engineering. ACM-CSUR 48(2):1–41. https://doi.org/10.1145/2794381
119. Augusto JC, Kramer D, Alegre U, Covaci A, Santokhee A (2017) The user-centred intelligent environments development process as a guide to co-create smart technology for people with special needs. Univ Access Inf Soc 17(1):115–130. https://doi.org/10.1007/s10209-016-0514-8
120. Jones SP, Hara S, Augusto JC (2014) eFRIEND: an ethical framework for intelligent environments development. Ethics Inf Technol 17(1):11–25. https://doi.org/10.1007/s10676-014-9358-1
121. Augusto JC, Gimenez-Manuel JG, Quinde M, Oguego CL, Ali SM, James-Reynolds C (2020) A Smart Environments Architecture (SEArch). Appl Artif Intell 34(2):155–186. https://doi.org/10.1080/08839514.2020.1712778
122. Augusto JC, Quinde M, Oguego CL, Manuel JGG (2021) Context-Aware Systems Architecture (CASA). Cybernetics and Systems, 1–27. https://doi.org/10.1080/01969722.2021.1985226
123. Augusto JC, Quinde M, Kahn N (2019) Using Formal Methods to Guide the Development of an Asthma Management System. In 2019 10th International Conference on Dependable Systems, Services and Technologies (DESSERT) (pp. 57–62). https://doi.org/10.1109/dessert.2019.8770017
124. Rodrigues FF, Matalonga S, Travassos GH (2016) CATS Design. Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing. https://doi.org/10.1145/2993288.2993293
125. Augusto JC, Quinde M, Oguego CL (2019) Context-aware Systems Testing and Validation. In 2019 10th International Conference on Dependable Systems, Services and Technologies (DESSERT) (pp. 7–12). https://doi.org/10.1109/dessert.2019.8770048
126. Chin J, Callaghan V, Allouch SB (2019) The Internet-of-Things: reflections on the past, present and future from a user-centered and smart environment perspective. J Ambient Intell Smart Environ 11(1):45–69. https://doi.org/10.3233/ais-180506
127. Manuel JGG, Augusto JC, Stewart J (2020) ANABEL: towards empowering people living with dementia in ambient assisted living. Univ Access Inf Soc 21(2):457–476. https://doi.org/10.1007/s10209-020-00760-5
128. Quinde M, Augusto JC, Khan N, Van Wyk A (2020) ADAPT: Approach to Develop context-Aware solutions for Personalised asthma ManagemenT. J Biomed Inform 111:103586. https://doi.org/10.1016/j.jbi.2020.103586
129. Augusto JC, Muñoz A (2019) User preferences in Intelligent Environments. Appl Artif Intell 33(12):1069–1091. https://doi.org/10.1080/08839514.2019.1661596
130. Oguego CL, Augusto JC, Springett M, Quinde M, Reynolds CJ (2019) An Interface for Managing users' Preferences in AmI. In 2019 15th International Conference on Intelligent Environments (IE) (pp. 56–59). https://doi.org/10.1109/ie.2019.00009
131. Ali SM, Augusto JC, Windridge D, Ward EV (2022) A user-guided personalization methodology to facilitate new smart home occupancy. Univ Access Inf Soc 22(3):869–891. https://doi.org/10.1007/s10209-022-00883-x
132. Sakanga N, Augusto JC, Brodie L, Marzano L (2022) Quality Traceability for User-centric Context-aware Systems in

Intelligent Environment. In 2022 IEEE 8th World Forum on Internet of Things (WF-IoT) (pp. 13–20). https://doi.org/10.1109/wf-iot54382.2022.10152209
133. Fortino G, Savaglio C, Palau CE, De Puga JS, Ganzha M, Paprzycki M, Montesinos M, Liotta A, Llop M (2017) Towards multi-layer interoperability of heterogeneous IoT platforms: The INTER-IoT approach. In Integration, Interconnection, and Interoperability of IoT Systems (pp. 199–232). Springer. https://doi.org/10.1007/978-3-319-61300-0_10
134. Ameller D (2009) Considering Non-Functional Requirements in Model-Driven Engineering [Master's Thesis]. Universitat Politècnica de Catalunya, Barcelona, Spain
135. Sachdeva V, Chung L (2017) Handling non-functional requirements for big data and IoT projects in Scrum. In Proceedings of the 7th International Conference Confluence 2017 on Cloud Computing, Data Science and Engineering (pp. 216–221). https://doi.org/10.1109/confluence.2017.7943152
136. Jones TS, Richey RC (2000) Rapid prototyping methodology in action: a developmental study. Education Tech Research Dev 48(2):63–80. https://doi.org/10.1007/bf02313401
137. Nast B, Sandkuhl K (2023) Methods for Model-Driven Development of IoT Applications: Requirements from Industrial Practice. In Proceedings of the 18th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2023) (pp. 170–181). https://doi.org/10.5220/0011973500003464
138. Ahmad E (2023) Model-based system engineering of the Internet of Things: a bibliometric literature analysis. IEEE Access 11:50642–50658. https://doi.org/10.1109/access.2023.3277429
139. Ghannem A, Salah Hamdi M, Ammar H, Soui M (2017) A systematic classification of requirements engineering approaches for adaptive systems. In Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing (ICC 2017) (pp. 1–9). https://doi.org/10.1145/3018896.3018939
140. Bouanaka C, Benlahrache N, Benhamaid S, Bouhamed E (2020) A review of IoT systems engineering: application to the smart traffic lights system. In Proceedings of the 4th International Conference on Advanced Aspects of Software Engineering (ICAASE 2020) (pp. 1–8). https://doi.org/10.1109/icaase51408.2020.9380114
141. Varga P, Blomstedt F, Ferreira LL, Eliasson J, Johansson M, Delsing J, De Soria IM (2017) Making system of systems interoperable – the core components of the Arrowhead framework. J Netw Comput Appl 81:85–95. https://doi.org/10.1016/j.jnca.2016.08.028
142. Arm (2023) What are IoT devices. Arm | The Architecture for the Digital World. https://www.arm.com/glossary/iot-devices
143. Perera C, Zaslavsky A, Christen P, Georgakopoulos D (2014) Context Aware Computing for the Internet of Things: a survey. IEEE Commun Surv Tutorials 16(1):414–454. https://doi.org/10.1109/surv.2013.042313.00197
144. Omar YM, Minoufekr M, Plapper P (2019) Business analytics in manufacturing: current trends, challenges and pathway to market leadership. Oper Res Perspect 6:100127. https://doi.org/10.1016/j.orp.2019.100127
145. Maciel P, Dantas J, Melo C, Pereira P, Oliveira F, Araújo J, Matos R (2021) A survey on reliability and availability modeling of edge, FOG, and cloud computing. J Reliable Intell Environ 8(3):227–245. https://doi.org/10.1007/s40860-021-00154-1
146. Miller T, Hoffman RR, Amir O, Holzinger A (2022) Special issue on Explainable Artificial Intelligence (XAI). Artif Intell 307:103705. https://doi.org/10.1016/j.artint.2022.103705
147. Palade A, Cabrera C, Li F, White G, Razzaque MA, Clarke S (2018) Middleware for Internet of Things: an evaluation in a small-scale IoT environment. J Reliable Intell Environ 4(1):3–23. https://doi.org/10.1007/s40860-018-0055-4
148. Bonawitz K, Kairouz P, McMahan B, Ramage D (2022) Federated learning and privacy. Commun ACM 65(4):90–97. https://doi.org/10.1145/3500240
149. Ogbuabor G, Augusto JC, Moseley R, Van Wyk A (2021) Context-aware support for cardiac health monitoring using federated machine learning. In Lecture Notes in Computer Science (Vol. 13101, pp. 267–281). https://doi.org/10.1007/978-3-030-91100-3_22
150. Sánchez S, Machacuay J, Quinde M (2023) Federated Learning for Human Activity Recognition on the MHealth Dataset. In Lecture Notes in Computer Science (Vol. 14125, pp. 215–225). https://doi.org/10.1007/978-3-031-42505-9_19
151. Hosseini S, Sardo SR (2022) Network intrusion detection based on deep learning method in internet of thing. J Reliable Intell Environ 9(2):147–159. https://doi.org/10.1007/s40860-021-00169-8
152. Perera C, Barhamgi M, Bandara AK, Ajmal M, Price B, Nuseibeh B (2020) Designing privacy-aware internet of things applications. Inf Sci 512:238–257. https://doi.org/10.1016/j.ins.2019.09.061
153. Bahaa A, Abdelaziz A, Sayed A, El-Fangary LM, Fahmy H (2021) Monitoring real time security attacks for IoT systems using DevSecOps: a systematic literature review. Information 12(4):154. https://doi.org/10.3390/info12040154
154. Zambare P, Liu Y (2023) Understanding security challenges and defending access control models for Cloud-based Internet of Things network. In Internet of Things. Advances in Information and Communication Technology (IFIPIoT 2023) (Vol. 684, pp. 179–197). Springer. https://doi.org/10.1007/978-3-031-45882-8_13
155. Mohan D, Alwin L, Neeraja P, Lawrence KD, Pathari V (2021) A private Ethereum blockchain implementation for secure data handling in Internet of Medical Things. J Reliable Intell Environ 8(4):379–396. https://doi.org/10.1007/s40860-021-00153-2
156. Shaheen Y, Hornos MJ, Rodríguez-Domínguez C (2023) IoT security and privacy challenges from the developer perspective. In Lecture Notes in Networks and Systems (Vol. 770, pp. 13–21). https://doi.org/10.1007/978-3-031-43461-7_2
157. Amraoui N, Zouari B (2021) Securing the operation of Smart Home Systems: a literature review. J Reliable Intell Environ 8(1):67–74. https://doi.org/10.1007/s40860-021-00160-3
158. Bertino E (2016) Data privacy for IoT systems: Concepts, approaches, and research directions. In IEEE International Conference on Big Data (pp. 3645–3647). https://doi.org/10.1109/BigData.2016.7841030
159. Yu J, Guo L, Zhang J, Wang G (2024) A survey on graph neural network-based next POI recommendation for smart cities. Appear Anniversary Issue J Reliable Intell Environ 10(3)
160. Sun S, Yen J (2005) Information Supply Chain: A Unified Framework for Information-Sharing. In Lecture Notes in Computer Science (pp. 422–428). https://doi.org/10.1007/11427995_38
161. Stephanidis C, Salvendy G, Antona M, Chen JYC, Dong J, Duffy VG, Fang X, Fidopiastis CM, Fragomeni G, Fu LP, Guo Y, Harris D, Ioannou A, Jeong K, Konomi S, Krömker H, Kurosu M, Lewis JR, Marcus A, Zhou J (2019) Seven HCI grand challenges. Int J Hum Comput Interact 35(14):1229–1269. https://doi.org/10.1080/10447318.2019.1619259
162. Bök P-B, Micucci D (2024) The future of human and animal digital health platforms. Appear Anniversary Issue J Reliable Intell Environ 10(3)
163. Le Guilly T, Nielsen MK, Pedersen TG, Skou A, Kjeldskov J, Skov MB (2016) User constraints for reliable user-defined smart home scenarios. J Reliable Intell Environ 2(2):75–91. https://doi.org/10.1007/s40860-016-0020-z
164. Saidi A, Kacem MH, Tounsi I, Kacem AH (2023) A formal approach to specify and verify Internet of Things architecture. Internet Things 24:100972. https://doi.org/10.1016/j.iot.2023.100972

165. Ashouri M, Lorig F, Davidsson P, Spalazzese R (2019) Edge computing simulators for IoT system design: an analysis of qualities and metrics. Future Internet 11(11):235. https://doi.org/10.3390/fi11110235
166. Prabha P, Chatterjee K (2022) Design and implementation of hybrid consensus mechanism for IoT based healthcare system security. Int J Inform Technol 14:2081–2093. https://doi.org/10.1007/s41870-022-00880-6
167. Birolini A (2017) Reliability Engineering: Theory and Practice (8th Edition). Springer, Berlin
168. Tang S, Xie Y (2021) Availability modeling and performance improving of a healthcare Internet of Things (IoT) system. IoT 2(2):16. https://doi.org/10.3390/IOT2020016
169. Xing L (2020) Reliability in Internet of Things: current status and future perspectives. IEEE Internet Things J 7(8):6704–6721. https://doi.org/10.1109/jiot.2020.2993216
170. Yuen KKF (2024) Fuzzy Cognitive Network Process for software reliability and quality measurement: comparisons with Fuzzy Analytic Hierarchy Process. Appear Anniversary Issue J Reliable Intell Environ 10(3)
171. Corno F, De Russis L, Mannella L (2022) Helping novice developers harness security issues in cloud-IoT systems. J Reliable Intell Environ 8(3):261–283. https://doi.org/10.1007/s40860-022-00175-4
172. Nasiri S, Adadi A, Lahmer M (2023) Automatic generation of business process models from user stories. Int J Electr Comput Eng 13(1):809–822. https://doi.org/10.11591/ijece.v13i1.pp809-822
173. Omer OSD, Eltyeb S (2022) Towards an automatic generation of UML class diagrams from textual requirements using case-based reasoning approach. In 4th International Conference on Applied Automation and Industrial Diagnostics (ICAAID 2022) (pp. 1–5). https://doi.org/10.1109/icaaid51067.2022.9799502
174. Refactoring.Guru (2023) Design patterns. https://refactoring.guru/design-patterns (Accessed: 3 November 2023)
175. Lu Z, Afridi I, Kang HJ, Ruchkin I, Zheng X (2024) Surveying neuro-symbolic approaches for reliable Artificial Intelligence of Things. Appear Anniversary Issue J Reliable Intell Environ 10(3)