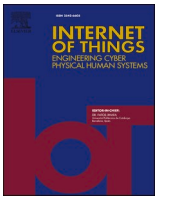
Internet of Things 36 (2026) 101887

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.sciencedirect.com/journal/internet-of-things



BPSmart-CARE: a framework for managing contextualized actions in IoT systems through the integration of business process modelling and complex event processing

Adrian Bazan-Muñoz [a,*], Cesare Pautasso [b], Guadalupe Ortiz [a], Alfonso Garcia-de-Prado [a]

[a] University of Cádiz, Puerto Real, Cádiz, Spain
[b] Faculty of Informatics University of Lugano, Lugano, Switzerland

ARTICLE INFO

Keywords:
Context awareness
Internet of Things
Business process model
Complex event processing

ABSTRACT

The exponential growth of Internet of Things (IoT) sensors and data has led to multiple software applications which aim at collecting, monitoring, controlling, and automating decisions based on such data in several domains such as smart cities. Even though early IoT frameworks and software architectures relied on correlating data from a single application domain, missing opportunities to improve contextualisation; in the last years several Complex Event Processing (CEP)-based frameworks and applications have been provided to improve the correlation of data from heterogenous domains, facilitating context-aware decision making. However, several issues remain unsolved: on the one hand, frameworks should not only provide the means for data correlation and contextualisation, but also for the definition of customisable real-time actions based on the context. On the other hand, CEP patterns update remains static in these frameworks, while there is a need for dynamically updating CEP pattern logic to adapt to changing contexts in real time. Even more, CEP patterns logic understanding might also be a handicap for developers. To face these challenges, in this paper, we enhance an existent CEP-based framework and software architecture by incorporating Business Process Modelling (BPM) to facilitate the definition of CEP patterns rationale, manage interdependencies between patterns, define and handle contextualised user actions, and enable real-time updating of CEP pattern logic based on detected situations. The proposed approach is illustrated through a case study, assessed via usability and performance evaluations, which show improvements in comprehensibility, maintainability, and updatability of CEP-based software applications for context-aware IoT scenarios.

1. Introduction

This introduction first motivates the need to enhance context awareness in order to enable more accurate, adaptive, and timely decision-making in IoT systems, where system behaviour and user-related actions are shaped by dynamic environmental and situational conditions. It then presents the underlying research hypothesis and formulates the Research Questions (RQs) that guide this study, thereby delimiting its scope and clarifying the aspects under investigation. Finally, it outlines the main contributions of the

* Corresponding author.
E-mail addresses: adrian.bazan@uca.es (A. Bazan-Muñoz), cesare.pautasso@usi.ch (C. Pautasso).

https://doi.org/10.1016/j.iot.2026.101887

Available online 9 February 2026

paper.

1.1. Motivation

In recent years, cities have faced exponential growth as an increasing percentage of the world's population has become urbanised [1]. Such a population growth places increased pressure on critical infrastructure such as transport, housing, water, electricity and utilities. As a consequence, major challenges arise in several domains, such as in energy consumption, waste, pollution, traffic, etc. To address these challenges through technology and improved management of urban resources, Smart Cities [2] have emerged. In this scope, the Internet of Things (IoT) enables the collection of data from heterogeneous sensor devices, the real-time monitoring of such data, the automation of decisions, and cost reduction, thereby improving the services provided in smart cities. Similarly, IoT technologies, and more specifically smart devices, have recently been boosted by reduced communication costs and the emergence of new technologies for smart environments. This evolution has led to numerous applications that enhance decision-making and improve the services delivered to users in these domains.

However, most early smart environment applications rely on data from a single application domain, monitoring such data, and generating intelligent actions limited to that domain [3,4]. By disregarding information from other application domains, these systems miss a significant opportunity to improve contextualisation in smart environments [5]. Furthermore, enhancing the contextualisation of smart IoT applications and smart environments requires a software architecture capable of performing data fusion across multiple domains, allowing data to be integrated and correlated. Through the homogeneous integration and correlation of data from devices in different application domains, contextualisation can be achieved across multiple domains rather than a single one, therefore improving decision-making.

In addition, citizens expect more than the detection of generic event patterns for the different situations they may face. They demand personalised services and event patterns that generate contextualised and customised actions, i.e. systems that provide intelligent responses based on specific situations of interest. A situation of interest is defined as a sequence of events and/or contextual conditions occurring within a given time period, which is relevant to a specific application domain and a specific user [6]. Thus, situations of interest emerge from the combination of individual events, such as environmental or user-related data, when information from multiple sources reveals meaningful conditions requiring attention or actions. For example, consider a smart home system that uses temperature sensors and wearables to monitor both indoor temperature and user activity. One situation of interest could be when the indoor temperature drops below 18°C and the user, who is usually active at this time of the day, has remained inactive for more than two hours. This situation may indicate that the user is asleep, or experiencing discomfort or a health issue, such as hypothermia in elderly people. In this case, an alert could be sent to a caregiver, or the heating system could be activated automatically. Situations of interest are inherently domain-dependent, as they must be relevant to the stakeholders of each domain. Therefore, the ability to recognise domain-specific situations of interest enables the generation of personalised contextual actions, the automation of responses and improved decision-making in intelligent systems.

These challenges have been addressed by Smart-CARE a context-aware real-time smart environment framework recently proposed [7,8]. Although [8] presents Smart-CARE through an Ambient Assisted Living (AAL) case study, the framework itself is domain-agnostic and applicable to a wide range of IoT and smart-environment domains, as illustrated in [7]. Moreover, from [8] onwards, the name “Smart-CARE” is adopted to consistently refer to the overall framework. Within this framework, a taxonomy was introduced to support the homogeneous description of data across multiple IoT and smart environment application domains. This taxonomy provides mechanisms that facilitate adaptation to different domains, while providing a common structure for information exchange between IoT domains and smart environments. It also enables the definition of contextual actions required for intelligent decision making in IoT systems. Moreover, the taxonomy simplifies data processing by allowing developers to define data structures using the JSON standard. In addition, a software architecture was proposed as part of the framework, aimed at receiving, processing and correlating data from different application domains, previously defined according to the taxonomy, in a homogeneous manner. This software architecture incorporates a Complex Event Processing (CEP) engine, which supports real-time data processing and the detection of relevant situations of interest for users. Particularly, the conditions to be met to detect a situation of interest are defined through an event pattern; once the event pattern is detected a contextualized action can be triggered to address the situation of interest. Therefore, by enabling information sharing across domains, the framework improves contextualisation and enhances decision-making capabilities.

Nevertheless, the definition of contextualised actions remains complex due to the large variety of possible contextualised actions that must be specified for each user. Defining a large number of CEP event patterns and contextualised actions significantly increases system complexity [9], making systems harder to maintain, adapt and evolve. Furthermore, CEP event patterns are static rules that offer limited flexibility once deployed, which complicates their adaptation to context changes. As a result, users may find it difficult to design event patterns that can be dynamically updated at runtime. For instance, the time required to dispose of rubbish varies depending on contextual factors, such as whether a user is alone or accompanied, or whether the activity occurs during the day or at night. Addressing this issue requires runtime adaption of event patterns, thereby avoiding the need to define a separate event pattern for every possible situation of interest.

1.2. Research hypothesis and research questions

To tackle the challenges identified in the previous subsection, our research is driven by the hypothesis that Business Process Management (BPM) models can effectively contribute to bridge the identified gaps. This hypothesis is motivated by prior work

showing that BPM-based visual representations have been successfully applied to model IoT systems and intelligent environments [10]. Moreover, BPM supports the analysis, design, implementation and continuous monitoring of system processes [11–13] and it enables system optimisation and error reduction, improving productivity and supporting rapid adaptation to evolving requirements.

Accordingly, we hypothesise that a novel integration of BPM models and Business Process Model and Notation (BPMN) standard with CEP-based architectures within the Smart-CARE framework may provide additional benefits over existing solutions. In particular, BPM models may allow the representation of both the behaviour of a context-aware architecture that integrates data from multiple domains, as well as the behaviour of a CEP engine that detects different stakeholder-specific situations of interest. This BPM-CEP integration within the Smart-CARE framework may reduce the complexity of defining and managing personalised contextual actions within IoT systems and smart environments. Furthermore, BPM may facilitate the explicit representation of the rationale underlying CEP event patterns, improving their understanding. BPM models may capture not only the behaviour detected by the smart devices, but also the reactions triggered by the CEP event patterns. In addition, BPM may enable the modelling of situations in which CEP event patterns must be updated dynamically based on runtime context. For example, BPM could depict the dynamic adaptation of a CEP event pattern that controls the expected time for a user to return home after disposing the rubbish, updating this temporal parameter in real time according to evolving conditions. The time required would differ, for example, if the usual bin were full and the user needed to locate another one.

To guide the investigation and structure the evaluation of these extensions, we formulate the following RQs:

RQ1. Can the extension of the Smart-CARE taxonomy with BPM improve the representation and personalisation of contextualised user actions in IoT systems? And if so, how effectively can the integration of BPM within Smart-CARE, beyond the taxonomy layer, support the definition, reuse and management of personalised contextualised actions for different users? Previous context-based proposals [14] often relied on domain-specific data models or taxonomies that were too rigid or complex to allow interoperability and customisation. While Smart-CARE offered a lightweight, extensible structure for integrating heterogeneous data, it did not provide a clear, structured way to specify how detected situations of interest should trigger and organise user actions. This RQ therefore examines whether extending the Smart-CARE taxonomy with BPM-based representations enables more precise and understandable definitions of contextualised actions and improves personalisation across heterogeneous IoT domains. Beyond the taxonomy layer, it also assesses whether embedding BPM in the Smart-CARE architecture supports the scalable definition, reuse, and evolution of personalised contextualised actions as users and contextual parameters grow, by leveraging modularity (e.g., reusable subprocesses) and parameterisation to reuse action logic while allowing user-specific adaptations in multi-user scenarios.

RQ2. To what extent does BPM support the definition of the rationale behind CEP event patterns and the representation of their interdependencies? The logic behind event patterns in many existing CEP-based systems—i.e., how they relate to each other and the conditions under which they should be evaluated— is implicitly encoded in the event schema and event pattern definitions. This may hinder the comprehensibility and maintainability of CEP-based systems and can complicate validation and development [15]. To address these limitations, BPM offers mechanisms for modelling dependencies, information flows and decision logic. This RQ examines whether BPM can effectively model the logic and interdependencies of CEP event patterns, thereby improving comprehensibility and transparency, supporting evolution, and facilitating collaboration between developers and domain experts.

RQ3. Can the integration of BPM into the Smart-CARE context-aware software architecture improve the overall management of contextualised actions and event patterns in terms of orchestration, traceability, and maintainability? Although CEP-based architectures excel at event correlation, they are limited in their ability to orchestrate the behaviour of complex systems, document designs, or manage evolution over time [16]. As systems become more complex, these limitations can have a negative impact on usability, clarity and maintainability. This RQ explores whether integrating BPM into the Smart-CARE context-aware architecture can improve the management of event patterns and contextualised actions. In particular, it examines whether BPM models can provide explicit traceability between detected situations of interest, CEP event pattern logic, and the contextualised actions they trigger, while providing a high-level, process-oriented view that complements CEP logic and preserves the performance advantages of CEP.

RQ4. How effective is BPM in supporting runtime updates to CEP event patterns, based on detected situations of interest, while preserving real-time performance constraints? The requirements and needs of domains in IoT environments are constantly changing. Consequently, systems must be able to adapt their behaviour at runtime in response to evolving context and user needs. Although CEP engines support dynamic rule updates, these are usually carried out manually and are not explicitly managed as part of higher-level system logic. BPM could provide a process-level mechanism to explicitly represent and govern such adaptations. This RQ explores whether BPM-based approaches can support the dynamic adaptation of CEP event patterns in response to detected situations of interest, and whether such adaptations can improve system adaptability and responsiveness in real IoT scenarios without introducing unacceptable latency or overhead.

1.3. Contributions

In response to the research hypothesis and questions outlined above, this paper introduces BPSmart-CARE, a BPM-enhanced version of the Smart-CARE context-aware real-time framework for smart environments. The main contributions of this work are as follows. (1) An extension of the taxonomy proposed in [7] to support the description of contextualised user actions using BPM. (2) An enhanced context-aware software architecture, supported by BPM, which adds new functionalities to the architecture in [7]. Specifically, BPM enables (2.a) the definition of CEP event pattern rationale, as well as the interdependencies between event patterns; (2.b) the definition and management of contextualised user actions and (2.c) the real-time updating of CEP event patterns logic based on detected situations of interest. (3) A case study illustrating how CEP event patterns and contextualised user actions can be represented using BPM, including runtime event pattern updates when particular conditions are met. These contributions are evaluated through

usability studies, the implementation of CEP event patterns in a practical exercise with and without BPM support and CEP and BPM performance evaluation with different workloads, in order to assess applicability, scalability, and real-time feasibility in smart environment scenarios.

The remainder of the paper is structured as follows. Section 2 presents the theoretical background of the proposal. Section 3 presents the related work in IoT data processing and correlation, with an emphasis on CEP, BPM-based, and CEP-BPM approaches. Section 4 describes the BPM-enhanced taxonomy and software architecture. Then, Section 5 explains in detail the design of a BPM model for a case study using the proposed software architecture. Section 6 reports a developer-centred evaluation while Section 7 reports a system-centred evaluation. Finally, Section 8 discusses the results obtained in this proposal, while Section 9 presents the conclusions and future work.

2. Theoretical background

In order to provide the conceptual foundations for the approach proposed in this paper, this section introduces CEP, which constitutes the core technological pillar of the Smart-CARE framework, and BPM, which is presented as a complementary paradigm that can enrich Smart-CARE. Together, they provide the theoretical basis required to motivate their integration and to contextualise the subsequent analysis of related work.

Context-aware systems aim to adapt their behaviour according to the current context, which may include environmental conditions, user state, location, time, and other situational factors. In this work, context-awareness is operationalised through the detection of situations of interest derived from event streams and contextual conditions, and through the execution of contextualised actions that respond to those situations of interest.

In this context, CEP can be positioned within the broader landscape of real-time stream processing technologies commonly used in IoT and smart environments. Accordingly, in IoT and smart-environment settings, real-time analytics is commonly supported by stream processing frameworks (e.g., Apache Flink, Apache Spark Streaming, and Kafka Streams) as well as by CEP engines such as Esper. While stream processing frameworks typically provide general-purpose operators over continuous dataflows, CEP focuses on the specification and detection of event patterns over event streams.

CEP [17] is a technology designed for real-time analysis of large volumes of data from various sources. In CEP-based systems, incoming data items are treated as simple events, which may originate from IoT sensors, wearable devices, or information-system records. The detection of situations of interest (also known as complex events) depends on the prior configuration of event patterns. These event patterns act as rules that inspect the incoming flow of simple events, searching for specific conditions and sequences of the same or different types of simple event, with possible temporal constraints between them. Thus, unlike traditional databases, CEP can identify relevant event patterns in real time by continuously correlating input data streams. This continuous evaluation is performed by a CEP engine, which serves as the computational core where event patterns are deployed and executed at runtime. There are multiple CEP engines available. Among them, Smart-CARE adopts the Esper CEP engine [18] due to its high performance [19,20] and technological maturity. In addition, Esper provides an expressive Event Processing Language (EPL), which supports flexible and efficient rule definition.

While CEP provides powerful mechanisms for event correlation and situation of interest detection, it does not, on its own, provide explicit constructs for defining the contextualised actions that such situations of interest should trigger, nor for graphically representing CEP event pattern logic. BPM [21] is a discipline that aims to model, execute and manage the behaviour of information systems through the use of explicit representations of business processes. Unlike implementations based exclusively on code, BPM provides mechanisms for describing the flow of activities, decisions and interactions. This approach is particularly valuable in environments where system logic must be understandable, adaptable and maintainable, since it provides an explicit description of control flow, responsibilities and decision-making processes.

In practice, these BPM capabilities are typically realised through the BPMN standard [22], which provides visual modelling constructs for process representation. This specification provides a graphical notation that allows key elements, such as activities, events, gateways and message exchanges, to be represented. This facilitates a shared understanding between technical and management profiles, providing a high-level abstraction. Additionally, these models have well-defined executable semantics that enable direct interpretation by BPM engines, which manage process lifecycles at runtime.

Based on these foundations, the following section reviews related work on CEP, BPM, and their integration in IoT and smart environments.

3. Related work

This section reviews related work in three strands. First, it summarises comparative studies of stream processing frameworks and CEP engines for real-time analytics in IoT and smart environments, and discusses limitations of CEP-centric approaches in supporting personalised responses. Second, it reviews BPM-based proposals for orchestrating system behaviour in IoT settings and surveys CEP–BPM approaches that aim to bridge event detection and process orchestration. Finally, it outlines complementary organisational and model-driven perspectives, including recent advances in intelligent and knowledge-based BPM. Finally, it discusses broader research on adaptability and process reconfiguration

3.1. CEP and stream processing for real-time analytics in IoT and smart environments

Recent work has evaluated and compared stream processing frameworks (e.g., Apache Flink, Apache Spark Streaming, and Kafka Streams) and CEP engines such as Esper for real-time analytics in IoT and smart environments [23]. Although Apache Flink achieves low latency and high throughput, its scalability is limited, whereas Apache Spark offers excellent scalability at the cost of higher latency. Kafka Streams demonstrates high performance [24] depending on the specific application context, while CEP, and specifically Esper CEP [25] outperforms other systems when processing and correlating event streams, both in single-core [19] and distributed settings, although its distributed implementation faces challenges in sharing information across different complex event processing engines [26].

Nevertheless, CEP engines like Esper primarily focus on processing, correlating and detecting situations of interest, and their standalone use is limited in providing contextualised decision-making and personalised adaptation in complex scenarios. Consequently, several studies have explored CEP-based approaches in smart environments and IoT, yet they still provide limited support for specifying how detected situations of interest should lead to personalised responses. For example Daum et al. [27] proposed using CEP for event detection, yet their approach does not address the execution of contextualised actions, referred to as 'reactions'. Similarly, Li et al. [28] suggested integrating CEP with BPMN by adding a new activity type to BPMN that allows non-technical users to define CEP event patterns and their technical details in an abstract manner. This new activity type facilitates communication with the CEP engine, sending input and receiving output to continue the BPMN process. However, this approach does not integrate data from different application domains, nor does it consider dynamically updating event patterns so that they adapt to changing contexts. Additionally, the graphical representations provided are highly simplified and abstract the underlying CEP event patterns. This can improve accessibility for non-expert users, but it may also reduce the transparency of the underlying event-detection logic. Providing personalised responses based on stakeholder requirements remains a complex challenge, as it depends on various contextual factors, such as user location, profile, environmental conditions, and the state of other interacting devices. Therefore, effective context-aware systems require mechanisms that can connect situations of interest detected by CEP with appropriate contextualised actions. These actions must be able to adapt flexibly to evolving circumstances while maintaining alignment with the objectives of IoT systems.

3.2. BPM-based orchestration and CEP-BPM integration approaches

While some proposals combine CEP with BPM primarily for the abstract definition of event patterns, several studies have explored using BPM as a central mechanism for analysing, designing, implementing and continuously monitoring processes within different IoT systems and smart environments. For instance, BPM models have been employed to orchestrate healthcare processes, such as monitoring patients with hypertension [29], and to integrate various sensor types with actuators for automated system behaviour [30]. Similarly, intelligent distribution centres have used BPM to coordinate system operations without human intervention [31]. These proposals illustrate that BPM provides key benefits for context-aware systems, including traceability of actions, standardisation of procedures, and the ability to adapt processes dynamically according to context. Moreover, the graphical and formal nature of BPMN facilitates the representation of system behaviour in a way that is understandable to both domain experts and the system's developers [32,33]. While these proposals demonstrate BPM's potential to manage contextualised actions in IoT and smart environments, they lack integration with real-time event processing. This means they fail to exploit the adaptive capabilities provided by CEP, leaving opportunities for enhancing context-awareness and dynamic responsiveness in heterogeneous IoT environments.

More recently, research has focused on combining CEP and BPM in order to detect situations of interest and orchestrate the corresponding contextualised actions. Early efforts focused on enhancing BPM models with real-time event monitoring, exemplified by Ruhkamp et al. [34], who proposed executing declarative BPM processes by mapping constraints directly to CEP queries. This approach highlighted that integrating CEP with declarative BPM models allows not only the detection of critical situations of interest but also guides human operators in decision-making during system failures, bridging the gap between process logic and event-driven execution. However, its primary focus is on monitoring and verifying constraints, rather than on how CEP logic can be maintained, adapted or customised for different users or contexts. In line with this reasoning Janiesch et al. [35] proposed modelling CEP engine operations within BPM in terms of event pattern deployment and the handling of simple input events. This approach further enhances the flexibility and responsiveness of business processes by establishing a close relationship between event processing and process execution. However, CEP event patterns are treated as technical artefacts embedded in process models, offering limited support for semantic transparency or the dynamic adaptation of event patterns based on contextual changes.

Other proposals have focused on scalability and distribution aspects of CEP–BPM integration. Fardbastani et al. [36] proposed a decentralised CEP mechanism integrated with BPM, which distributes the processing of monitoring rules and event streams across multiple nodes, enabling scalable and adaptive process monitoring. Although this approach addresses concerns relating to performance and scalability, it places little emphasis on usability or maintainability, nor on ensuring that event-detection logic is understandable to stakeholders beyond system developers. Domain-specific frameworks have also leveraged CEP–BPM integration to manage complex IoT environments. For example, Gonçalves et al. [37] proposed the REFlex Water framework, which integrates CEP and BPM to manage water supply systems in real time. BPM is used to model system behaviour and control unforeseen situations of interest with high reliability. While it is effective within its domain, the framework is tightly coupled to a specific application context and does not address cross-domain data integration or user-specific adaptation. More recent proposals aim to improve interoperability and adaptability. Valderas et al. [38] proposed a microservice-based architecture integrating CEP and BPM, which uses ontologies and event-based communication to improve decoupled interoperability and adaptability in heterogeneous IoT environments. Although this approach improves modularity and semantic integration, it does not explicitly explain how CEP event patterns can be represented,

understood and developed by users at a process level. Similarly, Bazán et al. [39] made a conceptual proposal suggesting the integration of CEP and BPM in Industry 4.0. The proposal emphasises the potential benefits of incorporating CEP into BPM, such as increased agility, context-sensitivity, and responsiveness to IoT devices. However, their work remains highly abstract and does not offer specific solutions for event pattern management, personalisation or runtime adaptation.

3.3. Complementary organisational and model-driven perspectives on adaptability

Beyond these technical integrations, several complementary research strands motivate the need for adaptive and process-oriented systems from organisational and model-driven perspectives. In addition to technical CEP and BPM-based proposals, recent research has also analysed business-oriented and organisational factors that justify the need for intelligent and adaptive process-oriented systems. In this line, Changalima et al. [40] examine and empirically demonstrate how technological absorptive capacity influences the business performance of SMEs. While this work focuses on why technological absorptive capacity is essential for business outcomes, it also reinforces the need for concrete process-oriented architectures capable of operationalising digital and technological innovation in complex, data-intensive environments.

In parallel, recent research has increasingly emphasised the importance of intelligent, adaptive and knowledge-based approaches in BPM. In this line, Cano-Marín [41] highlights the transformative potential of generative artificial intelligence (GenAI) in business environments, emphasising its ability to support informed decision-making and enhance adaptability and productivity through the exploitation of diverse data sources. While these approaches do not explicitly address BPM or real-time event processing, they reinforce the need for flexible, semantically transparent process architectures capable of evolving based on contextual information. This perspective is complemented by model-based engineering approaches, which have been proposed to bridge the gap between unstructured data from intelligent environments and business process models. For instance, El Kodssi et al. [42] suggest using a model-based architecture alongside machine learning techniques to convert sensor data into a form that is suitable for process mining. This highlights the significance of formal models in ensuring the maintainability, usability and semantic clarity of IoT-enabled BPM systems. Furthermore, the emergence of GenAI-based solutions such as ProcessGPT [43] further illustrates the trend towards adaptive, data-centric BPM. ProcessGPT uses generative models and process knowledge graphs to automatically generate, adapt and maintain process models, thereby improving usability and supporting continuous evolution. However, these approaches remain largely disconnected from real-time event correlation mechanisms, leaving open the challenge of systematically linking event detection with executable process logic in context-aware IoT systems.

Taken together, these works highlight the need for integrated frameworks that link real-time event correlation, cross-domain contextual data and executable process orchestration. From complementary technical and conceptual perspectives, research lines have addressed system adaptability and process reconfiguration. Technical approaches based on Service-Oriented Architecture (SOA) and Business Intelligence (BI) such as the proposal made by Mekimah et al. [44], focus on enabling dynamic process reconfiguration and visualisation through RESTful services, supporting flexible integration of system components across domains. However, these approaches often operate at an architectural or service level, without explicitly integrating real-time event correlation mechanisms into executable process logic. In Jundt et al. [45] from a human-centric perspective, adaptive performance research analyses how individuals and teams adjust their behaviour in uncertain and evolving environments, highlighting that effective adaptability requires

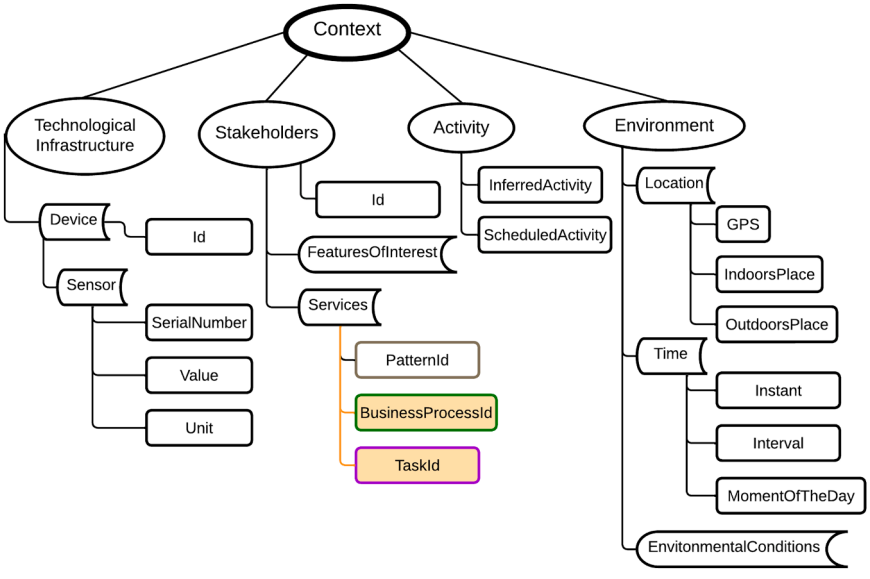


Fig. 1. Taxonomy modification introducing BPM processes representing the CEP pattern behaviour (BusinessProcessId) and the corresponding actions (TaskId) triggered when the complex event (situation of interest) is detected.

not only technical mechanisms but also support for real-time decision-making. While these studies provide valuable insights into adaptive behaviour, they do not address how such adaptability can be operationalised through executable, event-driven process models. Other BPM-oriented proposals, such as empirical analyses of BPM capabilities [46] and the concept of Processes-of-Processes (PoP) [47], emphasise hierarchical and evolutionary process structures as a means to support long-term system evolution. Nevertheless, these approaches remain largely conceptual or operate at higher abstraction levels, leaving open the challenge of systematically linking real-time CEP logic with concrete, executable BPM models. These proposals highlight that, while existing approaches address adaptability from architectural, organisational or conceptual perspectives, they lack integrated frameworks combining real-time event processing, cross-domain contextual data and executable process orchestration.

4. BPSmart-CARE

In this section we introduce BPSmart-CARE, our BPM enhanced Context-Aware Real time framework for smart environments. The fist subsection explains how the Smart-CARE taxonomy has been extended to support the description of contextualised user actions using BPM. The second subsection describes how the Smart-CARE software architecture has been extended to integrate a module for defining and managing contextualised user actions using BPM as well as to define the logic for real-time CEP event pattern updates based on detected situations of interest.

4.1. BPM-enhanced taxonomy

A detailed description of the Smart-CARE taxonomy categories and attributes is provided in [7]; additionally some application examples in the domain of water supply network management can be found in [48]. Building on the taxonomy in [7], we extended it to more precisely define contextualised user actions by incorporating attributes that describe the actions to be taken when a situation of interest to the user is detected (contribution 1). Fig. 1 shows the resulting taxonomy, with the new attributes introduced in this paper shaded in orange.

As shown in the figure, the taxonomy has been extended with two new attributes, *BusinessProcessId* and *TaskId*, which replace the previous *Action* attribute. As a result, the taxonomy now allows the description of contextualised user actions using BPM. *BusinessProcessId* enables the identification of the BPM process that represents the behaviour of the CEP event patterns deployed in the CEP engine, as well as the information exchange with it, in which situations of interest are detected, as explained later. Early detection of a situations of interest enables intelligent systems to improve their responses and the services provided to users. *TaskId* allows the identification of the BPM process that specifies the actions to be executed when a particular situation of interest is detected. It is important to note that, to adapt the previous proposal to the use of BPM, only minor modifications to the taxonomy are required to

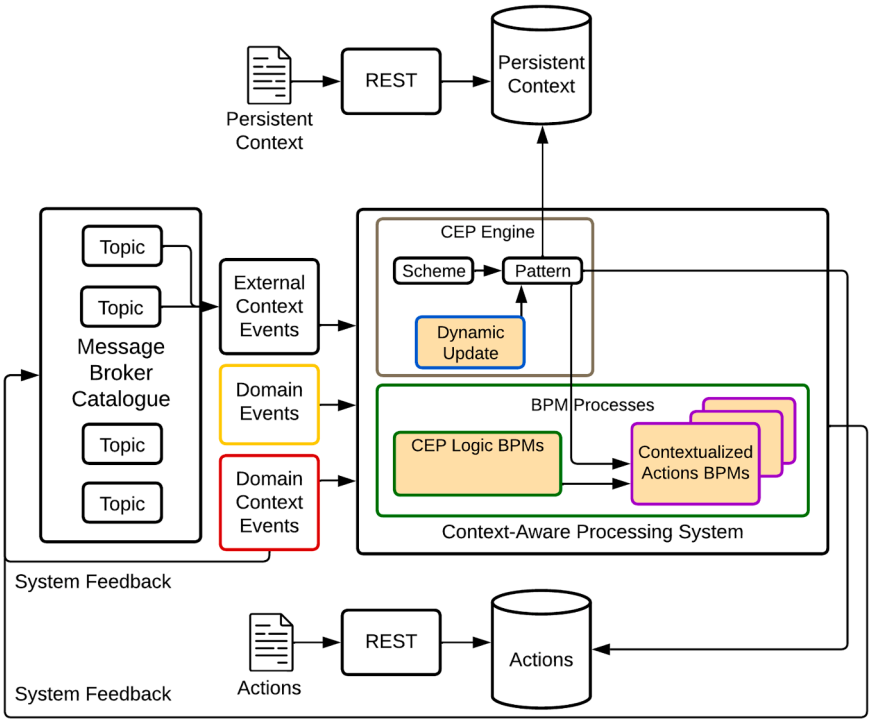


Fig. 2. Software architecture integrating CEP and BPM, highlighting the BPM processes used to represent CEP pattern logic and to manage contextualised actions, as well as highlighting a new CEP module that enables real-time updates of event patterns in context-aware IoT systems.

enable a more detailed representation of contextualised actions. No further modifications are needed to represent event patterns.

4.2. BPM-enhanced software architecture

Fig. 2 shows the software architecture proposed in [7], including the new elements introduced in this paper shaded in orange (contribution 2). The existing elements of the architecture are briefly described below to keep the paper self-contained, with additional detail provided for the new components:

- Three types of input events are considered: Domain Events (box with yellow border), Domain Context Events (box with red border) and External Context Event. As explained in [7], Domain Events are those that occur within a particular domain. For example, in the domain of an elderly person's home, these include all sensor events generated within that home. Domain Context Events are those generated by interaction between domains; for example when the elderly person interacts with another domain, such as the electricity provider, while consuming electricity. External Context Events are those that are not associated with a specific stakeholder, such as weather conditions or ambient temperature.
- Context-Aware Processing System. This component represents the core of the software architecture proposed. It consists of a CEP engine (box with brown border) and multiple BPM processes (box with green border), which represent the logic of CEP event patterns, their interdependences and their real-time update, as well as the definition of contextualised actions. The CEP engine has different schemas and event patterns deployed that trigger the execution of the corresponding Contextualised Actions BPMs (boxes with purple border). A situation of interest detected by a set of CEP event patterns can lead to one or multiple contextualised actions, depending on contextual conditions; in some cases, only a single action may be associated with each situation of interest. Therefore, when a complex event is detected in the CEP engine, the associated BPM process specifies how the systems should react according to the user's personal configuration parameters. In addition, the Context-Aware Processing System applies real-time updates (box with blue border) to selected CEP event patterns based on situations of interest detected at runtime, enabling dynamic adaption to changing conditions. Please note that although the system is implemented by receiving simple events in both the CEP engine and the BPM processes, this has been done for illustration purposes only. While the initial design considered routing events through BPM and exchanging complex events with the CEP engine, this integration is optional; in some deployments CEP can be used only for situation of interest detection, and BPM can be used to represent event patterns graphically and execute contextualised actions without direct CEP–BPM event exchange. Section 4 illustrates how separating simple events and complex events in different BPM processes clarifies the implementation of contributions 2.a and 2.c.
- Context and actions databases. The architecture includes two databases that store persistent context information and user-defined actions. The persistent context database stores stakeholder-related information, while the actions database stores the actions defined for each user.
- REST services. REST services provide interfaces that allow users to insert and retrieve information from the databases.
- Actions and persistent context represent the information that users stored in the databases through the corresponding REST interfaces.
- Message broker. The architecture includes a message broker with a catalogue of topics that enable information sharing across domains using a publish/subscribe mechanism.

4.3. BPM-smart CARE operation

The proposed software architecture works as follows:

- Information generated by smart sensor and intelligent systems enters the system as one of the defined event types, encoded in JSON format and structured according to the taxonomy. As previously said, Domain Events occur within a particular domain and are processed only by the corresponding Context-Aware Processing System (i.e. sensor data from an elderly person's home should not be shared externally). Domain Context Events result from interactions between domains, and may be shared both within the particular domain system and through message broker, as they may be relevant to other domains (e.g., electricity consumption data could be useful to the council department responsible for energy consumption). Finally, External Context Events are already available in the message broker catalogue and are forwarded only to the appropriate Context-Aware Processing Systems.
- All events, regardless of their type, are sent to the Context-Aware Processing System, specifically to the CEP engine, which processes them based on the deployed event patterns, correlates them and detects situations of interest or complex events. In parallel, events are sent to the BPM processes, where CEP event pattern execution and real-time updates are explicitly represented.
- For certain user-defined situations of interest, CEP event pattern updates can be dynamically performed at runtime.
- Persistent context information of the stakeholder, as well as user-defined actions are introduced into the system via REST services, encoded in JSON format according to the taxonomy.
- Persistent context and custom actions are stored in their respective databases. The engine queries the persistent context database to detect event patterns involving multiple users, while the actions database is queried only when a complex event is detected, allowing the system to retrieve the appropriate contextualised action.
- Once a situation of interest is detected, the relevant information, along with the associated contextualised action, are sent to the Contextualised Action BPM, where the action is processed and executed. All actions must be predefined by users within the corresponding BPM processes.

- Finally, the information generated by the CEP engine can be shared with other domains or fed back into the system through the message broker topics.

To facilitate comprehension of the proposal, the following section presents a case study detailing the BPM design and the data flows that occur during execution.

5. Case study

In this section, a case study is used to provide a concrete and reproducible illustration of how the proposed framework operates end-to-end. It is explained how the previously defined framework is used to design a BPMN process that fulfils four main objectives. Firstly, these BPM processes represent the actions performed by the different stakeholders, which are detected by their intelligent devices. Secondly, they define how individual events received by the CEP engine are evaluated and processed. Consequently, the BPM processes represent the sequence of interdependent event patterns deployed in the CEP engine, as well as the input required for each event pattern. This enables the identification of the user actions that are triggered by the detection of specific complex events. Thirdly, BPM processes manage the contextualised actions to be executed in response to detected situations of interest. Once a complex event is detected, the BPM process defines how the system should react according to the user's personal configuration parameters. Finally, BPM processes define real-time updates to CEP event pattern logic based on specific runtime situations of interest. Thus, during the execution of the event pattern sequence in the CEP engine, the BPM process may represent runtime updates to certain event patterns when specific conditions are met, enabling dynamic adaptation. At the same time, BPM reduces the complexity of representing CEP event patterns by providing a graphical notation that facilitates the understanding of the CEP engine behaviour, the relationships between event patterns, and the conditions required to detect situations of interest.

5.1. Case study description

To illustrate the proposed approach, an Ambient Assisted Living (AAL) case study was selected. This domain was chosen because it is representative of context-aware IoT systems that must combine heterogeneous sensing, real-time situation of interest detection, and personalised responses, often under safety-critical constraints. Moreover, the scenario naturally involves temporal conditions and interdependent event patterns, making it suitable to demonstrate the proposed CEP–BPM modelling and runtime orchestration capabilities. Although the case study is AAL-oriented, the modelling and integration principles are domain-agnostic and can be transferred to other smart-environment domains.

More in detail, the case study involves an elderly person living alone. This person performs a daily activity: he/she collects the rubbish at home, takes it outside to a street bin, and then returns home. Based on this simple scenario, several event patterns are processed by a CEP engine to detect anomalous situations of interest. Specifically, the system monitors whether the user fails to return home within the expected time after disposing of the rubbish. This may indicate an abnormal situation of interest, such as a fall, disorientation or becoming lost. Such situations of interest are particularly critical for elderly individuals, especially those with cognitive impairment or mental health conditions, as these factors increase the risk of incidents during routine activities.

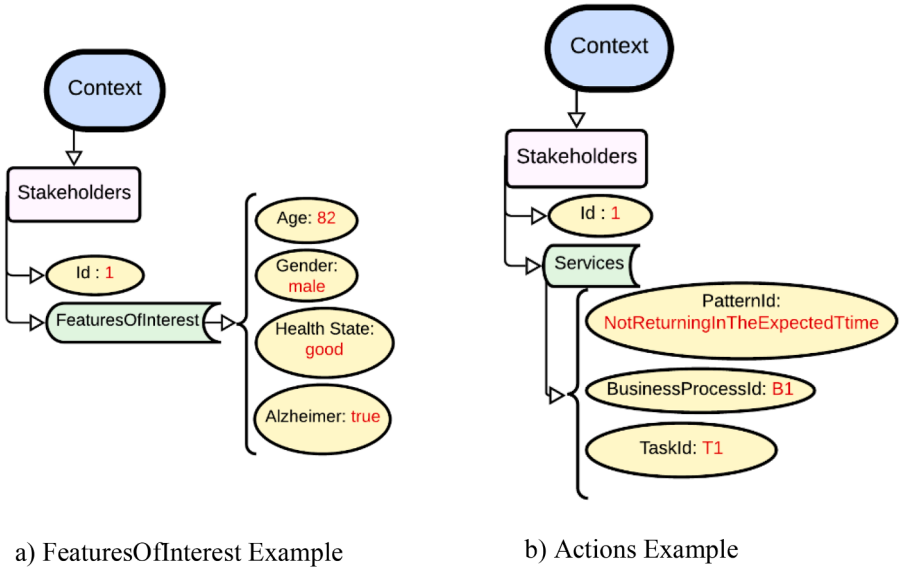


Fig. 3. Taxonomy usage examples illustrating how features of interest and user-specific contextualised actions are structured, and how the latter are associated with BPM processes.

5.2. Case study taxonomy

Before the detection of any situation of interest can take place, the features of interest and the contextualised actions associated with the user must be defined. Fig. 3 shows two examples of how the taxonomy can be used to structure these data. Fig. 3(a) shows how the *FeaturesOfInterest* are introduced into the system, while Fig. 3(b) illustrates how contextualised actions are defined using the new taxonomy attributes introduced in this paper. It is important to note that, as service customisation is user-specific, each assisted person in the AAL scenario has an individual BPM process that defines all associated event patterns. Consequently, each user's BPM includes the event patterns and the corresponding contextualised action processes, as users may require different operations to be performed for the same type of situation of interest.

5.3. Case study software architecture component and interaction flows

Fig. 4 shows the core components of the Context-Aware Processing System and illustrates a simplified information flow among the elements of the proposed architecture. Note that, prior to sending data to the system, the event patterns and input data schemas had to be deployed in the CEP engine, and their logic had to be represented in BPM models. Once deployed, data from various information sources (i), such as a weight sensor in the user's waste bin, magnetic and motion sensors on the house door, and beacon devices installed on the street waste container, are sent in parallel to the CEP engine (ii) and the BPM processes (iii). The CEP engine correlates simple events and detects situations of interest using predefined event patterns. When applicable, an alert (iv) is generated and sent to the Contextualised Actions BPM (vi). Besides, when a context change is detected by the CEP engine, a real-time update to a particular CEP event pattern might be performed (v). In parallel, the BPM processes, through the CEP Logic BPM, represent the information received from the sources and the execution of CEP event patterns. When an alert reaches the Contextualised Actions BPM, it is evaluated using the user's contextual information to determine the most appropriate action (vi). Finally, the action is executed, which may include notifying caregivers, activating an alarm, or sending information to emergency services. These interactions are modelled in greater detail in the BPM diagram shown in Fig. 5 where four pools can be distinguished as *Users, CEP Engine, Trash Company* and *Actions*, with the following functions:

- *User* pool (yellow shaded). This pool represents the different actions performed by the user and detected by intelligent devices or inferred from the user's behaviour. This pool is divided into two actors. The first one corresponds to actions performed by the user inside the home, *Home* (yellow shaded), which are detected by sensors and represented as domain events. These BMP tasks are modelled as receive tasks, since the information is delivered via a message broker. Once detected, the user's behaviour can be inferred. The user also interacts with another domain by disposing of rubbish, *Trash* (red shaded). In this case, a beacon network detects the user interaction with the bin, resulting in the generation of a domain context event.
- In this way, when the household bin becomes full, a sensor detects that the user is collecting the rubbish, allowing the inference of the activity of preparing to leave the house. Then, magnetic and motion sensors on the door then detect the action of leaving the house, which leads to the inference of the activity of walking. Sensors and beacons at the bin identify the action of disposing the rubbish, followed by the inference of walking back home. Finally, door sensors confirm that the user has returned home.
- *Trash Company* pool (red shaded). This pool models how the Trash Company periodically sends updated information about the status of the user's container in parallel to the CEP engine and BPM processes. This information is stored in the persistent context database for future queries.

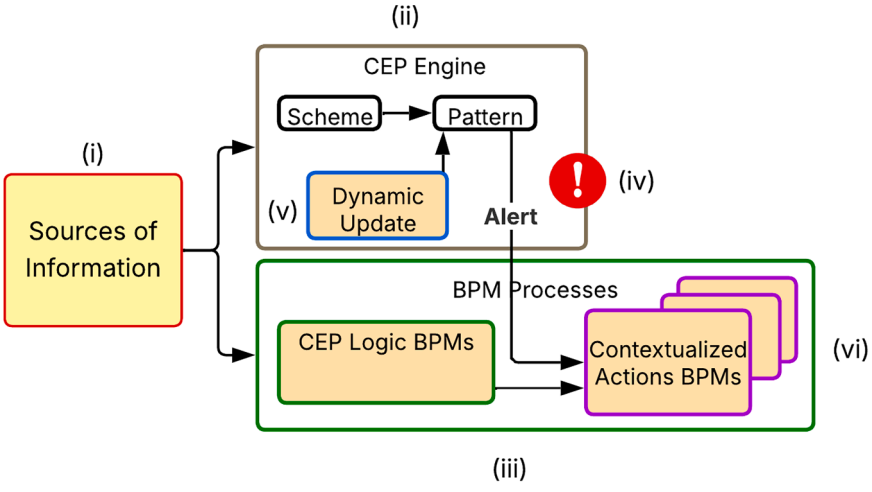


Fig. 4. Information flow within the Context-Aware Processing System, illustrating parallel event handling by CEP and BPM engines, situation detection, dynamic CEP pattern updates, and the triggering of contextualised actions.

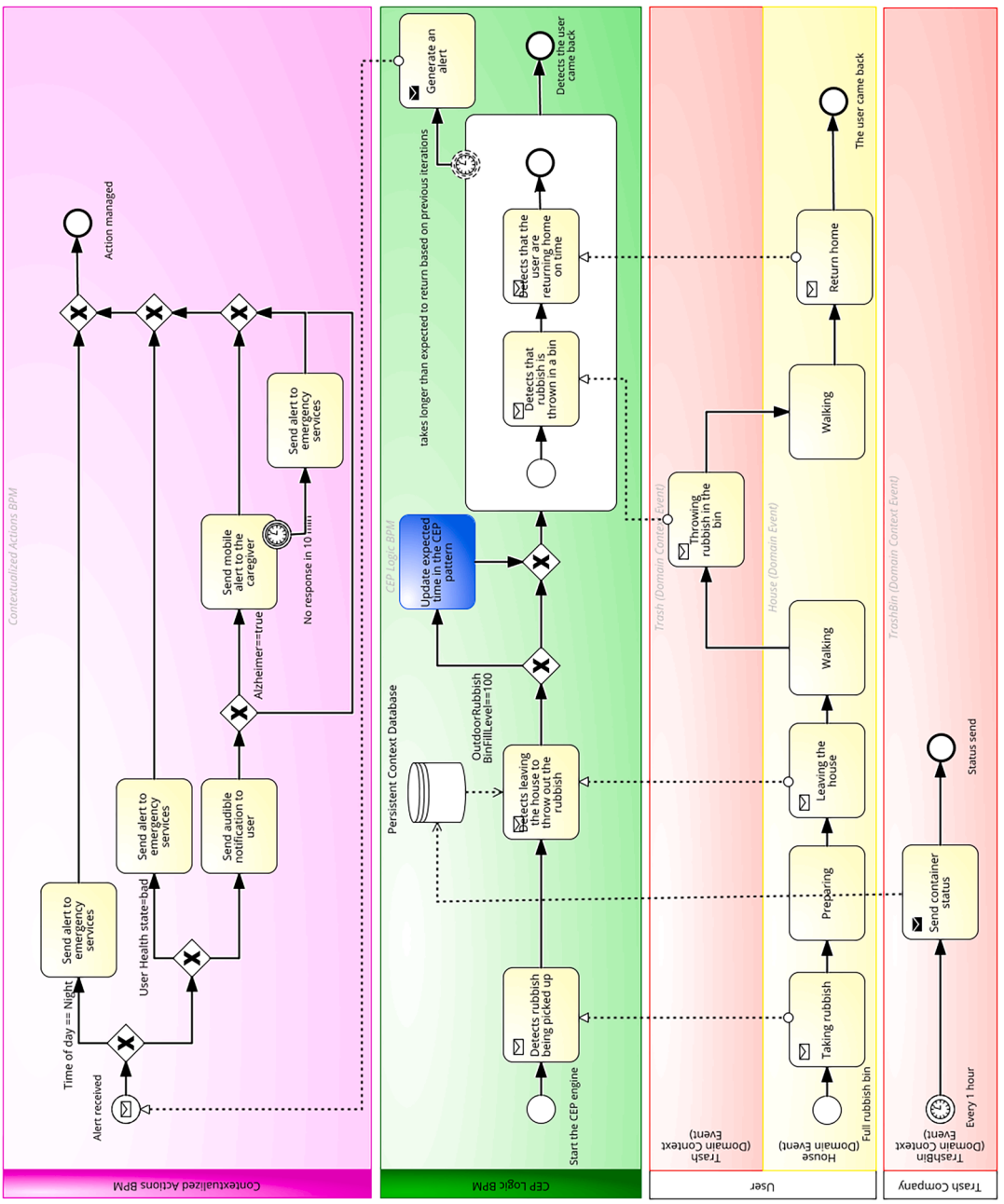


Fig. 5. BPM model detailing user interactions, CEP logic representation, and contextualised action execution in the case study.

- *CEP Logic BPM* pool (green shaded). All the actions performed by the user and detected by the sensors are sent to the CEP engine, and represented in the BPM model as messages sent to the *CEP Logic BPM*. This reflects the architecture, in which sensors data are sent via a message broker to both the CEP engine and BPM processes (contribution 2.a). When multiple simple events are received, the CEP engine correlates them to infer the user's action. In this case study, 5 are deployed: detecting rubbish collection, detecting that the user leaves the house to throw out the rubbish, updating the expected return time, detecting rubbish disposal and detecting the user's return to the house or generating an alert if the user does not return within the expected time. In parallel, the event pattern *detecting that the user leaves the house to throw out the rubbish* query the persistent context database to check the status of the user's usual bin. If the bin is full when the user leaves the house to throw away the rubbish, the expected return time is updated dynamically (blue shadow), since he/she will have to walk to a more distant bin (contribution 2.c). Then, the CEP engine starts a timer based on this updated value. If the user does not return before the timer expires an alert is generated, as explained below.
- *Contextualised Actions BPM* pool (purple shaded). When the CEP engine generates an alert (e.g. the user has not returned home in the expecting time after disposing the rubbish), it is sent to the *Contextualised Actions BPM,* where all possible user actions are defined (contribution 2.b). For instance, if an alert occurs at night or the user is in poor health, emergency services may be notified. Otherwise, an audible alert may be sent to the user, followed by notification to a caregiver if the user has Alzheimer's disease. If the caregiver does not respond within a defined period, emergency services are contacted. The exact execution details are retrieved from the contextualised action database.

This case study illustrated how the proposed architecture combining BPM with CEP can effectively detect situations of interest and trigger contextualised actions. Starting from a common daily activity, the example illustrates how the system components collaborate to correlate data from multiple domains, provide personalised services, and adapt to changing conditions. Furthermore, the feasibility of modelling both system behaviour and dynamic CEP event patterns updates using BPM has been demonstrated, thereby facilitating the understanding of data and event flows for developers and domain experts.

6. Developer-centred evaluation: usability and task performance

This section reports the results of an empirical study conducted to evaluate the usability and task performance of the proposed approach from a developer perspective. The examines how developers understand the rationale and dependencies of CEP event patterns when represented using BPM, how easily they can modify and adapt CEP logic through BPM processes in response to changing situations of interest, and whether BPM facilitates implementation and maintenance of CEP event patterns.

6.1. Context, participants and materials

The evaluation was conducted as a controlled classroom experiments within a university course on software engineering and intelligent systems during the final year of Computer Science and Engineering bachelor's degree. A total of 16 students participated in the study. However, as the evaluation was conducted over several sessions and 2 participants did not attend the final session, they were excluded from the final questionnaire, resulting on 14 students.

All students received the same training and completed the same exercises, and during the first two sessions of the experiment they all completed the same usability questionnaires on BPM and CEP, respectively. In the third session, as explained in more detail below, the students were divided into two groups to complete a practical CEP exercise with or without BPM support. To ensure comparable skill levels across experimental conditions, participants were assigned to groups based on academic performance in the course. Students were first ranked according to their grades and then distributed alternately between groups, resulting in balanced groups with similar average academic performance. This strategy mitigated the influence of prior knowledge and individual competence on the results.

Participants completed the experiment using their own laptops. All questionnaires and materials were distributed through the course's virtual campus, ensuring centralised data collection and time tracking. All the students had access to all course materials during the experiment. All questionnaires completed by the participants, the practical exercises proposed during the study, as well as the corresponding results obtained from both the questionnaires and the practical exercises, are provided as part of the supplementary materials (see Supplementary Materials section).

6.2. Experimental procedure and metrics

The study was conducted over three separate days:

Day 1. BPM usability questionnaire. After learning BPM concepts and completing several practical exercises using the Signavio Academic Initiative platform [49], students completed a questionnaire assessing their perception of BPM usability (see Supplementary Material section). The aim was to assess their understanding of BPM as a modelling tool. The metric used for this purpose was the System Usability Scale (SUS) [50].

Day 2. CEP usability questionnaire. After learning CEP concepts and completing several practical exercises, students completed a questionnaire assessing their experience with Esper CEP version 9.0.0 [51], focusing on the usability and ease of use (see Supplementary Materials section). The questionnaire assessed participants' understanding of using CEP for event detection and processing. Again, the metric used was the SUS.

Day 3. Practical exercise implementation and post-experiment questionnaire. All participants had to implement the same practical

exercise with several predefined situations of interest using CEP event patterns; in this case, the scenario involves monitoring temperature and humidity to detect potential fire situations and trigger alerts to emergency response systems. Both received a natural language description of the scenario to be implemented; however, while one group received a BPMN model describing the logic and dependencies between event patterns (*BPM group* from now on), the other group implemented the patterns without BPM-based support (*non-BPM group* from now on). All the materials provided to the students can be found at the supplementary material (see Supplementary Materials section). The final implementation of the practical exercise was evaluated using two metrics: effectiveness and efficiency. On the one hand, the assessment focused on analysing how effectively participants implemented CEP event patterns derived from a practical exercise. In this context, effectiveness of the implementation was evaluated based on correctness and completeness of CEP event patterns as reflected by their score obtained in the practical exercise. This ranged from 0 (incorrect implementation) to 10 (fully correct implementation). Six CEP event patterns were evaluated: the first two event patterns weighted at 1 point each, while the remaining four at 2 points each due to their higher complexity. Scores were assigned as follows: 100% for correct detection of all situations of interest, 50% for partial detection, and 0% if most situations of interest were not detected. The total effectiveness score was computed as the sum of the six event pattern scores. On the other hand, efficiency was evaluated based on the time taken to complete the task, it was calculated as the ratio of the effectiveness score and the time required to complete the task, providing insight into implementation speed. Besides, after completing the practical exercise, participants filled out a questionnaire assessing the perceived difficulty of implementing CEP event patterns, either with or without BPM support.

6.3. Results

This subsection presents the results obtained from the initial usability questionnaires, the practical exercise implementation and the post-experiment questionnaires (see Supplementary Materials section). The analysis of the practical exercise implementation compares the results of participants using BPMN support with those without BPM guidance. It also analysis the results of the usability evaluation of BPM and CEP, based on data collected via the questionnaires.

6.3.1. Initial questionnaires: perceived usability of BPM and CEP

The initial questionnaires captured participants' perceived usability of BPM and CEP using the SUS, one of the most widely adopted standardised instruments for usability evaluation. SUS provides a reliable method for assessing users' subjective perceptions of system usability, including aspects such as perceived complexity, ease of use, learnability, confidence, and the need for technical assistance. Given the small sample size, confidence intervals were computed using an 85% confidence level.

In line with the original SUS formulation, the questionnaire consisted of ten items, rated on a five-point scale, with responses ranging from strong disagreement to strong agreement or from low to high perceived difficulty, depending on the item formulation. As defined in SUS, odd-numbered items are positively worded, while even-numbered items express negative perceptions, allowing a balanced assessment of usability. In our experiment, the standard SUS items were adapted by explicitly referring to BPM or CEP, for

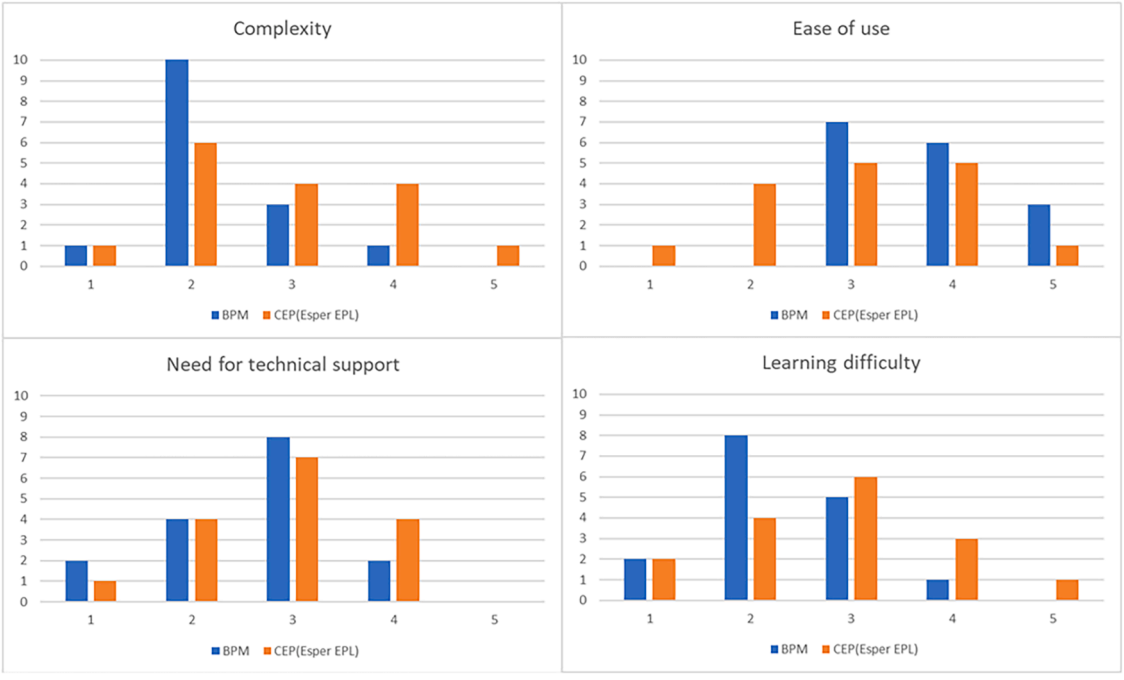


Fig. 6. Results of the initial questionnaire on the perceived usability of BPM and CEP.

Day 1 and Day 2, respectively, while preserving the original intent and structure of each question. The collected responses were later used to compute the SUS index for each participant, enabling a quantitative comparison of perceived usability for both technologies.

Fig. 6 and the SUS results in Table 1 collectively reveal a clear distinction in perceived complexity and usability between BPM and CEP. Most participants perceived BPM as a less-complex approach, with responses largely concentrated at the lower end of the scale for perceived complexity, whereas working directly with CEP rules and event patterns was perceived as more complex, with higher scores clustered around the middle and upper ranges. In terms of ease of use, the majority of participants agreed that BPM is easy to use, with responses leaning towards the higher end of the agreement scale, while responses regarding CEP were more heterogeneous and tended towards neutral or lower agreement values, reflecting a comparatively lower perception of ease of use. Similarly, participants reported a limited need for technical assistance when using BPM, in contrast to CEP, which was associated with higher perceived difficulty and greater reliance on support. Regarding learning difficulty, BPM was perceived as easier to learn, with responses concentrated in the lower and middle ranges of the scale, whereas CEP required greater effort to achieve competence. These qualitative impressions are confirmed by the SUS results: BPM achieved an overall SUS score of 69.81 (grade C), compared to CEP's 52.35 (grade D). According to standard SUS interpretation guidelines, letter grades provide a qualitative assessment of usability, where scores around C indicate acceptable usability, while grades D or lower reflect poor usability and a need for improvement. The Usable subscale also favours BPM (59.96, C) over CEP (50.78, D), and the Learnable subscale indicates that BPM is easier to master (63.28, $C+$) than CEP (53.91, $D+$), with CEP exhibiting a wider spread of participant scores. Overall, these findings suggest that while CEP provides powerful event-processing capabilities, BPM offers a more accessible, understandable, and learnable environment for managing contextualised actions in IoT scenarios. Given the small sample size, reliability estimates should be interpreted cautiously; nevertheless, the results show a clear and consistent trend.

Taken together, these results provide a descriptive comparison of participants' initial perceptions of the usability of BPM and CEP before the final experimental task.

6.3.2. Practical exercise results: effectiveness and efficiency

This subsection compares performance of the Day 3 practical exercise with and without BPMN support (see Supplementary Materials section). Table 2 summarises effectiveness, completion time and efficiency results.

On average, participants using BPMN support achieved a higher average effectiveness (7.79) than the non-BPM group (4.14), indicating that the BPMN model facilitated the correct implementation of CEP event patterns. In terms of completion time, both groups performed similarly, although the BPM group was slightly faster (1 h 47 min) than the non-BPM group (1 h 51 min). Despite this marginal time difference, the BPM group achieved a significantly higher efficiency rate (0.0738 points per minute) compared to the non-BPM group (0.0388 points per minute), suggesting that BPMN support enhanced both accuracy and productivity in the practical exercise tasks. These results demonstrate the positive impact of BPMN guidance on participants' task performance.

6.4. Final questionnaires: perceived difficulty of the practical exercise

Post- experiment questionnaires assessed perceived difficulty on implementing CEP event patterns with and without BPM support (see Supplementary Materials section). Complementing the SUS results with task-specific insights into comprehension and experienced workload.

As shown in Fig. 7, participants with BPMN model support generally reported lower difficulty levels with most responses ranging from easy to normal and only a few rated the task as difficult. In contrast, the non-BPM group reported higher levels of difficulty more frequently, indicating a greater perceived challenge in defining CEP event patterns. Regarding the clarity of the information provided, those supported by the BPMN model largely found the combination of textual description and visual BPMN models to be *sufficient* or *fully sufficient* for understanding the system's behaviour. Conversely, participants without BPMN model support provided a broader range of responses, including *insufficient* and *partially sufficient*. This suggest that the absence of a structured model hindered their ability to perceive the information as complete.

7. System-centred evaluation: real-time performance of the CEP and BPM Components

This subsection evaluates the performance of the proposed architecture in terms of real-time event processing. The experiments focus on assessing the behaviour of the CEP and BPM engines separately under increasing event rates to characterize their individual latency, throughput, and stability characteristics. Although one activity requires synchronization—waiting for messages from both

Table 1
SUS Rating obtained for CEP and BPM in the initial questionnaires.

SUS rating	SUS index mean	Standard deviation	CI lower limit (85%)	CI upper limit (85%)	Mean grade
CEP	51.41	9.66	47.93	54.88	D
BPM	60.63	5.59	58.61	62.64	C
CEP usable	50.78	10.05	47.16	54.40	D
BPM usable	59.96	6.96	57.45	62.47	C
CEP learnable	53.91	20.78	46.43	61.39	D+
BPM learnable	63.28	17.36	57.03	69.53	C+

Table 2
Effectiveness and efficiency analysis.

Group	No BPM	BPM
Participants	7	7
Average effectiveness (0–10)	4.14	7.79
Effectiveness standard deviation	2.48	2.34
Average time to completion	1 h 51 min	1 h 47 min
Time to completion standard deviation (min)	10.86	9.53
Average efficiency (points/min)	0.0388	0.0738
Efficiency standard deviation	0.0270	0.0235

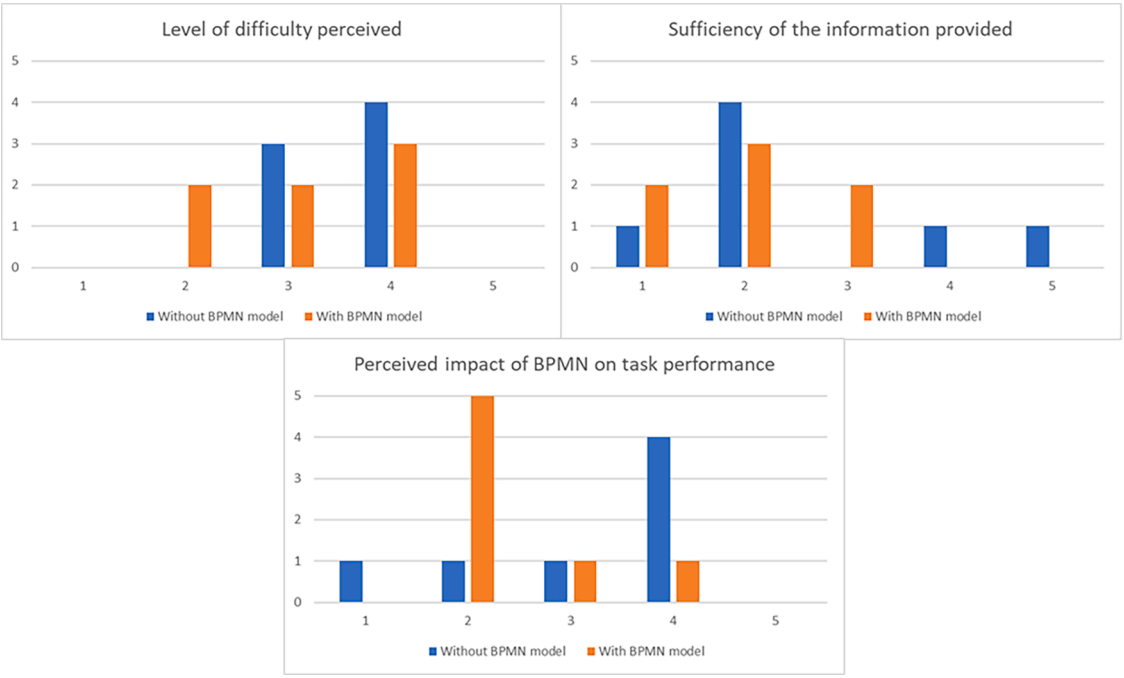


Fig. 7. Results of the final questionnaire on the perceived difficulty of the practical exercise.

CEP and BPM—the analysis isolates each engine's contribution. Thus, these tests do not fully represent the integrated architecture, but provide insight into the real-time capabilities of each component and the potential impact of combining them. It also aims to assess whether integration preserves real-time processing capabilities.

The proposed software architecture was fully implemented in order to carry out the performance tests. Sensor data were simulated using an event generator and published to RabbitMQ. Both CEP (Esper version 9.0.0) and BPM (Camunda BPM 7.24.0) engines independently subscribed to RabbitMQ event streams and processed the incoming data accordingly. Camunda BPM version 7.24.0 was selected despite not being the latest release, as it is the most recent freely available version; newer commercial releases report performance improvements that could further enhance the system efficiency. When both components detected a situation of interest, the BPM engine executed the corresponding contextualised action.

7.1. Computer resources and methods

Performance evaluation was conducted using a single computing device, although the architecture allows for deployment across multiple nodes when necessary. The experiments were run on a PC with an Intel® Core™ i7-–8750H CPU running at 2.20GHz and with 16GB of RAM. This represents a mid-range configuration typical of various real-world scenarios [7].

All components involved in the evaluation —including the event data simulator, RabbitMQ message broker, and both the CEP and BPM engines— were collocated on this machine. This setup ensured that performance data reflected the intrinsic architectural overhead, avoiding the interference of network-related variability.

The performance evaluation involved deploying the BPMN model introduced in Section 4, along with its corresponding CEP event patterns. While the BPMN model was executed on a BPM engine and the CEP event patterns on a CEP engine; both were integrated via the messaging infrastructure according to the architecture described in Section 3.2.

Three scenarios were evaluated: single-user, multi-user with shared logic, and multi-user with individualised logic. In the single-

user scenario, the system processes events generated by a single individual, serving as a baseline to measure the minimum architectural overhead by isolating the interaction between a single CEP engine and a single BPM engine, without the influence of engine concurrency or personalisation by different users. In this case, one CEP engine and one BPM engine operate together. The multi-user with shared logic scenario involves the system processing events from multiple users who share the same contextualised response logic, consequently a single CEP engine and a single BPM engine are used. In contrast, the multi-user with individualised logic scenario requires the system to handle events from multiple users, each with unique contextualised actions defined. This configuration deploys one CEP engine and one dedicated BPM engine for each user.

For each scenario, the incoming event rate was increased progressively to assess system behaved under increasing workloads. Experiments began with an input rate of 10 events per second (events/s), progressing up to 15 000 events/s, when the engine supported this rate under the given scenario. This incremental approach continued until either the CEP or BPM engine could no longer sustain the load during execution. Once a component failed to handle a given input rate, higher rates were not evaluated for that specific configuration.

7.2. Metrics

The following performance indicators were measured:

- Latency. Average time in milliseconds (ms) taken by either the CEP or BPM engine to process individual input event. Due to measurements having a millisecond-level resolution, any processing time below 1 ms is recorded as 0 ms. The values shown in Tables 3 and 4 are computed by averaging the processing times of all events received during each one-second interval.
- Throughput time. Total time, in minutes (min), required by the engine to process all incoming events. This metric serves to identify overload situations.
- Contextualised actions handling time. Average time in ms required to trigger and complete a contextualised action once a complex event has been detected. This indicator evaluates the system's responsiveness in executing real-time actions.
- Total complex event detected: total number of complex events identified throughout the test.

Each experiment was conducted over a fixed period of 10 min. If the throughput time for a given scenario and input rate exceeded

Table 3
Results of the short-term performance tests.

Incoming rate (events/s)	Throughput time (min)		Processing time (ms) (Latency)		Contextualised actions handling time (ms)	Total complex events detected	
	CEP	BPM	CEP	BPM		CEP	BPM
Single-user scenario							
10	10	10	0.634	4.188	8.679	3 000	3 000
25	10	10	1.388	220.303	5.554	7 500	7 500
50	10	10	0.948	121.47	8.217	15 000	15 000
75	10	10:10	0.772	2 009.259	7.758	23 077	23 077
100	10	11:30	1.02	53 109.525	7.878	30 000	30 001
1 000	10	-	0.471	-	-	299 998	-
10 000	10	-	2.068	-	-	3 000 014	-
15 000	10	-	3.731	-	-	4 546 637	-
Multi-user shared scenario							
10	10	10	2.216	162.594	9.856	3 005	3 005
25	10	10	1.448	191.141	9.074	7 503	7 503
50	10	12	0.98	42 793.002	6.294	15 002	15 005
1 000	10	-	0.576	-	-	300 005	-
10 000	10	-	12.348	-	-	2 999 997	-
15 000	10	-	18.989	-	-	4 499 927	-
Multi-user individualised scenario							
10	10	10	2.216	15.761	19.185	3 005	3 003
25	10	10	1.448	13.736	16.705	7 503	7 504
50	10	10	0.98	111.208	18.231	15 002	15 005
75	10	10	0.669	327.128	16.395	22 560	22 560
100	10	10	0.855	433.671	15.157	30 004	30 004
125	10	10	0.862	915.217	15.616	37 505	37 505
150	10	10	0.745	1 928.031	13.754	44 780	44 780
175	10	10	0.87	3 026.07	14.077	52 635	52 635
200	10	10	0.604	4 667.771	13.548	60 002	60 005
225	10	10:20	0.744	15 299.326	13.626	68 185	68185
1 000	10	-	0.576	-	-	300 005	-
10 000	10	-	12.348	-	-	2 999 997	-
15 000	10	-	18.989	-	-	4 499 927	-

Table 4
Results of long-term performance tests.

Incoming rate (events/s)	Throughput time (min)		Processing time (ms) (Latency)		Context actions handling time	Total complex events detected	
	CEP	BPM	CEP	BPM		CEP	BPM
Single-user scenario							
50	60	60	0.788	406.141	8.101	90 000	90 000
Multi-user shared scenario							
25	60	60	0.371	65.999	8.225	45 005	45 005
Multi-user individualised scenario							
200	60	64	0.726	33 538.802	12.22	360 005	360 005

this 10 min window, the system was considered unable to sustain that particular event rate.

The input data were generated using a custom Java-based simulator. The simulator produces event streams tailored to the scenario, ensuring the generation of the event sequences necessary to produce pairs of simple events whose correlation results in the detection of a complex event and the triggering of subsequent actions. The generated data are deterministic and identical across all executions, ensuring full reproducibility of the experiments. All the code used for the evaluation is provided as supplementary material (see Supplementary Material section).

Finally, after the 10 min tests were completed, an extended 60-minute stress test was performed using the peak sustainable input rate identified in the initial tests. This long-duration run was conducted to verify the system's stability under sustained load conditions.

7.3. Performance results

This subsection presents the results of the performance evaluation across the three defined scenarios. Tables 3 and 4 summarise the short- and long-duration experiments, respectively. All raw data obtained during the evaluation, as well as processed data, are provided as supplementary material (see Supplementary Material section).

As explained in the previous subsection, the short-duration tests (Table 3) were used to identify the maximum sustainable rate for each configuration. In the single-user scenario, the system effectively processes input rates up to 75 events/s, maintaining a throughput time of 10 min for both the CEP and BPM engines. However, at 100 events/s per second, while the CEP engine remains within the expected timeframe, the BPM engine exceeds the experiment's duration, reaching a throughput time of 11 min and 30 s. This behaviour is reflected in a sharp increase in BPM processing latency, which rises from under 3 s to over 53 s. Consequently, higher input rates were not evaluated for the BPM engine in this configuration. In contrast, the CEP engine was successfully evaluated up to 15 000 events/s. Additional load tests on the CEP engine in isolation were not considered necessary as the primary objective was to compare the relative performance of both engines under equivalent conditions.

Similar behaviour is observed in the multi-user scenario with shared contextualised actions, although the BPM engine reaches its saturation point earlier. Performance remains stable up to 25 events/s, with throughput times of 10 min and moderate processing latencies. However, at 50 events/s, the BPM engine throughput time climbs to 12 min, accompanied by a significant latency increase. This is because the BPM engine must manage the representation and evaluation of the logic of multiple CEP event patterns concurrently, which creates a computational bottleneck in the process logic itself. As in the previous scenario, the CEP engine continues to operate within acceptable latency ranges despite the increased load.

Notably, the multi-user scenario with individualised actions demonstrates the highest overall processing capacity among the evaluated configurations. This behaviour can be attributed to the fact that event detection remains centralised in the CEP engine, while decision logic and action execution are distributed among independent BPM process instances. This separation reduces the number of messages to be managed in each BPM engine, as each user-specific process manages a smaller, more specific set of contextualised actions, allowing the system to better absorb higher input rates before exceeding the experimental time window. As a result, this configuration handle input rates of up to 200 events/s within the 10-minute window in both engines. At 225 events/s, the BPM engine exceeds the experiment's duration by a marginal 20 s, while CEP processing remains stable. Beyond this point, BPM engine was discontinued. Throughout all evaluations, CEP latency remained consistently low, showing only a gradual increase relative to the input rate.

Across all scenarios, the processing time for contextualised actions remains relatively stable while the BPM engine operates within its load capacity. Values generally stay below 20 ms, showing that the system is able to trigger and complete contextualised actions efficiently once a complex event is detected.

With regard to the long-duration experiment (Table 4), all three scenarios were executed for 60 min. Both the CEP and BPM engines completed the full experiment successfully in the single-user scenario at a rate of 50 events/s and in the multi-user scenario with shared contextualised actions at a rate of 25 events/s, maintaining stable latency and consistent contextualised action handling times throughout the execution. These results indicate that, under low input rates, the system behaves reliably over extended execution periods, regardless of whether contextualised actions are managed for a single user or shared across multiple users. In contrast, the multi-user scenario with individualised contextualised actions, evaluated at 200 events/s, exhibits different behaviour. While the CEP engine maintained stable processing throughout the experiment, the BPM engine exceeded the intended execution window, reaching a

throughput time of 64 min. In this experiment, BPM latency degraded significantly under sustained high load, whereas CEP performance remained largely unaffected.

In addition to the results reported in Tables 3 and 4, Figs. 8–10 show how the average processing latency for the CEP and BPM engines evolves over time in each of the three scenarios evaluated during the long-duration tests. The average latency shown in these figures corresponds to the average processing time of all events handled per second of execution, enabling detailed comparison of the performance of both engines under sustained load conditions.

Fig. 8 illustrates how latency evolves in the single-user scenario when evaluated at 50 incoming events per second. As can be seen, the CEP engine maintains a consistently low and stable average latency throughout the entire 60-minute execution period. In contrast, the BPM engine exhibits greater variability, with noticeable latency peaks that become more pronounced towards the end of the experiment. Specifically, several peaks exceed 100 ms at the end of the test, suggesting that the BPM engine is experiencing greater processing overload, although overall performance remains within the established throughput time.

Fig. 9 shows the results for the multi-user scenario with shared contextualised actions, with an input rate of 25 events/s. Once again, the CEP engine demonstrates consistently low average latency. Although the BPM engine remains stable throughout the experiment, it consistently shows higher average latency than the CEP engine. This reflects the additional overhead introduced by the BPM logic, which is responsible for representing and evaluating multiple CEP patterns simultaneously. Nevertheless, the system remains within its sustainable operating range for this input rate.

Fig. 10 illustrates the evolution of latency in the multi-user scenario, in which individualised, contextualised actions are evaluated at a rate of 200 events/s. As in previous scenarios, the CEP engine maintains stable, low latency throughout the experiment. In contrast, the BPM engine cannot sustain this high input rate and exhibits irregular behaviour. A significant peak in average latency is particularly notable around the midpoint of the experiment, where BPM latency abruptly increases, reaching values close to 80 000 ms. This behaviour indicates clear saturation of the BPM engine under sustained high load, despite CEP processing remaining unaffected.

Finally, Fig. 11 reports the contextualised actions handling time for the three scenarios during the long-duration experiments. This metric remains relatively stable and consistently low across all configurations, showing that, once a situation of interest is detected, the execution of contextualised actions is handled efficiently. This suggests that the primary performance limitations observed in BPM are related to event processing and pattern logic management rather than the execution of contextualised actions.

Overall, these results show that the CEP component consistently exhibits low latency and robust behaviour across all configurations. In contrast, the BPM component emerges as the primary bottleneck under high event rates and long-running executions. This performance pattern remained consistent across both short- and long-running experiments.

8. Discussion

This section discusses the results obtained in the evaluation presented in Section 6 and Section 7, analysing the extent to which the RQs formulated in the introduction are answered and linking them to the contributions of this paper. Besides, this section analyses the benefits and lessons learned from the proposal contributions, evaluating the advantages, limitations and lessons learned from combining CEP and BPM in IoT and smart environments through BPSmart-CARE.

8.1. Response to research questions

RQ1. Can the extension of the Smart-CARE taxonomy with BPM improve the representation and personalisation of contextualised user actions in IoT systems? And if so, how effectively can the integration of BPM within Smart-CARE, beyond the taxonomy layer, support the

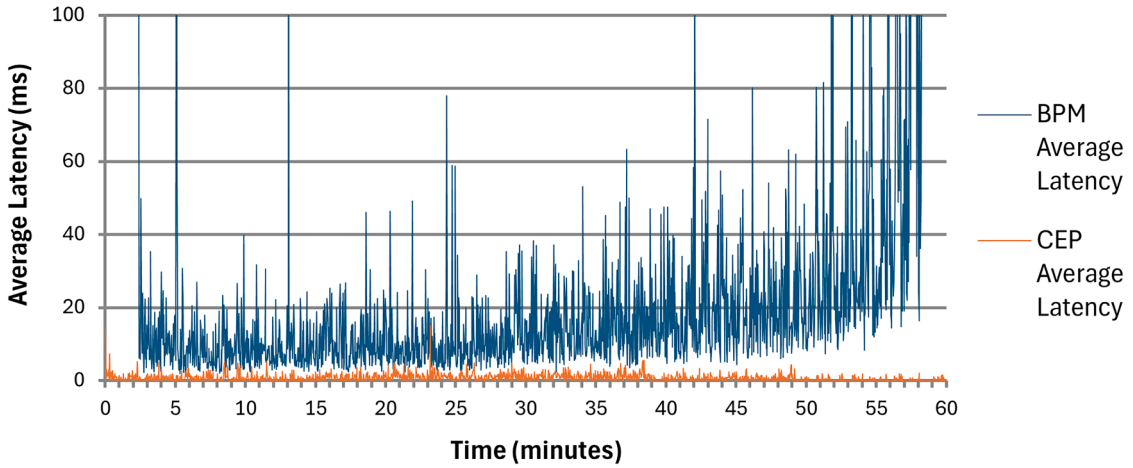


Fig. 8. Results of long-term performance tests for the single-user scenario for 50 events/s input rate.

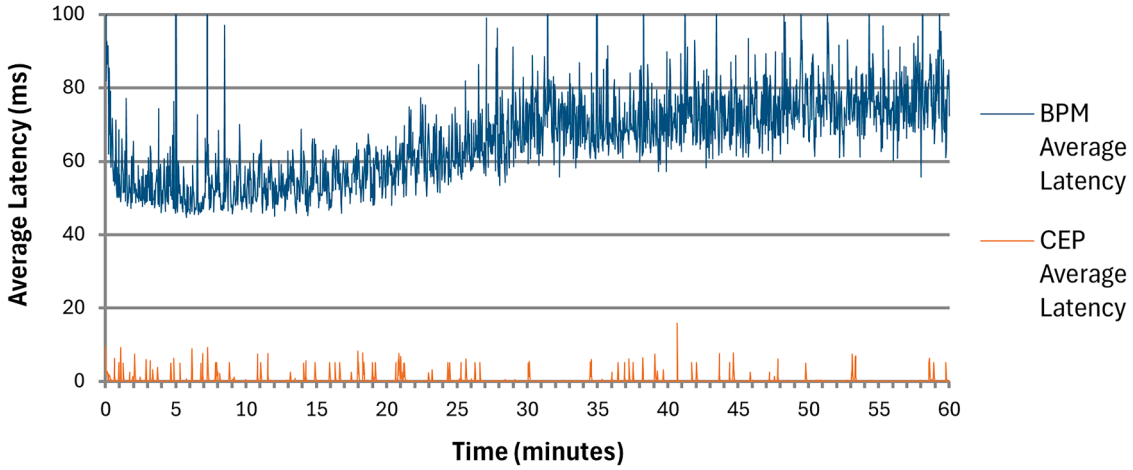


Fig. 9. Results of long-term performance tests for the multi-user shared scenario for 25 events/s input rate.

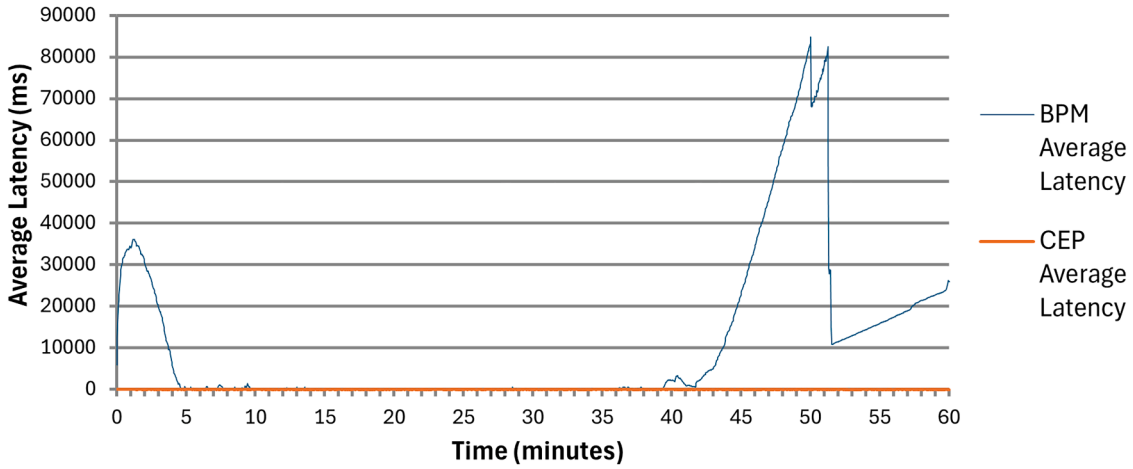


Fig. 10. Results of long-term performance testing for the individualised multi-user scenario for 200 events/s input rate.

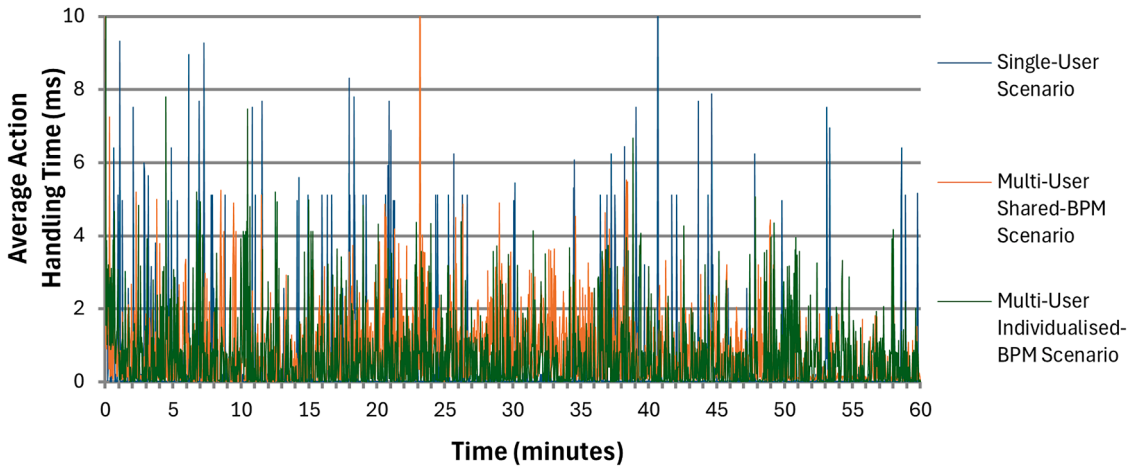


Fig. 11. Results of the contextualised actions handling time for the long-term performance tests of the three proposed scenarios.

definition, reuse and management of personalised contextualised actions for different users? The implemented case study shows the feasibility of extending the taxonomy with new attributes (Contribution 1) that reference BPM models used to define and manage users' contextualised actions. Besides it enables contextualised actions to be related to situations of interest and represented in a structured and understandable manner, facilitating action personalisation through the integration of contextual data originating from heterogeneous application domains. Beyond the taxonomy layer, the proposal shows that BPM can effectively support personalised contextualised actions to be defined and managed in a structured way (Contribution 2.b). This facilitates their integration through BPM models and enable systems to offer personalised services to users, even when data originates from various application domains. However, contribution 2.b also raises scalability concerns, particularly in scenarios with a large number of users and situations of interest. One potential challenge lies in the initial perception that it might be necessary to define a BPM model for each user or situation of interest, which would be impractical at scale. The proposal mitigates this issue by enabling the reuse of generic BPM processes that can be adapted through contextual parameters. For instance, in a nursing home with hundreds of residents, generic BPM models can be reused and customised based on individual user profiles, significantly reducing modelling and maintenance effort. To end with, the proposal was illustrated through a case study, in which a series of BPM event patterns and user-defined contextualised were modelled and executed in response to CEP event patterns detection (Contribution 3). The case study shows that integrating CEP and BPM permits representing the situations of interest and triggers the associated contextualised actions according to the user, and performs its execution effectively. Besides, performance evaluations confirm that individualised BPM models remain stable and efficient under moderate event rates, supporting the scalability of the approach.

RQ2. To what extent does BPM support the definition of the rationale behind CEP event patterns and the representation of their interdependencies? BPM effectively supports the definition of the rationale behind CEP event patterns and the representation of their interdependence with other event patterns directly within BPMN models (Contribution 2.a), offering significant advantages in terms of clarity, manageability and updatability. The results of the evaluation tests conducted with participants in the proposed experiment shows that the use of BPM greatly simplifies developers' tasks when developing event patterns, enabling them to visualise and manage such event patterns as part of an understandable process flow. It also improves communication with domain experts, who can collaborate in validating expected system behaviours. This capability is particularly valuable in environments where requirements frequently change and in large-scale IoT deployments with numerous event patterns, as the graphical process view eases understanding, review, and validation of updates, reduces misconfigurations, and mitigates complexity, therefore supporting reliable system evolution and CEP rule maintenance. The usability evaluation corroborates this observation: participants reported that BPM-based guidance allowed them to understand and manage CEP event patterns more easily than using CEP-only representations. However, this approach also presents certain challenges. Representing CEP event patterns in BPM is not always straightforward, as CEP is a flexible programming language, whereas BPM is more structured. For instance, event patterns that require data from multiple sensors may result in BPM models that are overly complex or difficult to interpret. Moreover, representing specific event sequences —such as enforcing event ordering constraints— can be difficult to express in BPM and may lead to incorrect event pattern evaluation if modelled improperly. Furthermore, certain CEP constructs, such as window clauses, pose significant modelling challenges in BPM. While some time- or event-counting-based windows can be represented by timers within tasks or threads, more advanced types, such as sliding windows, cannot. Moreover, the use of timers in BPM can lead to semantic ambiguity in CEP, as a single BPM timer may correspond to several different CEP event patterns. This is illustrated in the Section 4 case study, where an event pattern is defined to trigger an alert if an elderly person fails to return home within a certain time. Although this behaviour can be modelled in BPM as an external task with a timer, the underlying execution logic requires a CEP rule to monitor the related input events in real time.

RQ3. Can the integration of BPM into the Smart-Care context-aware software architecture improve the overall management of contextualised actions and event patterns in terms of orchestration, traceability, and maintainability? The integration of BPM within the context-aware software architecture (Contribution 2) represents a substantial advancement in the management of contextualised actions and event patterns. This integration allows the complementary strengths of both technologies: CEP ensures high-performance event correlation, while BPM supports process modelling, execution and maintainability. The results of the evaluation conducted with participants in the proposed practical exercise show that those using BPM guidance were both more effective and more efficient when implementing CEP event patterns. This proves that BPM improves comprehension and reduces implementation errors. The performance evaluation also indicates that BPM can be integrated without affecting real-time behaviour at low to moderate event rates—often sufficient for many practical scenarios. However, BPM does not provide the same processing capacity as the CEP engine: under sustained high loads, it becomes a bottleneck and may increase end-to-end response times. In such cases, BPM can be used in a descriptive role (i.e., as documentation and for stakeholder alignment) without being functionally coupled to the CEP execution. For each scenario, the most convenient option should be weighed accordingly.

RQ4. How effective is BPM in supporting runtime updates to CEP event patterns, based on detected situations of interest, while preserving real-time performance constraints? The proposal enables the real-time updating of CEP event patterns logic based on situations of interest detected through the processes defined in BPM (Contribution 2.c). This capability significantly enhances the adaptability and decision-making capacity of context-aware IoT systems which require adapting to new requirements in real time. An issue raised by contribution 2.c concerns BPM's limited ability to represent unknown or highly dynamic behaviour. While BPM effectively models predefined behaviour, representing highly dynamic runtime conditions based on user behaviour can be challenging. Nevertheless, the combination of BPM, CEP and a cross-domain taxonomy enables more precise definition and supports greater flexibility and adaptability. The case study shows that BPM can successfully coordinate the execution of contextualised actions triggered by detected CEP event patterns, supporting structured updates to system behaviour in response to changing situations of interest. This confirms the practical value of integrating BPM with CEP for adaptive management of context-aware IoT systems.

8.2. Comparison of BPSmart-care with existing BPM-CEP approaches

Following the analysis of the research questions in Section 8.1, it is important to situate BPSmart-CARE in the context of existing CEP-BPM proposals. The comparison presented in this subsection focuses on how the proposed framework addresses limitations observed in the literature and highlights its practical advantages over prior approaches.

Despite the advances found in the related work in integrating CEP and BPM, several limitations persist. Current CEP-BPM proposals provide only partial support for key dimensions required by context-aware IoT systems. Table 5 summarises the main features and limitations of representative CEP-BPM approaches and contrast them with BPSmart-CARE. We evaluate the following aspects of CEP-BPM integration: *Domain scope* indicates whether the proposal is independent of the domain or is tailored to a specific application area; *Usability* reflects how easily stakeholders can define, understand and manage contextualised actions; *Maintainability* refers to the effort required to evolve CEP event patterns and BPM models over time; and *Adaptability* assesses the system's ability to adjust its behaviour in response to contextual changes. In addition, *Real-time processing* is the ability to handle high volumes of event streams with low latency, which is essential in IoT environments; *Semantic transparency* [52] is a measure of how clearly the logic of event detection and decision-making is represented, and how understandable it is to both developers and domain experts; *CEP event pattern logic representation* refers to whether the internal logic and dependencies of CEP event patterns can be modelled and documented within BPM; while *Dynamic update of CEP event patterns* assesses the ability to modify event patterns at runtime based on contextual or process-level changes. In the table, a + indicates full support or a feature that can be easily achieved, $\pm$ denotes partial or limited support, and - represents either a lack of support or a feature that is highly complex or impractical to implement.

The comparison shows that many existing approaches focus on specific domains, such as smart water systems [37] or Industry 4.0 [39], limiting generalisation across applications. While several proposals achieve strong real-time processing capabilities [34–36,38], Table 5 highlights recurring weaknesses in terms of semantic transparency, maintainability and dynamic adaptability. This is largely due to CEP logic being represented implicitly or in a way that lacks semantic transparency, meaning that the logic of event detection and decision-making is not clearly understandable to developers or domain experts. Consequently, maintaining and evolving CEP event patterns can be challenging, since even minor modifications necessitate an in-depth understanding of the underlying rule definitions, which reduces maintainability. Furthermore, dynamic adaptability is limited because changes to event detection logic or system behaviour are typically carried out manually, which makes it challenging to respond quickly to changes in context. Overall, these studies highlight an existent gap: existing proposals rarely combine real-time, high-performance event processing with a representation of CEP logic in a way that is both understandable and manageable for users. Approaches that deliver low-latency event correlation, typically rely on non-transparent specifications, limiting transparency and usability. In contrast, approaches that prioritise understandability often abstract away real-time processing details, reducing control over event detection and adaptation. In this context, BPSmart-CARE addresses some of these limitations by combining BPM-based usability, maintainability and adaptability with real-time processing, ensuring that CEP event patterns can be effectively represented, managed, and updated in context-aware IoT environments. However, BPSmart-CARE still present some limitations regarding adaptability and maintainability, as explained in Section 8.4.

8.3. BPSmart-CARE practical implications

The findings of this work have several practical implications for the design, development, and maintenance of context-aware IoT systems.

First, extending the Smart-CARE taxonomy with BPM offers practitioners a clear and structured way to define contextualised user actions with greater precision and adaptability. Developers can specify actions at a finer level of detail while still accommodating variations across users and contexts, avoiding the need for domain-specific or hard-coded solutions. Because the taxonomy is domain-agnostic, the same modelling approach can be applied across heterogeneous IoT domains, supporting interoperability and reuse in real deployments. This cross-domain applicability is also reflected in ongoing work beyond the AAL scenario used in this paper for illustration. We are currently implementing BPSmart-CARE in a smart-port environment, where heterogeneous streams—such as vessel traffic, road traffic, bulk cargo operations, odour monitoring, and atmospheric conditions—are correlated to detect situations of interest and trigger contextualised actions.

Second, the integration of BPM into the CEP-based architecture simplifies the practical management of event patterns and system behaviour. BPM provides an intuitive and visual representation of how situations of interest lead to specific actions, which improves understanding and reduces the likelihood of modelling and implementation errors. Despite certain limitations, using BPM to define the logical dependencies among CEP event patterns provides significant practical value. It improves developers' understanding of event sequences and their relationship with contextualised actions. Moreover, this integration enhances the representation of event flows and interdependencies, improving the accuracy and maintainability of CEP rules, especially during system evolution. BPM can therefore be used both during early design stages, as a tool for planning the system's logic, and during maintenance phases, enabling accurate documentation, contextualisation, and extension of system functionalities.

Third, BPM-based representations facilitate collaboration between technical developers and domain experts by providing an intuitive view of system behaviour, supporting validation, communication, and system evolution in environments with frequently changing requirements.

Finally, enabling real-time updates of CEP event patterns through BPM processes has important practical consequences for adaptive systems. System behaviour can be adjusted dynamically in response to detected situations of interest, without disrupting real-time operation. This makes the proposed approach suitable for long-running, adaptive IoT systems. Nevertheless, the identified

Table 5
Comparison of CEP-BPM proposals (U: Usability, M: Maintainability, A: Adaptability, RT: Real Time Processing, ST: Semantic Transparency, LR: CEP Event Pattern Logic Representation, DU: Dynamic Update of CEP Event Patterns).

Approach	Domain scope	U	M	A	RT	ST	LR	DU
Ruhkamp et al. [34]	IoT and smart environments	-	±	-	+	-	-	-
Janiesch et al. [35]	IoT and smart industry	±	±	±	+	-	-	-
Fardbastani et al. [36]	Distributed systems	-	-	-	+	-	-	-
Gonçalves et al. [37]	Smart water	+	±	±	+	+	-	-
Valderas et al. [38]	IoT	+	±	±	+	-	±	-
Bazán et al. [39]	Industry 4.0	-	-	-	+	-	-	-
Smart-CARE [7]	IoT and smart environments	-	-	-	+	-	-	-
BPSmart-CARE	IoT and smart environments	+	±	±	+	+	+	+

limitations, summarised in the following subsection, highlight the need for further research to improve BPM expressiveness for complex CEP logic and highly dynamic behavioural event patterns.

8.4. BPSmart-CARE limitations

Despite the advances demonstrated by BPSmart-CARE, several limitations remain that impact system performance, adaptability, and practical deployment in complex IoT environments. These limitations can inform future research and guide further improvements to the framework.

First, although BPM offers a structured and visual approach to representing CEP event patterns, certain complex event constructs remain difficult to model. For example, window-based event patterns (particularly sliding windows), event ordering constraints and multi-sensor aggregated events can result in BPM models that are overly complex or ambiguous in meaning. This limitation may reduce the system's adaptability, as extending or modifying event patterns in response to evolving contexts could necessitate significant redesign. Future work could explore BPM extension to better represent different CEP clauses, thereby increasing expressiveness without sacrificing comprehensibility.

Second, individualised BPM models allow for customised actions for different users; however, scenarios with hundreds or thousands of users could place a significant burden on modelling and computation. Although the reuse of generic BPM processes mitigates this to some extent, maintaining a large set of contextualised actions can still pose a challenge for system performance and management. Possible solutions include developing automated adaptation strategies, parameterised templates, or hierarchical BPM modelling techniques to reduce redundancy and preserve customisation.

Finally, the ability to scale horizontally may be limited by the interdependencies between CEP event patterns processed on different engines. This can hinder the system's efficiency when distributing workloads across multiple nodes, particularly in environments with numerous complex event patterns. These limitations highlight areas for future research and improvement towards an efficient, multi-node implementation that does not compromise consistency or responsiveness.

9. Conclusions and future work

This proposal addresses one of the main challenges in IoT systems and smart environments: the definition of personalised, contextualised actions triggered from the analysis of large volumes of heterogeneous data across multiple application domains. To this end, an improved framework for context-aware real-time smart environments has been proposed through an extension to an existing taxonomy, along with the incorporation of BPM processes into a CEP-based context-aware software architecture. As a result, this proposal facilitates the contextualisation of IoT systems by: (i) representing and documenting CEP event patterns in BPM, which reduces the complexity of maintaining CEP rules and improves the understanding of event pattern logic; (ii) facilitating the user-specific and context-aware customisation of system responses across multiple domains, tailoring contextualised actions to the needs of each user and increasing interoperability and adaptability; (iii) improving collaboration between developers and domain experts through BPM as a common language to represent both user actions and CEP event patterns; and (iv) paving the way for the dynamic adaptation of system behaviour at runtime by defining through BPM models, the logic that orchestrates CEP event patterns updates, thus allowing system decisions to evolve in response to changes in the context.

The evaluation conducted —including usability studies, a practical exercise implementation and performance testing—, confirmed the effectiveness and feasibility of the proposed approach. Results show that the BPM-enhanced taxonomy improves understanding, personalisation and management of contextualised actions. In terms of performance, the results indicate that BPM can complement CEP without hindering real-time responsiveness at low to moderate event rates, which is sufficient for many realistic deployments, showcasing the practical applicability of the approach in real-world smart environments. For sustained high event loads, the most convenient option should be weighed on a case-by-case basis: CEP should remain the primary processing engine and BPM may be best dedicated to running complex action flows triggered by the CEP detection to avoid introducing additional latency.

As discussed in Section 8.4, limitations remain regarding the representation of advanced CEP construct in BPM and the impact of inter-pattern dependencies on horizontal scalability. Motivating further research.

Future work will involve investigating bidirectional synchronisation between the BPM model and the CEP code, to support semi-automated co-evolution. Additional research will also explore mechanisms to enhance BPM expressiveness for CEP rule representation.

Moreover, GenAI will be investigated as a support tool to assist in the creation, evolution, and explanation of BPM models and CEP event patterns, with the aim of improving maintainability, usability, and semantic transparency without replacing the explicit modelling foundations of BPM and CEP.

CRediT authorship contribution statement

Adrian Bazan-Muñoz: Writing – original draft, Methodology, Investigation, Conceptualization. **Cesare Pautasso:** Writing – review & editing, Methodology, Conceptualization. **Guadalupe Ortiz:** Writing – review & editing, Methodology, Funding acquisition, Conceptualization. **Alfonso Garcia-de-Prado:** Writing – review & editing, Methodology, Conceptualization.

Declaration of competing interest

The authors declared that they have no conflicts of interest to this work.

Acknowledgements

This publication is partially funded by MICIU/AEI/10.13039/501100011033 and by ERDF/EU (grant PID2021-122215NB-C33, AwESOMe). A. Bazan-Muñoz acknowledges the support from the pre-doctoral research staff training contract 2022-006/PU/EPIF-FPI-CT/CP of the University of Cadiz. G. Ortiz and A. Garcia-de-Prado respectively thank the financial support from European Regional Development Fund (ERDF) Operational Programme/Programa Operativo FEDER Andalucía 2021-2027 and Consejería de Universidad, Investigación e Innovación, Junta de Andalucía by grants FEDER-UCA-2024-A2-17 (PANORAMA project) and FEDER-UCA-2024-A1-21 (PICOC project).

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.iot.2026.101887, and also at doi:10.17632/x2brytbx4s.1.

Data availability

We have uploaded all the data as supplementary material.

References

[1] D. Kundu, A.K. Pandey, World urbanisation: trends and patterns, in: D. Kundu, R. Sietchiping, M. Kinyanjui (Eds.), Dev. Natl. Urban Policies Ways Forw. Green Smart Cities, Springer Nature, Singapore, 2020, pp. 13–49, https://doi.org/10.1007/978-981-15-3738-7_2.
[2] L. Hrytsenko, D. Krawczyk, L. Derkach, S. Kolomiiets, The role of smart City in achieving sustainable development: google trends analysis and exponential time series smoothing models, Bus. Ethics Leadersh. 8 (2024) 190–202, https://doi.org/10.21272/bel.8(1).190-202.2024.
[3] S. Mishra, S.K. Sharma, Advanced contribution of IoT in agricultural production for the development of smart livestock environments, Internet Things 22 (2023) 100724, https://doi.org/10.1016/j.iot.2023.100724.
[4] G. Sinnapolu, S. Alawneh, Integrating wearables with cloud-based communication for health monitoring and emergency assistance, Internet Things 1–2 (2018) 40–54, https://doi.org/10.1016/j.iot.2018.08.004.
[5] B. Cengiz, I.Y. Adam, M. Ozdem, R. Das, A survey on data fusion approaches in IoT-based smart cities: smart applications, taxonomies, challenges, and future research directions, Inf. Fusion 121 (2025) 103102, https://doi.org/10.1016/j.inffus.2025.103102.
[6] B.A. Mozzaquatro, C. Agostinho, D. Goncalves, J. Martins, R. Jardim-Goncalves, An ontology-based cybersecurity framework for the internet of things, Sensors 18 (2018) 3053, https://doi.org/10.3390/s18093053.
[7] A. Bazan-Muñoz, G. Ortiz, J.C. Augusto, A. Garcia-de-Prado, Taxonomy and software architecture for real-time context-aware collaborative smart environments, Internet Things 26 (2024) 101160, https://doi.org/10.1016/j.iot.2024.101160.
[8] A. Bazan-Muñoz, G. Ortiz, A. Garcia-de-Prado, Smart-CARE: integrating context awareness into smart health systems, in: J.C. Augusto (Ed.), Smart Health Handb. Stud. Health Technol. Inform. Ser., Sage/IOS Press, Amsterdam, The Netherlands, 2025.
[9] A. Alakari, K.F. Li, F. Gebali, A situation refinement model for complex event processing, Knowl.-Based Syst 198 (2020) 105881, https://doi.org/10.1016/j.knosys.2020.105881.
[10] A. Ivanchikj, S. Serbout, C. Pautasso, From text to visual BPMN process models: design and evaluation, in: Proc. 23rd ACMIEEE Int. Conf. Model Driven Eng. Lang. Syst, Association for Computing Machinery, New York, NY, USA, 2020, pp. 229–239, https://doi.org/10.1145/3365438.3410990.
[11] P. Gómez-Valiente, J.P. Benedí, J.M. Lillo-Castellano, M. Marina-Breysse, Smart-IoT business process management: a case study on remote digital early cardiac arrhythmia detection and diagnosis, IEEE Internet Things J 10 (2023) 16744–16757, https://doi.org/10.1109/JIOT.2023.3269820.
[12] D. Lübke, C. Pautasso, Empirical research in executable process models, in: D. Lübke, C. Pautasso (Eds.), Empir. Stud. Dev. Exec. Bus. Process., Springer International Publishing, Cham, 2019, pp. 3–12, https://doi.org/10.1007/978-3-030-17666-2_1.
[13] T. d'Hondt, A. Wilbik, P. Grefen, H. Ludwig, N. Baracaldo, A. Anwar, Using BPM technology to deploy and manage distributed analytics in collaborative IoT-driven business scenarios, in: Proc. 9th Int. Conf. Internet Things, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1–8, https://doi.org/10.1145/3365871.3365890.
[14] E. Mosqueira-Rey, D. Alonso-Ríos, V. Moret-Bonillo, Usability taxonomy and context-of-use taxonomy for usability analysis, 2009 IEEE Int. Conf. Syst. Man Cybern. (2009) 812–817, https://doi.org/10.1109/ICSMC.2009.5346929.
[15] L. Burgueño, J. Boubeta-Puig, A. Vallecillo, Formalizing complex event processing systems in Maude, IEEE Access 6 (2018) 23222–23241, https://doi.org/10.1109/ACCESS.2018.2831185.

[16] F. Freire Scattone, K.Rosa Braghetto, A microservices architecture for distributed complex event processing in smart cities, 2018 IEEE 37th Int. Symp. Reliab. Distrib. Syst. Workshop SRDSW (2018) 6–9, https://doi.org/10.1109/SRDSW.2018.00012.
[17] D.C. Luckham, Event Processing for Business: Organizing the Real-Time Enterprise, John Wiley & Sons, 2011.
[18] Espertech, Esper Documentation. https://www.espertech.com/esper/(accessed, 2024. January 1, 2026).
[19] G. Ortiz, A. Bazan-Muñoz, W. Lamersdorf, A. Garcia-de-Prado, Evaluating the integration of Esper complex event processing engine and message brokers, PeerJ Comput. Sci. 9 (2023) e1437, https://doi.org/10.7717/peerj-cs.1437.
[20] D. Corral-Plaza, G. Ortiz, I. Medina-Bulo, J. Boubeta-Puig, MEdit4CEP-SP: a model-driven solution to improve decision-making through user-friendly management and real-time processing of heterogeneous data streams, Knowl.-Based Syst. 213 (2021) 106682, https://doi.org/10.1016/j.knosys.2020.106682.
[21] M. Weske, W.M.P. van der Aalst, H.M.W. Verbeek, Advances in business process management, Data Knowl. Eng. 50 (2004) 1–8, https://doi.org/10.1016/j.datak.2004.01.001.
[22] R.K.L. Ko, S.S.G. Lee, E.W. Lee, Business process management (BPM) standards: a survey, Bus. Process Manag. J 15 (2009) 744–791, https://doi.org/10.1108/14637150910987937.
[23] H. Nasiri, S. Nasehi, M. Goudarzi, Evaluation of distributed stream processing frameworks for IoT applications in smart cities, J. Big Data 6 (2019) 52, https://doi.org/10.1186/s40537-019-0215-2.
[24] S. Langhi, R. Tommasini, E.D. Valle, Extending Kafka streams for complex event recognition, in: 2020 IEEE Int. Conf. Big Data Big Data, 2020, pp. 2190–2197, https://doi.org/10.1109/BigData50022.2020.9378217.
[25] M. Dayarathna, T. Suzumura, A performance analysis of System S, S4, and Esper via two level benchmarking, in: K. Joshi, M. Siegle, M. Stoelinga, P. R. D'Argenio (Eds.), Quant. Eval. Syst, Springer, Berlin, Heidelberg, 2013, pp. 225–240, https://doi.org/10.1007/978-3-642-40196-1_19.
[26] M.S. Aktas, M. Astekin, Provenance aware run-time verification of things for self-healing Internet of Things applications, Concurr. Comput. Pract. Exp. 31 (2019) e4263, https://doi.org/10.1002/cpe.4263.
[27] M. Daum, M. Götz, J. Domaschka, Integrating CEP and BPM: how CEP realizes functional requirements of BPM applications (industry article), in: Proc. 6th ACM Int. Conf. Distrib. Event-Based Syst, Association for Computing Machinery, New York, NY, USA, 2012, pp. 157–166, https://doi.org/10.1145/2335484.2335503.
[28] Q. Li, J. Wang, G. Chen, Y. Yuan, S. Ye, Business-oriented IoT-aware process modeling and execution, in: C. Xing, X. Fu, Y. Zhang, G. Zhang, C. Borjigin (Eds.), Web Inf. Syst. Appl., Springer International Publishing, Cham, 2021, pp. 747–755, https://doi.org/10.1007/978-3-030-87571-8_65.
[29] D. Ruiz-Fernández, D. Marcos-Jorquera, V. Gilart-Iglesias, V. Vives-Boix, J. Ramírez-Navarro, Empowerment of patients with hypertension through BPM, IoT and remote sensing, Sensors 17 (2017) 2273, https://doi.org/10.3390/s17102273.
[30] A. Fedeli, F. Fornari, A. Polini, B. Re, V. Torres, P. Valderas, FloBP: a model-driven approach for developing and executing IoT-enhanced business processes, Softw. Syst. Model. 23 (2024) 1217–1246, https://doi.org/10.1007/s10270-024-01150-8.
[31] P. Valderas, V. Torres, E. Serral, Modelling and executing IoT-enhanced business processes through BPMN and microservices, J. Syst. Softw. 184 (2022) 111139, https://doi.org/10.1016/j.jss.2021.111139.
[32] G. Jošt, J. Huber, M. Heričko, G. Polančič, An empirical investigation of intuitive understandability of process diagrams, Comput. Stand. Interfaces 48 (2016) 90–111, https://doi.org/10.1016/j.csi.2016.04.006.
[33] A. Dikici, O. Turetken, O. Demirors, Factors influencing the understandability of process models: a systematic literature review, Inf. Softw. Technol. 93 (2018) 112–129, https://doi.org/10.1016/j.infsof.2017.09.001.
[34] N. Ruhkamp, S. Schönig, Execution of multi-perspective declarative process models using complex event processing, Bus. Inf. Syst. (2021) 95–104, https://doi.org/10.52825/bis.v1i.51.
[35] C. Janiesch Jörn Kuhlenkamp, Enhancing business process execution with a context engine, (2021). https://doi.org/10.48550/arXiv.2110.04061.
[36] M.A. Fardbastani, F. Allahdadi, M. Sharifi, Business process monitoring via decentralized complex event processing, Enterp. Inf. Syst. 12 (2018) 1257–1284, https://doi.org/10.1080/17517575.2018.1522453.
[37] R. Gonçalves, J.J.M. Soares, R.M.F. Lima, An IoT-based framework for smart water supply Systems management, Future Internet 12 (2020) 114, https://doi.org/10.3390/fi12070114.
[38] P. Valderas, V. Torres, Towards a semantic interoperability in IoTEnhanced business processes. An event-driven solution based on microservices, 2023 19th Int. Conf. Intell. Environ. IE (2023) 1–8, https://doi.org/10.1109/IE57519.2023.10179096.
[39] P. Bazan, E. Estevez, Industry 4.0 and business process management: state of the art and new challenges, Bus. Process Manag. J. 28 (2021) 62–80, https://doi.org/10.1108/BPMJ-04-2020-0163.
[40] I.A. Changalima, I.J. Ismail, D. Amani, Driving SME performance through technological absorptive capacity and e-business innovation, Sustain. Technol. Entrep. 4 (2025) 100089, https://doi.org/10.1016/j.stae.2024.100089.
[41] E. Cano-Marin, The transformative potential of Generative Artificial Intelligence (GenAI) in business : a text mining analysis on innovation data sources, ESIC Mark. 55 (2024) e333–e333. https://doi.org/10.7200/esicm.55.333.
[42] I. Kodssi, H. Sbai, M. Kabil, Applying process mining to generate business process models from smart environments, Int. J. Comput. Digit. Syst. 15 (2024) 705–717, https://doi.org/10.12785/ijcds/160152.
[43] A. Beheshti, J. Yang, Q.Z. Sheng, B. Benatallah, F. Casati, S. Dustdar, H.R.M. Nezhad, X. Zhang, S. Xue, ProcessGPT: transforming business process management with generative artificial intelligence, 2023 IEEE Int. Conf. Web Serv. ICWS (2023) 731–739, https://doi.org/10.1109/ICWS60048.2023.00099.
[44] S. Mekimah, R. Zighed, K. Mili, I. Bengana, Business intelligence in organizational decision-making: a bibliometric analysis of research trends and gaps (2014–2024), Discov. Sustain 5 (2024) 532, https://doi.org/10.1007/s43621-024-00692-7.
[45] D.K. Jundt, M.K. Shoss, A process perspective on adaptive performance: research insights and new directions, Group Organ. Manag. 48 (2023) 405–435, https://doi.org/10.1177/10596011231161404.
[46] P.Q. Huy, V.K. Phuc, Unveiling how business process management capabilities foster dynamic decision-making for effectiveness of sustainable digital transformation, Bus. Process Manag. J. 31 (2025) 67–103, https://doi.org/10.1108/BPMJ-06-2024-0467.
[47] M.I. Cagnin, E.Y. Nakagawa, Towards dynamic processes-of-business processes: a new understanding, Bus. Process Manag. J. 27 (2021) 1545–1568, https://doi.org/10.1108/BPMJ-08-2020-0349.
[48] A. Bazan-Muñoz, G. Ortiz, A. Garcia-de-Prado, M. Cano-Crespo, J. Boubeta-Puig, E. Daneri-Vias, J.-M. Mariscal-Chavinet, Intelligent real-time anomaly detection for optimization of water monitoring systems, in: D. Gutiérrez, P. Millán, J. Blasco (Eds.), Smart Water Qual. Monit. Artif. Intell. Autom. Anal. Chem., Springer Nature Switzerland, Cham, 2026, pp. 157–182, https://doi.org/10.1007/978-3-032-03049-8_6.
[49] SAP Signavio Process Transformation Suite - Academic Login | Signavio, n.d, accessed, https://academic.signavio.com/p/login, 2026. accessed.
[50] P.W. Jordan, B. Thomas, I.L. McClelland, in: B. Weerdmeester (Ed.), Usability Evaluation In Industry, CRC Press, London, 1996, https://doi.org/10.1201/9781498710411.
[51] Esper Documentation, EsperTech (n.d.). https://www.espertech.com/esper/esper-documentation/(accessed, 2026.
[52] P. Caire, N. Genon, P. Heymans, D.L. Moody, Visual notation design 2.0: towards user comprehensible requirements engineering notations, 2013 21st IEEE Int. Requir. Eng. Conf. RE (2013) 115–124, https://doi.org/10.1109/RE.2013.6636711.