Computers and Electronics in Agriculture 154 (2018) 347–360

Contents lists available at ScienceDirect

Computers and Electronics in Agriculture

journal homepage: www.elsevier.com/locate/compag

Original papers

Rules engine and complex event processor in the context of internet of things for precision agriculture

Bertha Mazon-Olivo[a,b,*], Dixys Hernández-Rojas[a,b], José Maza-Salinas[a], Alberto Pan[b]

[a] Universidad Técnica de Machala, Academic Unit of Civil Engineering, AutoMathTIC Research Group, 5.5 Km Panamerican Av, Machala, El Oro, Ecuador
[b] Universidade da Coruña, Facultade de Informática, Campus de Elviña s/n, 15071 A Coruña, Spain

ARTICLE INFO

Keywords:
Complex event processing
Internet of Things (IoT)
Precision Agriculture
Rules engine
Real-time event processing

ABSTRACT

The Internet of Things (IoT) applications monitor large data flows and events in real time, some raw data is captured from devices located in wireless sensor networks (WSN) and used to make control decisions about actuators. This can be a major problem when the devices grow in number as well as the data that is captured. In this paper, we propose an architecture called "RECEP" for the dynamic processing of events generated in the context of IoT and Precision Agriculture (PA); it is made up of two components: Rules Engine (RE) and Complex Event Processor (CEP). RE allows you to configure dynamic rules conditioning input data from different sources and planning control actions on actuators, alerts, and notifications for end users or applications. The CEP component fuses the input data at the rate at which they arrive, with the rules established in the RE and it performs a prescriptive analysis that consists not only in predicting or detecting patterns of events, but in making automatic decisions. RECEP was implemented in a virtual machine with a 1.9 GHz CPU and 6 GB RAM, then it was integrated into an intelligent irrigation system of an experimental banana plot located in Machala-Ecuador. A WSN simulator was also used to generate sensor data in large quantities, the CEP was evaluated with several test cases, and results show that it consumes computational resources with a growth trend, represented by a logarithmic regression model (r-squared > 0.9); that is, the more events are processed, there is a minimum consumption of resources. It was tested for fifteen days; around 25 thousand events/s were processed. Our RECEP can be implemented in low-cost infrastructure typical of small and large banana producers.

1. Introduction

Today, the Internet of Things (IoT) is having great success in many areas, allowing computers to see, hear and detect parameters of the world (Qin et al., 2016). The number of connected devices is increasing, and according to CISCO, it is projected about 50 billion objects connected by 2020 (Evans, 2011). Morgan Stanley; however, forecasts that by the end of 2020, around 75 million objects will be connected (Rose et al., 2015). On the other hand, Huawei (2015) projects that there will be 100 billion of IoT devices installed, connected, and autonomously managed by 2025.

IoT is based on a system of interconnecting heterogeneous devices to the Internet; these devices have a unique identity, physical attributes, virtual personality, and use intelligent communication interfaces to transfer data to the network without human interaction (Vermesan and Friess, 2015). The devices constitute Sensor Networks and Wireless Actuators (WSN or WSAN) and at the same time, they are connected through a Gateway to a larger technological infrastructure such as a data center or cloud computing which is responsible for the storage, processing, analysis, and other data management processes. IoT has several fields of application such as Healthcare, Logistics; Manufacturing, Smart Home; Smart Cities, Precision Agriculture (PA); among others. In the case of PA, it consists of crop integration or farm management that combines information technology with rational agricultural industries and tries to provide quantities and types of supplies based on the real needs of a crop (Far and Rezaei-Moghaddam, 2017). In the context of IoT, raw data obtained from sensors and transmitted to a Cloud Computing is abundant, reaching high speeds in real time or near real-time; consequently, the task of monitoring and controlling WSN devices becomes complex and almost impossible to carry out by humans. It is for this reason that the need for a Complex Event Processor (CEP) appeared.

There are several CEP tools for IoT. For example Google offers its Real Time Stream Processing for IOT tool (Google Cloud Platform,

* Corresponding author at: Universidad Técnica de Machala, Academic Unit of Civil Engineering, AutoMathTIC Research Group, 5.5 Km Panamerican Av, Machala, El Oro, Ecuador.
E-mail address: bmazon@utmachala.edu.ec (B. Mazon-Olivo).

https://doi.org/10.1016/j.compag.2018.09.013
Received 16 February 2018; Received in revised form 3 July 2018; Accepted 10 September 2018
0168-1699/ © 2018 Elsevier B.V. All rights reserved.

2017) that deals with probabilistic events that are stored and processed by machine learning techniques to predict patterns. There is also WSO2 (WSO2, 2017), consisting of the CEP and BRS components; the CEP component identifies significant events and patterns of multiple data sources, analyzes their impact and generates real-time alerts; while BRS manages business rules. Companies like Microsoft, IBM, AWS, Oracle and others, also offer similar tools. In the scientific community there are several proposals such as: An intrusion detection systems (IDS) architecture based on the mechanism of event processing in IoT environments for real-time data computing (Jun and Chi, 2014). Bruns et al. (2015) show how CEP can be used as the key technology for intelligent M2M systems. Another example is a rule engine for smart buildings that guarantees real-time response and quick match between events and rules (Sun et al., 2015). Zhou et al. (2017) raised the following: Knowledge-infused and consistent CEP over real-time and persistent streams; a semantic CEP model (x-CEP) is introduced to query across real-time and persistent streams applied to case studies from Smart Grid. Akbar et al. (2017) worked an architecture which exploits historical data using machine learning for prediction in conjunction with CEP for Complex IoT Data Streams. Gökalp et al. (2018) proposed a visual programming framework for distributed Internet of Things centric complex event processing to users who have limited or no experience in these technologies.

After reviewing related works and software tools according to the topic discussed, although there are several options of RE and CEP for IOT, they are partial solutions still which are complex to develop, integrate and deployment; it requires a specialist in IoT and PA to take it to the actual applications. In addition, some of these tools are expensive and inaccessible to farmers in a developing country such as Ecuador. Another difficulty is the demands to process large data-events in IoT, the existing options require high-performance hardware or monthly payments for cloud service; or simply, they don't adjust to the needs of Precision Agriculture and Internet of Things applications. In the context of PA-IoT, the work proposed by Kamilaris et al. (2017), it is a semantic framework for IoT, based on smart farming applications, which support reasoning over various heterogeneous sensor data streams in real-time and it can integrate multiple cross-domain data streams. This work focuses only on rule design and detection of event patterns, but does not integrate prescriptive analysis, i.e. it does not make decisions.

This work proposes a kind of architecture for the automatic management of structured decisions generated in Precision Agriculture applications that use telemetry technologies and control of systems based on Internet of Things. For example the following: irrigation systems in crops, fertigation or plant nutrition; disease or pest control, animal monitoring; agricultural robots, machinery, greenhouses, etc. This architecture has been called "RECEP" because it integrates two Rules Engine components (RE) and Complex Event Processor (CEP); the contributions made are described in greater detail below:

A. The design of the RECEP architecture in order to facilitate user tasks that manage, monitor and control the processes of systems based on IoT as a multi-tenant SaaS service model. For example, software as a service for agricultural enterprises is organized in one or more plots. The implementation of RECEP intends to automate the complexity of raw data processing in WSN and manage the dynamism of the structured decisions to be generated. The architecture consists of two components: (1) RE that manages event models, decision rules models, configuration of control actions, alerts and notifications that are sent to applications or users; and (2) CEP is conceived to work as a service in the background performing tasks in parallel and in real time; its function is to process incoming events, and complete rules with data flows (from WSN, internal databases, external data such as web services), evaluate real-time rules and make decisions involving output events.
B. The proposed architecture has been verified in two ways: (1) Trough a use case focused on the monitoring and control of an irrigation system of a banana plot located on an experimental farm belonging to the Technical University of Machala (UTMACH) in Ecuador. RECEP was designed, built and integrated as a subsystem in the IoTMach platform that is an IoT platform created by the AutoMathTIC research group of UTMACH. And (2) by running functionality and performance tests considering the use of computational resources, carried out with raw data taken from the experimental banana farm and also generated by a simulator to evaluate large data flows at high speeds.

This paper is organized in the following aspects: related work, proposed RECEP architecture and its components; use case of the architecture in a monitoring and irrigation control system for a banana plot, evaluation and presentation of results, discussion and conclusions.

2. Related work

2.1. Internet of Things and Precision Agriculture

Precision Agriculture (PA) combines diverse Information and Communication Technologies (ICTs) for process and data management in an agricultural production system, allowing it to increase efficiency and productivity, applying a scientific approach during all phases of the production cycle of plants and animals. Data that can be collected or generated with ICTs (Bendre et al., 2015; Wolfert et al., 2017) can be obtained from: field (physical-chemical soil characteristics, topography, productivity data), climate, interpretation of camera or satellite images (space and/or temporal variability of the crop), yield maps, maps with input application prescriptions, historical data, other internal or external data. The integration and analysis of this data can be used to generate automatic decisions (structured decisions that are executed in devices, robots or machinery) or to support human decisions (semi-structured decisions). PA and IoT have the purpose to optimize resources (energy, water, fertilizers, pesticides, etc.) in order to produce more food with less effort, costs and environmental impact; all this in order to help farmers achieve higher productivity, economic benefits, greater sustainability and environmental protection (Ojha et al., 2015; Ray, 2017).

PA and IOT technologies have the challenge of optimizing agricultural work, automating processes such as: precision soil preparation, precision seeding, automatic irrigation, precision crop management, greenhouse management, phenotype measurement, integrated pest management, precision harvesting, data analysis and evaluation (Greenwood et al., 2014; Juul et al., 2015; Minet et al., 2017; Nikolidakis et al., 2015; Ojha et al., 2015; Srbinovska et al., 2015; Talavera et al., 2017; Wolfert et al., 2017). To monitor and control these processes, a variety of technologies have been created and applied such as: drones for field monitoring, machines and robots for routine operations; sensors and remote sensing, high precision positioning systems; Geomapping, automated information systems; variable rate technology (VRT), integrated electronic communications; cloud computing, mobile and fixed computing devices, data storage; big data, data science, etc. (Kaloxylos et al., 2013; Ray, 2017; Wolfert et al., 2017).

In all IoT domains (Smart Cities, Smart Home, Smart Buildings, Precision Agriculture, etc.), the connection of new devices (motes grouping sensors and/or actuators) to several WSNs is increasing each time, and through a IoT Gateway (Hernández-Rojas et al., 2018c) and the Internet, they are integrated into a cloud computing system that captures, stores, and processes data streams from a variety of sources. Depending on the number of devices and sensors and the frequency of data reading, the transport of raw data can be increased in volume and reach high speed (Zhou et al., 2017), generating inconveniences such as unpredictable latency, bottlenecks, incomplete or erroneous data, storage and processing difficulties, among other problems (Hao et al., 2017); more efficient processing systems are needed to deal with dynamic and diverse data, complex events and automation of decisions.

Such systems are known as Complex Event Processor (CEP).

2.2. Complex Event Processor (CEP)

CEP is a technology that allows processing, analyze and correlate large amounts of heterogeneous data in the form of events to detect patterns with direct implications in making decisions relevant to a particular domain (Boubeta, 2014). According Wang et al. (2018), CEP interprets and combines flows of primitive events, captured in most cases in real time, to identify composite events of higher level, known as event patterns or complex events. In addition, a CEP allows the processing of data flows from multiple data sources (Akila et al., 2016) and interprets a set of rules written in an ad hoc language for defining rules (Cugola and Margara, 2012). The CEP has a direct communication with a Rules Engine (RE), data server, communication protocols, notification server and other services and applications internal or external to the cloud (Wang et al., 2011). A CEP is reactive if it processes a complex pattern of events and generates automatic decisions, such as: sending control actions to devices, triggering alerts and sending notifications to applications and end users (Cugola et al., 2014; Wang and Cao, 2014; Zappia et al., 2011; Zhou et al., 2017). A CEP is considered proactive, predictive or probabilistic if it applies Machine Learning algorithms or data mining techniques to predict new events or generate trends (Akbar et al., 2017; Wang et al., 2018; Wang and Cao, 2014). In Table 1 some important milestones of the CEPs can be observed (Dayarathna and Perera, 2018; Luckham, 2007; Wang et al., 2016).

CEP uses and Benefits.

The cases of application of a CEP are focused on areas where it is essential to process large amounts of information that flow from outside the system; and, it is necessary to respond immediately with new complex events according to the pattern that has been triggered. A CEP has been applied in the continuous analysis of stocks to detect trends, supervision of business processes (Mayer et al., 2015), fraud detection in financial services, operational business intelligence, air traffic management or maritime, security supervision, as well as in Internet applications of Things, among other cases (Wang et al., 2016). Some examples of IoT domains that use a CEP are: Smart City, Smart Energy, Smart Manufacturing, Smart transport & logistics, Smart Health, Precision Agriculture, etc.

CEP Architectures

Several CEP architecture proposals have been found, these are detailed in Table 2.

CEP platforms and solutions. There are some open sources and commercial CEP platforms to consider, the most popular ones are described in Table 3.

CEP challenges in the context of IoT

- Traditional CEP is very much limited to the processing of queries on relational data; however, IoT requires the processing of data from multiple sources and formats (Zhou et al., 2017).
- For Predictive Analytics, is necessary to correlate raw data - present and historical events (Zhou et al., 2017).
- IoT devices and applications generate massive, real-time data that must be processed at high speed (Wang and Cao, 2014).
- In IoT applications, a CEP must capture events with some level of latency due to the integration of devices and applications from different networks (WSN, WAN, LAN) (Mayer et al., 2015).

2.3. Rules Engine (RE)

A rule engine is a component that allows you to create or modify business logic in a Business Process Management (BPM) system (Sun et al., 2015). RE provides rule models for a CEP to perform its function; through a user interface, it manages complex structures of simple and nested rules that involve arithmetic-logical operators and relational data or other content depending on their application.

There are several RE platforms that allow us to easily get a rule engine. Table 4 shows some known options.

3. The proposed architecture for the management of a Rules Engine and Complex Event Processor in the context of Internet of Things for Precision Agriculture (RECEP)

There are several reasons that justify the decision to propose this new architecture called RECEP. The purpose of this architecture to respond to the challenge of processing large data flows (throughput events/s) that arrive at high speed in real time or almost in real time from several WSN. The use case for which the architecture was focused is for precision agriculture applications and the Internet of things. Another challenge addressed is the use of minimal computational resources (CPU and RAM usage) in the CEP and therefore lower costs. In addition, a rules engine (RE) with an intuitive and easy-to-use user interface was designed to create rule models and establish complex event patterns that generate actuator control actions and notifications to users in PA-IOT applications. It is designed for users without knowledge of query or programming languages. The RECEP architecture is shown in Fig. 1.

The RECEP architecture is organized by the following components: (1) Gathering Data-Flow (GDF), (2) Rules engine (RE), and (3) Complex Event Processor (CEP). Each component is described below:

Gathering data-flow (GDF). This component corresponds to the source data that can be obtained from different sources: WSN, internal databases (agricultural company, plots; crops, soil types; irrigation system, configuration of IoT domains; WSN, devices or nodes with their respective sensors and actuators), web services or external databases that provide data such as meteorological information, production quotas or expected yields of a crop, etc. This component is located at the primary level (sensor level) of the IoTMach telemetry system, developed by the AutoMathTIC research group of the UTMACH. RECEP is also part of that system, located instead at the highest level (application level) and hosted on the servers of IoTMach. Therefore, RECEP takes advantage of all current benefits in this telemetry system, developed by

Table 1
Evolution of CEPs.

Year	Description	Fields of application
1960	Modeling and simulation of physical systems e.g. weather, avionics, etc.	Simulation of discrete events
1969	Transmitting packets of data-protocols developed to control the sequence, ordering, processing of the events. TCP, Ethernet, etc.	Networks
1980	Databases that could detect change and raise events/triggers, Event Condition Action (ECA)	Active DBs
1985	Intermediate layer to facilitate the integration of applications under Pub/Subscribe models	Middleware/MOM/SOA
2000–2007	Event processing evolved. First Simple Event Processing, then Event Stream Processing and finally Complex Event Processing	Business Activity Monitoring, financial trading applications
2008–2018	Propagation of CEP applications and evolution of processing techniques. Complex Event Processing (CEP) Engines, Event Processing Software, Distributed Stream Computing Platforms (DSCP), Event Processing Platforms (EPP)	IoT Applications and Intelligent Data Analysis (data mining, machine learning and other techniques). Big data stream processing from multiple sources and identifying patterns to generate a real-time response

Table 2
Some examples of CEP architectures.

Name	Description
Client-server architecture	Processes and extracts data for separate detection and reasoning procedures, reducing network consumption and computational load (Akbar et al., 2016)
SCEPter System	Semantic CEP on end-to-end event streams incorporating χ-CEP event and query models. It consists of three parts: the real-time query engine, the semantic archive subsystem, and the query planner (Ding et al., 2016; Zhou et al., 2017)
Proactive CEP	A proactive CEP method is proposed for large-scale transportation IoT, based on a multilayered adaptive dynamic Bayesian model (Wang and Cao, 2014)
Architecture: Real-Time Stream Processing for IoT	Infrastructure that manages real-time data streams from IoT smart devices and handles the processing, storage and analysis of data from hundreds of millions of events/hour (Google Cloud Platform, 2017)
Context-Aware Stream Processing for Distributed IoT applications	Approach based on the automatic, unsupervised learning method to group the underlying data into different events (Akbar et al., 2016)

the authors in previous works. One of the featured features that GDF incorporates is the Plug-and-play Human-centered Virtual Transducer Electronic Data Sheet (VTEDS)-based framework, which makes the GDF highly scalable, allowing unlimited new nodes to be added to the system dynamically through plug-and-play mechanisms. This framework is based on the IEEE 21451 VTEDS, which enable the interoperability of various intelligent sensors with different communication interfaces. A detailed explanation of this architecture and principle of operation of the plug-and-play mechanisms used are described in (Hernández-Rojas et al., 2018a). In addition, the sensor nodes make use of a Lightweight Protocols for Sensors (LP4S) family of sub-protocols (LP4S-6, LP4S-X, and LP4-J) that allow sensors to self-describe, autocalibrate, and auto-register. With the use of these protocols, an IoT solution with low latency, low power consumption and quality of service (QoS) are guaranteed. More details on the LP4S family can be found in (Hernández-Rojas et al., 2018b).

Rules engine (RE). This component is the graphical interface to interact with the user; it allows the management of events, rules, actions, alerts, and notifications. It is composed of the following subcomponents:

- **Event Engine.** It is the integrator subcomponent in charge of event management. An event consists of: rules, control actions for actuators, alerts and notifications to users. The RE sends a message to the CEP to update the models of events and rules loaded in memory.
- **Rules.** This module is designed to create, edit or delete rule patterns considering the data sources, actions and notification associated when the rule is fulfilled. A rule is composed of source data (input variables) and mathematical operators (arithmetic, logical, relational, etc.). A rule may also contain nested rules. Typical input variables are the sensor data, which are often contaminated with noise from the signal, the medium or elements of the data acquisition system. These noises must be treated at the level of the sensor node, incorporating active or passive filters by software or hardware depending on the smart transducer used. Techniques of averaging several sampled data and sending this data only when the value varies one percent (predefined) difference with the previous measurement are commonly used by IoT developers. When the noise is well characterized, a hysteresis can be incorporated into the rules, thus avoiding undesired oscillations in the output variables (actions). In the next versions of our system, the hysteresis management will be added in the rules editor. However, the current graphical interface allows us to demonstrate the functionality of our proposal.
- **Actions.** This module is designed to create, edit or delete control actions that must be executed on analog or digital actuators (electrovalve, motor, hydraulic bomb, etc.). Actions are classified as: automatic (running when a rule is fulfilled), repetitive (running a number of times every certain time) and planned (running every certain time during a period of time).
- **Notifications.** To configure alerts and messages for users and/or applications. Messages can be displayed in the RE log and sent via email, SMS or other means.
- **Storage of rules and events.** Module that allows you to store rules and events in a database with their respective action and notification settings.

Complex Event Processor (CEP). It is in charge of completing the rules created in the RE, fusing them with the data streams collected through GDF; if any rule is completed with the source data, it is automatically evaluated and if it is fulfilled, the associated events are triggered. The CEP is designed to work with a configurable time window consisting of a very small waiting period (ms) to re-analyze all rules with the source data, distributing the thread processing (parallel processing). The sub-components are described below:

Protocol adapter. Used as a communication protocol, it is responsible for sending and receiving messages (reading data or control actions) between cloud computing, gateway and WSN. This module is based on an MQTT messaging broker. In our case, we use the Mosquito (Eclipse Mosquitto, 2018) open broker. This module is of vital importance in the operation of the entire IoT telemetry system, of which RECEP is part. MQTT (Message Queuing Telemetry Transport) is an ISO standard based on pub/sub messaging mechanism (publish/subscribe), designed for remote connections in constraining sensor nodes with reduced memory capacity. Among many advantages MQTT allows the use of different manufacturers and technologies of gateways and sensor nodes of the WSN, for this they must only incorporate in their firmware MQTT clients. An intrinsic feature of this protocol is the three levels of quality of service (QoS) it includes. These are QoS 0, QoS 1 and QoS 2, which allow delivery to message at most once (fire and forget), delivery

Table 3
Some popular CEP solutions.

Product	Description
Apache Flink	An open-source stream processing framework for distributed, high-performing, always-available, and accurate data streaming applications (Apache Flink, 2017)
WSO2 CEP	Identifies events and patterns from multiple data sources, analyzes their impacts, and acts on them in real time (WSO2, 2018)
Stream Analytics (Microsoft Azure)	Develop and run massively parallel real-time analytics on multiple IoT or non-IoT streams of data using simple SQL like language (Microsoft Azure, 2017)
Amazon Kinesis Data Streams	It collects, processes, and analyzes real-time, streaming data so you can get timely insights and react to new events (Amazon, 2018a)
IBM InfoSphere Streams	A real-time analytics solution with development environment, runtime and analytic toolkits (Sakr, 2013)

Table 4
Rules engine platforms.

Product	Description
Drools (JBoss)	Drools is a business rules management system (BRMS) solution. Provides a core Business Rules Engine (BRE), a rule management and web creation application (Drools Workbench) and an Eclipse IDE plugin for core development (RedHat, 2006)
WSO2 BRS	It is a Business Rules Management System (BRMS) that provides the ability to define, implement, monitor and maintain an organization's business decisions and expose them as Web services (WSO2, 2017)
Rules for AWS IoT	Enables devices to interact with AWS services. Rules are analyzed and actions are performed based on the flow of MQTT topics. (Amazon, 2018b)

at least once (acknowledged delivery) and delivery exactly once (assured delivery) respectively. The WSN-IoT typically consists of low resource devices and power constraint, exposed to failures, shutdowns, and disconnections continuously. In these cases, it is recommended to implement clients with QoS 1 or QoS 2 with a persistent session, which allows for saving all relevant information for the sensor node on the broker. Thus, this feature allows achieving a robust and stable system against this type of failure. An excellent starting point for the study of MQTT, QoS and persistent session can be found in (HiveMQ, 2018).

– **Data Streaming**. Manages the transfer of data readings (real time or intermittently) from IoT devices or sensor nodes (Arduino, Raspberry Pi, Waspmote, Beaglebond, etc.) and arrive through an IoT Gateway to the Protocol Adapter located in the cloud. This subcomponent also performs the function of a log management system, designed for data transfer at high speeds and large volumes; that is, the data is stored temporarily until it is processed. For the storage of large volumes of data that arrive at high speed in real time, the use of Apache Kafka is proposed. Kafka is a Distributed Messaging System for Log Processing (Apache S. F., 2017; Díaz et al., 2016), tested and used by many companies for data streams and distributed storage.
– **Data Collector**. It is in charge of capturing the source data that generates input events and it is necessary to complete the rule patterns created in the RE. Source data can be collected from different sources: IoT devices that are part of a WSN, IoT Gateway, internal or external databases; they can be in a variety of formats: structured or relational and semi-structured (JSON: JavaScript Object Notation, CSV: comma-separated values, etc.); in addition, they can arrive in large volumes, at high speeds, in real time or near real time.
– **Event Extraction.** It updates the event models in main memory, as long as you receive a RE message stating that a change has been made to the rule models.
– **Event matching**. This sub-component, when input events have been generated, is in charge of fusing the values of the preconfigured source data with the rule patterns and verifying whether any rules have been completed; if this happens, it automatically invokes Maker Decision to evaluate the rule. Rule processing is designed to run on several tasks in parallel (multi-threaded execution) and work with a very small configurable time window (ms) waiting, before repeating the rule checking process again.
– **Decision maker (DM).** It manages output events through automatic control actions. It is in charge of evaluating a rule; if all its conditions are fulfilled, it performs a prescriptive analysis generating the most appropriate decisions in exit events and transferring the order to the Actions Dispatcher. The rules formed with sensor data as input, are always evaluated with the information available in the database. In case, that the sensor node associated to the input of the rule is failing or simply is in sleep mode, simply the rule is not

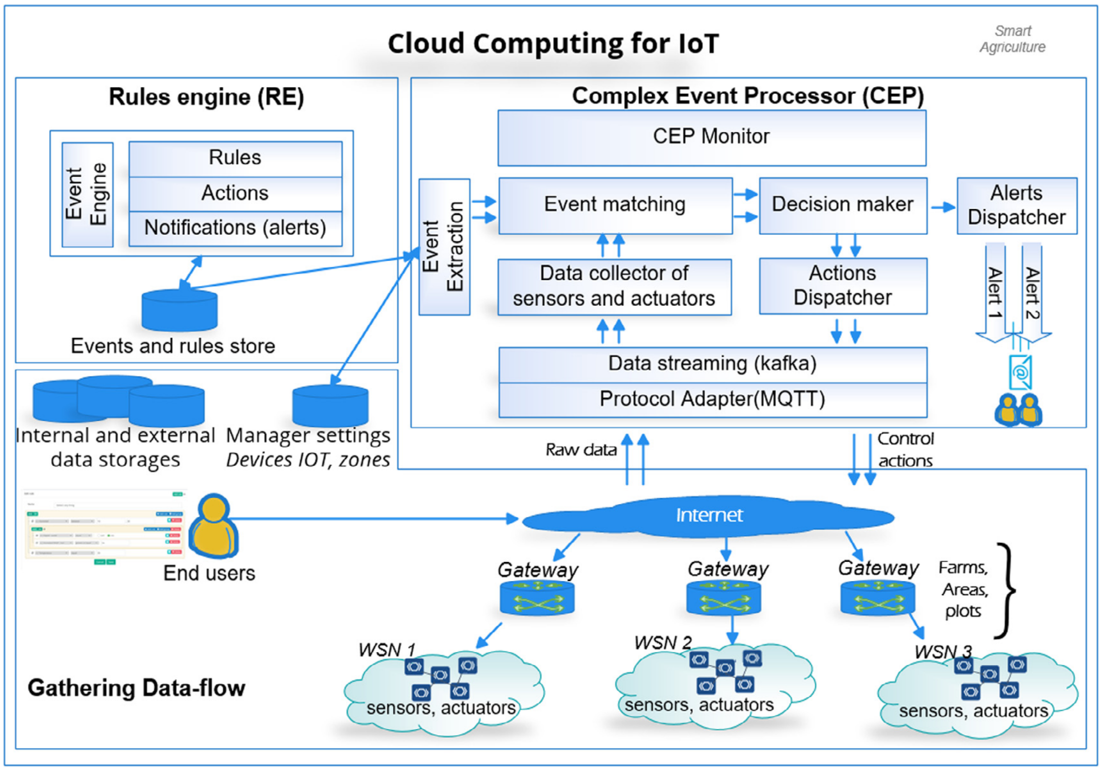


Fig. 1. RECEP Architecture.

evaluated until said sensor can update said input.

– **Actions dispatcher**. It manages control actions such as packets pending shipment and even has a history of actions already executed for a period of time. When you receive the DM order, it automatically sends the respective control action to an IoT device in the WSN to change the status of an actuator (ON, OFF or other value depending on the type of actuator). It also dispatches repetitive and planned actions that are not dependent on DM. At the same time that control actions are dispatched, the Alerts Dispatcher is also activated if the case calls for it. When an action is directed towards constraining sensor nodes, these may be damaged or just in sleep mode. In these cases, if the node was programmed with a QoS 1 or 2 client with a persistent session, the action must be executed regardless of the moment that occurs. Then when the node is active again, it will receive the respective order. In the case of QoS 0 clients, the dispatcher will retry the order three times and then it will be eliminated, to prevent the action from occurring outside the time defined for the given application domain.
– **Alerts dispatcher**. It manages the activation process - sending alerts and notification to users and/or applications by email or other means.

The interaction between each component and the way they communicate with each other is shown in Fig. 2.

```
{
    package_id:"1000",
    mac: " 11:11:11:11:11:ab",
    data:[
           {interface: "ind0", value:"1"},
           {interface: "ina0", value:"45"},
           {interface: "outa0", value:"14"},
         ],
    date: 2018/02/06 14:00"
}
```

Fig. 3. Data format received from an event.

4. Application of the RECEP architecture, use case: irrigation control system in a banana plot

4.1. Description of operation of the RECEP architecture-based subsystem

The GDF component

For this case, message interchange format based on subprotocol Lightweight Protocol for Sensor - Json (LP4S-J) was used between the IoT Gateway and the CEP as shown in Fig. 3. A data flow (raw data) sent by a IoT device and transmitted through a gateway to the CEP, is identified by "package_id", "MAC" is the physical address of the device that sends the data flow, "interface" is the pin where a sensor (in) or actuator (out) is connected; the interfaces are of various types: "ind" = digital input, "ina" = analog input, "outa" = analog output, "outd" = digital output and "date" is the date and time of the reading.

In order to make the processing of rules easier, the data flow in Fig. 3 is transformed into simple and easily processable input events as shown in Fig. 4, where it differs by interface and value.

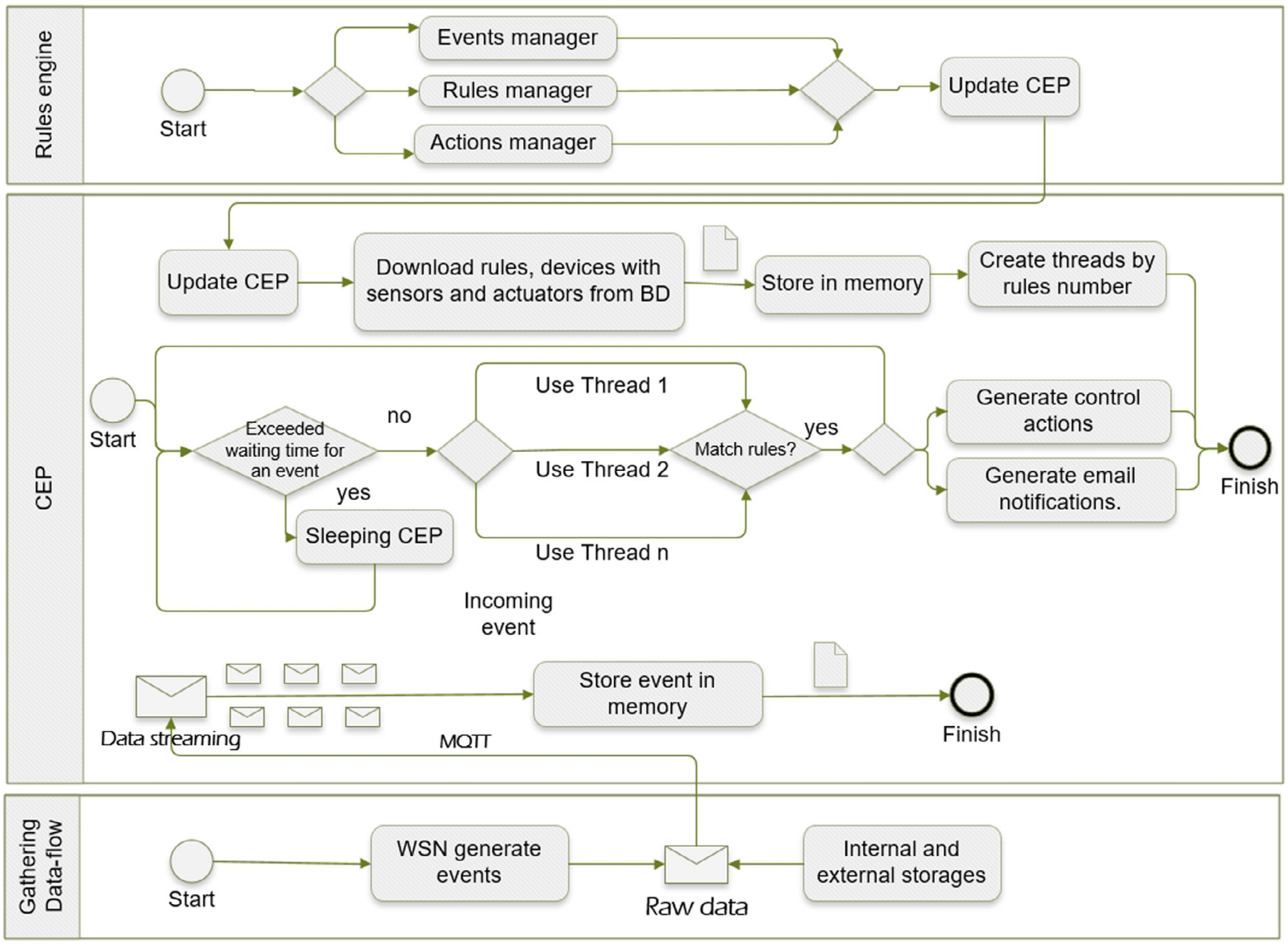
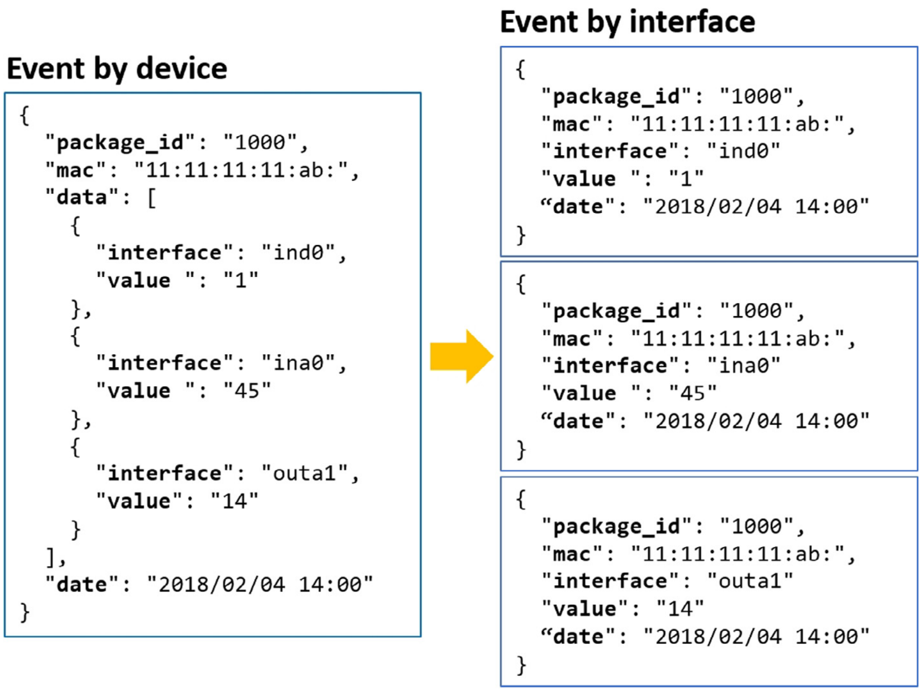


Fig. 2. Flow Diagram of GDF, CEP and RE.



```
{
  "package_id": "1000",
  "mac": "11:11:11:11:ab:",
  "data": [
    {
      "interface": "ind0",
      "value ": "1"
    },
    {
      "interface": "ina0",
      "value ": "45"
    },
    {
      "interface": "outa1",
      "value": "14"
    }
  ],
  "date": "2018/02/04 14:00"
}
```

```
{
  "package_id": "1000",
  "mac": "11:11:11:11:ab:",
  "interface": "ind0"
  "value ": "1"
  “date": "2018/02/04 14:00"
}
```

```
{
  "package_id": "1000",
  "mac": "11:11:11:11:ab:",
  "interface": "ina0"
  "value ": "45"
  “date": "2018/02/04 14:00"
}
```

```
{
  "package_id": "1000",
  "mac": "11:11:11:11:ab:",
  "interface": "outa1"
  "value": "14"
  “date": "2018/02/04 14:00"
}
```

Fig. 4. Parse from Event by device to event by interface.

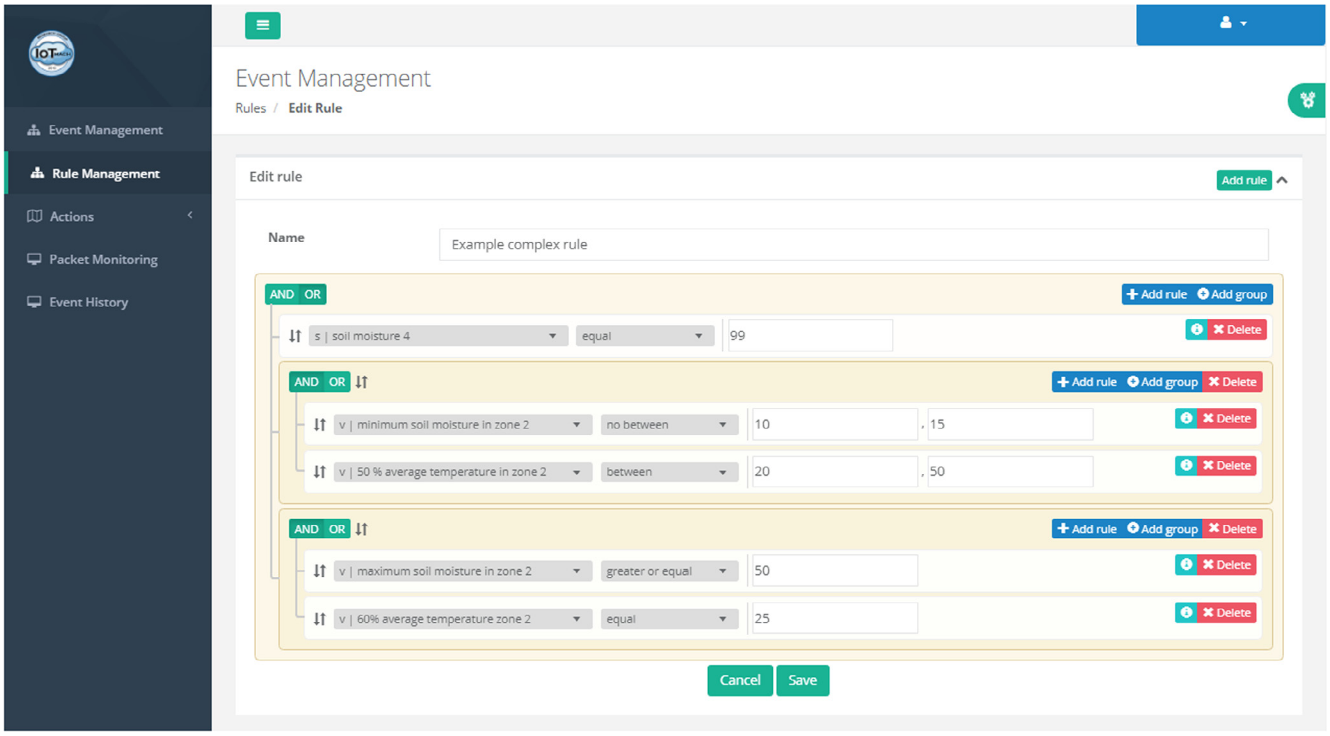
To this complete data structure, we divide it in such a way that now the event will no longer be by classified by device, but rather by interface or sensor as shown in facilitating the processing of rules.

The RE component

Fig. 5 shows the graphical interface of RE, on the left side you can see a menu of options to navigate between Event Manager, Rule Manager, Actions, Package Monitoring and Event History. In the middle of the screen, an example of how the user edits a rule is displayed.

Events Manager. It creates or modifies an event by integrating rules, control actions and alerts/notifications. Fig. 6 shows a simple example of an event, where the decision rule checks if Soil moisture1 is less to lower threshold; if the rule is fulfilled, the irrigation system in zone 1 is activated and the administrator is notified of the event occurring.

Rules management. An example of how the CEP reads a rule is shown in Fig. 7. The rule is made of input variables (sensors id/actuators: *x1,:x2,:y1,:y2*), comparison operators (= =, > , <), logical operators (*and, or, not*) for sub-rules, values with which they are



Name: Example complex rule

s | soil moisture 4 — equal — 99
v | minimum soil moisture in zone 2 — no between — 10 , 15
v | 50 % average temperature in zone 2 — between — 20 , 50
v | maximum soil moisture in zone 2 — greater or equal — 50
v | 60% average temperature zone 2 — equal — 25

Fig. 5. RE graphical interface.

```
Event:
Rule         => Soil moisture1 of zone1 is less to lower threshold
Actions      => Soil moisture1 of zone1 is less to lower threshold
Actions      => - A1: Open solenoid valve2 of zone1
                - A1: Open solenoid valve2 of zone1
                - A2: Turn on bomb2 which extracts water from a well
Notification => Sends a message to admin indicating that the
                irrigation system has been activated in zone1
```

Fig. 6. Event structure (rule, actions, alerts/notifications).

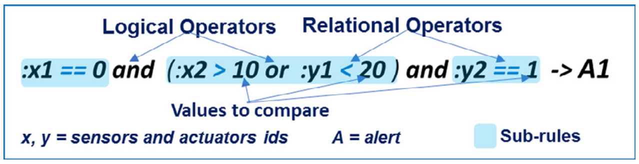

Logical Operators
Relational Operators
:x1 == 0 and (:x2 > 10 or :y1 < 20) and :y2 == 1 -> A1
Values to compare
x, y = sensors and actuators ids
A = alert
Sub-rules

Fig. 7. Example of a rule.

compared (*0, 10, 20, 1*), the symbol "- > " denotes "then", the expected decision is written down below, that is, *A1* is the id of the action that needs to be executed if the rule is fulfilled; several control actions can be placed in the consequent and a multi-user alert/notification.

A more complex rule can integrate data collections where it is possible to apply aggregation functions (minimum, maximum, average, etc.). For example, to average the values generated by two or more humidity sensors installed at different depth levels in the same irrigation area or plant. It is also possible to add new variables resulting from a custom mathematical expression that generates a calculated data; for example, the percentage applied to a simple input variable.

Management of control actions, alerts and notifications. If a rule is fulfilled, actuator control actions, alert and notification messages for users to be sent by email are configured.

The CEP component

It runs as a background service and is responsible for processing events that integrate rules, control actions and alerts/notifications. The events are loaded into RAM memory and a configurable runtime time window (tw) is used, which by default was set to 100 ms, and it represents the waiting time before fusing the rules with the source data again; once the rules are completed, it proceeds to evaluate them and then automatically generates the decision of which action, alert or notification to execute.

5. Experiments

This section describes the first case study to which the proposed RECEP architecture has been applied. This case was implemented as a subsystem of the IoTMach platform that was created for the field of Precision Agriculture, specifically for banana cultivation in Ecuador. Although tested on one experimental banana field, it is designed to grow and support multiple companies in the Ecuadorian banana sector.

5.1. Experimental setup

The Experimental Place was carried out at the Universidad Técnica de Machala (UTMACH), located in the province of El Oro, south of Ecuador, in the geographical coordinates: latitude 3°15′30″ S and longitude: 79°57′37″ O.

5.1.1. Materials and tools used for the development and implementation of the subsystems based on the RECEP architecture

The GDF component, was worked in two scenarios: (1) A WSN simulator was developed in the NodeJS language to generate large volumes of random sensor data; then, this tool was used to simulate two WSNs, with four Waspmote and four Raspberry PI 3 + WIFI devices, also two IoT Gateway devices deployed with two Raspberry PI 3 devices (see Fig. 8). (2) A physical WSN was implemented in the experimental banana field located in Santa Inés farm managed by the public company of UTMACH with the following resources: An IoT Gateway using Raspberry PI 3, and sensor nodes: Arduino Mega for the solenoid valve control panel and tree Raspberry PI 3 for sensor nodes; one temperature sensor, 6 soil moisture sensors, and 8 solenoid valves.

The RE component was developed in the programming language Python + Framework Django and the Nginx web server.

The CEP component was developed in the NodeJS language as a

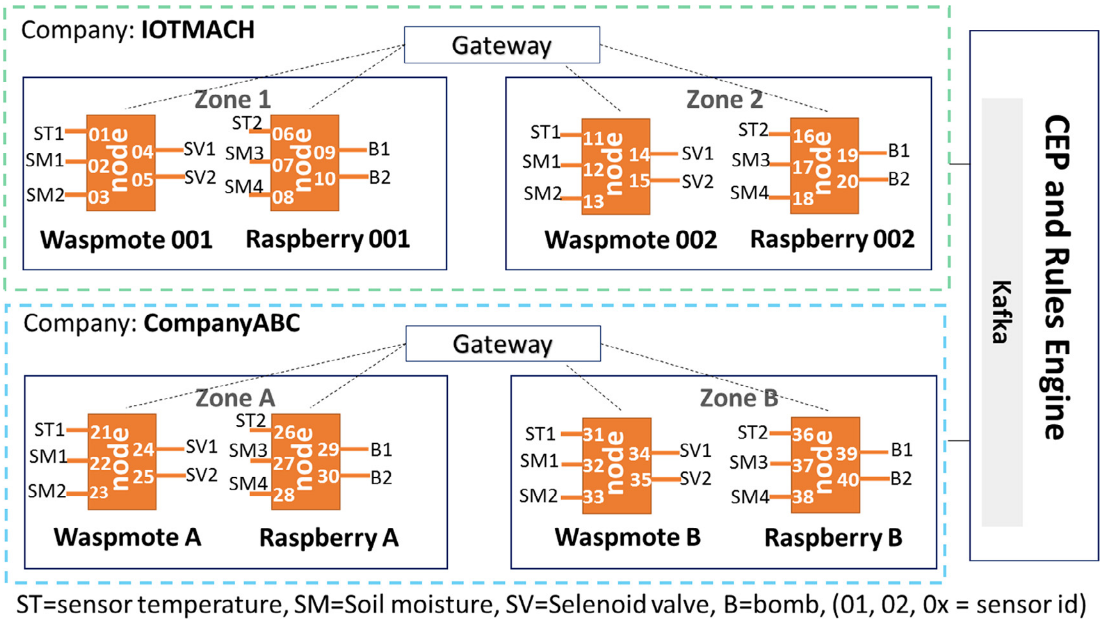

Company: IOTMACH
Gateway
Zone 1
Zone 2
ST1 SM1 SM2 01 02 03 node 04 05 SV1 SV2
ST2 SM3 SM4 06 07 08 node 09 10 B1 B2
Waspmote 001
Raspberry 001
ST1 SM1 SM2 11 12 13 node 14 15 SV1 SV2
ST2 SM3 SM4 16 17 18 node 19 20 B1 B2
Waspmote 002
Raspberry 002
Company: CompanyABC
Gateway
Zone A
Zone B
ST1 SM1 SM2 21 22 23 node 24 25 SV1 SV2
ST2 SM3 SM4 26 27 28 node 29 30 B1 B2
Waspmote A
Raspberry A
ST1 SM1 SM2 31 32 33 node 34 35 SV1 SV2
ST2 SM3 SM4 36 37 38 node 39 40 B1 B2
Waspmote B
Raspberry B
Kafka
CEP and Rules Engine
ST=sensor temperature, SM=Soil moisture, SV=Selenoid valve, B=bomb, (01, 02, 0x = sensor id)

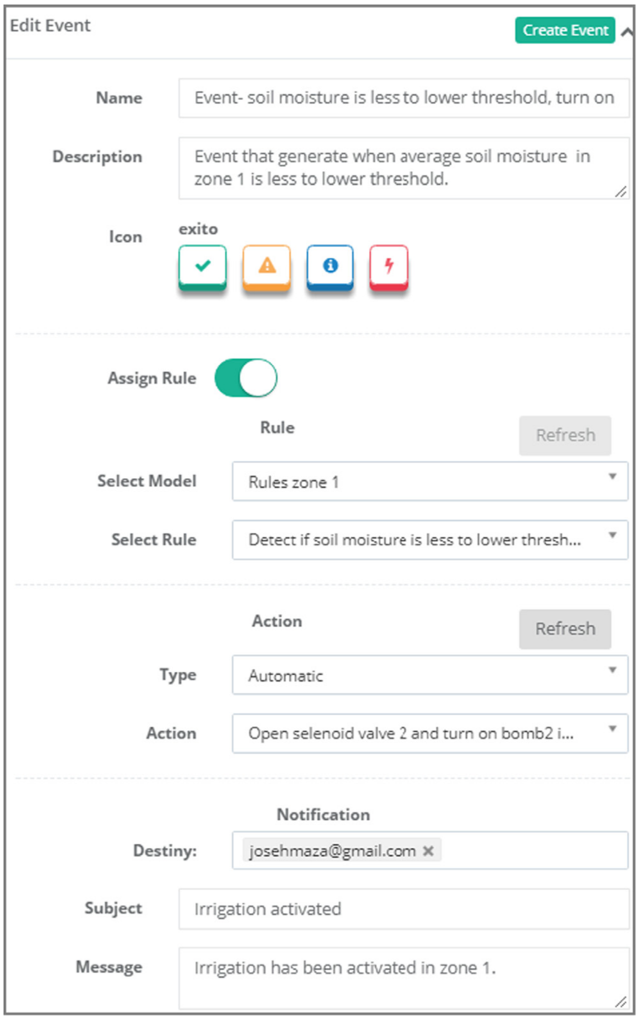
Fig. 8. Scenario of use case.

```
z1: Irrigation zone 1
h:  Average humidity of sensors located in zone 1 (soil moisture 1, soil
    moisture 2, soil moisture 3, soil moisture 4)
Oi: Lower irrigation threshold (30)
Os: Upper irrigation threshold (50)
Cc: Soil field capacity (52)

Event 1: Average humidity is less than the lower irrigation threshold.
Rules
R1: If  h < Oi then
- Open solenoid valve 2 of z1
- Activate bomb 2 which extracts water from a well.
- Send a message to the administrator, indicating that irrigation has
  been activated in z1, at a specific date and time

Event 2: Average humidity is higher than upper threshold or greater than
          field capacity
R2: Si h>= Os or h >= cc then
- Turn bomb 2 off and close solenoid valve 2
- Send a message to the administrator to end irrigation in z1,
  indicating: date and time of end.
```

Fig. 9. Examples of events designed for an automation scenario of an irrigation system.



Name	Event- soil moisture is less to lower threshold, turn on
Description	Event that generate when average soil moisture in zone 1 is less to lower threshold.
Icon	exito
Assign Rule	✓
Select Model	Rules zone 1
Select Rule	Detect if soil moisture is less to lower thresh...
Type	Automatic
Action	Open selenoid valve 2 and turn on bomb2 i...
Destiny:	josehmaza@gmail.com
Subject	Irrigation activated
Message	Irrigation has been activated in zone 1.

Fig. 10. Graphical interface for configure a complex event. A complex event consists of rules models, data sources, control actions and user notifications.

background service, which analyzes the events and rules, distributing the processing of events in tasks or threads in parallel. The MQTT Mosquitto broker was used for the sub-component Protocol Adapter, and Apache Kafka was used for the Data Streaming sub-component. The PostgreSQL database manager was used for both RE and CEP. The implementation of this sub-system was carried out in a virtual machine (with 1.9 GHz single-core CPU, 6 GB of RAM and 100 GB of Hard Disk) of the cloud computing server located in the Academic Unit of Civil Engineering of UTMACH.

5.2. Functionality demonstration using WSN simulator

The following scenario was designed to obtain results: Two companies were created with their respective WSN, gateway and irrigation zones (IoTMach: Zone1 and Zone2) and (Company ABC: Zone A and Zone B); and the following aspects were integrated: 2 sensor nodes, 2 temperature sensors (ST1, ST2), 4 humidity sensors (SM1 to SM4), two solenoid valves (SV1, SV2), and two hydraulic bombs (B1, B2). As shown in Fig. 8 two examples of complex events that group: actions and alerts-notifications are shown in Fig. 9; these events were recorded in the RE and updated in the CEP through the *Event Extraction* sub-component.

The CEP collected data flows (*Data Collector*) through the MQTT *Protocol Adapter* and the *Data Streaming* system with Apache Kafka.

The CEP then completes the rule patterns stored in memory with incoming data streams coming from the WSN (*Event Matching*). If a rule is fulfilled, output events are managed via *Decision Maker* and the automatic execution of control actions (*Dispatcher actions*) that affect actuators and the sending of alerts and notifications (*Dispatcher Alerts*) to users.

Fig. 9 shows two examples of event models applied in the automation of an irrigation system; *Event1* and rule *R1*: if the average humidity of sensors located in *Zone1* is less than or equal to the lower irrigation limit(*Oi*), open the solenoid *valve2*, activate *bomb2* and send a notification to the user admin informing the event, date and time; *Event2* and rule *R2* are also created: if the average humidity of sensors located in *Zone1* is higher than the upper threshold(*Os*) or field capacity (*cc*) then turn off *bomb2*, close solenoid *valve2* and notify the user admin of the event occurred.

The *Event Complex 1* configuration in RE is shown in Fig. 10, where

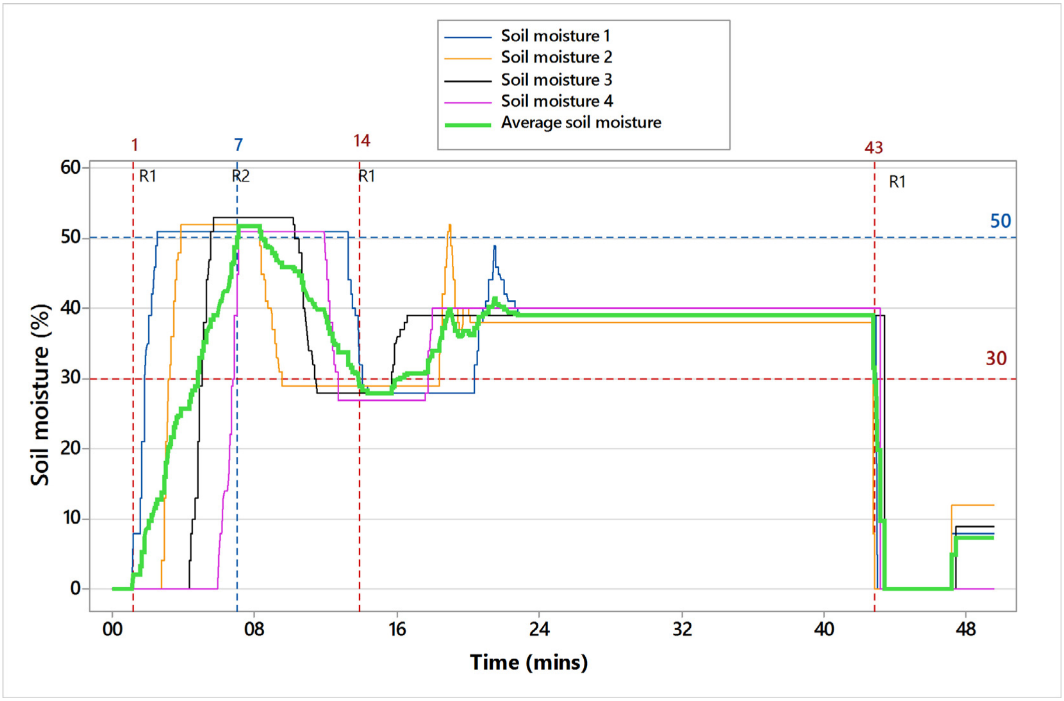


Fig. 11. Evaluation of the R1 and R2 rules as completed by the average data streams of the moisture. The green line represents the average soil moisture sensor data used in rules R1 that activates the irrigation system and R2 to turn off. The vertical and horizontal dashed lines in red indicate that the rule R1 has been fulfilled. Also, dashed blue lines indicate that rule R2 has been met.

Table 5
Sensors and actuators used in the experimental plot of banana.

Type of sensor	Number sensors/he	# sensor readings/h	Total measurements/h
Two-level soil moisture sensors	12	60	720
Temperature sensor	1	60	60
Water pressure meter	1	60	60
Actuators (Electro-valves)	8	1	8

a rule model is selected, then the rule associates whether it is simple or complex; finally, the control action and the alert/notification to be sent by email to the users when the rule is fulfilled are specified. In a similar way other events are created.

Fig. 11 shows how rules *R1* and *R2* are completed with the sensor data streams, and are immediately evaluated to determine if these are fulfilled. The vertical and horizontal dashed lines in red indicate that the rule *R1* has been fulfilled. Also, dashed blue lines indicate that rule *R2* has been met and consequently, the planned control actions in the actuators as well as the alerts/notifications to users must be generated. The green line represents the average soil moisture in *Zone1*.

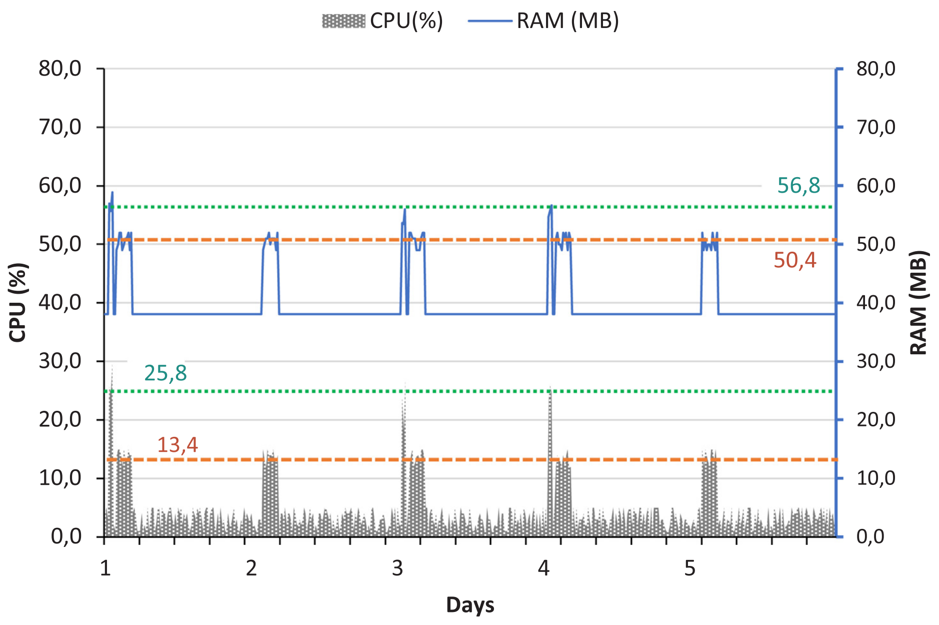


Fig. 12. Use of CPU and RAM in the CEP applied in the experimental banana crop. The horizontal reference lines in red indicate: Avg CPU consumption (13.4%) and Avg RAM used (50.4 MB), when processing incoming data flows (throughput). The horizontal reference lines in green represent Avg CPU consumption (25.8%) and Avg utilization of RAM (56.8 MB), when the rules models are updated in Rule Engine (RE).

Table 6
Measurement of CPU and RAM usage for several cases of Throughput (events/s). CEP Window size is the time window in seconds or milliseconds, which the CEP uses to evaluate rules and data flows (throughput).

Events/s	Events/day	Total events	Rule models	CEP Window size (s)	Rules Evaluated/day	Avg CPU (%)	Avg RAM (MB)
0,2	**1696**	**25 434**	**10**	**1**	**72 000**	**13.4**	**50.4**
1	7200	108 000	14	0.1	1 008 000	25.6	55.6
3	21 600	324 000	16	0.1	1 152 000	32.7	59.3
10	72 000	1 080 000	20	0.01	14 400 000	40.4	63.2
100	720 000	10 800 000	50	0.01	36 000 000	55.2	70.9
1200	8 640 000	129 600 000	600	0.001	4 320 000 000	71.2	79.1
4200	**30 240 000**	**453 600 000**	**1200**	**0.001**	**8 640 000 000**	**75.4**	**83.2**
10 000	72 000 000	1 080 000 000	5000	0.001	36 000 000 000	84.8	86.1
20 000	144 000 000	2 160 000 000	10 000	0.001	72 000 000 000	89.3	88.4
25 000	180 000 000	2 700 000 000	10 500	0.001	75 600 000 000	90.7	89.1

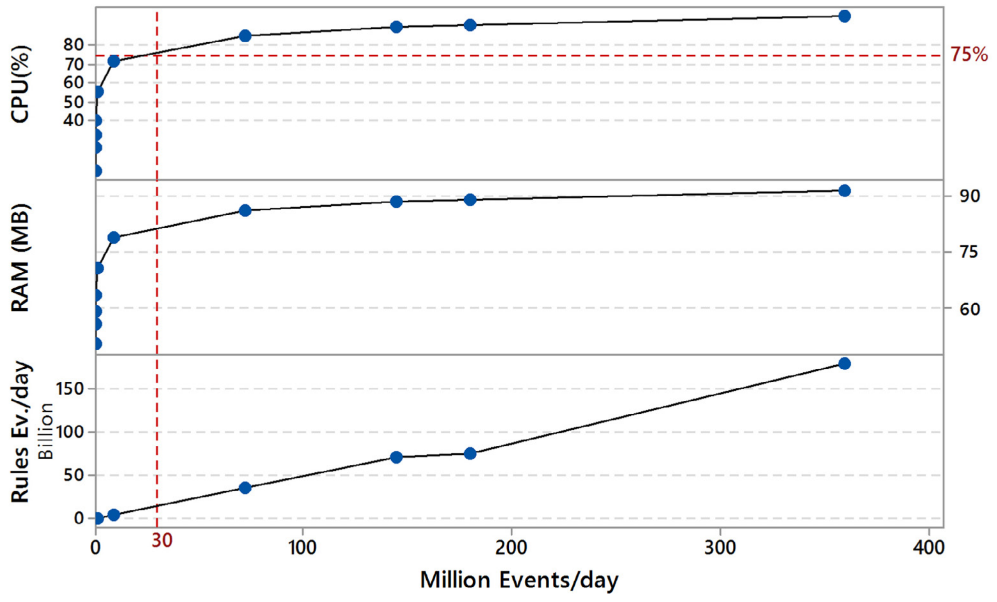


Fig. 13. Throughput with CPU and RAM usage in Test Case 2.

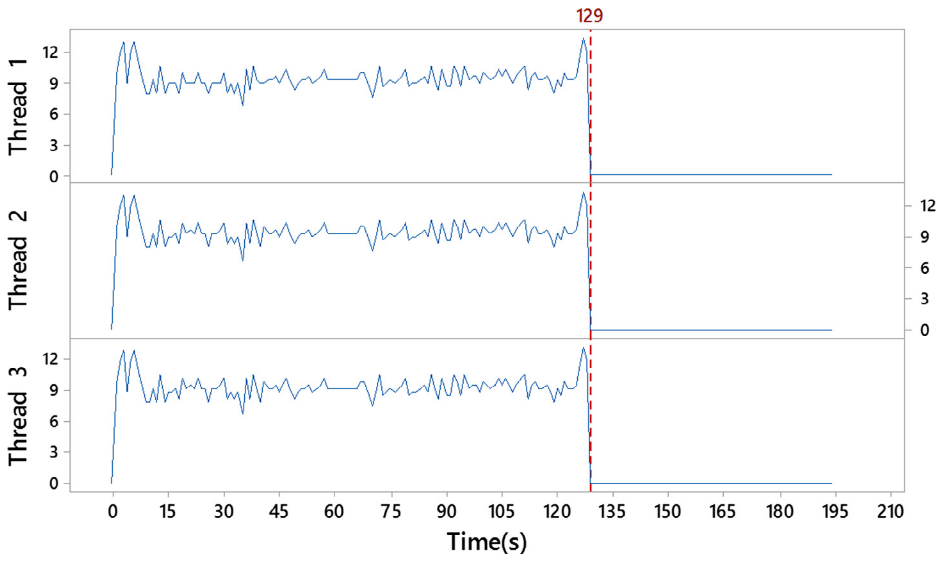


Fig. 14. Evaluated rules by thread in Test Case 2.

5.3. Evaluation of CEP performance

Some test cases were designed to verify the performance of the RE-CEP subsystem. The metrics that were analyzed are:

– RAM Usage: The amount of RAM memory that the CEP consumes while it is active.
– CPU Usage: The percentage of CPU that consumes while the CEP is active.

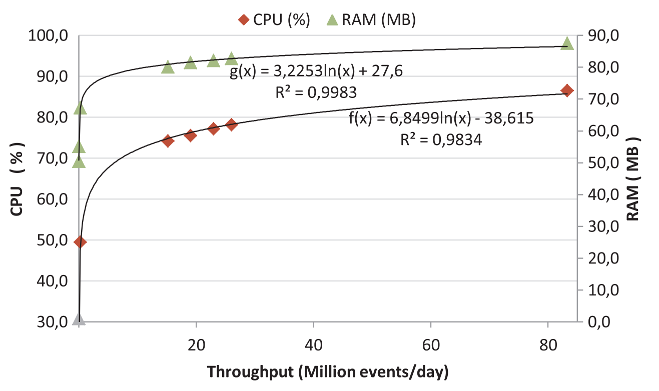


Fig. 15. Logarithmic regression model for predicting the consumption of CPU and RAM resources, with a logarithmic regression model $R^2 > 0.9$.

Table 7
Predictive throughput (events/day) by hectares, CPU and RAM usage results.

Banana producers	# Hectares	Throughput (events/day)	CPU (%)	RAM (MB)
Santa Inés	3.4	1696	5.3	51.9
Small producer	10	4988	26.0	55.1
Large-scale producer	500	249 412	49.5	67.2
Other provinces	30 489	15 208 674	74.2	80.1
Guayas	38 186	19 048 323	75.5	81.5
El Oro	46 051	22 971 261	77.2	82.1
Los Ríos	52 246	26 061 304	78.1	82.8
Total production surface	166 972	83 289 562	86.5	87.5

– Throughput (# received events/day and #rules evaluated/day): The number of events and rules that it is capable of evaluating per day.

Each test case is described below:

(a) Test case 1

The irrigation in the experimental banana plot was planned to be carried out for two hours daily, with a throughput of 1696 events/day. A total of 8478 events were analyzed during the evaluation period of five days. Table 5 shows the sensors and actuators used and the number of measurements collected per day in the experimental plot.

The results obtained according to the use of resources and performance metrics are shown in Fig. 12. The horizontal reference lines red represent the average consumption of CPU (13.4%) and the used average of RAM (50.4 MB), when processing incoming data flows (throughput). If the user registers new rule models or updates an existing rule in the Rules engine (RE), the CEP immediately detects it and incorporates the change. This process of updating the monitoring scheme of both rules and incoming event flow, results in a slight consumption of CPU and RAM. In the graphic, the rule models are updated, the horizontal reference lines green represent the average CPU consumption (25.8%) and average RAM utilization (56.8 MB).

(b) Test case 2

In this test case, a built-in WSN simulator for testing was used to send several throughput (events/s), as can be seen in the first column of Table 6. The second column shows the events collected in two hours a day, which normally lasts the irrigation in banana cultivation. The third column corresponds to the total of events collected during a 15-day evaluation period. The sixth column shows the amount of Rules Evaluated/day that is obtained according to the CEP Windows size and Rules models. And in the last two columns it shows the average CPU and RAM usage in each applied throughput.

In Fig. 13, it can be seen that the CEP shows an acceptable CPU performance of 75.4% up to 30 million events/day (an approximate of 4200 events/s), considering the few computational resources that the VM has where it was implemented. To improve performance, 2 cores or 4 cores CPU would suffice. Regarding the use of RAM, there is no significant increase compared to the 6 GB allocated to the VM.

When increasing the rules, it is indispensable to use parallel execution threads to distribute the rule evaluation load; for example, if there are 3 rules, each rule is executed in a different thread. Fig. 14 shows # events/s (X axis) that arrives to fuse with the rules represented by the Thread (Y). Note that there are no idle threads, as all rules are checked at the same time.

(c) Test case 3

Several blocks of events were tested, as seen in the first column of Table 6. Applying the logarithmic regression method, a predictive function with a Coefficient of Determination, r-squared > 0.9 (see Fig. 15) was obtained, considered an optimal prediction model both for CPU Consumption and use of RAM.

5.4. Analysis of the results

In the previous section, certain initial conclusions were carried out

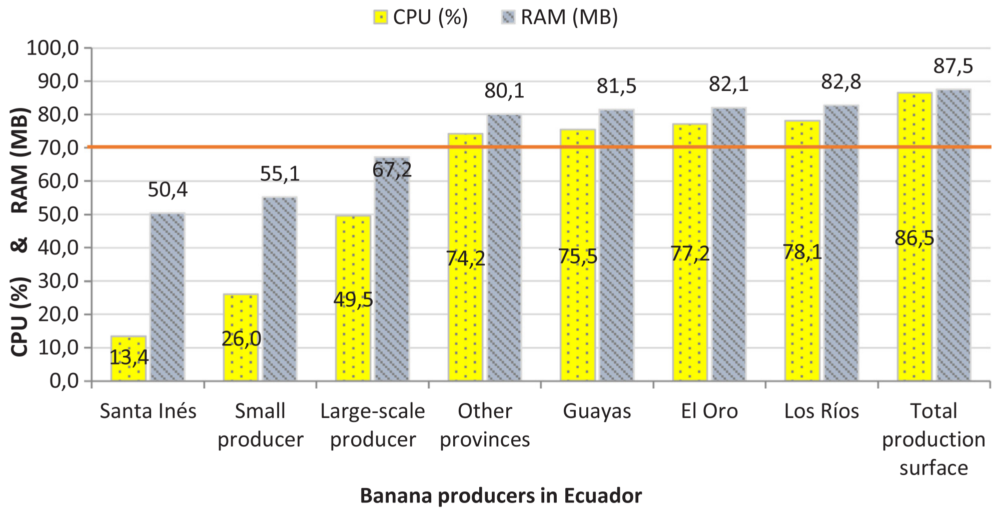


Fig. 16. Predicting consumption of CPU and RAM resources by planted hectares.

when analyzing the performance of the RECEP regarding CPU, RAM and Throughput usage. However, it is interesting to evaluate what would be the behavior of the proposed system in real scenarios of precision agriculture. From Fig. 15 the following expressions were extracted:

$$RAM\_used(MB) = 3.2253 * ln(troughput) + 27.6 \quad (1)$$

$$CPU\_consumption(\%) = 6.8499 * ln(troughput) - 38.615 \quad (2)$$

Expressions (1) and (2) were used to obtain the predictions shown in Table 7, where the variable throughput was estimated from the real results obtained in the experiments carried out at the Santa Inés farm, which constitutes an experimental station of banana production with an extension of 3.4 ha, located within the premises of the UTMACH. It is important to note that the product obtained from this farm is currently exported.

Fig. 16 shows a performance comparison of the RECEP in different scales of banana production, organized according to the number of hectares planted, which in Ecuador on average, is a direct proportion with the volume of output obtained. Therefore, small producers (which is the large productive mass not only in Ecuador, but in most developing countries), companies or associations of small producers with more than 500 ha and the equivalent of several companies grouped by provinces and even the total production at the country level, were analyzed. In Ecuador, banana production is concentrated mainly in three regions, Guayas, El Oro, and Los Ríos, which by themselves exceed the production obtained by the rest of the provinces, as can be seen in Table 5. The quantity of the hectares by regions used in our analysis was taken from the last report issued by the National Institute of Statistics and Censuses of Ecuador and can be found in (INEC, 2017).

Two interesting findings were found, as can be seen in Fig. 16. The first, show that the RAM required by our RECEP is low, allowing even the management of events of the entire area planted in Ecuador (82 MB), using a modest hardware infrastructure like ours, described in the experimental setup section. In contrast, the necessary CPU resources vary depending on the throughput. A referential value of 70% of CPU usage was added to the figure to represent an acceptable performance limit. There, we can appreciate that hardware architecture similar to the one we have used in our experiments are sufficient to deal with experimental environments and companies of small and large producers. But when it is required to manage several companies, with a more significant number of hectares, or management at a regional or national level, then an upgrade of the server with more CPUs must be done.

6. Conclusions

The Internet of Things is a technology that is making incursions in many fields and the agricultural field is no exception. IoT applied in Precision Agriculture, allows the optimization of resources and the increase of agricultural production. However, more and more wireless sensor networks that integrate many IoT devices with a variety of sensors and actuators are connecting, producing an increase in the transmission of large volumes of data flow and in real time; thus, the need to automate event processing, pattern identification and decision making is also growing.

In this work, we propose the architecture called RECEP for the processing of events generated in the context of Precision Agriculture and Internet of Things (PA-IoT), is formed by: Rules Engine (RE) and Complex Event Processor (CEP). The purpose of RE is to manage event templates (rules, incoming data flows, control actions and alert-notifications). While the CEP carries out the task of fusing incoming data, at the rate it arrives, with the rules established in the RE and performs a prescriptive analysis that consists of detecting event patterns and making automatic decisions. RECEP was implemented in a VM of a cloud with limited computing resources and integrated into an intelligent irrigation system of an experimental banana field located in Machala-Ecuador.

In order to determine the performance of the RECEP, it was evaluated using a WSN simulator was also used to generate sensor data in large volumes. The CEP was evaluated with several test cases, where a logarithmic regression model was found (r-squared > 0.9) to predict the use of CPU and RAM according to the number of events to be processed. The predictive values obtained when evaluating said regression model allow defining the hardware infrastructure necessary for the use of RECEP with different banana producing companies, categorized according to the number of hectares planted. The results obtained demonstrate the feasibility of using our RECEP in low-cost infrastructures for small and large producers.

Acknowledgments

This research received funding from the Universidad Técnica de Machala - Ecuador, through the AutoMathTIC research group. This work was also supported by the Universidade da Coruña, Spain, through an interinstitutional agreement.

References

Akbar, A., Carrez, F., Moessner, K., Sancho, J., Rico, J., 2016. Context-aware stream processing for distributed IoT applications. In: 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT). IEEE, pp. 663–668. https://doi.org/10.1109/WF-IoT.2015.7389133.

Akbar, A., Khan, A., Carrez, F., Moessner, K., 2017. Predictive analytics for complex IoT data streams. IEEE Int. Things J. 4, 1571–1582. https://doi.org/10.1109/JIOT.2017.2712672.

Akila, V., Govindasamy, V., Sandosh, S., 2016. Complex event processing over uncertain events: Techniques, challenges, and future directions. In: 2016 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC). IEEE, pp. 204–221. https://doi.org/10.1109/ICCPEIC.2016.7557198.

Amazon, 2018. Amazon Kinesis Data Streams [WWW Document]. URL < https://aws.amazon.com/kinesis/data-streams/ > (accessed 7.3.18).

Amazon, 2018. Rules for AWS IoT - AWS IoT [WWW Document]. URL < https://docs.aws.amazon.com/iot/latest/developerguide/iot-rules.html > (accessed 7.3.18).

Apache Flink, 2017. Scalable Stream and Batch Data Processing [WWW Document]. URL < https://flink.apache.org/ > (accessed 7.3.18).

Apache S. F., 2017. Apache Kafka. A distributed streaming plataform [WWW Document]. URL < https://kafka.apache.org/ > (accessed 7.3.18).

Bendre, M.R., Thool, R.C., Thool, V.R., 2015. Big data in precision agriculture : weather forecasting for future farming. In: 2015 1st International Conference on Next Generation Computing Technologies, Dehradun, India, pp. 744–750. https://doi.org/10.1109/NGCT.2015.7375220.

Boubeta, J., 2014. Model-Driven Development of Domain-Specific Interfaces for Complex Event Processing in Service-Oriented Architectures. Doctoral dissertation. Universidad de Cadiz.

Bruns, R., Dunkel, J., Masbruch, H., Stipkovic, S., 2015. Intelligent M2M: complex event processing for machine-to-machine communication. Expert Syst. Appl. 42, 1235–1246. https://doi.org/10.1016/j.eswa.2014.09.005.

Cugola, G., Margara, A., 2012. Low latency complex event processing on parallel hardware. J. Parallel Distrib. Comput. 72, 205–218. https://doi.org/10.1016/j.jpdc.2011.11.002.

Cugola, G., Margara, A., Matteucci, M., Tamburrelli, G., 2014. Introducing uncertainty in complex event processing: model, implementation, and validation. Computing 97, 103–144. https://doi.org/10.1007/s00607-014-0404-y.

Dayarathna, M., Perera, S., 2018. Recent advancements in event processing. ACM Comput. Surv. 51, 1–36. https://doi.org/10.1145/3170432.

Díaz, M., Martín, C., Rubio, B., 2016. State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. J. Netw. Comput. Appl. 67, 99–117. https://doi.org/10.1016/j.jnca.2016.01.010.

Ding, T., Gao, Y., Ma, Y., 2016. A semantic preserving CEP based approach for business data processing. In: 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD). IEEE, pp. 1808–1815. https://doi.org/10.1109/FSKD.2016.7603453.

Eclipse Mosquitto, 2018. Eclipse Mosquitto [WWW Document]. URL < https://mosquitto.org/ > (accessed 7.3.18).

Evans, D., 2011. The Internet of Things – How the Next Evolution of the Internet is Changing Everything [WWW Document]. CISCO white Pap. URL < https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf > (accessed 7.3.18).

Far, S.T., Rezaei-Moghaddam, K., 2017. Impacts of the precision agricultural technologies in Iran: an analysis experts' perception & their determinants. Inf. Process. Agric. 1–12. https://doi.org/10.1016/j.inpa.2017.09.001.

Gökalp, M.O., Eren, E., Koçyigit, A., 2018. A visual programming framework for distributed Internet of Things centric complex event processing. Comput. Electr. Eng. 1–24. https://doi.org/10.1016/j.compeleceng.2018.02.007.

Google Cloud Platform, 2017. Architecture: Real-Time Stream Processing for IoT [WWW

Document]. URL < https://cloud.google.com/solutions/architecture/real-time-stream-processing-iot > (accessed 7.3.18).

Greenwood, P.L., Valencia, P., Overs, L., Paull, D.R., Purvis, I.W., 2014. New ways of measuring intake, efficiency and behaviour of grazing livestock. Anim. Prod. Sci. 54, 1796–1804. https://doi.org/10.1071/AN14409.

Hao, Z., Novak, E., Yi, S., Li, Q., 2017. Challenges and software architecture for fog computing. IEEE Int. Comput. 21, 44–53. https://doi.org/10.1109/MIC.2017.26.

Hernández-Rojas, D.L., Fernandez-Caramés, T., Fraga-Lamas, P., Escudero, C.J., 2018a. A plug-and-play human-centered virtual TEDS architecture for the web of things. Sensors 18, 1–40. https://doi.org/10.3390/s18072052.

Hernández-Rojas, D.L., Fernández-Caramés, T., Fraga-Lamas, P., Escudero, C.J., 2018b. Design and practical evaluation of a family of lightweight protocols for heterogeneous sensing through BLE beacons in IoT telemetry applications. Sensors 18, 1–33. https://doi.org/10.3390/s18010057.

Hernández-Rojas, D.L., Mazon-Olivo, B., Novillo-Vicuña, J., Escudero-Cascon, C., Pan-Bermudez, A., Belduma-Vacacela, G., 2018c. IoT android gateway for monitoring and control a WSN. In: In: Botto-Tobar, M., Esparza-Cruz, N., León-Acurio, J., Crespo-Torres, N., Beltrán-Mora, M. (Eds.), Technology Trends. CITT 2017. Communications in Computer and Information Science, vol. 798. Springer, Cham, pp. 18–32. https://doi.org/10.1007/978-3-319-72727-1_2.

HiveMQ, 2018. MQTT Essentials Part 6: Quality of Service Levels [WWW Document]. URL < https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels > (accessed 7.3.18).

Huawei, 2015. Global connectivity index 2015. Benchmarking Digital Economy Transformation [WWW Document]. Glob. Connect. Index. URL < https://www.huawei.com/minisite/gci/assets/files/gci_2015_whitepaper_en.pdf?v=20180716 > (accessed 7.3.18).

INEC, 2017. Encuesta de Superficie y Producción Agropecuaria Continua (ESPAC 2017) [WWW Document]. URL < http://www.ecuadorencifras.gob.ec/documentos/web-inec/Estadisticas_agropecuarias/espac/espac_2017/Presentacion_Principales_Resultados_ESPAC_2017.pdf > (accessed 7.2.18).

Jun, C., Chi, C., 2014. Design of complex event-processing IDS in internet of things. In: 2014 Sixth International Conference on Measuring Technology and Mechatronics Automation. IEEE, pp. 226–229. https://doi.org/10.1109/ICMTMA.2014.57.

Juul, J.P., Green, O., Jacobsen, R.H., 2015. Deployment of wireless sensor networks in crop storages. Wirel. Pers. Commun. 81, 1437–1454. https://doi.org/10.1007/s11277-015-2482-3.

Kaloxylos, A., Wolfert, J., Verwaart, T., Terol, C.M., Brewster, C., Robbemond, R., Sundmaker, H., 2013. The use of future internet technologies in the agriculture and food sectors: integrating the supply chain. Proc. Technol. 8, 51–60. https://doi.org/10.1016/j.protcy.2013.11.009.

Kamilaris, A., Gao, F., Prenafeta-Boldu, F.X., Ali, M.I., 2017. Agri-IoT: a semantic framework for Internet of Things-enabled smart farming applications. In: 2016 IEEE 3rd World Forum Internet Things, pp. 442–447. https://doi.org/10.1109/WF-IoT.2016.7845467.

Luckham, D., 2007. A Short History of Complex Event Processing Part 2 : The Rise of CEP [WWW Document]. URL < http://complexevents.com/wp-content/uploads/2008/07/2-final-a-short-history-of-cep-part-2.pdf > (accessed 7.3.18).

Mayer, R., Koldehofe, B., Rothermel, K., 2015. Predictable low-latency event detection with parallel complex event processing. IEEE Int. Things J. 2, 274–286. https://doi.org/10.1109/JIOT.2015.2397316.

Microsoft Azure, 2017. Stream Analytics - Real-time data analytics [WWW Document]. URL < https://azure.microsoft.com/en-us/services/stream-analytics/ > (accessed 7.3.18).

Minet, J., Curnel, Y., Gobin, A., Goffart, J.-P., Mélard, F., Tychon, B., Wellens, J., Defourny, P., 2017. Crowdsourcing for agricultural applications: a review of uses and opportunities for a farmsourcing approach. Comput. Electron. Agric. 142, 126–138. https://doi.org/10.1016/j.compag.2017.08.026.

Nikolidakis, S.A., Kandris, D., Vergados, D.D., Douligeris, C., 2015. Energy efficient automated control of irrigation in agriculture by using wireless sensor networks. Comput. Electron. Agric. 113, 154–163. https://doi.org/10.1016/j.compag.2015.02.004.

Ojha, T., Misra, S., Raghuwanshi, N.S., 2015. Review: Wireless sensor networks for agriculture: the state-of-the-art in practice and future challenges. Comput. Electron. Agric. 118, 66–84. https://doi.org/10.1016/j.compag.2015.08.011.

Qin, Y., Sheng, Q.Z., Falkner, N.J.G., Dustdar, S., Wang, H., Vasilakos, A.V., 2016. When things matter: a survey on data-centric internet of things. J. Netw. Comput. Appl. 64, 137–153. https://doi.org/10.1016/j.jnca.2015.12.016.

Ray, P.P., 2017. Internet of things for smart agriculture: technologies, practices and future direction. J. Ambient Intell. Smart Environ. 9, 395–420. https://doi.org/10.3233/AIS-170440.

RedHat, 2006. Drools - Business Rules Management System [WWW Document]. URL < https://www.drools.org/ > (accessed 7.3.18).

Rose, K., Eldridge, S., Chapin, L., 2015. The Internet of Things: an overview [WWW Document]. Internet Soc. (ISOC). URL < https://www.internetsociety.org/sites/default/files/ISOC-IoT-Overview-20151221-en.pdf > (accessed 7.3.18).

Sakr, S., 2013. An introduction to InfoSphere Streams [WWW Document]. URL < https://www.ibm.com/developerworks/library/bd-streamsintro/index.html > (accessed 7.3.18).

Srbinovska, M., Gavrovski, C., Dimcev, V., Krkoleva, A., Borozan, V., 2015. Environmental parameters monitoring in precision agriculture using wireless sensor networks. J. Clean. Prod. 88, 297–307. https://doi.org/10.1016/j.jclepro.2014.04.036.

Sun, Y., Wu, T.Y., Zhao, G., Guizani, M., 2015. Efficient rule engine for smart building systems. IEEE Trans. Comput. 64, 1658–1669. https://doi.org/10.1109/TC.2014.2345385.

Talavera, J.M., Tobón, L.E., Gómez, J.A., Culman, M.A., Aranda, J.M., Parra, D.T., Quiroz, L.A., Hoyos, A., Garreta, L.E., 2017. Review of IoT applications in agro-industrial and environmental fields. Comput. Electron. Agric. 142, 283–297. https://doi.org/10.1016/j.compag.2017.09.015.

Vermesan, O., Friess, P., 2015. Building the Hyperconnected Society. River Publishers doi:978-87-93237-99-5.

Wang, D., Rundensteiner, E., Ellison, R., 2011. Active complex event processing over event streams. Proc. VLDB Endow. 4, 634–645. https://doi.org/10.14778/2021017.2021021.

Wang, D., Zhou, M., Ali, S., Zhou, P., Liu, Y., Wang, X., 2016. A novel complex event processing engine for intelligent data analysis in integrated information systems. Int. J. Distrib. Sens. Netw. 2016, 1–14. https://doi.org/10.1155/2016/6741401.

Wang, Y., Cao, K., 2014. A proactive complex event processing method for large-scale transportation internet of things. Int. J. Distrib. Sens. Netw. 10, 1–8. https://doi.org/10.1155/2014/159052.

Wang, Y., Gao, H., Chen, G., 2018. Predictive complex event processing based on evolving Bayesian networks. Pattern Recognit. Lett. 105, 207–216. https://doi.org/10.1016/j.patrec.2017.05.008.

Wolfert, S., Ge, L., Verdouw, C., Bogaardt, M., 2017. Big data in smart farming – a review. Agric. Syst. 153, 69–80. https://doi.org/10.1016/j.agsy.2017.01.023.

WSO2, 2018. WSO2 Complex Event Processor [WWW Document]. WSO2. URL < https://wso2.com/products/complex-event-processor/ > (accessed 7.3.18).

WSO2, 2017. Business Rules Server [WWW Document]. URL < https://wso2.com/products/business-rules-server/ > (accessed 7.3.18).

Zappia, I., Parlanti, D., Paganelli, F., 2011. LiSEP: A lightweight and extensible tool for complex event processing. In: 2011 IEEE International Conference on Services Computing. IEEE, pp. 701–708. https://doi.org/10.1109/SCC.2011.63.

Zhou, Q., Simmhan, Y., Prasanna, V., 2017. Knowledge-infused and consistent Complex Event Processing over real-time and persistent streams. Futur. Gener. Comput. Syst. 76, 391–406. https://doi.org/10.1016/j.future.2016.10.030.