Intelligent M2M: Complex Event Processing for Machine-to-Machine Communication

Ralf Bruns*, Jürgen Dunkel, Henrik Masbruch, Sebastian Stipkovic

Hannover University of Applied Sciences and Arts, Department of Computer Science, Ricklinger Stadtweg 120, 30459 Hannover, Germany

Abstract

M2M (machine-to-machine) systems use various communication technologies for automatically monitoring and controlling machines. In M2M systems, each machine emits a continuous stream of data records, which must be analyzed in real-time. Intelligent M2M systems should be able to diagnose their actual states and to trigger appropriate actions as soon as critical situations occur.

In this paper, we show how Complex Event Processing (CEP) can be used as the key technology for intelligent M2M systems. We provide an event-driven architecture that is adapted to the M2M domain. In particular, we define different models for the M2M domain, M2M machine states and M2M events. Furthermore, we present a general reference architecture defining the main stages of processing machine data. To prove the usefulness of our approach, we consider two real-world examples 'solar power plants' and 'printers', which show how easily the general architecture can be extended to concrete M2M scenarios.

Keywords: Machine-to-machine communication, Complex event processing, Event-driven architecture

1. Introduction

An essential characteristic of technological progress is the increased digital interconnection and computerization of technical systems and components. In recent years, also intelligent machines have been integrated in digital networks: M2M (machine-to-machine) systems use different communication technologies to automatically monitor and control remote machines/devices with or without any manual interaction.

M2M systems in daily operations can be found in several application areas, for instance, remote monitoring (e.g., solar panels, commercial printer services), track & trace (e.g., rental bicycles, fleet tracking), facility management (e.g. elevators, energy management), vending segment (e.g., vending machines), or metering (e.g., utility meters). More and more application areas will arise in the future, see (Chen et al., 2012; Pandey et al., 2011). M2M exhibits the potential to serve as one core concept for an interconnected, sustainable, and mobile future (BMWi, 2011).

Figure 1 depicts an example of a conventional wireless M2M system: the monitoring of an industrial solar power plant. Each solar inverter (= machine) is equipped with a GPRS (General Packet Radio Service) terminal that transmits a continuous stream of solar panel data to a central M2M backend server via a wireless communication network. A M2M portal server provides a visualization of the collected data, which human experts can use to monitor the M2M system performance.

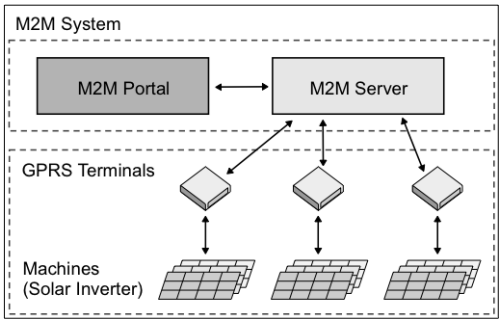

Figure 1: Conventional M2M system

Most conventional M2M systems support only the collection, aggregation, and presentation of low level machine data. Still they do not support in-depth and flexible analysis of M2M system data. Instead, machine data must mostly be correlated and interpreted manually by domain experts. Beyond that some more sophisticated M2M systems provide hard-coded processing of machine data, which is is difficult and expensive to adapt to changing application needs (e.g. changes of the M2M systems topology or new types of machines). Altogether, today's M2M systems still do not achieve its potential of exploiting the machine data in an integrated and flexible manner.

Therefore, a new class of M2M decision support systems is required allowing the analysis of machine data in an intelligent and flexible manner. Furthermore, such systems must be able to process high-frequent streams of machine data nearly in real-time, because usually machine problems must be identified as soon as possible to prevent major damage.

In this paper, we suggest a novel, event-driven approach for

*Corresponding authors:
Email addresses: ralf.bruns@hs-hannover.de (Ralf Bruns), juergen.dunkel@hs-hannover.de (Jürgen Dunkel), henrik.masbruch@hs-hannover.de (Henrik Masbruch), sebastian.stipkovic@hs-hannover.de (Sebastian Stipkovic)
URL: http://sw-architecture.f4.hs-hannover.de/ (Ralf Bruns)

the intelligent analysis of (wireless) M2M systems[1]. Main concept of our approach is the application of Complex Event Processing (CEP) for M2M applications. CEP is a rather new software technology (Luckham, 2002) for processing continuous streams of general data in (near) real-time. The basic concept of CEP is in-memory pattern matching, which means to identify those patterns in a stream of data that represent a meaningful situation in the application domain. Therefore, CEP can be used in M2M systems for enabling situation- and context-awareness and, consequently, leads to a new quality of M2M systems. Because event patterns are defined descriptively by rules, CEP can improve significantly the adaptability and flexibility of M2M solutions.

In the following, we present a general architecture of an intelligent decision support system for M2M using CEP. In particular, we provide an overall system architecture, various M2M-specific domain, state and event models, and a M2M-specific event processing network (EPN). In order to adapt our approach to a particular M2M domain, only the domain-specific knowledge, such as the events types and the specific event processing rules, has to be specified and integrated into the proposed general architecture. Experimental results obtained by two real-world case studies prove the feasibility of our approach.

The outline of the paper is as follows. In Section 2, we introduce the main concepts of our intelligent event-driven M2M system based on event processing principles. Section 3 presents how our approach can be applied in practice by means of two case studies: (a) solar power plants and (b) printer supply & maintenance service. In the subsequent section, some experimental results with real-world data sets are reported. In Section 5, we discuss some related work. Finally, in the last section we summarize the most important aspects of our approach and provide an outlook on future lines of research.

2. CEP for M2M

2.1. CEP - Overview

CEP is a software technology for processing high frequent event streams (Luckham (2002), Etzion & Niblett (2010)).[2] Everything that happens inside or outside of a system can be considered as an *event* (Luckham & Schulte, 2011). The stream of incoming events is processed in three basic steps:

1. occurred events are captured in the environment,
2. the event stream is analyzed to detect meaningful event patterns, and
3. an immediate reaction is triggered as a response.

Event stream processing systems manage the most recent set of events in-memory and process them in near real-time.

CEP analyzes streams of incoming events to detect the presence of event patterns with a particular meaning for the domain. The matching of an event pattern signifies a significant state of the system and causes either the generation of a new *complex event* or triggers a domain-specific action. Complex events correlate several simple, low level events to more meaningful business events and provide the real power of CEP.

CEP employ sliding windows and temporal operators to specify temporal relations between events. A core concept of CEP is a declarative *event processing language* (EPL) to express *event processing rules*. An event processing rule contains two parts: a *condition* part describing the requirements for firing the rule and an *action* part that is performed if the condition matches. The *condition* is defined by an event pattern using several operators and further constraints.

In this article, we employ a simplified pseudo language for specifying event processing rules supporting the following operators:

Operators

$\wedge, \vee$	boolean AND, OR operator for events or constraints.
`NOT`	negation of a constraint.
`->`	temporal sequence of events.
`Timer`	*Timer(time)* defines a time to wait.
`.win:time`	specifies a time window in which an event has to occur.

A CEP engine analyzes the stream of continuously incoming events and executes the matching rules. Event processing rules transform low level simple events into more complex events in order to gain insight into the current state of the environment. Popular open source CEP engines are *Esper* (ESPERTECH, 2014), *Drools Fusion* (JBoss, 2014), and *Triceps* (Babkin, 2014).

If the central architectural concept of a software system is the processing of events, then the system is called event-driven and the architectural style is called an Event-Driven Architecture (EDA) (Bruns & Dunkel (2010), Chandy & Schulte (2010)).

Although rather new, CEP has already been successfully applied in a variety of application domains with numerous documented use cases. Sample application domains are fraud detection, algorithmic stock-trading, sensor networks, or business activity monitoring (Bruns & Dunkel, 2010).

2.2. CEP for Intelligent Decision Support in M2M

We propose to extend the capabilities of M2M systems towards real-time intelligence by taking advantage of innovative event processing technologies. The requirements of a modern M2M system comprise

- processing of high volumes of continuously arriving machine data and
- correlation of low level technical machine data to derive more advanced domain information in real-time.

CEP has been designed to cope exactly with these requirements (Bruns & Dunkel, 2010; Luckham, 2002). In classic M2M systems, data processing is usually implemented by hard-coded algorithms, which are often scattered all over the source

[1] The focus of this article lies on wireless M2M systems, but most concepts hold for wired M2M systems as well.

[2] The terms Event Stream Processing (ESP) or Data Stream Processing are used too.

code and difficult to maintain. Instead in CEP, event processing is capsulated in rules that are clearly separated from other source code. Furthermore, due to their declarative nature, rules can be adapted and maintained in an efficient and agile manner.

Therefore, CEP has the potential to serve as key software technology for building intelligent, responsive and flexible M2M systems. Although, CEP alone is not sufficient for building decision support systems for M2M. Beyond that we need a software architecture and a blueprint that fosters the development of CEP-based M2M systems. In the following subsections we present such an architecture in more detail.

2.2.1. M2M Event-Driven Architecture

In our approach, the basic architecture of classic M2M systems is enhanced by a CEP component. As a result, we developed an event-driven software architecture for M2M systems (as depicted in Figure 2).

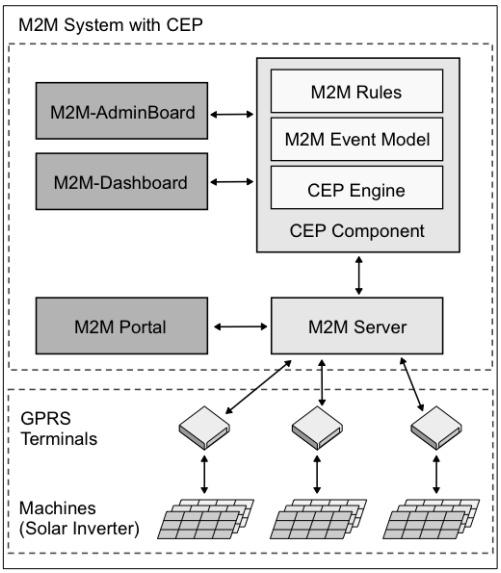

Figure 2: Event-driven architecture for M2M (see (Metzdorf et al., 2013))

Every data record that is transmitted in the M2M system is considered as an event. The CEP component is responsible to continuously analyze the incoming stream of machine data/ events. It consists of three parts:

- *Event model*: The event model specifies all possible types of events that can be processed by the CEP component.
- *Event rules*: The event processing rules define the event patterns to be detected in the event stream. Thus, the M2M knowledge of domain experts is explicitly represented in the rule base.
- *CEP engine*: A CEP engine tries to match the event patterns with the incoming data stream.

The event model and rules build the knowledge base of the system. In addition, M2M-AdminBoard and M2M-Dashboard are software components for the administration and monitoring of the CEP component.

The design principles of the introduced CEP component are discussed below for different kinds of M2M problems in general. Note that the approach is independent of any particular type of machines. In the following subsections, first we develop a simplified domain model for wireless M2M and corresponding M2M-specific state models. Then, we derive a M2M-specific event model with the event types that trigger state transitions, and finally we present a M2M event processing architecture.

2.2.2. M2M Domain Model

The purpose of the M2M Domain DSL is to model the M2M infrastructure, basically the installed machines and their characteristic properties. This information is required to formulate CEP rules. In the M2M domain, we can distinguish the following types of entities:

(a) application-independent components: the communication devices, e.g. the GPRS terminals,
(b) application-dependent components: the specific types of machines under investigation, e.g. a photovoltaic solar panel producing electricity.

Figure 3 shows a simplified general domain model, which summarizes the core entities important in most M2M applications.

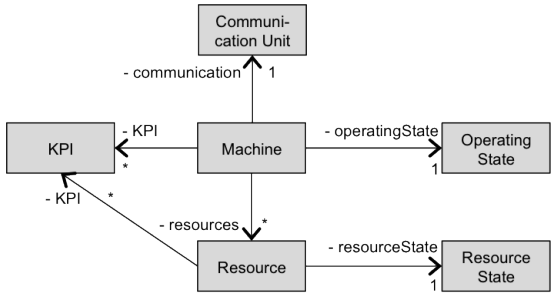

Figure 3: Domain model for M2M

The basic concept in the domain model is 'Machine' that defines a particular machine. Each machine has a distinct 'Operating State' indicating if it is running normally or if a malfunction or defect has occurred. Furthermore, M2M systems have to monitor the performance of the machines, e.g. how often they are used or how they behave. The machine behavior is characterized by Key Performance Indicators (KPI) that contain performance measures of an individual machine or aggregate other KPIs. Moreover, particular types of machines require certain resources to fulfill their tasks, e.g. paper and toner for printers, or goods held in vending machines. Thus, in the domain model, a machine may have a relation to one or more resources, each having its own resource state.

Table 1 lists different categories of M2M machines: the left column shows some machine types having only an operating state, because they do not consume any resources; the right column gives examples of those machine types that additionally require resources.

Table 1: Different categories of M2M machines

Machines with operating states only	Machines with operating and resource states
solar panels	printers/ copiers *resource:* toner, paper
elevators	industrial coffee machines *resource:* coffee
rental bikes	vending machines *resource:* beverages, snacks
rental vehicles	
utility meters	

Finally, the domain model in Figure 3 contains a 'Communication Unit' such as a GPRS terminal, which connects each machine to the M2M backend system.

2.2.3. M2M State Models

Monitoring of the machines' operating and resource states is one of the main issues in M2M systems. The state diagrams in Figure 4 and 5 illustrate the main states and the state transitions for the M2M domain. The transitions between states are defined by certain types of events (defined in 2.2.4).

Operating States

Figure 4 shows the model of operating states for M2M and the event types that trigger the transitions between states.

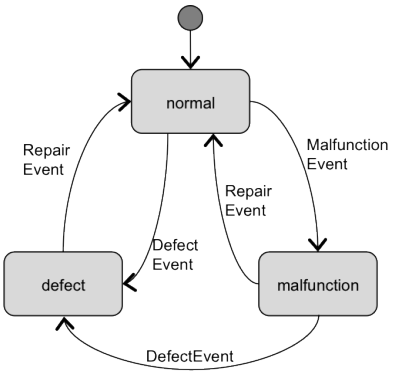

Figure 4: M2M operating state model

A machine usually operates in the 'normal' state meaning that the machine is running without any problems. If some malfunctions have occured but the machine is still running, a 'Malfunction Event' is emitted causing the transition to the 'malfunction' state. A serious problem, e.g. an elevator gets stuck between two floors, causes the transition to the 'defect' state indicating that the machine is not running anymore. After fixing the problem the machine gets back into 'normal' state.

Resource States

For machines that rely on the availability of sufficient resources, the filling level of their resources is crucial. A machine with a full resource starts in the state 'sufficient' (see Figure 5).

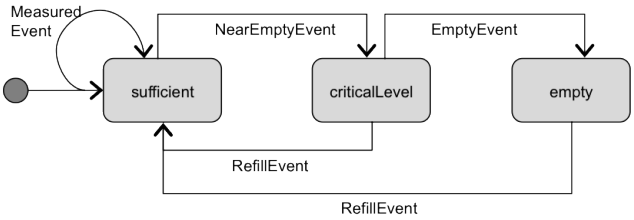

Figure 5: M2M resource state model

A running machine consumes continuously resources. For example, vending machines change to the 'empty' state when their goods are sold out. Usually, machines with empty resources cannot operate properly. Thus at a specified resource level, a machine should change to the 'critical level' state in order to initiate the refill of the near empty resource. After having refilled the resource, the machine returns to the 'sufficient' state.

Key Performance Indicators

Conventional M2M solutions store all collected data in a central database, which is periodically analyzed for processing appropriate KPIs. In contrast, our event-driven M2M system processes and analyzes the data immediately when the events arrive to provide up-to-date KPI information. In the M2M domain, the calculated KPIs are transformed to machine events or provided by aggregated KPI events.

The actual operating and resource states of the machines are derived from the KPIs. Consequently, the state transition events are inferred from KPI events. This has the advantage that the state transitions are derived in near real-time instead of determining the current state periodically at a fixed rate.

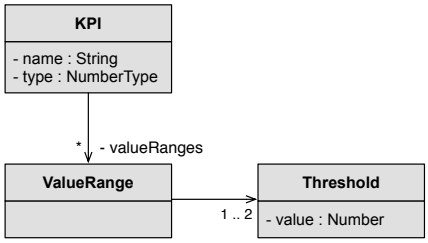

Figure 6: M2M KPI model

The general KPI model for M2M is depicted in Figure 6. Every KPI has a name and a type, e.g., a KPI for the resource 'candy bar' of a vending machine is an integer value which represents the KPI 'remaining number of items'. Every KPI has value ranges bounded by threshold values. The value range for the critical level state of the resource 'candy bar' is, for example, between the quantity of five and one. When the quantity falls under the upper threshold of the value range, the machine enters the 'critical level' state (see resource state model in Figure 5).

2.2.4. M2M Event Model

The state transitions of the presented M2M models are triggered by events. A general event model for the M2M domain is represented in Figure 7. According to the general M2M concepts introduced above, four different categories of event types can be distinguished:

1. *KPI Events* represent single or aggregated machine data.
2. *Operating Events* show that the operating state of a machine has changed.
3. *Resource Events* correspond to a change of a resource state.
4. *Communication Events* signalize the state change of the network connection. These types of events are application-independent and, consequently, are valid for all kinds of M2M problems.

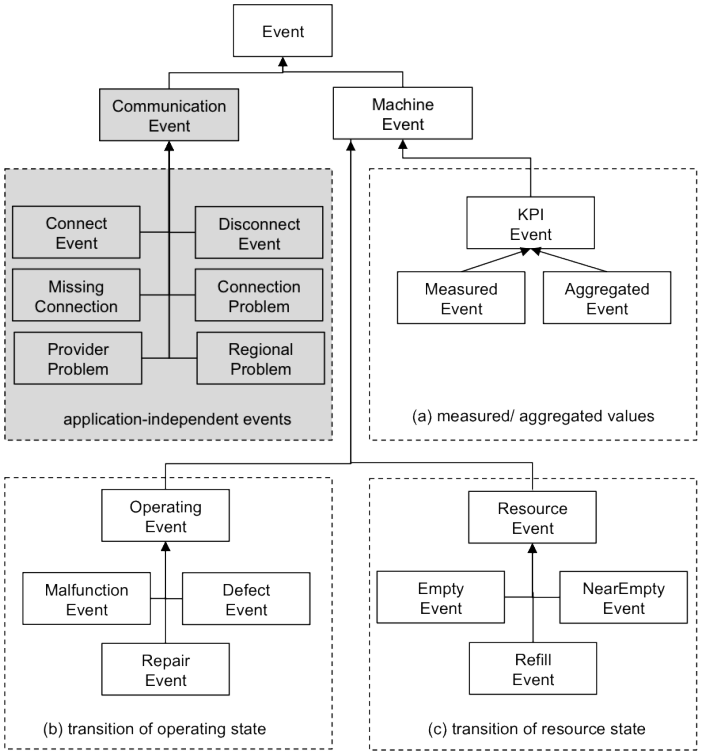

Figure 7: M2M event model

We can distinguish explicit from materialized events: *Explicit events* are events that are emitted by an event source. For instance, KPI events are directly emitted by a machine and contain the measured values of a particular machine parameter, e.g. number of usage, produced voltage, etc. In contrast, *materialized events* are not produced directly by the M2M components, but are generated by a CEP event processing rule, e.g. events that aggregate measured values of a certain machine.

2.2.5. M2M Event Processing Architecture

Rule-based systems with a large set of rules are hard to maintain and do not scale well. Therefore, Luckham (2002) introduced the concept of Event Processing Agents (EPA). An EPA is an individual CEP component with its own rule engine and rule base. Several EPAs can be connected to an Event Processing Network (EPN) that constitutes a software architecture for event processing. EPAs communicate with each other by exchanging events.

The CEP component of our event-driven M2M system (see Figure 2) consists of a two-layer, multi-staged architecture. Figure 8 shows the layers with their different EPAs and illustrates the flow of events. The EPAs in the top layer are responsible for processing the application-independent communications event. The bottom layer contains the EPAs for processing the M2M machine events.

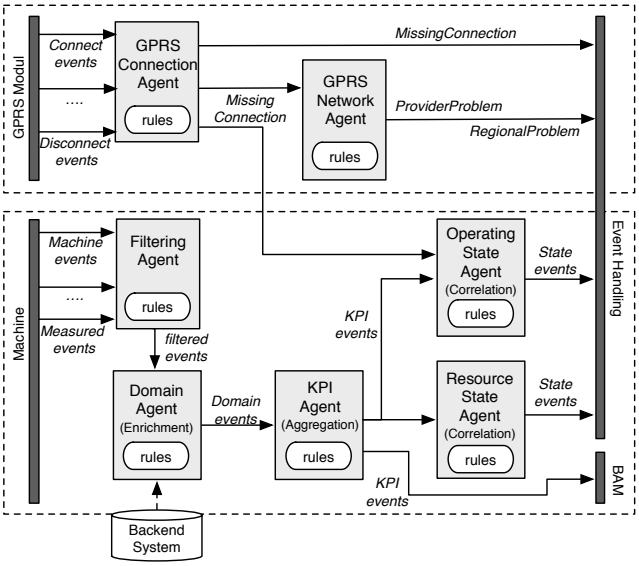

Figure 8: Event processing network for event-driven M2M

Top Layer: Application-independent EPN

The communication events emitted by the employed communication technology, e.g. GPRS terminals, are absolutely independent of the particular kind of machines that they connect. Therefore, this layer of the EPN stays the same for all M2M applications and is treated in detail below in Subsection 2.2.6.

Bottom Layer: M2M-specific EPN

For all kinds of M2M domains, the general responsibilities of the domain-specific EPAs are the same, however the particular event processing rules must be specified application-dependently.

- *Filtering Agent*: Due to technical problems or imprecise metering, event data is often inconsistent or redundant. Hence, in a filtering step all machine events are preprocessed to overcome inconsistencies or to filter out irrelevant events (Chandy & Schulte, 2010; Etzion & Niblett, 2010).
- *Domain Agent*: The raw machine events contain only low level technical data, e.g., the internal machine ID. The Domain Agent enriches machine events by mapping technical

data to domain concepts, e.g., the machine ID is related to a costumer who operates the machine and to its geographical location. The necessary information is retrieved from backend enterprise systems.

- *KPI Agent*: The KPI Agent receives the enriched machine events as input and aggregates them to domain-specific key performance indicators, e.g., by calculating average values over temporal or spatial related events. KPI events are used for determining the operating and resource states. Moreover, they provide valuable information about the current performance of the overall system used by BAM (business activity monitoring) applications. The resulting event types are subtypes of the KPI event types specified in the M2M event model in Figure 7, event category (a).
- *Operating State Agent*: The Operating State Agent manages the transitions of the M2M operating state model defined in Figure 4. For this purpose, the event processing rules analyze the incoming KPI events in order to detect an event pattern that results in an 'Operating Event' defined in the event model in Figure 7, event category (b).
- *Resource State Agent*: The Resource State Agent is responsible for the M2M resource state transitions (see Figure 5) and relies on the event types defined in Figure 7, event category c).

The M2M event processing architecture introduced in Figure 8 enhances significantly the structuring and modularization of M2M systems.

(a) The application-specific expert knowledge is represented in a declarative manner in the event processing rules, so that it is clearly separated from the infrastructural source code.
(b) Light-weight EPAs with a distinct set of few and coherent rules improve the understanding, agility, and maintainability of the entire system.

2.2.6. Application-Independent Communication Problems

The two EPAs in the top layer of the event processing architecture of Figure 8 are responsible for processing the application-independent communication events emitted by the GPRS terminals (or any other technology used for communication), as shown in Figure 7.

The first EPA (GPRS Connection Agent) recognizes problems of an individual GPRS terminal, i.e., a failure in the wireless communication connection. The second EPA (GPRS Network Agent) identifies more general problems either of the GPRS network or of the telecommunication provider (see also (Metzdorf et al., 2013)).

For instance, the following rule in Listing 1 detects a connection problem of a GPRS terminal; and belongs to the rule base of the GPRS Connection EPA (see Figure 8). The rule considers 'Connect' and 'Disconnect' events (see event model in Figure 7), which are emitted by a terminal when it establishes a GPRS connection or when the connection breaks down.

A connection problem is created, if a 'Disconnect' event has occurred and the same terminal does not emit a 'Connect' event

Listing 1: Connection problem

```
1 CONDITION
2   EVERY Disconnect AS d ->
3   (
4         NOT Connect(machineID = d.machineID)
5   )
6   TIMER(10 min)
7 ACTION create MissingConnection(d.machineID)
```

within the next 10 minutes. If this pattern is detected in the event stream, the rule matches and generates a new 'Missing Connection' event that is propagated to the subsequent GPRS Network EPA (see Figure 8).

The rule base of the GPRS Network EPA contains, among others, the following rule (Listing 2) that correlates 'Missing Connection' events in order to analyze if the telecommunication provider is responsible for the connection problems.

Listing 2: Provider problem

```
1 CONDITION
2   MissingConnection
3   TIMER(10 min)
4   GROUP BY mcc, mnc
5   GREATER THAN 20
6 ACTION create ProviderProblem(mcc, mnc)
```

A telecommunication provider problem exists, if different GPRS terminals of the same provider signal connection problems. In order to diagnose a provider problem, the rule summarizes all 'Missing Connection' events and groups them according to 'Mobile Country Code' (mcc) and 'Mobile Network Code' (mnc). If the number of occurred events is greater than 20 within a time window of 10 minutes, a 'Provider Problem' event is inferred.

Both rules illustrate how CEP derives high level complex events out of low level technical events. The high level events can lead to concrete actions, for example, the inspection of a particular GPRS terminal with respect to a hardware defect.

2.3. Evaluation of the Approach

The proposed event-driven M2M system enables the development of a new quality of M2M systems. CEP provides intelligent data correlation, low latency and high flexibility for M2M applications. Thus, CEP leads to situation- and context-awareness in M2M. In particular, the approach achieves:

- Intelligent data correlation:
  - Appropriate event processing rules observe the stream of incoming machine events for deriving automatically the machines' actual operating and resource states.
  - Event processing rules correlate machine data in order to detect temporal or spatial relationships. Further context information can be easily incorporated in the correlation as well.

  - Fine-grained simple technical events can be transformed into complex domain events that represent a significant meaning in the M2M domain.

- Real-time processing:

  - CEP engines have been designed to efficiently handle huge data streams directly in-memory. In-memory processing enables the detection of machine problems in near real-time and adequate reactions such as repair operations can be triggered automatically with low latency.

- High flexibility:

  - The specification of event patterns in an EPL enables the easy adjustment and extension of event processing rules.
  - The analysis of data streams by complicated and hard-coded algorithms is obsolete.
  - New types of machines can be seamlessly integrated in the event model as well as in correlation patterns specified in the EPL.

The presented event-driven architecture for M2M with its domain, state, and event models and its event processing architecture can serve as a blueprint for the design and structured development of intelligent M2M applications in practice.

However, in industrial practice, the specification of the event processing rules can still be a complicated task. On the one hand, the rules must take complex domain relationships between the events into account. Moreover, parameters such as the length of sliding windows or relevant correlation sets, have to be determined often experimentally in order to achieve adequate results.

3. Case Studies

In order to demonstrate how the proposed event-driven M2M system can be applied in practice, this section discusses two distinct M2M application scenarios:

- solar power plants (in Subsection 3.1) and
- professional printer supply & maintenance service (in Subsection 3.2).

Both case studies rely on real-world M2M scenarios with real-world sample data taken from commercial M2M systems.

Based on the main building blocks of event-driven M2M systems introduced in Section 2, for each use case only the particular event types and the application-specific event processing rules have to be specified.

3.1. Scenario 1: Solar Power Plants

Solar power plants play an important role in the future supply with renewable energy. Usually, solar power plants are geographically distributed to far-away remote locations with no on-site maintenance staff. Maintenance intervals are fixed according to a predefined schedule so that it is difficult to diagnose the current operating state of the solar panels. For instance, what is the reason that a solar panel does not produce any electricity: lack of sun, technical problems, or problems with the transmission of the metered data?

A solar power plant consists of a set of solar panels that convert sunlight into electricity. To feed the generated directed current (DC) electricity into the alternating current (AC) electricity grid a conversion is needed. For this purpose, the generated electricity of the solar panels is converted by an inverter and then fed into the public electricity grid. Multiple panels are attached to one inverter. Figure 9 sketches the schematic structure of solar power plants.

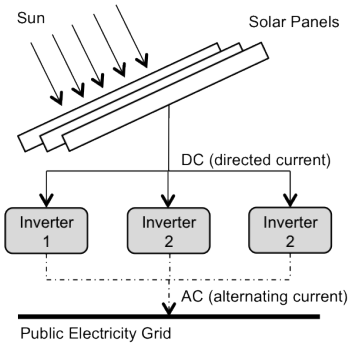

Figure 9: Schematic structure of solar power plant

In this solar power plant scenario an inverter device is considered as a machine (in M2M sense). It has no consumable resources[3], but KPIs for input and output power and voltage. The solar panels are not considered as machines, because they have no measurable KPI values on their own, although they might be malfunctioned or defect in the sense of a machine operating state.

Figure 10 shows the refinement of the general M2M event model (introduced in Figure 7) for the solar scenario. In event category (a) the measured events represent the KPI values for input and output power and voltage, respectively. The KPI event 'OutputPowerAVG' represents the regional average output power of all machines in a given radius around one machine. In event category (b) sample 'Malfunction' events are 'LowOutputPower' and 'LowOutputVoltage'.

Listing 3 shows the event processing rule that calculates a new regional average output power each time when the machine sends a new 'OutputPowerMeasure' event. It collects all 'Output Power' events that are emitted in a distance less than 40 km of a particular machine within the last 10 minutes, and calculates the average of the data using the 'avg'-operator.

[3]Consequently, neither any special resource event types nor a Resource State Agent are necessary.

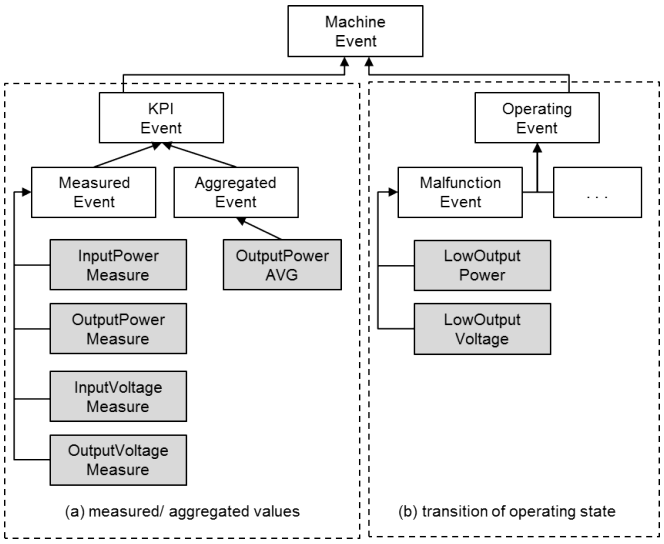

Figure 10: Event model for solar scenario

Listing 3: Solar scenario: Regional average output power

```
CONDITION
  (OutputPowerMeasure AS om → OutputPowerMeasure ↩
      AS or)
  om.distanceTo(or) ≤ 40 kilometers ∧
  om.machineId != or.machineId
  or.avg(data) as average
  [win:time:10 min]
ACTION create OutputPowerAVG(
  machineId = om.machineId,
  distance = 40 kilometers,
  value = average)
```

Because this rule yields an important indicator for the performance of the solar panel, it belongs to the rule base of the KPI Agent in Figure 8.

Usually, geographically adjacent inverters are supposed to produce similar electricity values. Therefore, if the measured output power of one inverter differs significantly from the average power output in the direct neighborhood, a malfunction of the inverter can be assumed.

The rule in Listing 4 specifies a particular 'Malfunction' event 'LowPutputPowerEvent' of the affected inverter (see Figure 4).

Listing 4: Solar scenario: Low output power compared to regional average

```
CONDITION
  (OutputPowerMeasure as om ∧ OutputPowerAVG as ↩
      oavg)
  [win:time:10 min]
  om.machineId = oavg.machineId ∧
  om.value < 0.5 ∗ oavg.value
ACTION create LowOutputPowerEvent(
  machineId = om.machineId)
```

The rule compares two KPI events: the measured output power of one inverter (OutputPowerMeasure) with the average output power in its geographical neighborhood (OutputPowerAVG generated by the rule in Listing 3). If the output power of the inverter is more than 50% below the regional average then the inverter may be disrupted. The rule is part of the rule base of the Operating State Agent in Figure 8.

Another indicator for a possible inverter malfunction is the case that the output voltage of an inverter decreases although the input voltage stays stable or even increases. One possible cause of such a malfunction may be a blown fuse.

Listing 5: Solar scenario: Low output voltage

```
CONDITION
  (OutputVoltage AS oa → OutputVoltage AS ob)
  ∧ (InputVoltage AS ia → InputVoltage AS ib)
  [win:time:15 min]
  oa.machineId = ob.machineId = ia.machineId = ib.↩
      machineId ∧
  oa.value < ob.value ∧
  ib.value ≥ ia.value
ACTION create LowOutputVoltageEvent(
  machineId = cl.machineId
)
```

The event processing rule in Listing 5 specifies the corresponding event pattern: the value of two consecutive 'InputVoltage' events is compared to two consecutive 'OutputVoltage' events. If the output voltage decreases while the input voltage increases then a 'Malfunction' event is derived causing a state transition. (Note that the 'LowOutputVoltage' event is a specialized 'Malfunction' event, see Figure 10)

The sample rules discussed above have been selected to illustrate the processing of solar events. Of course, the rule bases of the KPI Agent and the Operating State Agent comprise many other application-specific rules.

3.2. Scenario 2: Printer Supply & Maintenance Service

Companies, which are selling digital office communication products, are often offering additional services, like preventive maintenance contracts, consulting, carrying out repairs, and a cartridge supply service.

In this case study we discuss the remote monitoring and maintenance service for high performance multi-purpose business printer and copier units (= machines in M2M sense). Again the printers are geographically distributed among several customers without any on-site maintenance staff.

Of course, an office printer has an operational state too. However, in this section we focus on the resource state of the printer. High performance office printers are equipped with one or more cartridges for different colors. Every cartridge is a resource of the printer, having its own life cycle as explained in subsection 2.2.3. The purpose of the cartridge service is to exchange cartridges early enough so that the printer never runs out of resources. Furthermore, the customers wish a just-in-time delivery service so that no stock keeping (and thus no additional burden of administrative work) is necessary, e.g. managing and ordering the cartridges for the printers.

These service requirements can be best achieved by an event-driven M2M solution. The M2M system automatically monitors the resource state transition of a printer. If a critical level state is reached, the service contractor is timely and automatically informed to deliver spare cartridges.

In the following, some sample event processing rules illustrate the processing of printer events.

The event model in Figure 11 shows the KPI events emitted by the printers.[4] Whereas the value of the 'CartridgeLevelMeasure' event indicates the filling level of toner, the 'ConsumptionMeasure' event represents the difference between two measured cartridge levels, i.e. the consumed amount of toner.

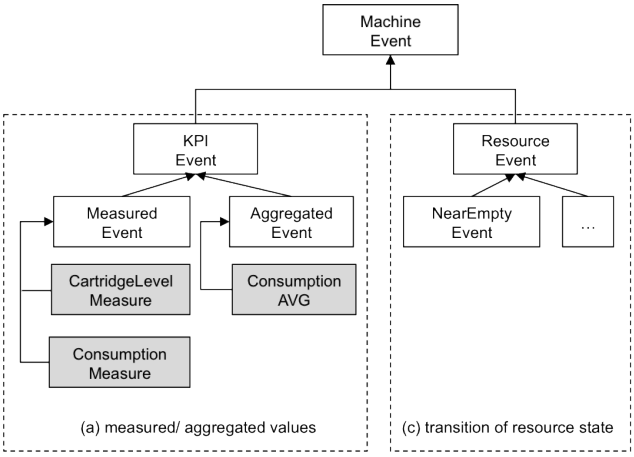

Figure 11: Event model for printer scenario

A new cartridge is starting in the 'sufficient' resource state (see resource state model in Figure 5). Every print job consumes some toner of the cartridges and causes a decreasing cartridge filling level. Corresponding 'CartridgeLevelMeasure' and 'ConsumptionMeasure' events are then transmitted to the M2M system.

The resource consumption and the filling level events are processed by the KPI Agent of the EPN shown in Figure 8. Because the consumption is oscillating, an appropriate average must be calculated. Therefore, all consumption events of a certain machine are aggregated over a longer period of time showing the long term consumption behavior. The calculated 'ConsumptionAVG' event represents the average consumption of a printer.

The corresponding event processing rule is specified in Listing 6. It defines a time window that contains all 'ConsumptionMeasure' events of the last week (line 2-3). These events are grouped by their machine id and aggregated by the *avg*-operator to the weekly average of each machine (line 4-5). A new calculation is triggered with every new event entering or leaving the sliding window.

The Resource State Agent of the EPN in Figure 8 is responsible for the determination of the correct resource state according

[4]The operating state event types (category (b)) are omitted for the sake of simplicity.

Listing 6: Printer scenario: Average consumption per day

```
1  CONDITION (
2      ConsumptionMeasure AS c ∧
3      [win:time:7 days]
4      c.avg(consumptionPerDay) as averageConsumption
5      GROUP BY c.machineId
6    )
7  ACTION create ConsumptionAVG(
8      machineId = c.machineId,
9      consumption = averageConsumption
10   )
```

to the state model in Figure 5. The transition from the 'sufficient' state to the 'criticalLevel' state is the most important one. The assumed time a cartridge will last is calculated by correlating 'CartridgeLevel' and 'ConsumptionMeasure' events.

A rule for a resource state transition is shown in Listing 7. The ConsumptionAVG and CartridgeLevel event streams are joined (line 2-4) in order to calculate the remaining days. If the remaining time is lower then 2 days, the state transition into the 'criticalLevel' state is recognized and a 'NearEmpty' event is created. In line 5, the cartridge filling level (in percentage terms) is divided by the consumption per day (in percentage terms) to determine the remaining time (in days) until the carting is empty.

Listing 7: Printer scenario: (Near)Empty rule

```
1  CONDITION (
2      ConsumptionAVG AS ac ∧
3      CartridgeLevel AS cl ∧
4      ac.machineId = cl.machineId ∧
5      cl.level / ac.consumption ≤ 2 days
6    )
7  ACTION create NearEmptyEvent(
8      machineId = cl.machineId
9  )
```

4. Experimental Results

The applicability and usefulness of our approach have been evaluated by sample implementations of the two case studies introduced in Section 3. All experimental evaluations are based on real-world data supplied by our industrial partner, i.e. from an existing photovoltaic solar power plant and a commercial printer supply & maintenance service, respectively.

In the solar case study, real-world operational data of an industrial solar power plant with 200 GRPS terminals and 2000 solar inverters has been used. The sample data set contains about 160.000 connection events, 100 million measurement events, and more than 30 million state records (see also (Metzdorf et al., 2013)).

Both case studies have been implemented with the open source CEP engine *Esper* (ESPERTECH, 2014). The Esper engine provides the essential features of typical CEP systems like

sliding temporal windows, event pattern operators, and external method calls. The event processing rules are specified in the Esper language EPL, a SQL-like rule language. In contrast to SQL the EPL queries are executed directly in-memory on continuously arriving event streams and not on a database (Bruns & Dunkel, 2010).

Several event processing rules have been implemented for connection problems as well as for application-specific problems. Some of these rules are presented in sections 2.2.6 and 3.

Figure 12 depicts some sample experimental results for the solar power plant case study. The figure visualizes the output power graphs of different solar inverters over time and the calculated regional average (see Listing 3). The marked time range indicates when 'LowOutputPower' events are thrown by the rule in Listing 4 , i.e. a malfunction of the affected inverter is diagnosed.

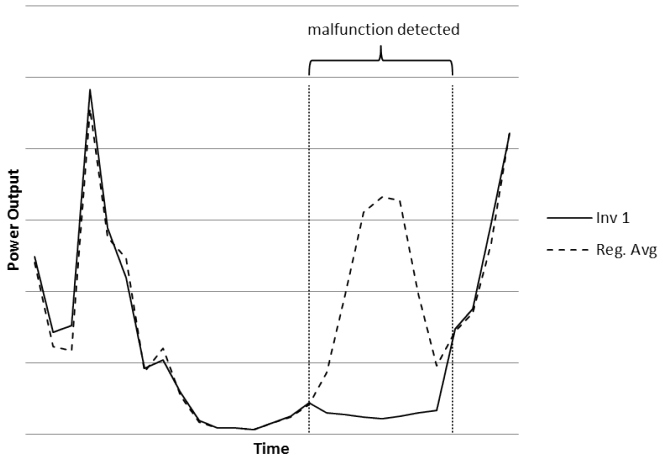

Figure 12: Low output power based on regional average

Our experimental experiences confirm the advanced sense-and-response capabilities of CEP achieving an intelligent correlation of M2M data. Our Esper implementation provides a real-time analysis and diagnosis of high-frequent M2M data streams. The Esper rule engine is capable of processing the extreme huge volume of events in our case studies, deriving without any significant delay the actual states of machines and upcoming problems.

5. Related Work

5.1. Classic M2M Systems

Most of the current M2M literature is focused on network and communication technologies, because a set of different technology components has to work together to build a particular M2M solution (Pandey et al., 2011). As a direct consequence, the technological standardization plays an indispensable role in future M2M development (Chen et al., 2012). Standardization efforts regarding architectural issues concentrate on the overall network architecture for M2M systems. The European Telecommunications Standards Institute (ETSI) has divided M2M systems into three interlinked parts: (1) M2M area domain formed by a M2M area network and M2M gateway, (2) communication network domain consisting of all kinds of wired and wireless networks, and (3) application domain with diverse application-specific services (Chen et al., 2012; ETSI, 2011).

Several approaches have investigated different aspects of the network and communication architecture, e.g., (Wu et al., 2011) and (Wu et al., 2013). In contrast, only very few investigations have been published about architectural considerations for the application part of M2M systems. Usually, the application services of classic M2M solutions store all collected data in a database and process it afterwards by batch processes. Recently, Kitagami et al. (2013) have proposed a data stream processing approach based on a SQLite-based database and the data statistical analysis software R. However, our event-driven approach offers significant advantages compared to database-based approaches (see the discussion in Section 2.3).

In this article, we intend to provide a substantial contribution to the architectural design of the application part that is still missing.

5.2. M2M Systems and Event Processing

In recent years, first attempts have been published that report on the deployment of event processing technologies in the M2M application domain.

Wan et al. (2012) present an event-based high-level architecture for the smart cities domain. It is based on a publish/subscribe middleware with an *Event Manager*. However, the *Event Manager* remains a black box component without any details about its design or technology. In a similar manner, Walczak et al. (2012), Glezer et al. (2007), and Isoyama et al. (2011) suggest to employ complex event processing for handling of streams of machine data. Walczak et al. (2012) report no concrete design details. Glezer et al. (2007) identify the components of a general M2M platform (M2MGen), but without presenting any further design details about the CEP part. Isoyama et al. (2011) present the SCTXPF platform architecture. However, this platform architecture is targeting towards scalability of M2M systems and not, as our approach, towards the concrete design of the M2M application system.

Recently, some vendors of CEP platforms are promoting the M2M area as a new application field for event stream processing (Ericsson, 2012; Kostukovsky & Bowers, 2013). However, without providing any system design guidelines.

Although these aforementioned approaches do not provide any solution concepts or architectural considerations, they show the significance and the general need of intelligent event stream processing and correlation in the M2M domain.

So far, M2M systems are typically tailor-assembled (BMWi, 2011). Generic reference architectures, building blocks, models, or design guidelines showing how CEP can be concretely applied on M2M systems are still missing.

5.3. Sensor Networks and Event Processing

Several approaches have been published that prove the suitability of CEP for processing data streams emitted by sensor

networks. Examples for different sensor domains can be found in Bhargavi et al. (2010), Dunkel et al. (2011), Jeffery et al. (2006), Mouttham et al. (2009), Selvakennedy et al. (2007), and Walczak et al. (2012).

In contrast to event-driven M2M systems, there already exist first design proposals and reference architectures for event processing and sensor networks, see for instance, Bruns & Dunkel (2013), Saleh & Sattler (2013), and Wang et al. (2008). Since sensor networks and M2M communication share some characteristic features, some of the architecture principles and design consideration can be applied on M2M systems as well.

However, sensor networks differ in many aspects significantly from M2M systems. In contrast to the machines of a M2M system, sensors are very restricted devices that are highly specialized just in collecting data of their local environments. Sensors do not provide any additional functionality and, therefore, do not possess operating or resource states. In sensor networks, not sensors are the objects of observation, but the environment that they monitor (Rizvi et al., 2005).

In this article, we extend the previous work on CEP and sensor networks towards the M2M domain by taking the special characteristics of M2M devices into account. We define M2M-specific domain and state models, as well as a general, but M2M-specific event processing network consisting of archetypal EPAs, which can be easily adapted to particular M2M application scenarios.

6. Conclusion

Intelligent M2M systems should be able to diagnose their actual states and to trigger appropriate actions as soon as critical situations occur. But current operating M2M systems are restricted on collecting, aggregating, and presenting low level machine data and have some crucial disadvantages: Processing of machine data is usually implemented by hard-coded algorithms, which are often scattered over the source code. With the consequence, that it is extremely difficult to deal with system changes. For instance, if new types of machines have to be integrated or problem finding algorithms should be changed. Furthermore, most classic M2M systems are not responsive, i.e. they cannot process the huge amount of continuously produced machine data in real-time.

In this paper, we propose a novel event-driven architecture for a decision support system that leads to a new quality of M2M systems, which are intelligent, responsive, adaptable, and flexible.

The main design principle of our software architecture is applying Complex Event Processing as key software technology, because CEP is specialized on processing high-frequent data streams so that we can achieve real-time behavior. Furthermore, stream processing is capsulated in CEP rules that are clearly separated from other source code, and therefore can be adapted and maintained in a flexible and agile manner. However, CEP alone yields only the basic mechanisms of data stream processing, but does not provide any guidelines for building M2M decision support systems. Beyond that we need a software architecture that can serve as a blueprint, which fosters the design and structured development of an entire CEP-based M2M system in industrial practice.

In order to guide the development of M2M decision support systems, we have presented a general software architecture, which distinguishes two different aspects.

First, we have presented some models that describe the M2M infrastructure, and which are required to formulate CEP rules. The *M2M domain model* specifies the basic entities of a all M2M applications such as machines, resources, and communication units. The *M2M event model* defines all explicit events that are directly emitted by the machines and the materialized events generated by CEP rules. Each machine event is related to a certain state model. The *M2M state model* defines the general machine lifecycle considering operating and resource consumption states, as well as Key Performance Indicators (KPIs).

The second view on the system provides the *M2M processing architecture* that consists of an Event Processing Network that determines the different archetypal stages of an event processing pipeline. The stages are realized by Event Processing Agents, who are - among other things - responsible for deriving the machines' operating and consumptions states, as well as for calculating relevant KPIs. Furthermore, we present the processing pipeline for monitoring the wireless communication infrastructure that is application-independent.

The proposed reference architecture defines only the general structure and processing pipeline for arbitrary M2M systems. In order to obtain a concrete architecture for a particular M2M application scenario, it has to be customized and adapted to the specific domain requirements. Two real world examples showed how the reference architecture can be adapted by extending the event types and implementing application-specific event processing rules.

Our research direction is analogous to the well established theory of industrialized software engineering: in particular production lines, reference architectures, and pattern-oriented software development, see for instance, Buschmann et al. (1998); Fowler (2002). Mature principles of professional software engineering practice are applied to the design of intelligent M2M systems in particular, but do also hold for intelligent expert systems in general.

In summary, the proposed architecture of the M2M decision support system meets the following characteristics: First, it provides real-time stream processing of machine data. Moreover, the architecture is highly adaptable and maintainable: Declarative CEP rules can be easily changed and added to meet new requirements. Additionally, various models describing the M2M machine infrastructure leads to a separation of concerns making the formulation of CEP rules easier. Finally, the real-world M2M case studies and the presented experimental results prove the feasibility, usefulness, and the performance of our approach. Thus, our reference architecture is one step towards the professional development of practical CEP-based M2M applications in industrial operation.

Of course, there are also some drawbacks: the proposed architecture is still rather general and, nevertheless, requires considerable efforts for adapting it to a concrete M2M system. In particular, the formulation of CEP rules in real event process-

ing languages like Esper is too complicated for domain experts. Therefore, as a future line of research, we will investigate, how the development of CEP-based M2M systems can be made easier.

One possible approach is the refinement of our architecture to more specific M2M application scenarios: M2M domain and state models specialized to particular M2M system types would further facilitate the adaption of our approach to concrete problems. A similar idea is developing a library of standard M2M rules that consider general phenomenons in M2M systems.

To benefit from our proposed M2M architecture, a developer needs a sound technical background. For non-technical users, who are the domain experts in the problem under investigation, even the understanding of the CEP rules is very difficult. Therefore, we are currently developing a Domain Specific Language (DSL), which simplifies the specification of our system architecture and allows the definition of M2M specific rules using M2M domain concepts. The M2M DSL would enable the developer to write event processing code on a higher level of abstraction, based on self-defined domain-specific language constructs.

Finally in our approach, the specification of the M2M infrastructure is still based on proprietary class and state models that must be transformed into appropriate programming code. As a future line of research, we will investigate how semantic web technologies can be used to standardize the specification of M2M systems. Furthermore, reasoning on appropriate M2M ontologies could prove the consistency of the specified infrastructure and derive automatically further characteristics of the M2M system.

Acknowledgement

This work has been funded by the Federal Ministry of Economics and Technology based on a decision of the German Bundestag with grant number: KF2425003MS2.

References

Babkin, S. A. (2014). Complex Event Processing with Triceps CEP v1.0. URL: `Retrieved from http://triceps.sourceforge.net/`.

Bhargavi, R., Vaidehi, V., Bhuvaneswari, P., & Chandra, G. (2010). Complex event processing for object tracking in wireless sensor networks. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology* (pp. 211–214).

BMWi (2011). *Machine-to-Machine-Kommunikation - eine Chance für die deutsche Industrie*. Technical Report Bundesministerium für Wirtschaft und Technologie, Nationaler IT Gipfel.

Bruns, R., & Dunkel, J. (2010). *Event-Driven Architecture: Softwarearchitektur für ereignisgesteuerte Geschäftsprozesse*. Berlin Heidelberg: Springer-Verlag. In German.

Bruns, R., & Dunkel, J. (2013). Towards pattern-based architectures for event processing systems. *Software: Practice and Experience*, . doi:10.1002/spe.2204.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1998). *Pattern-Oriented Software-Architecture Volume 1: A System of Pattern*. Addison Wesley.

Chandy, K., & Schulte, W. (2010). *Event Processing: Designing IT Systems for Agile Companies*. McGraw-Hill.

Chen, M., Wan, J., & Li, F. (2012). Machine-to-machine communications: Architectures, standards and applications. *KSII Transactions on Internet and Information Systems*, *6*, 480–497.

Dunkel, J., Fernández, A., Ortiz, R., & Ossowski, S. (2011). Event-driven architecture for decision support in traffic management systems. *Expert Systems with Applications*, *38*, 6530–6539.

Ericsson (2012). Bringing stream analytics to M2M-based smartlife scenarios. URL: `Retrieved August 26, 2014, from https://labs.ericsson.com/blog/bringing-stream-analytics-to-m2m--based-smartlife-scenarios`.

ESPERTECH (2014). Event processing with Esper and NEsper. URL: `Retrieved from http://esper.codehaus.org/`.

ETSI (2011). European Telecommunications Standards Institute TS 102 690 Machine-to-Machine communications M2M; Functional architecture. URL: `Retrieved August 26, 2014, from http://www.etsi.org/deliver/etsi_ts/102600_102699/102690/01.01.01_60/ts_102690v010101p.pdf`.

Etzion, O., & Niblett, P. (2010). *Event Processing in Action*. Manning.

Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley.

Glezer, C., Krishnamurthy, S., Schloeder, K., Anson, O., & Tahan, G. (2007). M2MGen- an application generator for machine to machine (M2M) applications. In *Proceedings of the 1st International Workshop on RFID Technology - Concepts, Applications, Challenges, IWRT 2007, Funchal, Madeira, Portugal* (pp. 133–140). INSTICC PRESS.

Isoyama, K., Sato, T., Yoshida, M., & Kida, K. (2011). Large-scale real-time processing technology for M2M service platform. *NEC Technical Journal*, *6*, 90–94.

JBoss (2014). Drools Fusion. URL: `Retrieved from http://www.jboss.org/drools/drools-fusion.html`.

Jeffery, S., Alonso, G., Franklin, M., Widom, J., & Hong, W. (2006). A pipelined framework for online cleaning of sensor data streams. In *Proceedings of the International Conference on Data Engineering (ICDE)* (pp. 140–142). IEEE.

Kitagami, S., Yamamoto, M., Koizumi, H., & Suganuma, T. (2013). An M2M data analysis service system based on open source software environments. In *Proceedings of the 27th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2013, Barcelona, Spain* (pp. 953–958). IEEE Computer Society.

Kostukovsky, O., & Bowers, W. (2013). Dynamic M2M event processing. URL: `Retrieved August 26, 2014, from https://www.eclipsecon.org/na2014/sites/eclipsecon.org.na2014/files/slides/EclipseCon2014-DynamicEventProcessing_draftV2.pdf`.

Luckham, D. (2002). *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Reading, MA: Addison-Wesley.

Luckham, D., & Schulte, R. (2011). Event processing glossary - version 2.0. URL: `Retrieved August 26, 2014, from http://www.complexevents.com/wp-content/uploads/2011/08/EPTS_Event_Processing_Glossary_v2.pdf`.

Metzdorf, M., Bruns, R., Dunkel, J., Masbruch, H., Hellwich, I., & Kasten, S. (2013). Complex Event Processing für intelligente mobile M2M-Kommunikation. In *ITG-Fachbericht der 18. ITG-Fachtagung Mobilkommunikation: Technologien und Anwendungen* (pp. 58–63). VDE Verlag.

Mouttham, A., Peyton, L., Eze, B., & El Saddik, A. (2009). Event-driven data integration for personal health monitoring. *Journal of Emerging Technologies in Web Intelligence*, *1*, 144–148.

Pandey, S., Choi, M.-J., Kim, M.-S., & Hong, J. W. (2011). Towards management of machine to machine networks. In *Proceedings of the 13th Asia-Pacific Network Operations and Management Symposium (APNOMS)* (pp. 1–7). IEEE.

Rizvi, S., Jeffrey, S., Krishnamurthy, S., Franklin, M., Burkhart, N., & Liang, L. (2005). Events on the edge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 885–887).

Saleh, O., & Sattler, K.-U. (2013). Distributed complex event processing in sensor networks. In *Proceedings of the 14th IEEE International Conference on Mobile Data Management (MDM)* (pp. 23–26).

Selvakennedy, W., Rohm, U., & Scholz, B. (2007). Event processing middleware for wireless sensor networks. In *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW)* (p. 65). IEEE.

Walczak, D., Wrzos, M., Radziuk, A., Lewandowski, B., & Mazurek, C.

(2012). Machine-to-machine communication and data processing approach in future internet applications. In *Proceedings of the 8th IEEE International Symposium on Communication Systems, Networks and Digital Signal Processing* (pp. 1–5). IEEE.

Wan, J., Li, D., Zou, C., & Zhou, K. (2012). M2M communications for smart city: An event-based architecture. In *Proceedings of the 12th IEEE International Conference on Computer and Information Technology, CIT 2012, Chengdu, Sichuan, China* (pp. 895–900). IEEE Computer Society.

Wang, W., Sung, J., & Kim, D. (2008). Complex event processing in EPC sensor network middleware for both RFID and WSN. In *Proceedings of the 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)* (pp. 165–169).

Wu, G., Talwar, S., Johnsson, K., Himayat, N., & Johnson, K. D. (2011). M2M: From mobile to embedded internet. *IEEE Communications Magazine*, *49*, 36–43.

Wu, H., Zhu, C., La, R. J., Liu, X., & Zhang, Y. (2013). FASA: Accelerated S-ALOHA using access history for event-driven M2M communications. *IEEE/ACM Transactions on Networking*, *21*, 1904–1917.