ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

Complex reactive event processing for assisted living: The Habitat project case study

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

*Published Version:*
Complex reactive event processing for assisted living: The Habitat project case study / Loreti, Daniela; Chesani, Federico; Mello, Paola; Roffia, Luca; Antoniazzi, Francesco; Cinotti, Tullio Salmon; Paolini, Giacomo; Masotti, Diego; Costanzo, Alessandra. - In: EXPERT SYSTEMS WITH APPLICATIONS. - ISSN 0957-4174. - STAMPA. - 126:(2019), pp. 200-217. [10.1016/j.eswa.2019.02.025]

*Availability:*
This version is available at: https://hdl.handle.net/11585/675075 since: 2019-03-04

*Published:*
DOI: http://doi.org/10.1016/j.eswa.2019.02.025

*Terms of use:*

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

28 June 2024

This is the final peer-reviewed accepted manuscript of:

**Loreti, D., Chesani, F., Mello, P., Roffia, L., Antoniazzi, F., Cinotti, T. S., . . . Costanzo, A. (2019). Complex reactive event processing for assisted living: The habitat project case study.** ***Expert Systems with Applications, 126*****, 200-217.**

The final published version is available online at: http://dx.doi.org/10.1016/j.eswa.2019.02.025

Rights / License:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

*This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/)*

***When citing, please refer to the published version.***

# Complex Reactive Event Processing for Assisted Living: the Habitat Project Case Study

Daniela Loreti[a,*], Federico Chesani[b], Paola Mello[b], Luca Roffia[b], Francesco Antoniazzi[b], Tullio Salmon Cinotti[b,d], Giacomo Paolini[c], Diego Masotti[c], Alessandra Costanzo[c]

[a] *CIRI - Health Sciences & Technologies, University of Bologna. Via Tolara di Sopra, 41/E, Ozzano dell'Emilia (BO), Italy*
[b] *DISI - Department of Computer Science and Engineering, University of Bologna. Viale Risorgimento 2. Bologna, Italy*
[c] *CIRI - Information and Communication Technologies, University of Bologna. Via Rasi e Spinelli 176. Cesena (FC), Italy*
[d] *ARCES - Advanced Research Center on Electronic Systems "Ercole De Castro", University of Bologna. Via Toffano, 2. Bologna, Italy*

**Abstract**

While the increasing average age of population is posing new challenges to societies and healthcare systems, the emergence of the Internet of Things research area is generating the hope for automated assisted environments, which could combine the advances in sensors networks with that of runtime monitoring systems, in order to create smart houses able to take care of their older inhabitants and delay the recourse to hospitals and nursing homes. However, although various assisted living systems have been proposed in the last decade, the goal of realizing an effective domestic support system for elderly is still far from reached.

In this work, we present a project aiming to re-engineer a set of everyday life objects, equipping them with environmental and wearable sensors, thus

*Corresponding author

*Email addresses:* daniela.loreti@unibo.it (Daniela Loreti ), federico.chesani@unibo.it (Federico Chesani), paola.mello@unibo.it (Paola Mello), luca.roffia@unibo.it (Luca Roffia), francesco.antoniazzi@unibo.it (Francesco Antoniazzi), tullio.salmoncinotti@unibo.it (Tullio Salmon Cinotti), giacomo.paolini4@unibo.it (Giacomo Paolini), diego.masotti@unibo.it (Diego Masotti), alessandra.costanzo@unibo.it (Alessandra Costanzo)

to monitor the condition of older people in their domestic residences and provide security while preserving the autonomy and independence of the subjects. The main focus of the paper at hand is on the requirements and solutions implemented to realize the backbone infrastructure of such system as regards both the adopted semantic message routing mechanism and the newly conceived approach to event analysis, which combines Complex Event Processing and a reactive implementation of Event Calculus.

*Keywords:* Internet of Things, Complex Event Processing, Reactive Event Calculus, Assisted Living

---

## 1. Introduction

Nowadays, as the average age of population increases, the need for solutions to support and lengthen the autonomy of elderly has become more and more pressing. In order to maintain the dignity of older people, it is crucial to allow them to continue living in their everyday environment, avoiding specialized centers and hospitals (Centers for Disease Control and Prevention, 2013).

Over the last decade, the availability of low-cost electronic devices, sensors and actuators, and the diffusion of Internet of Things (IoT) research area have been seen as enabling factors for the creation of assisted environments, which can turn common houses into flexible, adaptive and reactive places, where the autonomy of people is no longer opposed to security but tightly bounded to it. This would allow old or disabled subjects to remain in their houses longer, relaying on a hardware/software infrastructure to ensure a close interaction with family and healthcare professionals.

Besides this strategy positively affecting elderly's self-confidence and independence (and ultimately exponentially increases the quality of their life), it also brings positive consequences to the society from an economic point of view, in terms of reduction of the assistance costs. In fact, as also underlined by the 2015's *World report on ageing and health* by the Word Health Organization (World Health Organization, 2015; Beard et al., 2016), "healthy ageing is more than just the absence of disease" and the healthcare costs to society are not only related to the actions made to foster old people's functional abilities, but also to "the benefits that might be missed if countries fail to make the appropriate adaptations and investments". The recommended societal approach to population ageing includes the objective of building an

age-friendly world. The report also highlights the need to transform health systems away from disease-based curative models but towards the provision of *integrated care*, which should be centered on the needs of older people. The goal of developing IoT-based assisted environments goes precisely in this direction.

Although some enhancements has been made, the objective of realizing a domestic monitoring system (possibly provided with a nice and easy-to-use interface), able to give an effective runtime support to user's everyday life, still remains far from achieved.

In this framework, the Habitat project (HABITAT, 2016) has been designed to apply the advances of IoT research field to the creation of a smart assisted domestic environment, capable of adapting to the evolving requirements of aging or disabled people. The project aims to re-engineer a set of objects from everyday life, equipping them with environmental and wearable sensors and actuators, which should result non-invasive during daily user activities. The behavior of the system as a whole is required to adapt to the actual abilities of each user, thus helping (or preventing) him to carry out actions depending on her specific profile.

One of the main challenges of the project was the development of an effective backbone software infrastructure, to collect row monitoring data from sensors and extract elaborated information from them, including reaction strategies in case of emergency. Such infrastructure is also responsible for connecting the physical world with the information world: autonomous reasoning capabilities must be applied to a digital representation of the actual environment. Based on such capabilities, interactive services must be provided to a multitude of stakeholders. In the end, the infrastructure is required to manage the inter-play among objects (like furniture and appliances) and people (with different profiles), eventually taking into consideration "global knowledge" originated outside the surrounding environment.

From the technical point of view, the design of such reliable infrastructure must take into account various requirements. First of all, as the Habitat system consists of heterogeneous, multi-vendor objects and devices (sensors, actuators, human interfaces), which are not originally designed to work together but need to co-operate in order to achieve a common goal, *interoperability* is a primary and crucial requirement.

Furthermore, these entities may runtime connect or disconnect from the system, and new objects, appliances and services might be added during the application life-cycle (possibly avoiding any reconfiguration), thus making the

*support to a dynamic environment* another important requirement. Changes within the system must be detected, related to the context and notified to the interested users according to the application business logic. Eventually, context-dependent services need to be provided by interactive devices acting as mediators between people and the surrounding space (Bartolini et al., 2012).

As the re-engineered objects provide a large continuous stream of information about the environment and its guests, the Habitat platform is required to supply an *efficient and reliable processing mechanism*. The data stream from various sensors can be actually seen as a flux of occurred events, which might not be significant if considered individually, but might on the contrary provide precious information if correctly grouped. Depending on the complexity of properties that the system is required to check, the analysis of the data flowing from sensors can be very burdensome in terms of computing capabilities. The requirement of being able to provide sudden *runtime* reactions in case of detected critical conditions further exacerbates the problem.

Finally, as the Habitat system necessarily intertwines the information from several interconnected elements with a knowledge base of different user profiles, the nature of the concepts and properties that needs to be monitored is often very complex, demanding a *highly expressive notation* to represent the relevant situations that might occur in the domestic environment. Nevertheless, this expressive notation should not increase the complexity of temporal reasoning on the events and their effects.

In order to meet these multiple requirements, the Habitat project leverages three key technologies:

- a semantic message routing engine named SPARQL Event Processing Architecture (SEPA) (Roffia et al., 2018), to enable *interoperability* and *dynamicity*;
- a knowledge-based Complex Event Processing (CEP) (Luckham, 2008) engine, to provide an *efficient runtime* mechanism to monitor the flow of events from various sensors;
- Reactive Event Calculus (REC) (Chesani et al., 2010a; Bragaglia et al., 2012a), to support temporal reasoning and provide a *highly expressive* – yet *easy-to-manage* – mechanism to represent the properties going to be monitored.

In this paper, we discuss how the advances in semantic message routing, event processing and temporal reasoning has been applied to the real use case of the Habitat project to realize an effective Decision Support System (DSS) for elderly and healthcare professionals. Therefore, the main focus is on the platform's backbone infrastructure, responsible to collect row monitoring data from sensors, extract further information, and provide useful suggestions for an healthy life or reactions to dangerous situations.

The main contributions of the work at hand can be summarized as follows:

- a description – in the framework of the Habitat project requirements – of the proposed mechanism for semantic message routing, as well as the advantages brought by this approach;
- a discussion about how CEP and REC can be intertwined to realize an efficient and easy-to-manage knowledge-based DSS, which allows aging people to live in the their present domestic environments without giving up security;
- a practical overview of some interesting scenarios in which the application of REC onto a platform for CEP significantly simplifies the implementation of the temporal rules;
- a performance evaluation of the main components of the Habitat system.

In the following, we give a deeper insight of the Habitat project, its desired features and motivations for chosen technologies (Section 2). We also provide a description of the architecture of the system (Section 3), giving particular attention to the information routing engine and the implemented DSS. In Section 4, some real scenarios of the Habitat system are presented, whereas in Section 5 the Habitat project's performance is evaluated in real and simulated environments. Related work and conclusion follow.

## 2. Project aim and enabling technologies

Habitat has been designed to partially answer to the need of independence coming from aging people living alone or occasionally assisted by relatives and caregivers. The project, which has been developed over a period of two years from January 2016, started from the initial elicitation of the system requirements emerged from a series of interviews to older persons, their

relatives and healthcare professionals, conducted by means of ASC Insieme (ASC, 2018), a social cooperative specialized in the care of elderly. This initial requirement analysis confirmed that, in order to increase the quality of life of the aging population, it is crucial to reduce the use of nursing homes. The more an elderly person can continue to inhabit his normal spaces in autonomy, the more are the positive effects on his self-esteem and ultimately on physical and mental well-being.

For this reason, the project aims to enrich the house of the subject with a collection of re-engineered objects equipped with sensors and actuators, capable to interact with each other and realize an advanced assisted environment.

As discussed in the previous section, aiming to realize a robust backbone architecture for such system, *interoperability* and support to *dynamicity* are primary requirements to allow inter-object cooperation. With this purpose, Habitat builds on top of previous research in the domain of *smart spaces* introduced by Lassila (Lassila, 2007) and further developed in several European projects, including SOFIA (SOFIA, 2018). A smart space is a digital representation of the physical world (i.e., of the environment and the objects therein located), stored in an interoperable, machine understandable format, kept up to date and made available to interested applications (Ovaska et al., 2012). The adopted solution for information routing implements the smart space concept using Semantic Web technologies (Berners-Lee et al., 2001) i.e., through ontologies mapped onto a common data model represented as a direct labelled graph according to the Resource Description Framework (RDF). Smart spaces lead to an extremely simple and general approach to solve Habitat interoperability, evolvability and dynamicity issues, as well as its message exchange needs (Ovaska et al., 2012). Indeed, a platform hosting a smart space just requires a repository for storing and managing graphs and a graph query language – e.g., SPARQL Protocol and RDF Query Language (SPARQL) – to extract information, which might even be not explicitly stated. All objects and devices connected to the smart space share the same representation of the environment, co-operate and are called *smart objects*.

Starting from this background, the smart space solution proposed by Habitat (Antoniazzi et al., 2017) consists of the following components: (*i*) a shared ontology modeling all the Habitat players, their instances and the relevant static and dynamic relations between them. This model, based on the Semantic Web technologies, is needed to enable interoperability at information level i.e., mutual understanding of all Habitat entities; (*ii*) a

SPARQL endpoint hosting the smart space; (*iii*) a semantic engine with semantic event processing and notification capabilities named SEPA (Roffia et al., 2018). Through it, the smart space is continuously updated by the smart objects, which, in turn, may subscribe to be alerted upon a specific and semantically defined context-change event; (*iv*) a set of multi-language APIs, to enable the interaction between the smart space and the smart objects.

On top of ensuring information level interoperability, this approach has the great advantage of decoupling data and code, thus providing a quick and easy way to develop the desired application (Roffia et al., 2016). At the same time, evolvability and dynamicity are supported, as new devices and entities may be added without reconfiguring the existing software agents, as long as the ontology is properly designed. As all Habitat objects and devices (including interactive devices with man in the loop) are connected to the smart space, they share the same semantic and dynamic representation of the environment.

Besides interoperability, another crucial issue when developing the backbone infrastructure of the Habitat IoT system is the need for an *efficient runtime analysis* of the flow of data coming from the smart objects in order to extract complex knowledge about the state of the system and in particular, information about the monitored subject. Therefore, the chosen message routing engine must work alongside a solid and efficient data processing mechanism. Furthermore, as a key feature of the Habitat system is the possibility to adapt the monitoring rules to the needs of different subjects, the flexibility in configuration is another important requirement of the system. For example, in an environment where more than one elderly or disabled people is present, some may require assistance while doing a particular action – like walking down the stares –, while some others may not. Forcing everyone to have the same kind of assistance would inevitably bring useless help to autonomous persons and insufficient assistance to more disabled people: both cases compromise the quality of life. It is also important to consider that even the assistance needs of a single subject may change over time as his physical conditions evolve.

For this reason, the backbone infrastructure of the IoT system is enriched with a configurable and easy-to-use mechanism based on CEP to express the monitoring properties. This software technology has been conceived to process high-frequency streams of events. Everything that happens inside (e.g., signals form sensors) or outside (e.g., external input from the caregiver or family) the environment can be seen as an event (Etzion & Niblett, 2010;

Luckham, 2008). Some happenings, like for example, the presence of a subject in a certain position of the domestic environment, might not be relevant if considered individually. The role of CEP is to contextualize the events and relate them to other previous happenings, thus to create more complex and meaningful events. For example, if the subject is identified in the system database as an old person with initial physical decay, and the sensors reported the same position – correspondent to a particularly significant zone, like the bathroom – for a certain amount of time that is considered dangerous, the CEP engine can trigger another more complex event signifying that the subject is likely to be fallen or be in a risky situation. Consequently, other actions should be carried out, like for example, report the fact to a relative.

In other words, the synthetic complex event that is generated by the system correlates several simple, lower-level events in a more meaningful event, thus providing the real benefits of CEP. This technology employs sliding windows and temporal operators, often with a declarative fashion, to simplify the specification of temporal reasoning. Nevertheless, in a complex environment, such as Habitat, the variety of sensor signals and situations that require a notification to users, might exacerbate the difficulties of monitoring property specification. Actually, although the declarative nature of CEP rules is able to simplify the identification of significant situations, temporal reasoning remains a complex topic: the triggering of an alarm and the kind of reaction to be provided by the assisted environment might depend on the occurrence of several different events in more or less recent windows of time.

For example, a simple policy stating that different reactions should be carried out by the system whether the caregiver is present in the house or not when the subject is detected to be seated with an incorrect posture for more than $M$ minutes, entails a non-trivial reasoning on the presence and absence of low level events – like the signal from sensors revealing the caregiver presence, or the data about the correct/incorrect posture from other sensors – on different windows of time. Furthermore, the notifications might need to be repeated over time in case of no answer (from the subject or caregiver), while the continuous triggering of the same notification as long as the significant situation is not resolved is certainly to be avoided. These simple examples, already highlight how the specification of the properties to be monitored over time might quickly become very difficult. In order to manage this complexity, it is important to provide a *highly expressive* mechanism to represent the analysed situation.

To this end, we relay on a reactive and logic-based version of Event Calculus (EC), named REC. Since its introduction in 1986 (Kowalski & Sergot, 1989), EC has been recognized as an excellent framework for reasoning about time and events. Its logic-based formalism was expressly conceived for specifying in a declarative manner how the happening of events affects the state of the environment. While EC is basically deductive, and is typically used to reason about a fixed history of events, its extension, REC (Bragaglia et al., 2012a) was introduced to cope with monitoring tasks, where events continuously happen at a fast rate and restarting the reasoning from scratch at each event arrival – as would be required by EC – results unpractical. In the Habitat platform, the employment of REC has a twofold advantage: on one hand, allows to specify very complex action-reaction mechanisms thanks to its highly expressive logic; on the other, it greatly simplify the management of the several events and situations occurring in the system.

## 3. Architecture of the system

The architecture of the Habitat system has been conceived to integrate a collection of heterogeneous smart devices into a single assisted environment. As underlined in Fig. 1, some of these everyday life objects includes sensors to enable the collection of low level parameters, while some others have been created to work as interfaces between the system and the elderly subject or the caregiver. There are four types of sensor-equipped objects:

- a *smart brooch*, composed of an active Radio-Frequency Identification (RFID) tag that can be pinned to the clothes of the subject going to be monitored;
- a *wall lamp*, equipped with a RFID reader able to track the presence of the smart pin, thus to realize an indoor localization system;
- a *wearable belt*, able to analyze the quality of the subject's movements in the space (e.g., number, length and balance of steps) thanks to an integrated inertial sensor; and
- a *smart chair*, equipped with load cells under its feet and seat, to detect when a user is seated or attempts to stand-up, as well as incorrect positions (which can cause postural discomfort or even falls).

The data coming from these sensors are collected through the SEPA semantic message routing engine and analyzed with REC temporal reasoning over a knowledge based CEP engine, to finally realize a DSS, which provides suggestions for the subjects and caregivers or triggers alarms when critical situations are detected.

The monitored subject and the caregiver interact with the system through tree types of objects:

- a *smartphone* with a preinstalled *specific application*, useful to deliver messages to the environment inhabitants and, at the same time, to collect feedbacks from them;
- a *wall panel*, which can operate as a common television or radio but is useful to remark and emphasize the messages sent by the Habitat platform to the smartphone's application; and
- a *web interface* to monitor the state of the system and edit the user profiles.

### *3.1. Indoor localization system*

Among all the sensor-equipped objects, a crucial role in the monitoring mechanism is played by the indoor localization system, which is performed by a RFID system composed of a single reader and several active tags. The

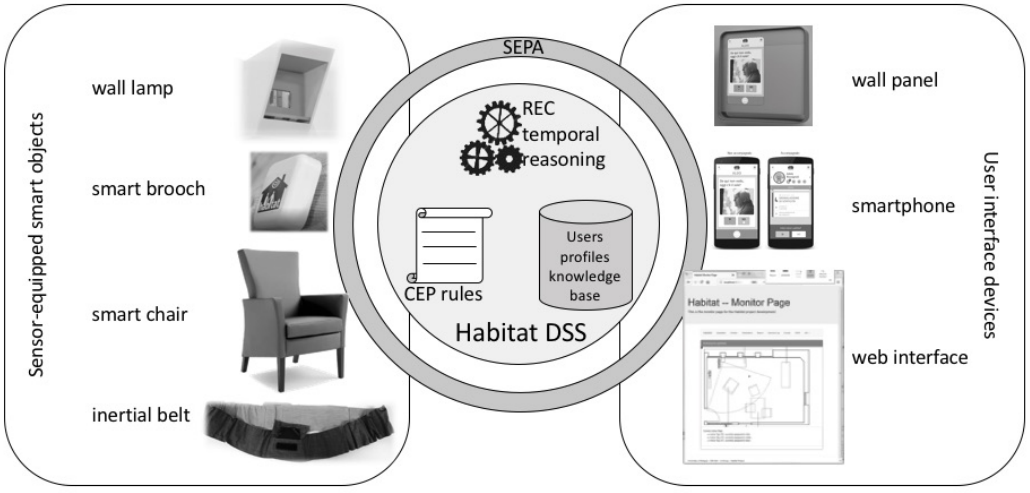


Figure 1: Architecture of the Habitat system

current prototype of the reader, called Remotely Identify and Detect (RID) (Del Prete et al., 2013), operates at a frequency of 2.4 GHz, belonging to the Industrial, Scientific and Medical (ISM) Radio Band, and is based on an array of two flag-type dipole antennas, able to create, according to the Monopulse radar technique, two different radiation patterns: the Sum ($\Sigma$), thanks to the in-phase feeding, and the Difference ($\Delta$), derived from the out-of-phase feeding, whose a-posteriori elaboration is able to guarantee the appropriate figure of merit (Maximum Power Ratio) for the detection of the tags. Finally, the electronic beam steering allows to retrieve the correct angular position of the tags at a certain time with centimeter-level accuracy.

For the calculation of the distance between the tag and the RID, the maximum value of Received Signal Strength Indicator (RSSI) at the $\Sigma$ Channel is exploited, using both the value of this RSSI at a reference distance (i.e., one meter) and a path-loss model to simply characterize the radio channel of the indoor scenario under evaluation. The RID-tag distance $d$ is therefore calculated as follows:

$$d = 10^{\frac{P_0 - P_R}{10^n}}, \tag{1}$$

where $P_0$ and $P_R$ are the maximum values of RSSI received at $\Sigma$ port during calibration at the reference distance and in real-time, respectively, and $n$ is the path-loss exponent (representing the radio propagation in the location under test: typically, 2 for free space, down to 1.6 for indoor environments) (Liang & Krause, 2016).

The room has been also sectorized into three calibration areas, characterizing them with different calibration and path-loss values: the reader is able to track and locate tagged people in a reading zone going from -45° to +45° with respect to the azimuthal plane. The interested reader can find further details about the indoor localization system in (Antoniazzi et al., 2017).

The RID-equipped wall lamp has been located approximately at 165 cm of height: this could prevent or minimize the effects of humane blockage or other unwanted interferences. After several trials, also some feasible positions of the active tags have been selected: in a cap on the head, on the chest as a pendant, or – as discussed in this work – on a shoulder as a brooch.

The angle and the distance of tags collected by the RID are converted into $(x, y)$ positions and sent to the Habitat DSS through the SEPA semantic routing. Then they are further elaborated into more complex events by the knowledge based CEP engine. The following sections clarify the working principles of the message broker and discuss how CEP and REC can be

intertwined to realize a reliable and easy-to-manage event processing system.

### *3.2. SEPA engine*

SEPA (SPARQL Event Processing Architecture) (Roffia et al., 2018) is a decentralized Web-based architecture designed to support the development of distributed, dynamic, context-aware and interoperable services and applications. As shown in Fig. 2, SEPA provides a layer over the Linked Data that enables the detection and notification of changes in the underpinning RDF graph(s).

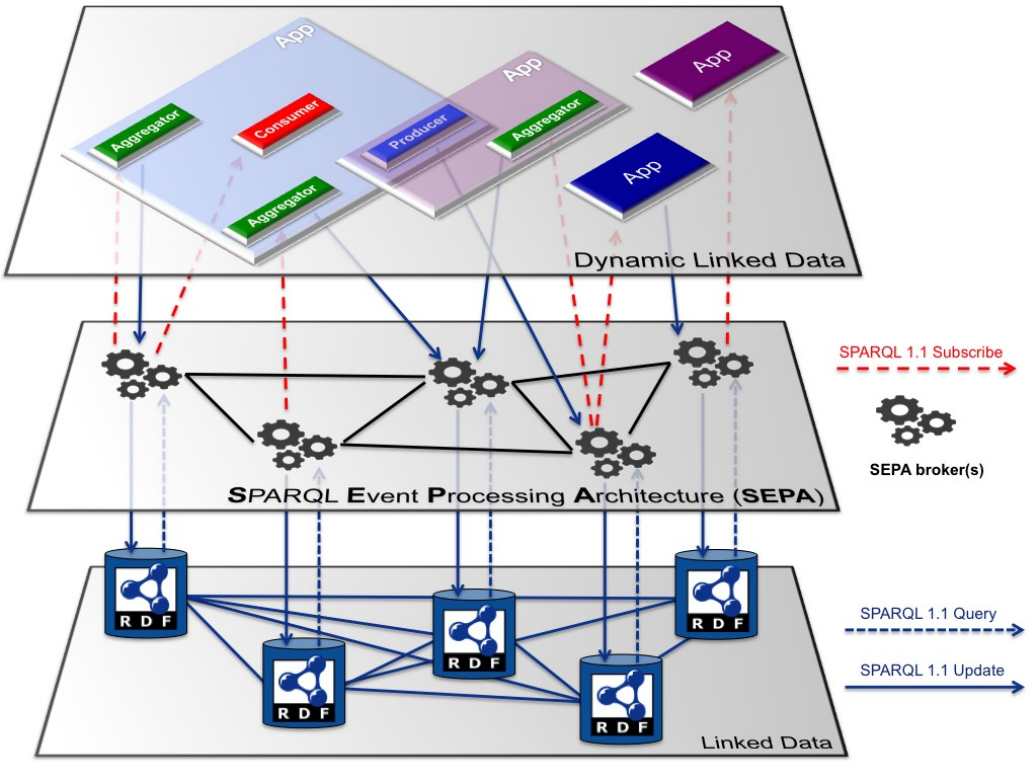


Figure 2: SEPA, SPARQL Event Processing Architecture

A core component of the architecture is the SEPA broker which implements a content-based publish-subscribe mechanism where the W3C SPARQL 1.1 Update (SPARQL Update, 2013) and Query (SPARQL Query, 2013) languages are fully supported and used respectively by publishers and subscribers. The architecture is built on top of the W3C SPARQL 1.1 Protocol

(SPARQL Protocol, 2013) and introduces the SPARQL 1.1 Secure Event (SE) Protocol (SPARQL SE Protocol, 2018) and the SPARQL 1.1 Subscribe Language (SPARQL Subscribe, 2018). The former allows agents to interact with a SEPA broker like with a standard SPARQL Protocol service (also known as SPARQL endpoint), but at the same time, it allows to convey subscriptions and notifications expressed according to the latter. The SPARQL 1.1 SE Protocol also support client and server authentication, data encryption and message integrity – i.e., based on JSON Web Tokens (JSON Web Tokens, 2015). The SEPA reference implementation is available on GitHub (SEPA, 2018) and uses the Websocket protocol (WebSocket, 2011) to convey subscription requests and notifications. APIs in several programming language are available, including Java, Python, JavaScript and Go.

SEPA provides a design pattern where an application can be seen as a collection of agents. As shown in the top layer of Fig. 2, each agent plays a specific role within an application i.e., *producer, consumer* or *aggregator*, and can be shared among different applications. While a producer publishes events by means of a SPARQL 1.1 Update, a consumer is subscribed to specific events through a SPARQL 1.1 Query. An aggregator plays both roles: it is subscribed to events and generates new events based on the content of the received notifications.

SEPA introduces the JSON SPARQL Application Profile (JSAP) (JSON, 2018) as a mean to describe an application. JSAP is a JSON file describing a SEPA application by including the parameters needed to interact with one (or more) SEPA broker instance(s), along with all the SPARQL 1.1 Updates and Queries used by the application.

### *3.3. Habitat Decision Support System*

In order to cope with the complexity of event stream analysis for IoT applications, it is crucial to arrange the flow of data from the smart environment according to a classification that can be employed at runtime to simplify the monitoring process. For this reason, the knowledge-based DSS implemented for the Habitat project organizes the flow of data according to two main directives:

- a *vertical* composition of events from the simplest to the most complicated – i.e., semantically pregnant –, as to realize a hierarchical organization of the facts reported by the IoT environment;

- a *horizontal* arrangement of happened (simple or complex) events according to their temporal occurrence.

Orthogonally to these vertical/horizontal directives, it is also important to consider that the Habitat monitoring process is further complicated by the need to perform temporal analysis not only on the status of the subject, but also on the generated notifications/answers (to/from the subject or caregiver), as they are an integral part of the set of events to be analyzed.

Thanks to its founding feature of enabling pattern matching over incoming events and their continuous transformations into semantically richer and richer events, CEP is the perfect candidate to perform the vertical organization of the observed facts. Actually, CEP is a technology (Luckham, 2008; Luckham & Schulte, 2011) to enable the continuous stream processing of general data flows in near real-time. The key element of CEP is the ability to provide context-awareness through in-memory pattern matching i.e., every raw information coming from the environment is considered as an event and CEP allows to specify those patterns in the event stream that represent meaningful situations in the application domain. According to CEP, the stream analysis follows three steps: 1. the data coming from sensors are captured and interpreted as events; 2. this event flow is inspected to identify patterns with a particular meaning for the domain; 3. a reaction to the identified situation is triggered as a response.

In particular, the matching of an event pattern means that the system is in a significant state from the point of view of the application domain. The reaction to this state can be either the triggering of a domain-specific action, or the generation of a novel, more complex event. The latter has the purpose to correlate several simple, low-level events into a semantically richer one, which is put back into the event flow and can be later involved in other matching operations. As these more complex events allow a progressive gain of insight into the current state of the system, they represent the key feature of CEP's context-awareness.

In order to apply CEP's theory, the Habitat DSS leverages Drools Fusion (Drools, 2018), an efficient and widely employed *forward chaining* engine for CEP based on the *Rete* algorithm (Forgy, 1982). According to Drools prescriptions, event processing rules are composed of two parts: a *logic condition*, that declares the pattern that should be matched in order to fire the execution of the next part; and a sequence of *actions*, that must be carried out when the condition is satisfied. Listing 1 exemplifies the structure of

Drools Fusion's rules.

Listing 1: Example of CEP rule according to Drools Fusion syntax

```
1 rule "Rule title/description"
2 when
3   logic condition on knowledge base facts or events from EventStream
4 then
5   actions to be executed
6 end
```

In order to express the temporal relations between events in the logic condition, Drools allows the employment of sliding windows and temporal operators, typically in a declarative fashion, which accounts for the clarity and ease-of-use of its approach. Indeed, as the domain of temporal reasoning is intrinsically characterized by high variability and necessity to deal with many execution alternatives, the declarative nature of the logic condition part represents a crucial advantage of Drools because it allows a simpler elicitations of the matching constraints with respect to procedural approaches.

Despite the well-known advantages brought by the employment of CEP in various domains, in the context of an IoT application like Habitat, event analysis remains an articulated issue. For this reason, REC (Chesani et al., 2010a; Bragaglia et al., 2012a) is used to enhance temporal reasoning through the horizontal arrangement of happened events.

REC is a *reactive* evolution of Kowalski and Sergot's EC (Kowalski & Sergot, 1989), who first proposed a novel approach to the reasoning about the evolution of the system: instead of thinking primarily in terms of system's state and actions as transitions between states, the authors proposed to think about the occurrences of actions – considering them as events – and the periods of time that they initiate and terminate (Sergot, 2002). In particular, the basic concepts of EC are those of *event*, anything that happens in the domain at a given point in time, and *fluent*, any measurable element of the environment that is subject to changes over time. Fluents are initiated or terminated by events and can be intended as the terms for describing the state of the environment i.e., the time intervals for which fluents hold describe the evolution of the system over time.

While EC is typically used to reason about a fixed history of events (thus, adding new events at runtime require the unpractical restart of the deductive reasoning from scratch), REC has been conceived to adapt EC to the dynamic context of runtime monitoring, where events happen (and are reported) at a

fast rate. It stems from the idea of caching the currently computed reasoning results for future use (Chittaro & Montanari, 1994). Every time a new event is delivered to the reasoner, the latter reacts by extending the narrative (i.e., the current set of happened events) and consequently revising the previously computed results in order to make them consistent with the newly arrived information.

To better understand the benefits brought by the combination of CEP and REC, consider the scenario of some temporal reasoning that must be carried out on the basis of the subject's permanence time in a certain room. The information regarding the room in which the subject is currently located can be deduced from the $(x, y)$ position of the wearable tag detected by the RID device placed in the wall lamp. Listing 2 exemplifies how CEP can be employed to convert the simple *PositionEvent* regarding the coordinates of the tag position (received in a certain instant of time) into a more meaningful *UserZoneEvent* signifying that the user is currently in a certain room or zone of the house.

Listing 2: First example of rule to generate more meaningful events from the active tag's position.

```
1 rule "UserZoneEvent generation from PositionEvent"
2 when
3   $pe : PositionEvent( $tag : tag , $x : x, $y: y,
4     $timestamp : timestamp) from entry−point "EventStream"
5 then
6   insert (new UserZoneEvent( DB.getUserByTag($tag)),
7     Map.getZoneFromCoord($x, $y), $timestamp );
8 end
```

Although the declarative nature of the logic condition simplifies the elicitation of the rule, with this approach, the reasoning on the permanence time in a particular room is not straightforward: a separation is needed between the concept of event – i.e., the $(x, y)$ position arriving at a particular instant of time – and environment/subject property – i.e., the condition of the subject presence in a particular room holding for a certain time as a consequence of the events' arrival. Therefore, the rule in Listing 2 must be employed besides other rules (see Listing 3) envisaging the horizontal arrangement of happened events as suggested by REC: when a first *UserZoneEvent* arrives (see the first rule in Listing 3), a novel *UserZoneFluent* is inserted in the knowledge base of facts reporting the timestamp of the event as to keep track of the instant in which the user was firstly detected in that zone of the house. Each

time a *UserZoneEvent* with a different zone arrives (second rule in Listing 3), the previous *UserZoneFluent* is modified – according to REC multi-value fluent theory – as regards both the zone and the entering timestamp.

Listing 3: REC theory applied to the reasoning on the user's permanence time in a particular room or zone of the house.

```
1  rule "UserZoneFluent first insertion"
2  when
3    $ure : UserZoneEvent( $user : user) from entry−point "EventStream"
4    not (UserZoneFluent( user == $user) )
5  then
6    insert (new UserZoneFluent( $user, $ure.getZone(),
7      $ure.getTimestamp() )); //since when the fluent is active
8  end
9
10 rule "The user moved to another zone: update UserZoneFluent."
11 when
12   $ure : UserZoneEvent( $user : user, $zone : zone,
13     $timestamp : timestamp ) from entry−point "EventStream"
14   $fl : UserZoneFluent(user==$user, zone!=$zone, $since : since, $timestamp>$since)
15 then
16   modify($fl) {setZone($zone), setTimestamp($timestamp)}
17 end
```

Fig. 3 highlights the vertical and horizontal arrangement of events according to our approach.

## 4. Relevant real-world scenarios

The initial project requirements analysis has led to the identification and the subsequent implementation of more than 40 different scenarios, in which the Habitat IoT system could be fruitfully exploited to simplify everyday life and improve security for elderly. In this section, we only focus on the most interesting scenarios from the point of view of our combined approach of CEP and REC.

### *4.1. Caregiver presence tracking*

As the Habitat project aims to support aging people with different requirements, it focuses on a scenario of a partially autonomous subject with small mental or physical decay, which might be living alone or occasionally assisted by a caregiver (e.g., a relative or a healthcare professional). The presence of the caregiver in the system is a crucial information for the DSS,

since different actions might be carried out in case of emergency whether the caregiver is just in the next room or outside the house. For this reason, a specific fluent is created to track the presence of the caregiver in the house according to the arrival of *UserZoneEvent*s. Listing 4 describes the rule for the creation/deletion of this presence fluent and Fig. 4 illustrates an example of fluent evolution in time.

Listing 4: Evolution of the fluent tacking the presence of a caregiver in the Habitat environment.

```
1  rule "Caregiver is now present"
2  when
3    $uze : UserZoneEvent( $user:user, $user.getRole()==CAREGIVER)
4    not CaregiverPresentFluent(caregiver == $user)
5  then
6    insert( new CaregiverPresentFluent($user, $uze.getTimestamp())
7  end
8
9  rule "Caregiver went away"
10 when
11   $uze1 : UserZoneEvent($user:user, $user.getRole()==CAREGIVER, $ts : timestamp)
12   $cpf : CaregiverPresentFluent(caregiver == $user, since < $ts)
13   not $uze2 : UserZoneEvent( $uze1!=$uze2, user==$user,
14     this after [0s, 15s] $uze1)
15 then
```

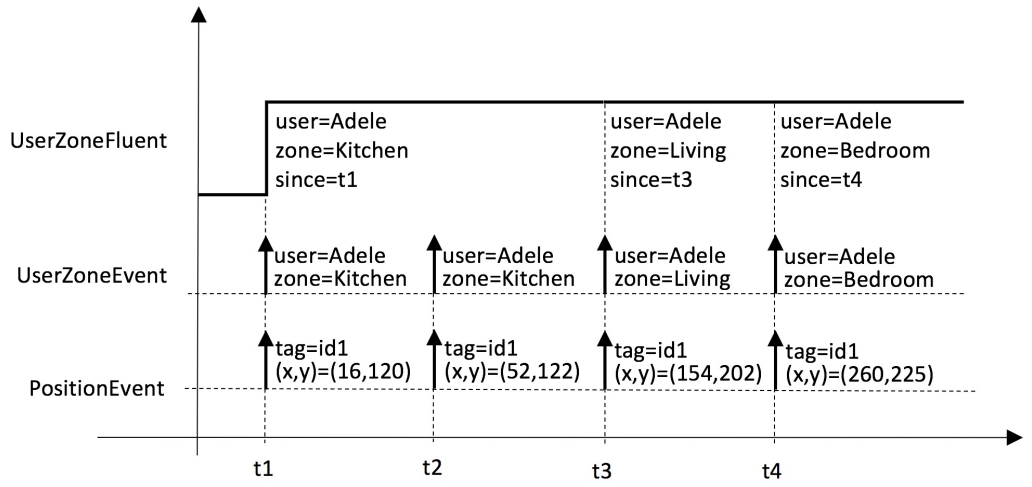


Figure 3: Example of events and fluents evolution to keep track of the user's permanence time in a particular zone of the house.

```
16   delete($cpf)
17 end
```

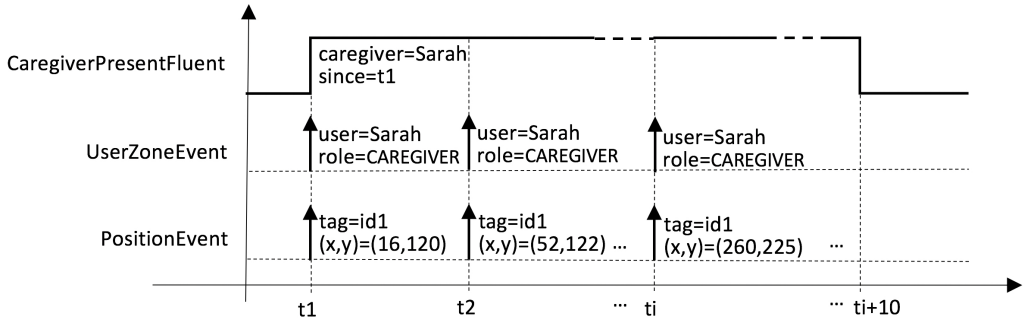


Figure 4: Example of fluent to track the presence of the caregiver in the Habitat system.

In particular, the second rule in Listing 4 employs a sliding window to express the fact that, if no *UserZoneEvent* related to the caregiver is received for a certain period (15 seconds in the presented example), we deduce that the caregiver's active tag must have been switched off as a consequence of the caregiver going away, thus the *CaregiverPresentFluent* must be deleted (line 16 in Listing 4).

### *4.2. Subject in relevant zone*

Consider the case of the monitored subject attempting to get out. As in the envisaged scenario the old person experiences only a slight cognitive decadence, she is not prohibited to do so alone but the caregiver, if present, must be notified of the situation. In case the caregiver is not in the house, it is instead important that the subject remembers to do some operations before going out e.g., take the apartment's keys, switch off all the lights, etc.

In order to generalize this scenario, we envisage that some rooms/zones of the house (e.g., the area around the main door) can be declared *relevant* for a certain subject (at configuration time), so that, if she is detected there, an advice is sent to the caregiver (if present) or subject herself. As show in the first rule of Listing 5, this logic is realized through the creation of a *UserRelZoneUpEvent* when the subject is detected in a zone that is relevant for her (line 5) for a configurable amount of time (line 6). Conditions in lines 8 and 9 reproduce a sort of histeresys cycle, thus to avoid the event is triggered again when a notification has been already created and within 20

seconds after the situation has been solved (see the following for a proper explanation of *UserRelZoneNotifFluent* and *UserRelZoneDownEvent*).

The *UserRelZoneUpEvent* triggers the creation of a *NotificationMessage*, which might be destined to the subject (line 20) or the caregiver (line 33) depending on the existence of a *CaregiverPresentFluent* in the knowledge base of facts. Either the caregiver is present or not, an appropriate *NotificationMessage* must be sent to all the interested interface devices through the SEPA engine and a *UserRelZoneNotifFluent* must be inserted in the Drools' knowledge base (lines 25 and 36) as to signify that a notification for the detected scenario has been sent and the situation is waiting to be managed (i.e., no answer has been received yet, nor the notification has been marked has ignored).

Listing 5: Detection of subject in a relevant zone of the house and notification sending.

```
1  rule "Detect the user in a relevant zone"
2  when
3    $ure : UserZoneEvent($user : user, $zone : zone, $ts : timestamp)
4    $urf : UserZoneFluent ($user==user,$user.getRole()==SUBJECT,
5      $zone==zone, $zone memberOf $user.getRelZones(),
6      $since : since, $ts - $since>$user.getTriggerTimeForRelZone($zone))
7    not UserRelZoneNotifFluent($user==user,$zone==zone,whenRaised<$ts,
8      ! this.isAck(), ! this.isIgn())
9    not UserRelZoneDownEvent($user==user, $zone==zone, this before [0s,20s] $ure)
10 then
11   insert(new UserRelZoneUpEvent( $user, $zone, $ts ) )
12 end
13
14 rule "Notification of user in relevant zone when caregiver absent"
15 when
16   $upEv: UserRelZoneUpEvent($user: user, $zone: zone, $ts: timestamp)
17   not CaregiverPresentFluent()
18 then
19   NotificationMessage $notification = new NotificationMessage(
20     $user,  // recipient
21     $user,  // refers to this user
22     $zone,  // regards this zone/room
23     $user.getMessageForRelZone($zone)); //e.g.,near exit door:''Remember your keys''
24   SEPA.produce($notification);
25   insert( new UserRelZoneNotifFluent($notification, $ts))
26 end
27
28 rule "Notification of user in relevant zone when caregiver present"
29 when
```

```
30    $upEv : UserRelZoneUpEvent($user : user, $zone : zone, $ts: timestamp)
31    $cpFl : CaregiverPresentFluent($caregiver : caregiver)
32  then
33    NotificationMessage $notification = new NotificationMessage($caregiver,
34      $user, $zone, $user.getName()+" is in "+$zone); //e.g.,"Adele is in exit zone"
35    SEPA.produce($notification);
36    insert( new UserRelZoneNotifFluent($notification, $ts) )
37  end
```

Note that the system was initially conceived for a domestic use, in which we envisage the presence of a single caregiver. The same approach could be also employed in a different environment, like for example, a day-care facility for elderly, where more than one caregiver is present because there are several old people to be assisted. In this case, the presented rules must be slightly modified to select the notification recipient according to specific policies (e.g., select the nearest caregiver or the most competent for the required operation).

When an answer is received for a *UserRelZoneNotifFluent* that was previously raised but not yet acknowledged nor set as ignored, the rule "UserRelZone Notification answer observed" (Listing 6) is executed. In this case, the *UserRelZoneNotifFluent* is set as acknowledged and saved in the database for logging purposes. The rule also triggers the creation of a *UserRelZoneDownEvent* (line 10), which is then used to remove the *UserRelZoneNotifFluent* from the knowledge base of facts (line 33 of Listing 6).

The rule "UserRelZone Notification ignored" triggers in case no answer is received within a time window of 30 seconds (line 18) for a certain notification. Analogously to the previous case, the notification is marked as ignored and saved in the database before a *UserRelZoneDownEvent* is generated. The occurrence of this latter event finally cause the deletion of the notification fluent (Last rule of Listing 6).

Listing 6: Handling of answer to a notification of subject in relevant zone.

```
1   rule "UserRelZone Notification answer observed"
2   when
3     $ans : AnswerMessage($notId : notId, $ts : timestamp)
4       from entry−point "EventStream"
5     $notFl : UserRelZoneNotifFluent($not : notification, $not.getId()==$notId,
6       whenRaised < $ts, ! this.isAck(), ! this.isIgn())
7   then
8     $notFl.setWhenAcknowledged($ts);
9     DB.save($notFl);
10    insert(new UserRelZoneDownEvent($not.getUser(),$not.getZone(),$ts));
```

```
11 end
12
13 rule "UserRelZone Notification ignored"
14 when
15   $upEv : UserRedZoneUpEvent($user : user, $zone : zone, $ts : timestamp)
16   $notFl : UserRelZoneNotifFluent($not : notification, ! this.isAck(), ! this.isIgn(),
17     $user == user, $zone == zone)
18   not $ans : AnswerMessage(notId == $not.getId(), this after [0s, 30s] $upEv)
19     from entry-point "EventStream"
20 then
21   Date now = System.currentTimeMillis();
22   $notFl.setWhenIgnored(now);
23   DB.save($notFl);
24   insert(new UserRelZoneDownEvent( $not.getUser(), $not.getZone(), now));
25 end
26
27 rule "UserRelZone Closing the notification fluent"
28 when
29   $ev: UserRelZoneDownEvent( $user : user, $zone : zone, $ts : timestamp)
30   $notFl : UserRelZoneNotifFluent ( this.notification.getUser() == $user,
31     this.notification.getZone() == $zone, whenRaised < $ts)
32 then
33   delete($notFl );
34 end
```

An example of evolution of the fluents involved in this scenario is presented in Figure 5.

### *4.3. Detection of static subject*

The aim of this scenario is to identify the situation in which the user results still in a certain position for too long. This condition indeed, might mask a fall and the need for help. The requirement can be more exhaustively expressed as follows: if the user is detected to be in a certain position for a considerable amount of time (user-dependent and configurable) and she does not result seated, then trigger a notification to the caregiver, if she is in the house. Alternatively, if the caregiver is not present, send the notification to the monitored subject herself, asking whether she is fine. A text message must be sent to a predefined relative's telephone number in case the user answers with an help request or does not answer to the notification within a certain (configurable) time.

The rules realizing this policy are reported in Listing 7.

Listing 7: Detection of subject too static in a certain position and notification sending.

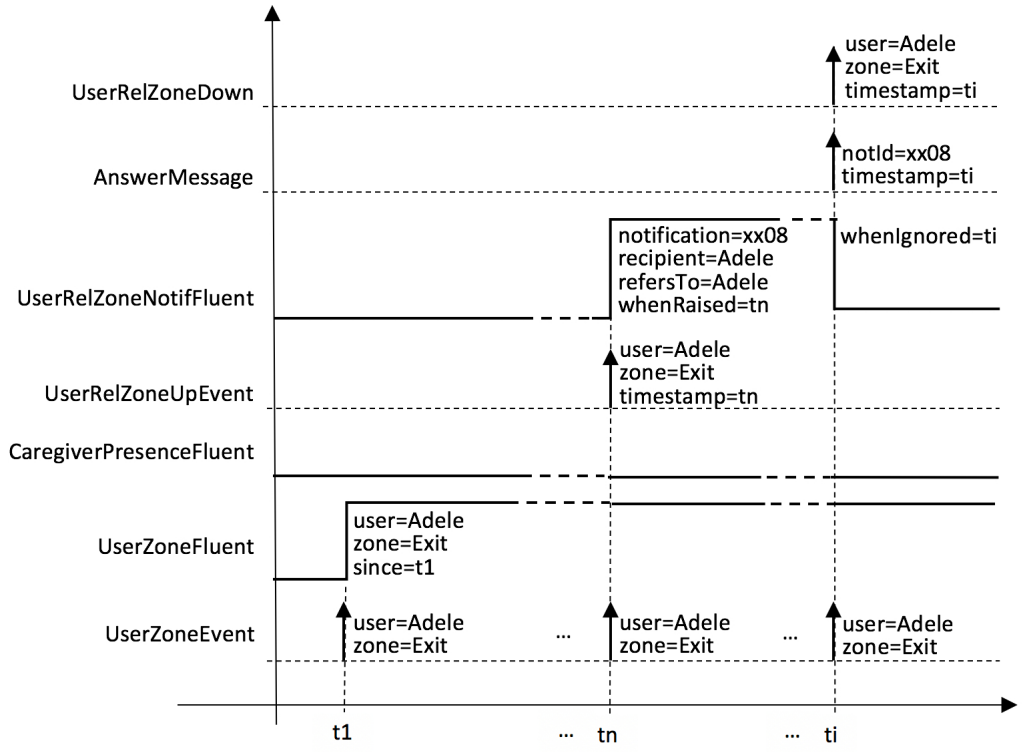


Figure 5: Example of fluent evolution for the scenario of a user detected in a relevant zone of the house. Here the caregiver is not present, so the notification is sent to the monitored subject Adele herself. After a certain time window without any answer from Adele, the NotificationMessage is marked as ignored at the time instant *ti*.

```
1  rule "Detect the user too static condition upon PositionEvents and UserPositionFluent"
2  when
3    $pe : PositionEvent ( $tag : tag, $x : x, $y : y, $zone : zone, $ts : timestamp )
4      from entry−point "EventStream"
5    $upf : UserPositionFluent ($user : user, $user==DB.getUserByTag($tag),
6      $user.getRole()==SUBJECT, $upf.isPositionInRange($x, $y),
7      $since : since, $ts − $since > $user.getTooStaticTriggerTime())
8    not UserSeatedFluent($user == user)
9    not $fl : UserTooStaticNotifFluent($user==user, whenRaised<$ts,
10     !this.Ack(), !this.Ign())
11   not $evDown : UserTooStaticDownEvent ($user == user, this before [0s,30s] $pe)
12 then
13   insert(new UserTooStaticUpEvent( $user, $zone, $ts));
14 end
```

```
15
16 rule "UserTooStatic Notification when caregiver absent"
17 when
18   $ev: UserTooStaticUpEvent ( $user : user, $zone : zone, $ts : timestamp)
19   not CaregiverPresentFluent()
20 then
21   NotificationMessage $notification = new NotificationMessage($user, $user, $zone,
22     $user.getTooStaticMsg()); //e.g.,"You are still for long time. Is everything fine?"
23   SEPA.produce($notification);
24   insert ( new UserTooStaticNotifFluent($notification, $ts) );
25 end
26
27 rule "UserTooStatic Notification when caregiver present"
28 when
29   $ev: UserTooStaticUpEvent ( $user : user, $zone : zone, $ts : timestamp)
30   $cg CaregiverPresentFluent($caregiver : caregiver)
31 then
32   NotificationMessage $notification=new NotificationMessage($caregiver, $user, $zone,
33       $user.getName()+" too static in "+$zone);//e.g.,"Adele is too static in bathroom"
34   SEPA.produce($notification);
35   insert ( new UserTooStaticNotifFluent($notification, $ts) );
36 end
```

The first rule employs two further fluents generated upon *PositionEvent*, namely *UserPositionFluent* and *UserSeatedFluent*. We omit the rules generating these fluents for brevity. The aim of *UserSeatedFluent* is to keep track of the fact that the user is seated on the smart chair (and consequently her static behavior should not trigger alarms), while the aim of *UserPositionFluent* is to maintain and update a centroid of the user's positions. Indeed, as the coordinates emitted by the RID-tag system are inevitably affected by an error, the exact value of the $x$ and $y$ sent to the system might slightly vary in time although the user is still. In order to ascertain that the subject is static while coping with uncertainity about the precise user position, we must consider the case in which the centroid remains the same for a certain amount of time. The centroid's coordinates in the *UserPositionFluent* are therefore initialized with the $(x, y)$ and timestamp of the first *PositionEvent*, and then updated only when a $(x, y)$ is detected outside the error range around the centroid.

In order to identify the stillness condition of the user, the first rule in Listing 7 checks if the coordinates in the *PositionEvent* are inside the centroid's range (line 6) for a period that exceeds the configured time (line 7) and the user is not seated (line 8). In that case, according to the prescriptions of REC

theory, a *UserTooStaticUpEvent* is generated, which is then used to create a notification and a *UserTooStaticNotifFluent*. Analogously to the scenario of the user in a relevant zone, different recipients and messages are set inside the *NotificationMessage* object whether the caregiver is present in the house (line 32) or not (line 21). The answer to this notification (or the lack of an answer within a certain amount of time) is managed similarly to the case of the user in a relevant zone and it is therefore omitted. Fig. 6 shows the evolution of fluents as prescribed by Listing 7.

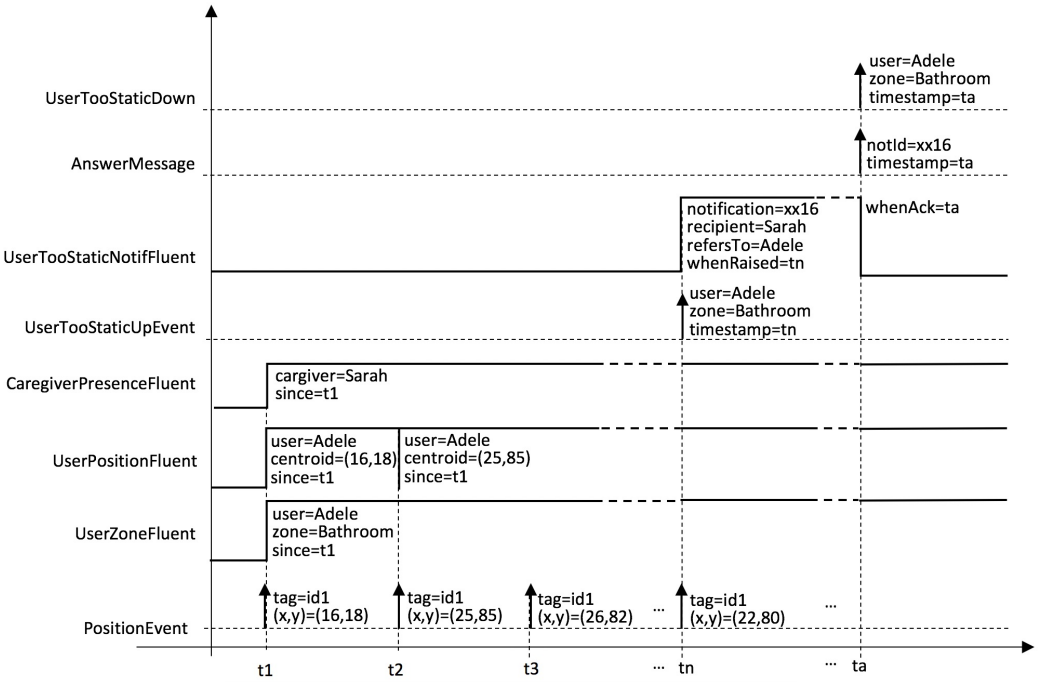


Figure 6: Example of fluent evolution for the scenario of a user that is detected too static. Here the caregiver is present and therefore chosen as notification recipient. She answers to the *NotificationMessage* at time *ta*, thus causing it to be marked as acknowledged.

### *4.4. Detection of subject standing up from chair*

When the elderly subject is experiencing a physical decay, some – otherwise simple – operations, like standing up from a chair, represent a dangerous behavior as they might pull her off balance and cause a fall. Thanks to a set of load cells under its seat and feet, the smart chair is able to recognize not only if a user is correctly or incorrectly seated, but also if she is trying to

stand up. The Habitat DSS is responsible for collecting the messages from the chair and understand if the subject going to stand up is allowed to do so alone or should be assisted instead. To this end, the first rule in Listing 8 is activated upon the arrival of a *ChairMessage*. It checks the profile of the involved subject through the *UserSeatedFluent* and determines her permission to stand up autonomously. If not allowed (line 6), a *UserStandingUpUpEvent* is generated, which is later used in the second rule of Listing 8 to produce a notification for the caregiver. After discussing the logic of this scenario with ASC Insieme, we assume that a user needing help when standing up is always assisted. Thus, no rule is provided in case of caregiver absence. Fig. 7 highlights the evolution of fluents for this scenario.

Listing 8: Detection of subject standing up from the smart chair.

```
1  rule "Detect the user standing up upon ChairMessages and UserSeatedFluent"
2  when
3    $chMsg : ChairMessage($idChair : idChair, type==STANDING_UP,
4      $ts : timestamp) from entry-point "EventStream"
5    $usf : UserSeatedFluent( $user : user, idChair == $idChair, $zone : zone
6      since < $ts, ! $user.allowedStandUpAlone() )
7    not $fl : UserStandingUpNotifFl($user==user, idChair == $idChair,
8      whenRaised<$ts, !this.Ack(), !this.Ign())
9    not $evDown:UserStandingUpDownEvent($user==user, this before[0s,30s]$chMsg)
10 then
11   insert(new UserStandingUpUpEvent( $user, $zone, $ts));
12 end
13
14 rule "UserStandingUp Notification, only when caregiver present"
15 when
16   $ev : UserStandingUpUpEvent ( $user : user, $zone : zone, $ts : timestamp)
17   $cg : CaregiverPresentFluent($caregiver : caregiver)
18 then
19   NotificationMessage $notification=new NotificationMessage($caregiver, $user, $zone,
20       $user.getName()+" is trying to stand up in "+$zone);
21       //e.g.,''Adele is trying to stand up in living room''
22   SEPA.produce($notification);
23   insert ( new UserStandingUpNotifFl($notification, $ts) );
24 end
```

In an analogous way, the Habitat DSS, notifies the subject in case of incorrect posture, or – employing a time window as in the scenario of a too static subject – in case she is detected to spend too much time on the chair. In the latter eventuality, aiming to improve the lifestyle of the monitored

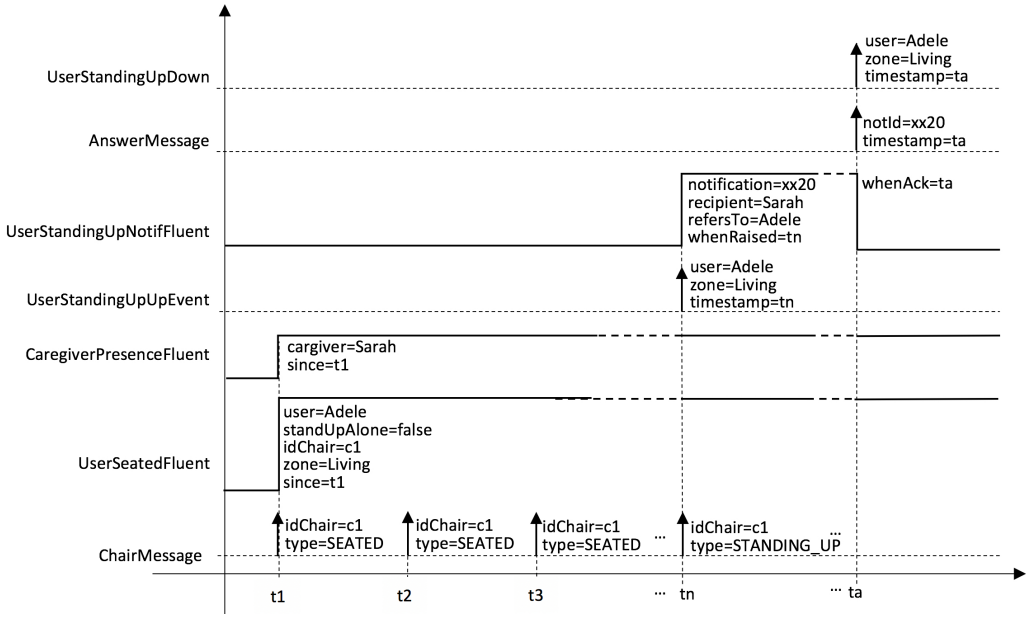


Figure 7: Example of fluent evolution for the scenario of a user standing up from the smart chair. In this case, since the user is not allowed to do so without assistance, a notification is sent to the caregiver. At time *ta*, she answers to the *NotificationMessage*, thus causing it to be marked as acknowledged.

subject, after a configurable amount of time, a message suggesting to have a walk is sent to the elderly.

## 5. Evaluation

The Habitat project has undergone two kind of evaluations:

- a qualitative test carried out in a day-care facility for elderly, in order to understand the degree of usability and attractiveness of the smart objects for elderly subjects, and;
- a quantitative test conducted in simulated environments, to determine the individual technical capabilities of the Habitat main components and point out which of them could represent a bottleneck in large use cases.

The qualitative test has been carried out by an external reviewer during May 2018 at the day-care facility for elderly *Borgo del Sasso* in Sasso Mar-

coni (Bologna, Italy), involving 19 participants (7 self-sufficient old persons, 4 caregivers together with 4 non-autonomous patients, 2 caregivers alone, and 2 elderly needing care alone), who have been given first experience of the various objects and smartphone applications composing the Habitat system. In particular, the room has been equipped with a smart-lamp, a wall panel, and a smart chair, as well as standard forniture so as to resemble a normal living room. During the interview, the participants were invited to wear the inertial belt and the smart brooch, and to familiarize with the Habitat application installed on a smartphone, as well as with the messages runtime delivered to the latter (e.g., notification of incorrect posture when the participant is seated on the chair, too static position, proximity to the doorstep as a relevant zone, etc.). The attractiveness of an alternative localization tag inserted into a smart pendant was also evaluated.

The main goal of the installation was a qualitative analysis – conducted according to the usability heuristics of (Nielsen, 1994) – of the experience of elderly and caregivers when they first come into contact with the interfaces of the smart objects (Mincolelli et al., 2019).

Nonetheless, the experiments were also the occasion to evaluate the Habitat project in a real environment from a technical and operational point of view. As expected, the installation of the testing scenario has highlighted a crucial role of the wi-fi signal and keepalive of smartphone services in the detection of the relevant situations. Moreover, the analysis have underlined that the wiring and recharging needs of the smart devices could represent a concern for the future industrial development and commercialization of the Habitat system. In view of this, the adoption of less power consuming wireless communication standards – such as Zigbee (Zigbee, 2018) –, will be considered.

In order to give a quantitative evaluation of the Habitat system performance, the indoor localization system, the SEPA routing system, and the DSS have been separately considered. The final aim of these tests is to analyze the behavior of the components in large scenarios, where several users need to be monitored. As the data collected during the usability tests were not sufficient to stress the system, we resort to simulation for the quantitative evaluation.

As regards the localization system, several measurements have taken place in a typical indoor scenario of about 31 square meters. Nine different positions in the room have been analyzed and reported, with the tag placed in the two different locations of the user's body: on the chest as a pendant or on

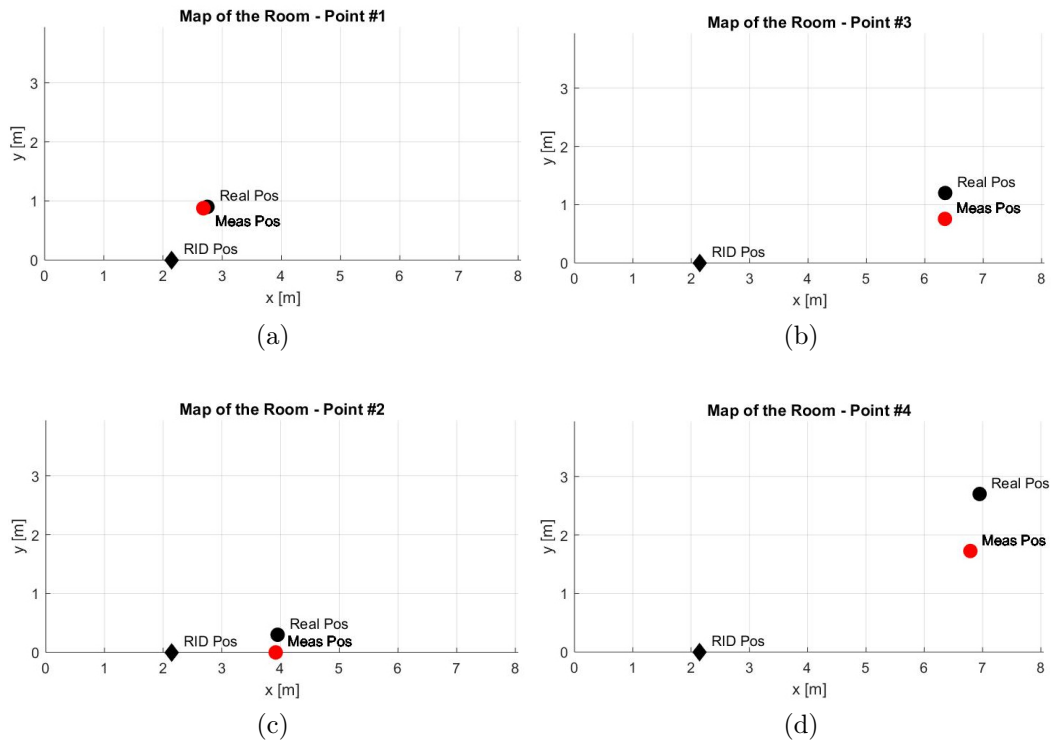


Figure 8: Representation of four real and measured positions for the points of minimum (left, point #1 in 8a: 1.08 m, point #2 in 8c: 1.82 m) and maximum (right, point #3 in 8b: 4.37 m, point #4 in 8d: 5.51 m) distance from the reader, with the active tag placed on the shoulder of the user. The average of ten consecutive measurements has been considered for the estimated points.

the shoulder as a brooch. A minimum error of 0.06 m and maximum of 0.56 m have been noticed in correspondence of the points of minimum (1.08 m) and maximum (5.51 m) RID-tag distance, respectively, with the tag placed on the shoulder of the user. In Fig. 8, real and retrieved positions of the tagged users are reported, taking into account the average of ten consecutive bi-dimensional localization measurements.

The Habitat's message routing and DSS have been tested through software simulated environments, which allows to reproduce particularly stressing conditions. Both the evaluations refers to the execution of the backbone infrastructure on an average hardware architecture: a quad-core 2,9 GHz Intel Core i7 CPU with hyperthreading, 16GB RAM and 256GB SSD.

The performance of the semantic message routing system SEPA has been evaluated by simulating the arrival of an increasing number of user position messages. For each message, the time between its insertion into the engine and its notification to the specific consumer has been recorded. Fig. 9 reports the evolution of the aforementioned times in the worst (dotted line), best (dashed line) and average (continuous line) cases, when the number of tracked subjects increases. The graph underlines that the SEPA engine is able to deliver the position messages with an almost constant average delay lower than 100ms. Nonetheless, a future implementation in large scenarios (e.g., a day-care facility for elderly) should take into account that, if the system is required to track more than 200 subjects, the delivery of a restricted number of position messages (around 6‰) might be delayed of more than 2 seconds (as underlined by the MaxTime line in Fig. 9).

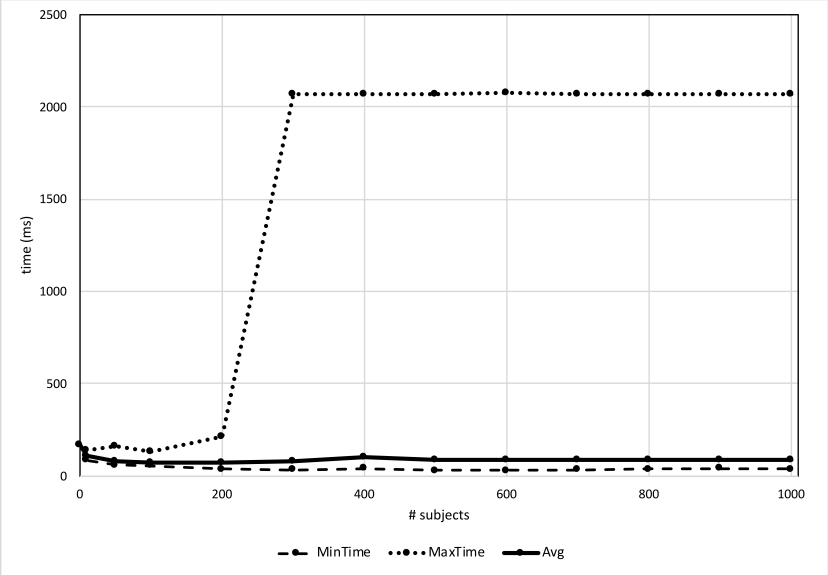


Figure 9: Evaluation of the SEPA delivery times when simulating an increasing number of observed subjects.

As regards the Habitat DSS, the performance evaluation has been carried out considering the scenario of a user in relevant zone, since it employs one of the most complicated chaining of Drools rules in order to decide if the notification needs to be sent (and which should be its content and recipient). Furthermore, different scenarios would call for different performance e.g., the notification of a user spending too much time on the chair can be delayed by

minutes, whereas the message reporting that the elderly is going out should be delivered within some seconds. For this reason, the testing scenario has been built with all the users requiring to be suddenly reminded of a specific sequence of actions to be executed before going out of the apartment. The notification is triggered when the subject is detected in the exit zone for a time window that has been specifically configured to be null in this test. Thus, to simulate the most difficult scenario, where the DSS has to react within seconds to the changing environment.

The performance of the DSS are tested by recording the time between the arrival of a position message from the SEPA engine (reporting a location inside the exit zone range) and the insertion (back into the SEPA engine) of the correspondent notification triggered by the DSS rules. Analogously to what presented for the semantic routing system, Fig. 10 reports these times in the worst (dotted line), best (dashed line) and average (continuous line) cases, while we increase the number of tracked subjects.

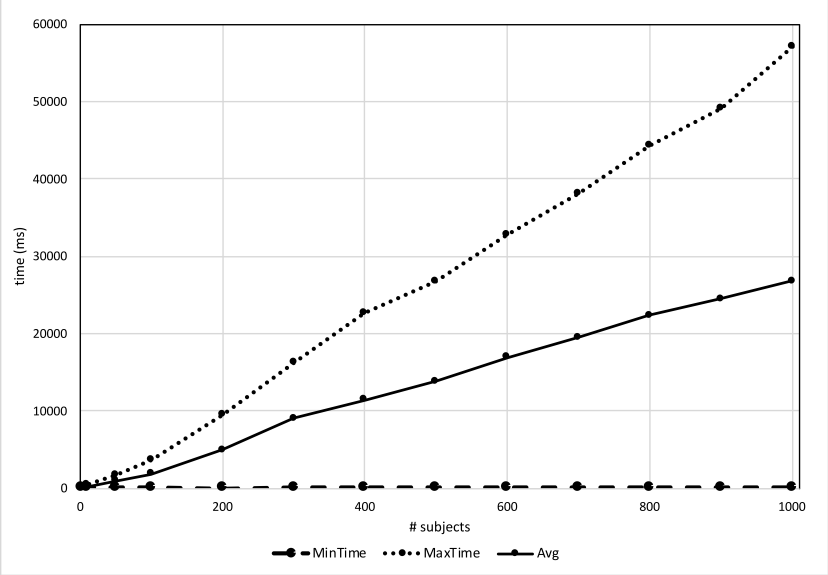


Figure 10: Evaluation of the time employed by Habitat DSS to trigger a notification in case of subject detected near the exit zone, when simulating an increasing number of observed subjects.

Realizing the main business logic of the application, the DSS is the most complicated and consequently slowest component of the Habitat system. The graph in Fig. 10 in particular, highlights that for example, if an average

time of 5s should elapse from the detection of a user in a relevant zone to the triggering of a notification, then no more than 200 subjects should be managed or, at least, no more than 200 subjects should be triggering such notification in the same instant of time. Clearly, the maximum number of subjects that can be simultaneously monitored depends on the maximum delay accepted in the specific context.

## 6. Related work

The Habitat project aims to create a flexible support system for elderly and disabled people combining the most recent advances in the field of IoT: Semantic Web technologies for message routing and a combination of CEP and REC theories for temporal reasoning.

The Semantic Web (Berners-Lee et al., 2001), intended as a global Web of Data (Bizer et al., 2009) which can be interpreted and directly processed by digital machines, is becoming a Web of Dynamic Data, where detecting, communicating and tracking the evolution of data changes play a crucial role and open new research questions (Umbrich et al., 2010; Sanderson & Van de Sompel, 2012). Works focused on enabling interoperability in the IoT through the use of Semantic Web technologies, like the ones presented in (Llanes et al., 2016; Schade et al., 2012; Boulos et al., 2015; Alti et al., 2016), emphasize the need for solutions on detecting and communicating data changes over the Web of Data.

The SPARQL Event Processing Architecture (SEPA) (Roffia et al., 2018) adopted by Habitat aims at providing such a solution by means of a content-based publish-subscribe mechanism where the W3C SPARQL 1.1 Update (SPARQL Update, 2013) and Query (SPARQL Query, 2013) languages are fully supported respectively by publishers and subscribers. SEPA can be framed within the research topics known as stream reasoning (Della Valle et al., 2009), linked stream data processing (Le-Phuoc et al., 2012) and content-based publish-subscribe (Eugster et al., 2003). Initial solutions for Semantic Web-based publish-subscribe are presented by Wang et al. (Wang et al., 2004) and Chirita et al. (Chirita et al., 2004), while a first attempt to use SPARQL as subscription language is presented in (Skovronski, 2006). Other early RDF stream processing approaches propose to extend SPARQL with time-windows like in streaming SPARQL (Bolles et al., 2008), C-SPARQL (Barbieri et al., 2010), SPARQLStream (Calbimonte et al., 2010), Event Processing SPARQL (EP-SPARQL) (Anicic et al., 2011), Continu-

ous Query Evaluation over Linked data Streams (CQELS) (Le-Phuoc et al., 2011) and Sparkwave (Komazec et al., 2012). Other research works focusing on the use of SPARQL as subscription language include the one by Groppe et al. (Groppe et al., 2007), EventCloud (Pellegrino et al., 2012, 2013), INSTANS (Abdullah et al., 2012; Rinne et al., 2012), semantic event notification service (SENS) (Murth, 2008; Murth & Kühn, 2009b,a, 2010) and Smart-M3 (Roffia et al., 2016; Honkola et al., 2010; Suomalainen et al., 2010; Galov et al., 2015; Viola et al., 2016; D'Elia et al., 2017; Morandi et al., 2012).

Research efforts on developing Linked Data, and in particular detecting Linked Data changes, would have an impact on the development of distributed, context-aware and interoperable applications/sevices as much as Web standards will be adopted and promoted. The main Web players are pushing in this direction by promoting standards – e.g., the W3C recommendation on Linked Data Notifications (Linked Data, 2017) – and defining ontologies and vocabularies – e.g., Schema.org founded by Google, Microsoft, Yahoo and Yandex. The SPARQL 1.1 Secure Event Protocol (SPARQL SE Protocol, 2018) and the SPARQL 1.1 Subscribe Language (SPARQL Subscribe, 2018) proposed by SEPA go in that direction.

As regards temporal reasoning on the messages coming from the Habitat sensors, CEP (Luckham, 2008) has been employed. As also pointed out by Mani Chandy et al. (Chandy & Schulte, 2010), thanks to its expressive power and notation clarity, this approach born to process high-frequency streams of events, has already been applied in various application domains, like Business Process Management (BPM), fraud detection, analysis of data from sensor networks, or algorithmic stock-tradings. In the field of IoT in general, where several messages coming from different sensors are supposed to be routed to a monitoring system and analyzed to extract a complex information about the overall environment, CEP is the perfect candidate to support event stream processing (Bruns et al., 2015; Qin et al., 2016; Akbar et al., 2017; Flouris et al., 2017; Terroso-Sáenz et al., 2015; Sheriff et al., 2015; Machado et al., 2017; Wickramasinghe et al., 2017).

The work by Sheriff et al. (Sheriff et al., 2015) in particular, presents a framework that integrates Big Data analytics techniques, IoT and CEP to realize a healthcare-specific analytics software. More similarly to our approach, those by Machado et al. (Machado et al., 2017) and Wickramasinghe et al. (Wickramasinghe et al., 2017) employ CEP to detect critical situations occurred in ambient assisted living environments on the basis of row data from

sensors.

Some other works deal with the analysis of events from smart objects in a generic domestic environment, overlooking CEP but particularly focusing on the aspects of process mining (Dimaggio et al., 2016; Leotta et al., 2015; Sztyler et al., 2015; Tax et al., 2018a,b).

In order to simplify the temporal reasoning on incoming events, our approach applies the REC theory (Chesani et al., 2010a; Bragaglia et al., 2012a) alongside the stream processing technology of CEP. Being firstly conceived to apply the EC paradigm to the more dynamic field of event flow runtime monitoring, REC has been employed in a variety of application domains e.g, BPM (Montali, 2010; Chesani et al., 2016), clinical guidelines and care-flow protocols (Bottrighi et al., 2011; Bragaglia et al., 2014), service oriented architectures (Chesani et al., 2008) and multi-agent systems (Chesani et al., 2010b).

In particular, the works (Bragaglia et al., 2012b; Bragaglia, 2013) are closely related to our research since they represent an attempt to enhance Drools framework by giving support to REC theory, thus testifying the increasing interest in the combination of CEP and REC for event monitoring purposes.

For the future, the work at hand can be enriched by profiling the subject's habits through machine learning techniques. Indeed, as also highlighted in the surveys by Rashidi et al. (Rashidi & Mihailidis, 2013) and Nabian et al. (Nabian, 2017), several works in this field have already shown good results in the identification of user's low-/high-level activities from the analysis of various types of sensor-originated information (Anguita et al., 2012, 2013; Fleury et al., 2013, 2010). Taking inspiration from these works, Habitat can be extended to further promptly identify behaviors that can cause health problems to the monitored subject in the long run.

## 7. Conclusion and future research directions

With the view to lengthen the autonomy of elderly and support their dignity, it is crucial to foster their stay at home and delay the resort to nursing homes as much as possible. The Habitat project aims to combine the recent advances in the field of environmental and wearable sensors, IoT technologies and event processing, to build an assistive system for domestic residences to support aging people.

In this work, we presented the backbone infrastructure of such system, particularly focusing on the adopted semantic message routing technology, called SEPA engine, and our approach to event analysis, which combines CEP and a reactive implementation of EC.

On one hand, the SEPA engine enables information level interoperability and give support to the evolvability and dynamicity of the underlying IoT sensor network. On the other hand, the event analysis approach, which intertwines CEP and REC, ensures an expressive mechanism for the formulation of the temporal properties to be monitored and corresponding reactions to be carried out. The present work shows these advantages through an accurate description of a subset of real-world scenarios, where decoupling the vertical dimension of semantic pregnancy from the horizontal one of temporal reasoning mostly highlights its benefits in terms of expressiveness and clarity.

The Habitat platform performance has been evaluated by separately analyzing the indoor localization system, the SEPA engine, and the DSS. Further tests have been carried out on the Habitat system as a whole by an external reviewer aiming to asses the degree of usability and acceptability of each smart object.

For the future, in order to improve reliability and decrease power consumption of the overall system, we plan to integrate energy efficient technologies for wireless communications. We also envisage an improvement of the features of the chair, through the introduction of an electomechanical system to lift the seat when an allowed autonomous attempt to stand up is detected. The adoption of CEP and REC make the extension of the DSS in this sense a rather straightforward task.

Finally, all the data collected during the usability evaluation in a real environment could represent the starting point for a future improvement of the Habitat project adaptability. Indeed, the system is already able to provide a report summarizing the main features of the elderly's lifestyle – in terms of e.g., the portion of time spent on the smart chair or in more dynamic activities, time spent in each room or outside the apartment, distance covered by feed, etc. This report has been conceived to allow the family or caregiver to have a quick glance at the subject's lifestyle and, eventually, modify the system configurations with custom parameters. We plan to exploit the data collected during the usability tests as a first training set to automate the profiling of user habits through machine learning techniques, and consequently enhance the quality of possible suggestions for a healthy lifestyle. For example, consider a user spending too much time on the smart chair e.g.,

four hours when the best for her health is two. If the system is be able to understand the usual amount of time spent on the chair and progressively reduce (e.g., 10% each day) the timeout after which (the corresponding scenario is triggered and) the user is delivered a message suggesting to have a walk, the acceptability of the suggestion could increase w.r.t. a system always notifying the message after two hours.

The application of machine learning techniques could also be useful for anomaly detection, whereas more sophisticated analysis could be conducted on data coming from several days of observation. For example, as Alzheimer's disease and dementia are frequently preceded by gait disorders, balance problems, abnormal motor behaviors, and degeneration of the sleep-waking cycle (Nams et al., 2010; Hoffmeyer et al., 2012; Pettersson et al., 2005, 2002), the application of machine learning techniques in the framework of the Habitat project could pave the way to enhancements in early detection of these diseases.

## 8. Acknowledgements

This research was partly funded by the EU-supported Regional Operative Project HABITAT - Home Assistance Based on Internet of Things for the AuTonomy (`http://www.eng.habitatproject.info`).

## 9. References

Abdullah, H., Rinne, M., Törmä, S., & Nuutila, E. (2012). Efficient matching of sparql subscriptions using rete. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing* SAC '12 (pp. 372–377). New York, NY, USA: ACM. `https://doi.org/10.1145/2245276.2245348`.

Akbar, A., Khan, A., Carrez, F., & Moessner, K. (2017). Predictive Analytics for Complex IoT Data Streams. *IEEE Internet of Things Journal*, *4*, 1571–1582. `https://doi.org/10.1109/JIOT.2017.2712672`.

Alti, A., Lakehal, A., Laborie, S., & Roose, P. (2016). Autonomic semantic-based context-aware platform for mobile applications in pervasive environments. *Future Internet*, *8*, 1–26. `https://doi.org/10.3390/fi8040048`.

Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2012). Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In J. Bravo, R. Hervás, & M. Rodríguez

(Eds.), *Ambient Assisted Living and Home Care - 4th International Workshop, IWAAL 2012, Vitoria-Gasteiz, Spain, December 3-5, 2012. Proceedings* (pp. 216–223). Springer volume 7657 of *Lecture Notes in Computer Science*. https://doi.org/10.1007/978-3-642-35395-6_30.

Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2013). A public domain dataset for human activity recognition using smartphones. In *21st European Symposium on Artificial Neural Networks, ESANN 2013, Bruges, Belgium, April 24-26, 2013*. http://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2013-84.pdf.

Anicic, D., Fodor, P., Rudolph, S., & Stojanovic, N. (2011). EP-SPARQL: A Unified Language for Event Processing and Stream Reasoning. In *Proceedings of the 20th International Conference on World Wide Web* WWW '11 (pp. 635–644). New York, NY, USA: ACM. https://doi.org/10.1145/1963405.1963495.

Antoniazzi, F., Paolini, G., Roffia, L., Masotti, D., Costanzo, A., & Cinotti, T. S. (2017). A web of things approach for indoor position monitoring of elderly and impaired people. In *2017 21st Conference of Open Innovations Association (FRUCT)* (pp. 51–56). https://doi.org/10.23919/FRUCT.2017.8250164.

ASC (2018). ASC Insieme. http://www.ascinsieme.it.

Barbieri, D. F., Braga, D., Ceri, S., & Grossniklaus, M. (2010). An execution environment for c-sparql queries. In *Proceedings of the 13th International Conference on Extending Database Technology* EDBT '10 (pp. 441–452). New York, NY, USA: ACM. http://doi.acm.org/10.1145/1739041.1739095.

Bartolini, S., Milosevic, B., D'Elia, A., Farella, E., Benini, L., & Cinotti, T. S. (2012). Reconfigurable natural interaction in smart environments: approach and prototype implementation. *Personal and Ubiquitous Computing*, *16*, 943–956. https://doi.org/10.1007/s00779-011-0454-5.

Beard, J. R., Officer, A. M., & Cassels, A. K. (2016). The world report on ageing and health. *The Gerontologist*, *56*, S163–S166. http://doi.org/10.1093/geront/gnw037.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American*, *284*, 34–43. http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, *5*, 1–22. https://doi.org/10.4018/jswis.2009081901.

Bolles, A., Grawunder, M., & Jacobi, J. (2008). Streaming SPARQL - Extending SPARQL to Process Data Streams. In S. Bechhofer, M. Hauswirth, J. Hoffmann, & M. Koubarakis (Eds.), *ESWC2008 - The Semantic Web: Research and Applications* (pp. 448–462). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-68234-9_34.

Bottrighi, A., Chesani, F., Mello, P., Montali, M., Montani, S., & Terenziani, P. (2011). Conformance checking of executed clinical guidelines in presence of basic medical knowledge. In F. Daniel, K. Barkaoui, & S. Dustdar (Eds.), *Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part II* (pp. 200–211). Springer volume 100 of *Lecture Notes in Business Information Processing*. https://doi.org/10.1007/978-3-642-28115-0_20.

Boulos, M. N., Yassine, A., Shirmohammadi, S., Namahoot, C. S., & Brückner, M. (2015). Towards an "internet of food": Food ontologies for the internet of things. *Future Internet*, *7*, 372–392. https://doi.org/10.3390/fi7040372.

Bragaglia, S. (2013). *Monitoring Complex Processes to Verify System Conformance: A Declarative Rule-Based Framework*. Ph.D. thesis alma. http://amsdottorato.unibo.it/5753/.

Bragaglia, S., Chesani, F., Mello, P., Montali, M., & Torroni, P. (2012a). Reactive event calculus for monitoring global computing applications. In A. Artikis, R. Craven, N. Kesim Çiçekli, B. Sadighi, & K. Stathis (Eds.), *Logic Programs, Norms and Action: Essays in Honor of Marek J. Sergot on the Occasion of His 60th Birthday* (pp. 123–146). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-29414-3_8.

Bragaglia, S., Chesani, F., Mello, P., & Sottara, D. (2012b). A rule-based calculus and processing of complex events. In A. Bikakis, & A. Giurca (Eds.), *Rules on the Web: Research and Applications - 6th International Symposium, RuleML 2012, Montpellier, France, August 27-29, 2012. Proceedings* (pp. 151–166). Springer volume 7438 of *Lecture Notes in Computer Science*. https://doi.org/10.1007/978-3-642-32689-9_12.

Bragaglia, S., Monte, S. D., & Mello, P. (2014). A distributed system using ms kinect and event calculus for adaptive physiotherapist rehabilitation. In *2014 Eighth International Conference on Complex, Intelligent and Software Intensive Systems* (pp. 531–538). https://doi.org/10.1109/CISIS.2014.77.

Bruns, R., Dunkel, J., Masbruch, H., & Stipkovic, S. (2015). Intelligent M2M: Complex event processing for machine-to-machine communication. *Expert Systems with Applications*, *42*, 1235 – 1246. https://doi.org/10.1016/j.eswa.2014.09.005.

Calbimonte, J.-P., Corcho, O., & Gray, A. J. G. (2010). Enabling ontology-based access to streaming data sources. In P. F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Z. Pan, I. Horrocks, & B. Glimm (Eds.), *The Semantic Web – ISWC 2010* (pp. 96–111). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-17746-0_7.

Centers for Disease Control and Prevention (2013). *The State of Aging and Health in America 2013*. Atlanta, GA: Centers for Disease Control and Prevention, US Dept of Health and Human Services. https://www.cdc.gov/aging/pdf/State-Aging-Health-in-America-2013.pdf.

Chandy, K. M., & Schulte, W. R. (2010). *Event Processing - Designing IT Systems for Agile Companies*. New York, NY, USA: McGraw-Hill, Inc. http://www.mhprofessional.com/product.php?isbn=0071633502.

Chesani, F., Ciampolini, A., Loreti, D., & Mello, P. (2016). Process mining monitoring for map reduce applications in the cloud. In J. S. Cardoso, D. Ferguson, V. M. Muñoz, & M. Helfert (Eds.), *CLOSER 2016 - Proceedings of the 6th International Conference on Cloud Computing and Services Science, Volume 1, Rome, Italy, April 23-25, 2016.* (pp. 95–105). SciTePress. https://doi.org/10.5220/0005864000950105.

Chesani, F., Mello, P., Montali, M., & Torroni, P. (2008). Verification of choreographies during execution using the reactive event calculus. In R. Bruni, & K. Wolf (Eds.), *Web Services and Formal Methods, 5th International Workshop, WS-FM 2008, Milan, Italy, September 4-5, 2008, Revised Selected Papers* (pp. 55–72). Springer volume 5387 of *Lecture Notes in Computer Science*. https://doi.org/10.1007/978-3-642-01364-5_4.

Chesani, F., Mello, P., Montali, M., & Torroni, P. (2010a). A logic-based, reactive calculus of events. *Fundam. Inform.*, *105*, 135–161. https://doi.org/10.3233/FI-2010-361.

Chesani, F., Mello, P., Montali, M., & Torroni, P. (2010b). Monitoring time-aware social commitments with reactive event calculus. In *Proceedings of the 20th European Meeting on Cybernetics and Systems Research, 7th International Symposium 'From Agent Theory to Agent Implementation' (AT2AI-7)* (pp. 447–452). https://pdfs.semanticscholar.org/8821/efa8d614288a18fc065383fe72458ab783e3.pdf.

Chirita, P. A., Idreos, S., Koubarakis, M., & Nejdl, W. (2004). Publish/-Subscribe for RDF-based P2P Networks. In *Lecture Notes in Computer Science* (pp. 1–15). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-25956-5_13.

Chittaro, L., & Montanari, A. (1994). Efficient handling of context dependency in the cached event calculus. In *Proc. of TIME'94 - International Workshop on Temporal Representation and Reasoning* (pp. 103–112). http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.1668.

Del Prete, M., Masotti, D., Arbizzani, N., & Costanzo, A. (2013). Remotely identify and detect by a compact reader with mono-pulse scanning capabilities. *IEEE Transactions on Microwave Theory and Techniques*, *61*, 641–650. https://doi.org/10.1109/TMTT.2012.2229290.

D'Elia, A., Viola, F., Roffia, L., Azzoni, P., & Cinotti, T. S. (2017). Enabling interoperability in the internet of things: A OSGi semantic information broker implementation. *International Journal on Semantic Web and Information Systems*, *13*, 146–167. https://doi.org/10.4018/IJSWIS.2017010108.

Della Valle, E., Ceri, S., Harmelen, F. V., & Fensel, D. (2009). It's a Streaming World! Reasoning upon Rapidly Changing Information. *IEEE Intelligent Systems*, *24*, 83–89. https://doi.org/10.1109/MIS.2009.125.

Dimaggio, M., Leotta, F., Mecella, M., & Sora, D. (2016). Process-based habit mining: Experiments and techniques. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)* (pp. 145–152). https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0043.

Drools (2018). Drools Business Rules Management System. http://www.drools.org.

Etzion, O., & Niblett, P. (2010). *Event Processing in Action*. (1st ed.). Greenwich, CT, USA: Manning Publications Co.

Eugster, P. T., Felber, P. A., Guerraoui, R., & Kermarrec, A.-M. (2003). The many faces of publish/subscribe. *ACM Computing Surveys*, *35*, 114–131. https://doi.org/10.1145/857076.857078.

Fleury, A., Vacher, M., & Noury, N. (2010). Svm-based multimodal classification of activities of daily living in health smart homes: Sensors, algorithms, and first experimental results. *IEEE Transactions on Information Technology in Biomedicine*, *14*, 274–283. https://doi.org/10.1109/TITB.2009.2037317.

Fleury, A., Vacher, M., Portet, F., Chahuara, P., & Noury, N. (2013). A french corpus of audio and multimodal interactions in a health smart home. *J. Multimodal User Interfaces*, *7*, 93–109. https://doi.org/10.1007/s12193-012-0104-x.

Flouris, I., Giatrakos, N., Deligiannakis, A., Garofalakis, M., Kamp, M., & Mock, M. (2017). Issues in complex event processing: Status and prospects in the big data era. *Journal of Systems and Software*, *127*, 217 – 236. https://doi.org/10.1016/j.jss.2016.06.011.

Forgy, C. L. (1982). Rete: A Fast Algorithm for the Many PatternIMany Object Pattern Match Problem. *Artificial Intelligence*, *19*, 17–37. https://doi.org/10.1016/0004-3702(82)90020-0.

Galov, I. V., Lomov, A. A., & Korzun, D. G. (2015). Design of semantic information broker for localized computing environments in the internet of things. In *2015 17th Conference of Open Innovations Association (FRUCT)* (pp. 36–43). https://doi.org/10.1109/FRUCT.2015.7117968.

Groppe, S., Groppe, J., Kukulenz, D., & Linnemann, V. (2007). A sparql engine for streaming rdf data. In *2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System* (pp. 167–174). https://doi.org/10.1109/SITIS.2007.22.

HABITAT (2016). The HABITAT project. http://www.eng.habitatproject.info.

Hoffmeyer, A., Yordanova, K., Teipel, S., & Kirste, T. (2012). Sensor based monitoring for people with dementia: Searching for movement markers in alzheimer's disease for a early diagnostic. In R. Wichert, K. Van Laerhoven, & J. Gelissen (Eds.), *Constructing Ambient Intelligence* (pp. 137–145). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-31479-7_21.

Honkola, J., Laine, H., Brown, R., & Tyrkko, O. (2010). Smart-M3 information sharing platform. In *The IEEE symposium on Computers and Communications* (pp. 1041–1046). IEEE. https://doi.org/10.1109/ISCC.2010.5546642.

JSON (2018). JSON SPARQL Application Profile. http://mml.arces.unibo.it/TR/jsap.html.

JSON Web Tokens (2015). RFC 7519, JSON Web Tokens. https://tools.ietf.org/html/rfc7519.

Komazec, S., Cerri, D., & Fensel, D. (2012). Sparkwave: Continuous schema-enhanced pattern matching over rdf data streams. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems* DEBS '12 (pp. 58–68). New York, NY, USA: ACM. https://doi.org/10.1145/2335484.2335491.

Kowalski, R., & Sergot, M. (1989). A logic-based calculus of events. In J. W. Schmidt, & C. Thanos (Eds.), *Foundations of Knowledge Base Management: Contributions from Logic, Databases, and Artificial Intelligence*

*Applications* (pp. 23–55). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-83397-7_2.

Lassila, O. (2007). *Programming Semantic Web Applications: A Synthesis of Knowledge Representation and Semi-Structured Data*. Ph.D. thesis Department of Computer Science and Engineering, Helsinki University of Technology (Espoo, Finland). http://lib.tkk.fi/Diss/2007/isbn9789512289851/.

Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., & Hauswirth, M. (2011). A native and adaptive approach for unified processing of linked streams and linked data. In L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. Noy, & E. Blomqvist (Eds.), *The Semantic Web – ISWC 2011* (pp. 370–388). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-25073-6_24.

Le-Phuoc, D., Parreira, J. X., & Hauswirth, M. (2012). Linked Stream Data Processing. In *Reasoning Web. Semantic Technologies for Advanced Query Answering: 8th International Summer School 2012, Vienna, Austria, September 3-8, 2012. Proceedings* (pp. 245–289). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-33158-9_7.

Leotta, F., Mecella, M., & Mendling, J. (2015). Applying process mining to smart spaces: Perspectives and research challenges. In A. Persson, & J. Stirna (Eds.), *Advanced Information Systems Engineering Workshops* (pp. 298–304). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-19243-7_28.

Liang, P. C., & Krause, P. (2016). Smartphone-based real-time indoor location tracking with 1-m precision. *IEEE Journal of Biomedical and Health Informatics*, *20*, 756–762. https://doi.org/10.1109/JBHI.2015.2500439.

Linked Data (2017). Linked Data Notifications protocol. https://www.w3.org/TR/ldn/.

Llanes, K. R., Casanova, M. A., & Lemus, N. M. (2016). From Sensor Data Streams to Linked Streaming Data: a survey of main approaches. *Journal*

*of Information and Data Management*, *7*, 130–140. https://seer.ufmg.br/index.php/jidm/article/view/1618.

Luckham, D. (2008). The power of events: An introduction to complex event processing in distributed enterprise systems. In N. Bassiliades, G. Governatori, & A. Paschke (Eds.), *Rule Representation, Interchange and Reasoning on the Web* (pp. 3–3). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-88808-6_2.

Luckham, D., & Schulte, R. (2011). Event processing glossary – version 2.0. http://www.complexevents.com/wp-content/uploads/2011/08/EPTS_Event_Processing_Glossary_v2.pdf.

Machado, A., Maran, V., Augustin, I., Wives, L. K., & de Oliveira, J. P. M. (2017). Reactive, proactive, and extensible situation-awareness in ambient assisted living. *Expert Systems with Applications*, *76*, 21 – 35. https://doi.org/10.1016/j.eswa.2017.01.033.

Mincolelli, G., Marchi, M., Giacobone, G. A., Chiari, L., Borelli, E., Mellone, S., Tacconi, C., Cinotti, T. S., Roffia, L., Antoniazzi, F., Costanzo, A., Paolini, G., Masotti, D., Mello, P., Chesani, F., Loreti, D., & Imbesi, S. (2019). UCD, Ergonomics and Inclusive Design: The HABITAT Project. In S. Bagnara, R. Tartaglia, S. Albolino, T. Alexander, & Y. Fujita (Eds.), *Proceedings of the 20th Congress of the International Ergonomics Association (IEA 2018)* (pp. 1191–1202). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-96071-5_120.

Montali, M. (2010). *Specification and Verification of Declarative Open Interaction Models - A Logic-Based Approach* volume 56 of *Lecture Notes in Business Information Processing*. Springer. https://doi.org/10.1007/978-3-642-14538-4.

Morandi, F., Roffia, L., D'Elia, A., Vergari, F., & Cinotti, T. S. (2012). RedSib: a Smart-M3 semantic information broker implementation. In *12th FRUCT Conference* (pp. 86–98). SUAI. https://doi.org/10.23919/FRUCT.2012.8122091.

Murth, M. (2008). A Semantic Event Notification Service for Knowledge-Driven Coordination. In *Proc. of 1st Int'l. workshop on emergent semantics and cooperation in open systems (ESTEEM), cooperation with the*

*2nd Int'l. Conf. on Distributed Event-Based Systems (DEBS 2008), Rome, Italy*.

Murth, M., & Kühn, E. (2009a). A heuristics framework for semantic subscription processing. In *The Semantic Web: Research and Applications* (pp. 96–110). Springer. https://doi.org/10.1007/978-3-642-02121-3_11.

Murth, M., & Kühn, E. (2009b). Knowledge-based coordination with a reliable semantic subscription mechanism. In *Proceedings of the 2009 ACM Symposium on Applied Computing* (pp. 1374–1380). ACM. https://doi.org/10.1145/1529282.1529588.

Murth, M., & Kühn, E. (2010). Knowledge-based interaction patterns for semantic spaces. In *Proceedings of the 4th International Conference on Complex, Intelligent and Software Intensive Systems* (pp. 1036–1043). https://doi.org/10.1109/CISIS.2010.31.

Nabian, M. (2017). A comparative study on machine learning classification models for activity recognition. *Journal of Information Technology & Software Engineering*, *07*. http://doi.org/10.4172/2165-7866.1000209.

Nams, V. O., Fozard, J. L., & Kearns, W. D. (2010). Tortuosity in movement paths is related to cognitive impairment. *Methods of Information in Medicine*, *49*, 592–598. http://doi.org/10.3414/ME09-01-0079.

Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. *Proceedings of the SIGCHI conference on Human factors in computing systems celebrating interdependence - CHI '94* , . http://doi.org/10.1145/191666.191729.

Ovaska, E., Cinotti, T. S., & Toninelli, A. (2012). The Design Principles and Practices of Interoperable Smart Spaces. In X. Liu, & Y. Li (Eds.), *Advanced Design Approaches to Emerging Software Systems: Principles, Methodologies and Tools* (pp. 18–47). Hershey, PA, USA: IGI Global. https://doi.org/10.4018/978-1-60960-735-7.ch002.

Pellegrino, L., Baude, F., & Alshabani, I. (2012). Towards a scalable cloud-based RDF storage offering a pub/sub query service. In *CLOUD COMPUTING 3rd International Conference on Cloud Computing and GRIDs Virtualization* (pp. 243—246).

Pellegrino, L., Huet, F., Baude, F., & Alshabani, A. (2013). A distributed publish/subscribe system for rdf data. In A. Hameurlain, W. Rahayu, & D. Taniar (Eds.), *Data Management in Cloud, Grid and P2P Systems* (pp. 39–50). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-40053-7_4.

Pettersson, A., Olsson, E., & Wahlund, L.-O. (2005). Motor function in subjects with mild cognitive impairment and early alzheimer's disease. *Dementia and Geriatric Cognitive Disorders*, *19*, 299–304. http://doi.org/10.1159/000084555.

Pettersson, A. F., Engardt, M., & Wahlund, L.-O. (2002). Activity level and balance in subjects with mild alzheimer's disease. *Dementia and Geriatric Cognitive Disorders*, *13*, 213–216. http://doi.org/10.1159/000057699.

Qin, Y., Sheng, Q. Z., Falkner, N. J., Dustdar, S., Wang, H., & Vasilakos, A. V. (2016). When things matter: A survey on data-centric internet of things. *Journal of Network and Computer Applications*, *64*, 137–153. https://doi.org/10.1016/j.jnca.2015.12.016.

Rashidi, P., & Mihailidis, A. (2013). A survey on ambient-assisted living tools for older adults. *IEEE J. Biomedical and Health Informatics*, *17*, 579–590. https://doi.org/10.1109/JBHI.2012.2234129.

Rinne, M., Abdullah, H., Törmä, S., & Nuutila, E. (2012). Processing heterogeneous rdf events with standing sparql update rules. In R. Meersman, H. Panetto, T. Dillon, S. Rinderle-Ma, P. Dadam, X. Zhou, S. Pearson, A. Ferscha, S. Bergamaschi, & I. F. Cruz (Eds.), *On the Move to Meaningful Internet Systems: OTM 2012* (pp. 797–806). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-33615-7_24.

Roffia, L., Azzoni, P., Aguzzi, C., Viola, F., Antoniazzi, F., & Salmon Cinotti, T. (2018). Dynamic Linked Data: A SPARQL Event Processing Architecture. *Future Internet*, *10*. https://doi.org/10.3390/fi10040036.

Roffia, L., Morandi, F., Kiljander, J., D'Elia, A., Vergari, F., Viola, F., Bononi, L., & Cinotti, T. S. (2016). A Semantic Publish-Subscribe Architecture for the Internet of Things. *IEEE Internet of Things Journal*, . https://doi.org/10.1109/JIOT.2016.2587380.

Sanderson, R., & Van de Sompel, H. (2012). Cool URIs and Dynamic Data. *IEEE Internet Computing*, *16*, 76–79. https://doi.org/10.1109/MIC.2012.78.

Schade, S., Ostermann, F., Spinsanti, L., & Kuhn, W. (2012). Semantic Observation Integration. *Future Internet*, *4*, 807–829. https://doi.org/https://doi.org/10.3390/fi4030807.

SEPA (2018). SEPA reference implementation. https://github.com/arces-wot.

Sergot, M. (2002). Bob kowalski: A portrait. In A. C. Kakas, & F. Sadri (Eds.), *Computational Logic: Logic Programming and Beyond: Essays in Honour of Robert A. Kowalski Part I* (pp. 5–25). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45628-7_2.

Sheriff, C. I., Naqishbandi, T., & Geetha, A. (2015). Healthcare informatics and analytics framework. In *2015 International Conference on Computer Communication and Informatics (ICCCI)* (pp. 1–6). https://doi.org/10.1109/ICCCI.2015.7218108.

Skovronski, J. (2006). *An Ontology-Based Publish-Subscribe Framework*. Master's thesis State University of New York at Binghamton.

SOFIA (2018). The SOFIA Project. https://cordis.europa.eu/project/rcn/106175_en.html.

SPARQL Protocol (2013). SPARQL 1.1 Protocol. https://www.w3.org/TR/sparql11-protocol/.

SPARQL Query (2013). SPARQL 1.1 Query Language. https://www.w3.org/TR/sparql11-query/.

SPARQL SE Protocol (2018). SPARQL 1.1 Secure Event Protocol. http://mml.arces.unibo.it/TR/sparql11-se-protocol.html.

SPARQL Subscribe (2018). SPARQL 1.1 Subscribe Language. http://mml.arces.unibo.it/TR/sparql11-subscribe.html.

SPARQL Update (2013). SPARQL 1.1 Update Language. https://www.w3.org/TR/sparql11-update/.

Suomalainen, J., Hyttinen, P., & Tarvainen, P. (2010). Secure information sharing between heterogeneous embedded devices. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume* ECSA '10 (pp. 205–212). New York, NY, USA: ACM. https://doi.org/10.1145/1842752.1842793.

Sztyler, T., Völker, J., Carmona, J., Meier, O., & Stuckenschmidt, H. (2015). Discovery of personal processes from labeled sensor data - an application of process mining to personalized health care. In *ATAED@Petri Nets/ACSD*.

Tax, N., Sidorova, N., Haakma, R., & van der Aalst, W. (2018a). Mining process model descriptions of daily life through event abstraction. In Y. Bi, S. Kapoor, & R. Bhatia (Eds.), *Intelligent Systems and Applications* (pp. 83–104). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-69266-1_5.

Tax, N., Sidorova, N., Haakma, R., & van der Aalst, W. M. P. (2018b). Event abstraction for process mining using supervised learning techniques. In Y. Bi, S. Kapoor, & R. Bhatia (Eds.), *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016* (pp. 251–269). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-56994-9_18.

Terroso-Sáenz, F., Valdés-Vela, M., Campuzano, F., Botia, J. A., & Skarmeta-Gómez, A. F. (2015). A complex event processing approach to perceive the vehicular context. *Information Fusion*, *21*, 187 – 209. https://doi.org/10.1016/j.inffus.2012.08.008.

Umbrich, J., Villazön-Terrazas, B., & Hausenblas, M. (2010). Dataset dynamics compendium: A comparative study. In *Proceedings of the First International Conference on Consuming Linked Data* (pp. 49–60). Aachen, Germany: CEUR-WS.org volume 665. http://dl.acm.org/citation.cfm?id=2878947.2878952.

Viola, F., D'Elia, A., Roffia, L., & Cinotti, T. (2016). A modular lightweight implementation of the Smart-M3 semantic information broker. In *2016 18th Conference of Open Innovations Association and Seminar on Information Security and Protection of Information Technology (FRUCT-ISPIT)* (pp. 370–376). IEEE. https://doi.org/10.1109/FRUCT-ISPIT.2016.7561552.

Wang, J., Jin, B., & Li, J. (2004). An ontology-based publish/subscribe system. In H.-A. Jacobsen (Ed.), *Middleware 2004* (pp. 232–253). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-30229-2_13.

WebSocket (2011). RFC 6455, The WebSocket Protocol. https://tools.ietf.org/html/rfc6455.

Wickramasinghe, A., Torres, R. L. S., & Ranasinghe, D. C. (2017). Recognition of falls using dense sensing in an ambient assisted living environment. *Pervasive and Mobile Computing*, *34*, 14 – 24. https://doi.org/10.1016/j.pmcj.2016.06.004. Pervasive Computing for Gerontechnology.

World Health Organization (2015). *World report on ageing and health*. Geneva, Switzerland: World Health Organization. http://www.who.int/ageing/events/world-report-2015-launch/en.

Zigbee (2018). Zigbee Alliance. https://www.zigbee.org.