Test event generation for a fall-detection IoT system

Lorena Gutiérrez-Madroñal, Inmaculada Medina-Bulo
UCASE Research Group
University of Cadiz, Spain
Email: {lorena.gutierrez, inmaculada.medina}@uca.es

Luigi La Blunda, Matthias F. Wagner
WSN and IOT Research Group
Frankfurt University of Applied Sciences
Email: {l.lablunda,mfwagner}@fb2.fra-uas.de

***Abstract*—The Internet of Things (IoT) is a very popular paradigm which has been applied to different areas such as smart cities, medicine, business process, etc. The IoT system main inconvenience is to make decisions in real time according to a huge amount of information that arrives as events. This information is filtered thanks to the Event Processing Languages (EPL), which use patterns to define the relevant situations. So, given that filter the correct information is crucial to carry out the established actions, testing these IoT systems is imperative. In the majority of the relevant situations to detect, the events have a specific behaviour which must be simulated to test the IoT system. Moreover, in several situations is quite difficult to obtain test events with specific values: adverse environment conditions, rise or fall in blood pressure, heart attack, falls... In this paper we introduce a fall analysis and the test event generation using IoT-TEG tool. The fall analysis has highlighted the special behaviour of the fall-involved events and the necessity to improve the IoT-TEG with a new functionality which allows to define the desired behaviour by defining behaviour rules.**

I. Introduction

Due to the progress of health care, the longevity of people increases and this leads to an ageing society. Elderly people are exposed to a higher risk of falling because of the growing age and multiple diseases, which cause serious injuries that require long convalescence and restriction of mobility. According to the survey of the *Robert Koch Institute* [1], 53.7% of accidents in the age group over 60 are caused by falls. Statistically, about one-third of the elderly people suffer severe lesions and the half of them suffer fall-events repeatedly [2]. Falls are not deployed by a single cause, 90% of them occurred from multiple factors. These factors refer to old-age or illness (intrinsic factors) or external factors e.g. hazards which occur at home, in traffic or during activities of daily life (extrinsic factors) [2]. The founder of *Vigilio Telemedical* reported that yearly more than 20 million elderly over the age of 65 in Europe experience fall-situations, that lead to traumatic based cases of death [3], [4]. Additionally, people affected by Dementia and Parkinson have a higher risk to fall. In accordance with [5] research proved that Dementia-patients have a 20 times higher risk and Parkinson-patients a 10 times higher risk of falling than healthy people of the same age. To counteract these life-threatening situations a fast and fully automated assistance is needed, because an unconscious person may not be able to call the emergency services. An approach could be continuous monitoring of medical and/or physical signals via a wearable sensor network (see Figure 2).

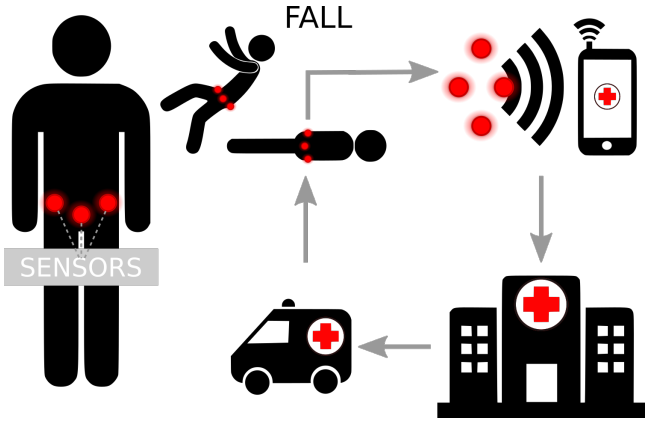

Figure 1. Scheme fall simulation.

A prototype in form of a belt was developed, which is worn on the hip by the patient and consists of a five sensor nodes Body Area Network (BAN) [6], [7]. The obtained data are used to define a EPL of EsperTech [8] pattern to detect falls. The first step of this prototype is to identify a fall from a fast movement, a siting move or lying move, but the main goal of the to be finished system is to prevent the falls and act to prevent them. Given that to test this system is crucial, test events which simulate falls are necessary. In the literature can be found different type of falls, and it is necessary to identify all of them in order to tell them apart from a no-fall; in this study two type of falls will be analysed.

IoT-TEG [9], [10] is a tool which automatically generates test events of many type. Thanks to the obtained data from the sensors we have checked that the measured parameter during a fall, the acceleration, has an specific behaviour. As a consequence, the test events must be generated according to its behaviour. This problem is solved with the new functionality that IoT-TEG includes which is introduced in this paper. Moreover, the ongoing fall detection prototype will be analysed and its improvements will be described; the new functionality of IoT-TEG can be adapted according to the improvements of the fall detection prototype. The main contributions of this paper are:

- **An analysis of the involved parameter in a fall**; while a person is falling, the acceleration is the parameter that can measure the movement of the body. This parameter is analysed in order to know its behaviour during two type

of falls.

- **A new functionality of the IoT-TEG system** which allows to simulate the behaviour of different event attributes in order to generate test events following a specific pattern.
- **A study of the fall detection prototype evolution**; the system which is been used has suffered some improvements. We describe the evolution its architecture, how the data is analysed and the founded problems.
- **New definitions of two type of falls**; after the analysis of the acceleration during two type of falls a new definition for each one has been done. The obtained data of the fall detection prototype is used to define a EPL of EsperTech pattern to detect those type of falls.

The remained of this paper is organised as follows. Section II describes not only the related work of event generators, but also the existing solutions for fall-detection. Section III provides the basic knowledge of falls, Event Processing Languages and IoT-TEG tool. The architecture of the fall detection system, the fall analysis and some founded problems are introduced in Section IV. Section V describes the improvements on the prototype, a new fall analysis and a comparison of the obtained results. Finally, in Section VI, we conclude our paper and make recommendations for future work.

II. Related work

An overview about event generators reveals that the first event generators [11], [12] were focused on extremely specific topics such as environmental conditions for the simulation of high energy physics events at particle colliders. Nowadays, the people and business need to control and monitor the things around them. The received information allows them to make decisions and to act according to it. This is the reason of the creation of the IoT Platforms, which are the key for the development of scalable IoT applications and services that connect objects, systems and people to each other. However, not every IoT Platform is an IoT Platform [13]; for instance, some event generators that are integrated in an enterprise software packages, which are increasingly allowing the integration of IoT devices, are often not advanced enough to be classified as a full IoT Platform. Examples are given in the following lines:

The Timing System [14] provides a complete timing distribution system including timing signal generation. Its event generator is responsible for creating timing events which are sent out as serialised event frames.

The company Starcom [15] has developed an event generator to solve the problem of managing a huge number of events. They state that their generator is capable of controlling the end event action, so the exact managers requirements can be filtered. The tool is included in a kit distributed with their system.

The WebLogic Integration Solutions [16] allows the managing and monitoring of entities and resources required for WebLogic Integration applications. This system contains an event generator module which allows the creation and deployment of the event generators included as part of WebLogic Integration. The mentioned events generators allow to define event types but they do not are capable to simulate a specific behaviour with a set of generated events. The relevant situations in IoT systems are a sequence of activities with a determined behaviour; that is why IoT-TEG [9], [10] includes this option.

Talking about fall-detection, there are several solutions that propose wearable sensors. A commercial solution which is available on the market is the VigiFall system [3], [17], supported by the European Commission. This system includes a wearable self-adhesive accelerometer, several motion sensors which are fixed in the living area and a calling unit to provide a fully automated emergency call. The self-adhesive patch communicates with the infrared motion detectors and in the case of a fall the wearable patch sends a signal out. Thereby the infrared sensors placed in the room are capable to recognise that there are no movements anymore and contacts the central unit. When the central node receives this flag, the emergency call is executed. The weak point of VigiFall [3] is the system structure which depends on the sensor-based room infrastructure. Once the patient leaves this area the system is not capable to provide the functionalities in case of a fall-event. Igual et al. [18] examined different fall-detection approaches which they categorised into context-aware systems and wearable systems. Context-aware systems depend on the living area of the patient, where sensors and actuators placed in the environment interact with the wearable node of the person to detect falls. Another solution based on this principle is a video-based method, which facilitates a reliable detection of falls but the patients would be exposed to a loss of privacy and this is not well accepted. Additionally the high purchase price is a barrier for many people and the dependency on the environment makes these system useless. The other system-type analysed by Igual et al. [18] comprises wearable systems, which are worn on the body and based on a BAN. This solution is able to detect falls independent from the environment in contrast to context-aware systems. They depict wearable systems which use the sensor fusion principle of accelerometer and gyroscope and built-in solutions that represent the usage of integrated smartphone sensors.

Taking into consideration wearable fall-detection solutions the approach proposed by Li et al. [19] will be explained subsequently, which was used for the development of our prototype. They introduce a fall-detection method based on a BAN that consists of two wearable sensor nodes. These two nodes comprise an accelerometer and gyroscope and are worn on the chest, node A, and thigh node B, (see Figure 2). The principle of this method differentiates between static postures and dynamic postures:

- Static postures: standing, sitting, lying and bending.
- Dynamic postures
  - Activities of daily life: walking, walk on stairs, sit, jump, lay down and run.
  - Fall-like motions: quick sit down upright and quick sit-down reclined.

– Flat surface falls: fall forward, fall backward, fall right and fall left.
– Inclined falls: fall on stairs.

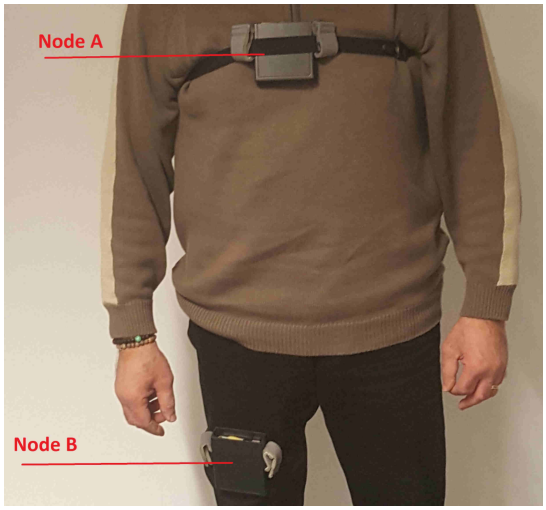

Figure 2. System architecture according to Li et al. [19]

To decrease the computational effort of the micro-controller a three-phase algorithm was proposed, which is structured as follows:

1) Phase activity analysis: check if person is in a static or dynamic position.
2) Phase position analysis: if existing posture coincides with static posture, check whether the actual position corresponds to lying.
3) Phase state transition analysis: if lying position, examine whether this transition was intentional or unintentional. The previous 5 seconds are used to analyse it. In the case that this position was unintentional, the system classifies it as a fall.

The weak point of this method is the differentiation between jumping into bed and falling against a wall with seated posture.

III. Background

A. Falls analysis

The approach used to detect falls is based on the basic idea proposed by [20], [21]. A typical acceleration pattern during a fall-event is a decreasing acceleration close to 0g (free fall), followed by an increasing acceleration value (see Figure 3). In a stationary position, the acceleration measured is around 1g ($9.81 m/s^2$) and during falling around 0g ($0 m/s^2$). Upon the impact to the ground (fall-event), the maximum acceleration (peak-value) is reached for a short time period and it is greater than 1g. This pattern can be used to detect fall-events using the integrated accelerometer of the sensor nodes (S1-S4). Important to know is that the acceleration magnitude which is illustrated in the Formula 1 was used for the fall-detection:

$$\overrightarrow{|\alpha|} = \sqrt{\alpha_x^2 + \alpha_y^2 + \alpha_z^2} \tag{1}$$

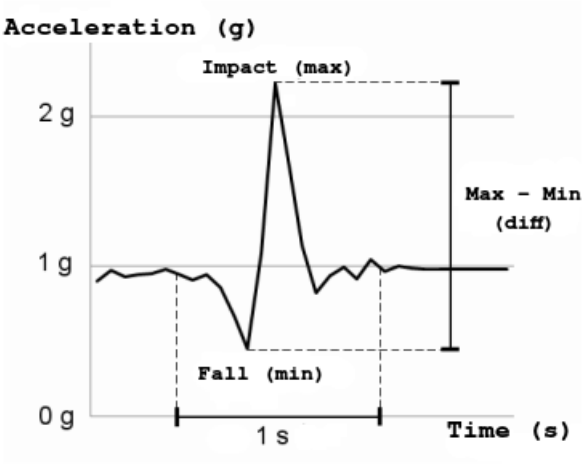

Figure 3. Acceleration during impact [21].

Taking into consideration the depicted Figure 3, we can apply the following rule to evaluate the incoming acceleration data:

$$\alpha_{max} - \alpha_{min} > 1g \tag{2}$$

in 1 second window and α_{max} came after α_{min}; where α_{max} is the maximum acceleration value, α_{min} is the minimum acceleration value and $1g \approx 9.81 m/s^2$. Looking at the rule an event is categorised as a fall when the difference between α_{max} and α_{min} is greater than 1g and α_{min} is followed by α_{max}. Important is that this should happen within a time window of 1 second due to the fact, that a fall can occur in less than 1 second [22]. To apply this rule we used the Event Processing Language (EPL) of EsperTech [8] to integrate it in a IoT system which detect falls.

B. Fall patterns with EPL of EsperTech

Etzion and Niblett [23] defined *event processing agents* as software modules that process events. Such agents are specified using an event processing language, and there are a number of styles of event processing languages in use. The following styles are included:

- Rule-oriented languages
  - Production rules: Production rules are rules of the type *if condition then action*: when the condition is satisfied, the action is performed.
  - Active rules: Active rules are rules of the type *event-condition-action*: when an event occurs, the conditions are evaluated and, if they are satisfied, it triggers an action.
  - Logic rules: A programming style based on logical assertions and deductive database.
- Imperative programming languages: The imperative programming languages define the operators which will be applied to the events. Each operator is a transformation in the event.

- Stream-oriented languages: The languages used to describe the queries are inspired by SQL and relational algebra, though not all of them are based on SQL.

The EPL in which our work is based on is EPL of Esper [8], an EPL stream-oriented language. The main reasons of its selection are: it is an extension of SQL (world famous), it can be embedded into Java applications and it is open source. On the other hand, it is executed by Esper, a CEP engine which can process around 500.000 events per second on a workstation, and between 70.000 and 200.000 events per second on a laptop (according to the company EsperTech).

As it was mentioned before, the EPL of EsperTech is a SQL like query language. However, unlike SQL that operates on tables, EPL operates on continuous stream of events. As a result, a row from a table in SQL is analogous to an event present in an event stream. An EPL statement starts executing continuously during runtime. While the execution is taking place, EPL queries will be triggered if the application receives pre-defined or timer triggering events.

Example 1. EPL of EsperTech query example

```
select A as temp1, B as temp2 from
  pattern [every temp1.temperature > 400
  -> temp2.temperature > 400]
```

In the above example (see Example 1) a "Central" needs to measure the temperature of its systems, its temperature gauges take a reading of the core temperature every second and send the data to a central monitoring system. The EPL of EsperTech query throws a warning if we have 2 consecutive temperatures above a certain threshold (400). This is a situation where it is needed a quick reaction to emerging patterns in a stream of data events. A quick reaction is also needed in fall detection, the pattern which describes this situation will be shown in the corresponding section.

Because the difficulties to simulate falls, IoT-TEG is used in order to get automatically the fall test events. IoT-TEG can be adapted or modified, if it is necessary, to generate any test events, even after the improvements in the fall-detection prototype.

C. IoT test event generator

IoT-TEG is a Java-based tool which takes an event type definition file and a desired output format (JSON, CSV, and XML, the most common across IoT platforms). IoT-TEG is made up of a *validator* and an *event generator* (Figure 4). The validator ensures the definition follows the rules set by IoT-TEG. The generator takes the definition and generates the indicated number of events according to it.

Previous studies suggested there were no differences in testing effectiveness between using events generated by IoT-TEG, or events recorded from various case studies [10]. These results confirm IoT-TEG can simulate many types of events occurring in industrial applications, and solve the main challenges developers face when they test event-processing programs:

1) Lack of data for testing,

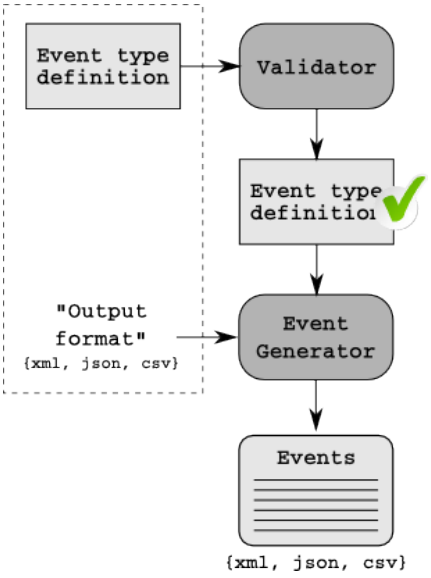

Figure 4. IoT-TEG Architecture.

2) needing specific values for the events, and
3) needing the source to generate the events.

For the sake of clarity, Example 2 shows an event type definition that could be used to test the queries of Example 1.

Example 2. Event type definition example

```
<?xml version="1.0" encoding="UTF-8"?>
<event_type name="TemperatureEvent">
  <block name="feeds" repeat="150">
    <field name="created_at" quotes="true"
     type="ComplexType">
     <attribute type="Date" format="yy-MM-dd">
     </attribute>
     <attribute type="String" format="T"></attribute>
     <attribute type="Time" format="hh:mm">
     </attribute>
    </field>
    <field name="entry_id" quotes="false"
     type="Integer" min="0" max="10000">
    </field>
    <field name="temperature" quotes="false"
     type="Float" min="0" max="500" precision="1">
    </field>
  </block>
</event_type>
```

The defined event type contains three properties: `created_at`, `entry_id` and `temperature`. These properties are defined as fields in the event type definition. The `created_at` field is complex type and `entry_id` and `temperature` are simple types. The property that is evaluated in the Example 2 queries is `temperature`.

IV. Prototype

A. Architecture

The system architecture is an important part to provide a precise and reliable fall-analysis. Additionally, the aspect of patient compliance should be taken into consideration, because the system should be developed not only from the developer point of view, but it should be accepted by the patients. An

important requirement of the elderly is that the hardware design should facilitate the freedom of movement. Furthermore, the system should guarantee a reliable functionality and a redundancy to protect the wearable system against a total system failure. With reference to these aspects a BAN in form of a belt was developed which includes a five sensor nodes which is based on ZigBee[1].

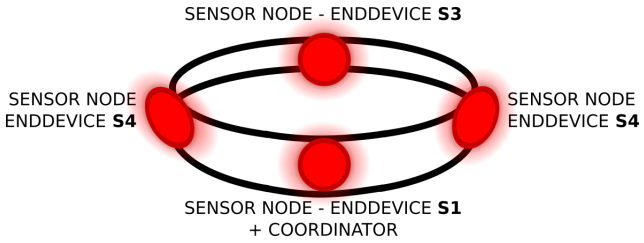

Figure 5. Fall-detection belt.

Four of the sensor nodes are acting as end-devices (S1-S4) and the other node as a coordinator. The end-devices have attached an accelerometer and gyroscope which acquire continuously sensor data and is sent wirelessly to the coordinator using the ZigBee protocol. The coordinator receives the incoming data and it has the function to evaluate the patient's status. To test the system's accuracy different fall-types were reproduced with several test persons. A test procedure based on Li et al. [19] and Pannurat et al. [24] was developed which includes several motions and fall-types which are typical in nursing homes and hospitals.

Taking into account the architecture of the prototype and the rule to define a fall (see Section III-A), the EPL of EsperTech pattern to define the fall situation is the one shown in the Example 3.

Example 3. Fall pattern

```
select a1.accelS1, a2.accelS1, a1.accelS2,
 a2.accelS2 from
 pattern [every(a1=BodyEvent(a1.accelS1 <= 9.81) ->
 a2=BodyEvent(a2.accelS1 -a1.accelS1 >= 9.81 and
 a1.PersonID = a2.PersonID)
 where timer:within(1sec)) or every
 (a1=BodyEvent(a1.accelS2 <= 9.81)
 -> a2=BodyEvent(a2.accelS2-a1.accelS2 >= 9.81
 and a1.PersonID = a2.PersonID)
 where timer:within(1sec))];
```

We should describe a little the EPL query

B. Fall simulation test events

The fall to generate the test events have been selected from [19], [24]. The fall consists on rolling into bed and fall (RBF). In this study one person have been analysed; he has been doing for a period of 2 minutes the RBF fall. The analysed data and videos can be found in []. In the following lines the steps to simulate the fall with the generated events are described:

Mirar el tiempo que estuvo cayendo

[1]http://www.zigbee.org/

1. Study of the values: Given that the sensor 1 is the one that suffer the impact, its acceleration values are the first to be analysed. The Figure 6 shows the acceleration data from sensor 1 while the person was falling. The goal is to study the acceleration behaviour while a fall in order to generate test events, so the acceleration values are normalised ($N(m/s^2)$). The normalised acceleration is shown in Y edge, and the time in milliseconds (ms) is in X edge.

While the data analysis, it has to be taken into account that the values suffer alterations because several factors: the person movement, the person bounces against something (floor, wall, etc), the collocation of the sensors to the original position after a fall, sensor pressure because an impact or the person is lying over it, etc.

2. Fall identification and analysis: After the previous study, the peaks of the acceleration are identified. These peaks, or maximum values, are when the sensor suffers the impact. Because the mentioned noise, two ranges of the obtained values are extracted in order to analyse data properly. Please, see the highlighted parts in Figure 6; according to X edge $[210, 360]$ (Fall 1) and $[900, 1050]$ (Fall 2). The range of extracted values are a set of data that happen is less than a time window of 1 second, to meet the fact described in [22]. So as to compare both RBF falls the acceleration values are normalised according to their impact value. The tables Table I and Table II show the values to analyse where the impact value is highlighted with colour green.

Mirar los datos originales del prototipo 1 para comprobar el timestamp The first columns of the Tables represent the milliseconds (ms) when the acceleration was measured. The second columns show the acceleration values (m/s^2) and in the third columns are deployed the normalised acceleration values ($N(m/s^2)$) according to the fall maximum value. In Figure 7 a comparison of the normalised acceleration behaviour while the previous RBF falls is shown. This comparison show a similar behaviour of the acceleration during the RBF falls. Moreover, the acceleration in these two RBF falls follow the rule that define a fall, see Equation 2.

The goal is to simulate this type of fall with events to test the IoT system, so we have to analyse the values of the acceleration before and after the impact. Taking into account that the maximum value of the acceleration is when the impact occurs (α_{max}), the acceleration behaviour during a RBF fall consists on:

1) from a value less than the half of α_{max}, the acceleration value increases to obtain a value in the range:
$$[\alpha_{max}/2 - 0.5, \alpha_{max}/2 + 0.5]$$
The person is rolling on the bed.
2) when the acceleration obtains a value in the previous range, its value decrease until the minimum value α_{min}. The person is falling, a free fall.
3) the acceleration value goes from the minimum value α_{min} to the maximum value α_{max}. The person suffers the impact.

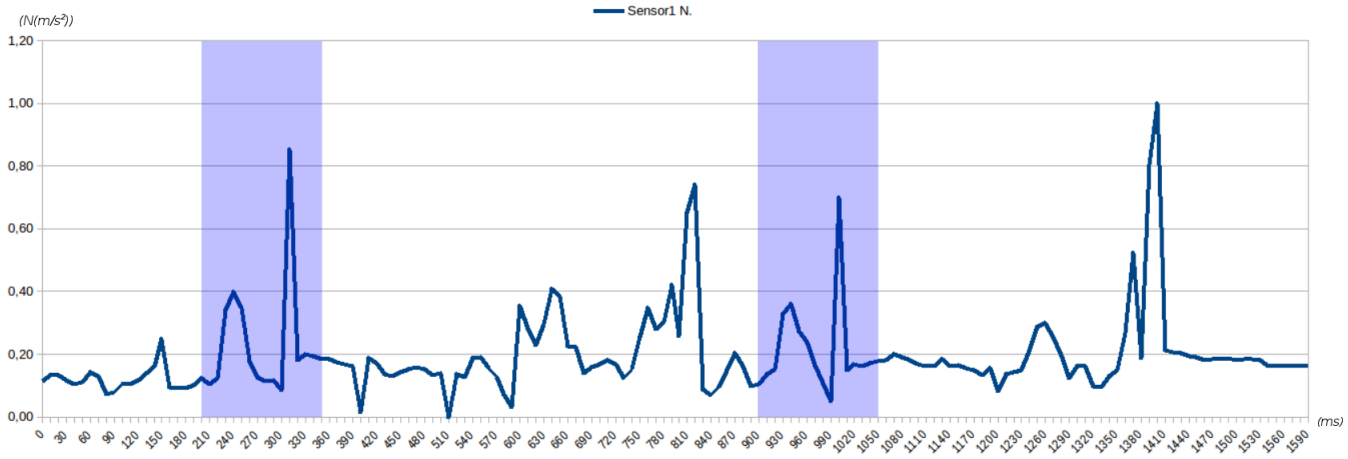

Figure 6. Sensor 1 acceleration.

TIEMPO (ms)	ACEL. (m/s^2)	ACEL. N ($N(m/s^2)$)
0	6,29	0,02
10	7,57	0,05
20	20,7	0,33
30	24,07	0,41
40	21,01	0,34
50	10,81	0,12
60	7,71	0,05
70	6,87	0,04
80	7,06	0,04
90	5,23	0,00
100	51,58	1,00
110	11,01	0,12
120	12,12	0,15
130	11,77	0,14
140	11,26	0,13
150	11,16	0,13

Table I
ACCELERATION - *Fall 1*

TIEMPO (ms)	ACEL. (m/s^2)	ACEL. N ($N(m/s^2)$)
0	5,98	0,08
10	6,31	0,08
20	8,2	0,13
30	9,21	0,16
40	19,92	0,43
50	21,8	0,48
60	16,52	0,34
70	14,41	0,29
80	9,97	0,18
90	6,54	0,09
100	3,01	0,00
110	42,34	1,00
120	8,96	0,15
130	10,14	0,18
140	9,81	0,17
150	10,45	0,19

Table II
ACCELERATION - *Fall 2*

4) the acceleration value is established with values around the half of α_{max}. The person is lying on the floor.

The same analysis process has been done to the rest of sensors, and the behaviour of the acceleration of all of them follows the same pattern.

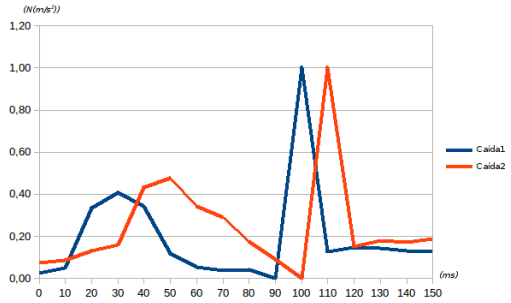

Figure 7. Acceleration comparative.

Taking into account the architecture of the prototype and the previous rules to define a RBF fall, the EPL of EsperTech pattern to the define a RBF fall is the one shown in the Example **??**.

We should include an EPL query to detect this RBF fall

3. To define the fall event: Once that the fall acceleration behaviour has been observed, the next step is to define the fall event in order to generate test events with IoT-TEG. As it was explained in Section III-C, the event type attributes have to be defined using the `<field>` element. The fall event contains one attribute, the acceleration, which is float, `type=''Float''`, and its values are not quoted, `quotes=''false''`. A new parameter in IoT-TEG has been defined as a consequence of the previous falls study. Given that the acceleration values follow a specific behaviour, it is necessary to include the `custom_behaviour` property in the `<field>` element to define the behaviour of any event attribute; in this study, the acceleration. In the `custom_behaviour` property the path to the file that includes the behaviour of the event attribute has to be written. The Example 4 shows the complete fall event definition (FallEventType).

Example 4. Fall event type definition

```
<?xml version="1.0" encoding="UTF-8"?>
<event name="FallEventType">
<block name="feeds" repeat="100">
 <field name="acceleration" quotes="false" type="Float"
 custom_behaviour="/Path/To/Rule/File"></field>
</block>
</event>
```

IoT-TEG includes a new functionality, which has been implemented to simulate the desired behaviour of an event attribute with a `custom_behaviour` property in its event type definition. This functionality allows to generate values of the event attribute following a behaviour that the user has described in a file. In order to explain how the user has to define the desired behaviour of an event attribute, we are going to use the RBF fall behaviour rules (see Example 5). In a XML file the number of simulations has to be indicated, the events involved in a simulation will be calculated according to the total number of events to generate and the desired simulations. For example, if the number of test events to generate is 100, number indicated in the event type definition file, `repeat="100"`, and the number of desired simulations is 5, `simulations="5"`, number indicated in the behaviour rules definition file, the number of events involved to simulate the behaviour is 20. In the RBF fall example, the user ask to generate 100 test events and 5 falls (simulations), so 20 events will be used to define a RBF fall.

Variables can be defined if they are needed in the behaviour rules. They can be defined in the file where the behaviour rules are included using the `<variables>` tags. To define them a name and a value have to be given to the variables. The value can be defined as a fixed value with the `value` property, or using a range with the `min` and `max` properties. Moreover, in some variables are involved in the value of another variables; this is indicated using the variable with an specific format `$(variable)`, see Example 5. In addition, arithmetic operations can be done in the definition of the variable values, let see how using them in the RBF fall example. To define the acceleration behaviour while a RBF fall three variables are defined: `Roll`, `Fall` and `Impact`. The `Impact` will be the maximum value α_{max}, the `Fall` will be the minimum value α_{min} and the `Roll` variable is defined by a range where the acceleration value will be obtained according to the `Impact` value. In order to obtain a low value, a range between 0 and the fifth part of `Roll` is assigned to the `Fall` variable.

Once the variables are defined, the `<rules>` tags are used to define the rules. A `weight` must be assigned to each one to calculate the number of events to generate for each rule for each simulation. Following the example and the assigned `weight` in Example 5, if 20 events simulate a RBF fall $20 * 0,25 = 5$ events will be generated for the first rule, another 5 events for the second rule, 1 event for the third rule, and the remained events for the fourth rule. We have to assign zero to the weight `weight="0"` to indicate how the remained events have to be generated.

To define the rules, `min`, `max` and `value` properties can be used as well as the arithmetic operations and the references to another variables. Moreover, the `sequence` property can be used to obtain values lower or higher than the one generated previously. The `sequence` property values are `inc`, to increase the value, or `dec`, to decrease the value.

Thanks to the included properties and parameters in the IoT-TEG new functionality, the desired behaviour rules can be defined. In the RBF fall rules, the involved event attribute is the acceleration, and its behaviour during the fall is defined by four rules; in the first rule the acceleration value increases to a value close or equal to `Roll`, in the second rule the acceleration value decreases to a value close or equal to `Fall`, in the third rule the acceleration value is equal to `Impact`, it obtains the highest value and in the fourth rule the acceleration value is established in a range which is lower than `Roll`.

Example 5. Rules to define a RBF fall

```
<?xml version="1.0" encoding="UTF-8"?>
<custom_conditions simulations="5">
<variables>
 <variable name="Impact" min="40.0" max="156.96"/>
 <variable name="Roll" min="$(Impact)/2-0.5"
                          max="$(Impact)/2+0.5"/>
 <variable name="Fall" min="0.0" max="$(Roll)/5"/>
</variables>
<rules>
 <rule weight="0.25" min="$(Roll)/4" max="$(Roll)"
                                    sequence="inc"/>
 <rule weight="0.25" min="$(Roll)" max="$(Fall)"
                                    sequence="dec"/>
 <rule weight="1" value="$(Impact)"/>
 <rule weight="0" min="$(Roll)/2-0.25"
                           max="$(Roll)/2+0.25"/>
</rules>
</custom_conditions>
```

It is needed to highlight that to obtain these rules to define the behaviour of the acceleration several test have been done. Once that we obtained the desired results, test events were generated as it was necessary. The Figure 8 shows the acceleration values of some of the generated RBF falls using IoT-TEG and the new functionality.

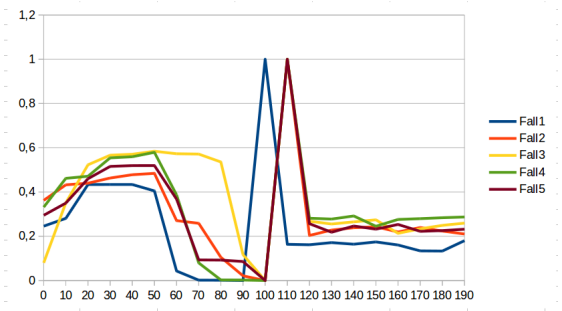

Figure 8. IoT-TEG generated RBF falls.

The generated events which simulate RBF falls follow the acceleration values that the original RBF falls contain. So, these generated events can be used to test the fall detection system.

V. Improved prototype

A. Architecture

To talk about the improvements in the prototype

B. Fall simulation test events

The second fall to generate the test events consists on the impact of the person with a wall and falling using the knees and then using the chest, *fall against wall* (FAW). In this study two persons have been analysed; they have been doing for a period of 2 minutes the FAW fall. The analysed data and videos can be found in [].

In this analysis the same steps that the ones described in Section IV have been done:

Study of the values: Given that the sensor 1 is the one that suffer the impact, its acceleration values are the first to be analysed. The acceleration values have been normalised ($N(m/s^2)$). After the normalisation the impacts of the falls, peaks, have to be detected ($N(m/s^2) > 0,7$). After applying the previous rule in all the fall data and taking into account the alterations because the mentioned factors, the impacts are detected.

Fall identification and analysis: Once the peaks are detected, a range of values, including the peaks, are selected in order to analyse data properly and to study the acceleration behaviour while FAW fall. The range of extracted values are a set of data that happen is less than a time window of 1 second, to meet the fall rule of [22] described in Equation (2). The Table III shows the acceleration value while one FAW fall of person 1 and person 2.

Time ($s.ms$)	Accel. (m/s^2)	N. Accel. ($N(m/s^2)$)	Time ($s.ms$)	Accel. (m/s^2)	N. Accel. ($N(m/s^2)$)
57.311	375,30	0,01	25.254	873,31	0,09
57.359	349,55	0,00	25.303	1096,02	0,19
57.408	795,96	0,17	25.352	1154,65	0,22
57.506	1124,36	0,30	25.401	1414,93	0,34
57.506	664,30	0,12	25.449	961,12	0,13
57.554	468,96	0,05	25.503	877,49	0,09
57.603	433,24	0,03	25.546	886,98	0,10
57.651	2334,45	0,77	25.596	824,89	0,07
57.700	2936,90	1,00	25.645	1423,64	0,35
57.749	2220,15	0,72	25.693	2323,01	0,76
57.800	915,78	0,22	25.742	1078,30	0,19
57.846	1095,01	0,29	25.791	882,43	0,09
57.900	1092,41	0,29	25.841	851,94	0,08
57.944	951,10	0,23	25.888	678,93	0,00
57.996	1340,62	0,38	25.937	1777,75	0,51
58.042	2144,14	0,69	25.986	2830,06	1,00
58.186	2881,04	0,98	26.082	1175,78	0,23
58.188	2800,97	0,95	26.084	1155,21	0,22
58.240	1016,71	0,26	26.134	1069,57	0,18
58.286	1123,59	0,30	26.181	895,96	0,10
58.334	1191,39	0,33	26.230	1306,94	0,29

Table III
FAW fall acceleration, person 1 and 2

The first and fourth columns of the Table III represent the seconds and milliseconds ($s.ms$) when the acceleration was measured. The second and fifth columns show the acceleration values (m/s^2) and in the third and sixth columns are deployed the normalised acceleration values ($N(m/s^2)$) according to the fall maximum value. In Figure 9 a comparison of the normalised acceleration behaviour while the previous FAW falls of person 1 and person 2 is shown. These comparisons show a similar behaviour of the acceleration during the FAW fall.

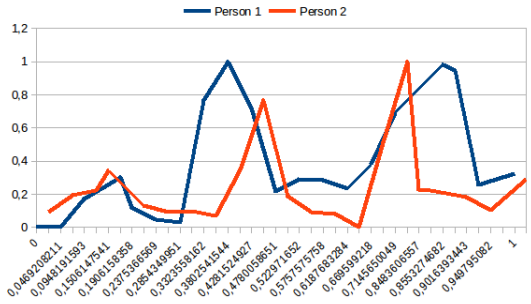

Figure 9. Acceleration comparison during FAW fall.

The acceleration behaviour during the FAW consists on:

1) the variation of its values while the person is walking, the acceleration has a normalised values between [0, 0,35].
2) as a consequence of the impact of the person against a wall, the acceleration normalised values has to be greater than 0,7.
3) the normalised acceleration value decrease. The values of the normalised acceleration are between [0, 0,35]; the number of times that the acceleration value is in the mentioned range is variable. That number of times depends on the high of the person; the higher person the longer range and if the person retains the fall thanks to the wall. Moreover, a subtle peak could appear as a consequence of a rebound.
4) a second impact happens when the person hit the ground, the acceleration normalised values has to be greater than 0,7.
5) finally, the person is lying on the ground and the normalised acceleration value decrease. The values of the normalised acceleration are between [0, 0,35] and no subtle peak appear.

The same analysis process has been done to the rest of sensors, and the behaviour of the acceleration of all of them follows the same pattern.

To define the fall event: Once that the fall acceleration behaviour has been observed, the next step is to define the fall event in order to generate test events with IoT-TEG. Given that the involved event attribute in this fall is the acceleration, the Example 4 in Section IV can be used to define the fall event (FallEventType). The rules to define the behaviour of the acceleration in this type of fall is shown in Example 6.

Example 6. Rules to define a FAW fall

```
<?xml version="1.0" encoding="UTF-8"?>
<custom_conditions simulations="5">
<variables>
```

```
 <variable name="Base" min="0.0" max="500.0"/>
 <variable name="Impact" min="$(Base)*0.7"
                         max="$(Impact)/2+0.5"/>
 <variable name="Fall" min="0.0" max="$(Roll)/5"/>
</variables>
<rules>
 <rule weight="0.25" min="$(Roll)/4" max="$(Roll)"
                                    sequence="inc"/>
 <rule weight="0.25" min="$(Roll)" max="$(Fall)"
                                    sequence="dec"/>
 <rule weight="1" value="$(Impact)"/>
 <rule weight="0" min="$(Roll)/2-0.25"
                         max="$(Roll)/2+0.25"/>
</rules>
</custom_conditions>
```

*** Like with the RBF fall, we should include a new fall definition of FAW type of fall.***

*** We must compare the prototype with and without the improvements***

C. Founded problems

After testing the used fall detection prototype, some problems were found. Moreover, some considerations will be applied in future tests.

First of all, we are going to explain the problems related to the prototype; problems related to its hardware and software. Talking about the software, the synchronisation in the prototype is an issue; we have founded the following problems:

- In one hand, the falls from the beginning always should be discarded because the sensors are sending a lot of data. Let us say that in "fall 1", there are more than 100 values in a second, and later in "fall 3", there are 35 values in a second. So, in order to analyse the falls and compare the acceleration behaviour, it is difficult to work with that information. A comparison of two FAW falls with the synchronisation problem is shown in Figure 10; the sensor 1 acceleration values for the first FAW fall are coloured in blue and the sensor 1 acceleration values for the second FAW fall are coloured in red. The first fall is one fall from the beginning of the simulation, and the second one is from the middle of the simulation.

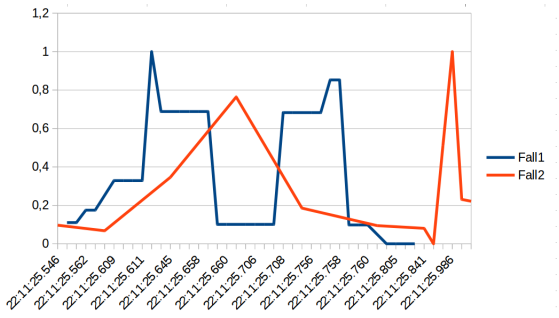

Figure 10. Synchronisation problem; comparison of acceleration values during two FAW falls.

- On the other hand, the sensors transmit more data than the others; no matter the fall, the sensors are not sending the same amount of data, even sometimes there is no data. The four sensors were working while the FAW fall simulation, but the obtained acceleration values were from three of them, one of the sensor does not transmit data in one moment of the simulation, see Figure 11.

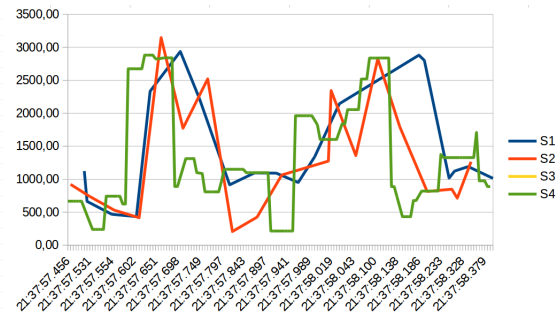

Figure 11. Synchronisation problem; acceleration values from the four sensors during a FAW fall.

If we focus our attention to the hardware, the duration of the battery is also something to improve. Nowadays the duration of the battery is X, and if we want use this IoT system in people, specially in elderly people, or people that need special treatment, we have to increase the duration of the battery in order to charge it as little as possible.

The study has revealed that while we were studying the acceleration data, some of its values could be misinterpreted. This is because the person that is falling for the simulation, stands up very fast. Given that we have been cautious in our analysis and we have been checking not only the values but also the videos and matching them, we have detected this issue. So, in our future tests, the person that is falling should wait at least 2 seconds lying on the floor after the fall for a better fall simulation. In a real situation, if a person falls and stand up means that the person is conscious and is able to move and call to the emergency services, if it is necessary. On the contrary, if the person falls and do not stand up means that maybe the person is unconscious or is not able to move and call to the emergency services. Therefore, waiting at least 2 seconds between falls will help not only to understand better the behaviour of the acceleration but also to do a better fall simulation.

VI. Conclusion and Future Work

We talk about new functionalities in IoT-TEG related with the future work in the fall detection prototype.

Acknowledgment

Paper partially funded by The Ministry of Economy and Competitiveness (Spain) and the FEDER Fund, under the National Program for Research, Development and Innovation, Societal Challenges Oriented, Project DArDOS TIN2015-65845-C3-3-R, and the Programa de Fomento e Impulso de la actividad Investigadora of the University of Cádiz.

REFERENCES

[1] G. Varnaccia, *Das Unfallgeschehen bei Erwachsenen in Deutschland: Ergebnisse des Unfallmoduls der Befragung Gesundheit in Deutschland aktuell 2010*, ser. Beiträge zur Gesundheitsberichterstattung des Bundes. Robert-Koch-Institut, 2013.

[2] N. Schott and A.-K. Kurz, "Stürze bei älteren Erwachsenen: Risikofaktoren – Assessment – Prävention: Ein Review," *Zeitschrift für Sportpsychologie*, vol. 15, no. 2, pp. 45–62, 2008.

[3] Vigilio, "Vigilio s.a. - Vigilio s.a.solutions télémédicales fiables pour la santé et la sécurité des plus fragiles," http://www.vigilio.fr/.

[4] APA-OTS, "Eu fördert Fallwatch, ein Erkennungssystem, das Senioren nach Stürzen schneller hilft," https://goo.gl/51o8X3, 2013.

[5] Monks - Ärzte im Netz GmbH, "Demenzkranke stürzen besonders häufig," https://goo.gl/rdkfL2.

[6] L. La Blunda and M. Wagner, "Fall-detection belt based on body area networks," in *Proceedings of Biotelemetry XXI (ISOB conference)*. Leuven (Belgium): Faculteit Ingenieurswetenschappen KU Leuven, 2016, vol. 1, pp. 21–24.

[7] ——, "The usage of body area networks for fall-detection," in *Proceedings of the eleventh International Network Conference (INC 2016)*. Frankfurt am Main: Alekseev, S. and Dowland, P. S. and Ghita, B. and Schneider, O., 2016, pp. 159–163.

[8] EsperTech, "Espertech website," http://www.espertech.com/esper/index.php, 2016.

[9] L. Gutiérrez-Madroñal, "Generación automática de casos en procesamiento de eventos con epl - authomatic generation of cases in event processing using epl," Ph.D. dissertation, University of Cadiz, 2017.

[10] L. Gutiérrez-Madroñal, I. Medina-Bulo, and J. Domínguez-Jiménez, "Iot–teg: Test event generator system," *Journal of Systems and Software*, 2017.

[11] M. A. Dobbs, S. Frixione, E. Laenen, K. Tollefson, H. Baer, E. Boos, B. Cox, R. Engel, W. Giele, J. Huston *et al.*, "Les houches guidebook to monte carlo generators for hadron collider physics," *arXiv preprint hep-ph/0403045*, 2004.

[12] M. L. Mangano and T. J. Stelzer, "Tools for the simulation of hard hadronic collisions," *Annu. Rev. Nucl. Part. Sci.*, vol. 55, pp. 555–588, 2005.

[13] IoT-Analytics, "White paper - iot platforms - the central backbone for the internet of things," https://iot-analytics.com/5-things-know-about-iot-platform/, 11-2015.

[14] Micro_Research_Finland_Oy, "Event generator," http://www.mrf.fi/index.php, 2016.

[15] Starcom_Systems, "The event generator," https://www.starcomsystems.com, 2016.

[16] Oracle, "Introducing the weblogic integration administration console," https://docs.oracle.com, 2016.

[17] European Commission, "New device vigi'fall will make growing older safer thanks to eu funding - horizon 2020 - european commission," https://goo.gl/6jLwsp, 2013.

[18] R. Igual, C. Medrano, and I. Plaza, "Challenges, issues and trends in fall detection systems," *Biomedical engineering online*, vol. 12, p. 66, 2013.

[19] Q. Li, J. A. Stankovic, M. A. Hanson, A. T. Barth, J. Lach, and G. Zhou, *Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information*. In Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on (pp. 138-143): IEEE, 2009.

[20] H. Gjoreski, A. Rashkovska, S. Kozina, M. Lustrek, and M. Gams, "Telehealth using ecg sensor and accelerometer," in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2014, pp. 270–274.

[21] S. Kozina, H. Gjoreski, M. Gams, and M. Luštrek, "Efficient activity recognition and fall detection using accelerometers," vol. 386, pp. 13–23.

[22] M. Lüder, G. Bieber, and R. Salomon, "Sturzerkennung mittels Luftdruck- und Beschleunigungssensorik Air Pressure- and Acceleration-Based Fall Detector: 2. Deutscher AAL-Kongress mit Ausstellung - Technologien, Anwendungen ; 27. - 28. Januar 2009 in Berlin; Tagungsbandbeiträge," 2009.

[23] O. Etzion and P. Niblett, *Event Processing in Action*, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2010.

[24] N. Pannurat, S. Thiemjarus, and E. Nantajeewarawat, "Automatic fall monitoring: A review," *Sensors*, vol. 14, no. 7, pp. 12 900–12 936, 2014.