Test event generation for a fall-detection IoT system

Lorena Gutiérrez-Madróñal†, Luigi La Blunda‡, Matthias F. Wagner‡ and Inmaculada Medina-Bulo*

†UCASE Software Engineering Research Group, University of Cádiz, Av. Universidad de Cádiz, 10, 11519 Puerto Real, Spain Email: {lorena.gutierrez, inmaculada.medina}@uca.es

‡WSN and IOT Research Group, Frankfurt University of Applied Sciences, Nibelungenplatz 1, 60318 Frankfurt am Main, Germany Email: {l.lablunda, mfwagner}@fb2.fra-uas.de

Abstract—The Internet of Things (IoT) is a popular paradigm which has been applied to different areas such as smart cities, medicine or business processes. One of the main drawbacks of an IoT system is the amount of information it has to manage and to monitor. This information arrives as events that need to be processed in real-time in order to make correct decisions. The Event Processing Languages (EPLs) were designed to handle this information by defining event patterns which describe relevant situations to be detected and filter the information. In the majority of the relevant situations to detect, the events have a specific behaviour which must be analysed not only to define the event patterns but also to simulate them to test the IoT system. Moreover, in several situations it is quite difficult to obtain test events with specific values: adverse environment conditions, rise or fall in blood pressure, heart attack, falls, among others. In this paper we introduce a complete study of falls as relevant situations; we show an analysis of the fall-involved events of two types of falls based on an IoT prototype, the event patterns to detect the falls and their test using the IoT-TEG (IoT - Test Event Generator) tool. The fall analysis has highlighted the necessity to improve IoT-TEG with a new functionality which allows defining the desired conduct 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. 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 caused 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]. Jean-Eric Lundy, an emergency doctor at the hospital Cochin in Paris 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]. Additionally, people affected by Dementia and Parkinson have a higher risk to fall. In accordance with [4] 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 1). 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) [5], [6].

![Fig. 1. Scheme fall-simulation where a person has a wearable sensor network in form of a belt. [5], [6]](image)

The EPLs have been designed to address the main problems of IoT systems. In particular, EPLs are used to define critical situations in order to filter the information and to make correct decisions according to the obtained data. These critical situations are defined using event patterns and event rules. Among the existing EPLs, the EPL of EsperTech [7] is used the most often. In our study, the critical situations are the falls, so the obtained data from the mentioned prototype are used to define an EPL of EsperTech pattern to detect falls.

The first step of this prototype is to identify a fall from a fast movement, a sitting or laying down movement, but the final goal of the system is to predict 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 different type of falls can be found, and it is necessary to identify all of them in order to tell them apart from a non-fall: in this study, two types of falls will be analysed.

IoT-TEG [8], [9] is a tool which automatically generates test events of many types. Thanks to the obtained data from the sensors we have checked that the measured parameter during a fall, the acceleration, has a 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. In this study of falls as relevant situations, our contributions are:

- **A study of the fall detection prototype evolution:** the system which is been used is continuously being improved. We describe the evolution of its architecture, how the data is analysed and the detected problems.
- **An analysis of the major 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 types of falls.
• New definitions of two types of falls: after the analysis of the acceleration during two types of falls a new definition for each one has been done. The obtained data of the fall-detection prototype is used to define EPL of EsperTech patterns to detect those types 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.

The rest 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 and EPLs. A brief introduction of IoT-TEG and its new functionality appear in Section IV. The architecture of the fall-detection system and the fall analysis are introduced in Section V. Section VI describes the improvements on the prototype, a new fall type analysis, a comparison of the obtained results and some detected problems. Finally, in Section VII we conclude our paper and make recommendations for future work.

II. RELATED WORK

An overview about event generators reveals that the first event generators [10], [11] were focused on extremely specific topics such as environmental conditions for the simulation of high energy physics events at particle colliders. Nowadays, people and business need to control and monitor the things around them in order to make decisions and to act according to the received information. 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 a real 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] allow 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 are not 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 [8], [9] includes this option.

Talking about fall-detection, there are several solutions that propose wearable sensors. We are going to focus our attention on the approach proposed by Li et al. [17]: their wearable fall-detection solutions, which will be explained subsequently, 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, laying 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.

To decrease the computational effort of the microcontroller a three-phase algorithm was proposed, and its phases are:

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 current position corresponds to laying.
3) Phase state transition analysis: if in laying 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 the activity jumping into bed and the fall type falling against a wall with seated posture.

III. BACKGROUND

A. Fall analysis

The approach used to detect falls is based on the basic idea proposed by [18], [19]. 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 (9.81m/s²) and during falling around 0g (0m/s²). 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. Important to know is that the acceleration magnitude which is illustrated in the Formula (1) was used for the fall-detection:

\[
\alpha = \sqrt{\alpha_x^2 + \alpha_y^2 + \alpha_z^2}
\]  

(1)

\[ \alpha_{max} - \alpha_{min} > 1g \]  

(2)

**B. Fall patterns with EPL of EsperTech**

Etzion and Niblett [21] defined event processing agents as software modules that process events. Such agents are specified using an EPL, and there are a number of styles of EPLs in use. The following styles are included: rule-oriented, imperative and stream-oriented.

The EPL our work is based on is EPL of Esper [7], a stream-oriented language. The main reasons of its selection are: it is an extension of Structured Query Language (SQL), it can be embedded into Java applications and it is open source.

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 tmp1, B as tmp2 from
pattern [every tmp1.temperature > 20 ->
tmp2.temperature > 20]
```

In the above example (see Example 1) of a “bacteria control system”, its temperature gauges take a reading of the 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 (20 degrees Celsius). This is a situation where a quick reaction to emerging patterns is needed in a stream of data events. A quick reaction is also needed in fall detection.

Because of the difficulties to simulate falls, IoT-TEG is used in order to get the fall-test events automatically. 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.

**IV. IOT TEST EVENT GENERATOR**

IoT-TEG [8], [9] is a Java-based tool which takes an event type definition file and a desired output format. 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.

**Example 2. Fall event type definition**

```
<event name="FallEventType">
<field>custom_behaviour</field>
</event>
```

Previous studies suggested there were no differences in testing effectiveness between using events generated by IoT-TEG, or events recorded from various case studies [8], [9]. Moreover, thanks to its implementation, IoT-TEG can be used to do different types of tests: functional, negative, integration, stress, etc; indeed an example of its usability can be found in [8], [22], where IoT-TEG has been used to apply mutation testing [23].

An event type is defined using XML language and its attributes have to be defined using the `<field>` element. A new property in IoT-TEG has been defined, the custom Behaviour, which is included in the `<field>` element to define the desired behaviour of any event attribute. In the `custom behavioural` property the path to the file that includes the behaviour of the event attribute has to be written.

For the sake of clarity, Example 2 shows the complete fall event definition (FallEventTime) that we are going to use. The fall event type has one attribute, the acceleration, which follows an specific behavior depending on the type of fall.
Example 3. Rules to define a fall

```xml
<xml version="1.0" encoding="UTF-8"/>
<custom_conditions simulations="5"/>
:variables>
  <variable name="Base" value="9.81"/>
</variables>
<rules>
  <rule weight="1" min="$(Base)*2" max="$(Base)*6"/>
  <rule weight="1" min="0.0" max="$(Base)/5"/>
  <rule weight="0" min="$(Base) / 9.81" max="$(Base) / 9.81"/>
</rules>
</custom_conditions>
```

Example 4. Fall pattern

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

...
future this query will be extended to four sensor nodes. The four node architecture (see Figure 5) is currently only used for redundancy purposes. With the select statement the event properties are selected to create a pattern for fall-detection. In the given example the following event properties are selected:

- a1.accelS1: starting acceleration value of node 1.
- a2.accelS1: subsequent acceleration value of node 1.
- a1.accelS2: starting acceleration value of node 2.
- a2.accelS2: subsequent acceleration value of node 2.

Taking into consideration the selected event properties the query checks if the starting acceleration of sensor node 1 is \( \leq 9.81 \text{ m/s}^2 \) which means the person is in a stationary position in which the earth's gravity of \( 1g \) (9.81 m/s\(^2\)) acts on the body. Additionally, the subsequent acceleration of the first node checks if the subtraction of the subsequent acceleration and the first acceleration within a time window of 1 second is \( \geq 9.81 \text{ m/s}^2 \) which means that the patient has suffered an impact to the ground. Using the OR disjunction the second sensor node can be added and the statement is able to detect a fall in case one of the nodes matches the EPL query and the values of the acceleration correspond to the same person.

**B. Fall simulation test events**

Our goal is to study the acceleration behaviour during a fall in order to generate test events. The selected fall consists on rolling in the bed and fall (RBF) [17], [26]. In this study we have used a healthy subject, and we have recorded falls with all possible realism while also trying to avoid risks. The person has been doing RBF type falls for a period of 2 minutes. The analysed data and videos can be found in [27]. In the following lines the steps to simulate the fall with the generated events are described, theses steps are very similar to the ones followed by [28], [29].

1. **Study of the values:** Given that the sensor 1 is the one that suffers the impact, its acceleration values are the first to be analysed. Figure 6 shows the normalised acceleration data \( (N(m/s^2)) \) from sensor 1 while the person was falling. The normalised acceleration is shown on the Y-axis, and the time in milliseconds (ms) on the X-axis.

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

The obtained data do not have a timestamp, so given that every 10 milliseconds the system sends the data, we have divided the values according to this time. To compare the RBF falls the time in the Table II starts in 0 ms and then increase in 10 ms, that is what it is shown in the first column of the Table. The second column shows the acceleration values \( (m/s^2) \) and the normalised acceleration values are deployed in the third column \( (N(m/s^2)) \) according to the fall maximum value. In Figure 7 a comparison of the normalised acceleration behaviour during the previous RBF falls is shown. This comparison shows 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 Formula (2).

![Fig. 6. Sensor 1 acceleration.](image-url)

**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. We have considered a peak when the normalised acceleration is greater than 0.7 \( (N(m/s^2)) > 0.7 \). Because of 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, with respect to the X-axis [210, 360] (Fall 1) and [900, 1050] (Fall 2). The range of extracted values are a set of data that happen in less than a time window of 1 second, to meet the fact described in [29]. So as to compare both RBF falls the acceleration values are normalised according to their impact value. The Table II shows the values to analyse where the impact value is highlighted with colour green.

<table>
<thead>
<tr>
<th>TIME (s.ms)</th>
<th>ACCEL. (m/s^2)</th>
<th>N. ACCEL. (N(m/s^2))</th>
<th>TIME (s.ms)</th>
<th>ACCEL. (m/s^2)</th>
<th>N. ACCEL. (N(m/s^2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6.29</td>
<td>0.02</td>
<td>0</td>
<td>5.98</td>
<td>0.08</td>
</tr>
<tr>
<td>10</td>
<td>7.57</td>
<td>0.05</td>
<td>10</td>
<td>6.31</td>
<td>0.08</td>
</tr>
<tr>
<td>20</td>
<td>20.7</td>
<td>0.33</td>
<td>20</td>
<td>8.2</td>
<td>0.13</td>
</tr>
<tr>
<td>30</td>
<td>24.07</td>
<td>0.34</td>
<td>30</td>
<td>9.21</td>
<td>0.16</td>
</tr>
<tr>
<td>40</td>
<td>21.01</td>
<td>0.34</td>
<td>40</td>
<td>19.92</td>
<td>0.43</td>
</tr>
<tr>
<td>50</td>
<td>10.81</td>
<td>0.12</td>
<td>50</td>
<td>21.8</td>
<td>0.04</td>
</tr>
<tr>
<td>60</td>
<td>7.71</td>
<td>0.05</td>
<td>60</td>
<td>16.52</td>
<td>0.34</td>
</tr>
<tr>
<td>70</td>
<td>6.87</td>
<td>0.04</td>
<td>70</td>
<td>14.41</td>
<td>0.29</td>
</tr>
<tr>
<td>80</td>
<td>7.06</td>
<td>0.04</td>
<td>80</td>
<td>9.97</td>
<td>0.18</td>
</tr>
<tr>
<td>90</td>
<td>5.23</td>
<td>0.00</td>
<td>90</td>
<td>6.54</td>
<td>0.09</td>
</tr>
<tr>
<td>100</td>
<td>51.58</td>
<td>0.00</td>
<td>100</td>
<td>3.01</td>
<td>0.00</td>
</tr>
<tr>
<td>110</td>
<td>11.01</td>
<td>0.12</td>
<td>110</td>
<td>42.34</td>
<td>0.00</td>
</tr>
<tr>
<td>120</td>
<td>12.12</td>
<td>0.15</td>
<td>120</td>
<td>8.96</td>
<td>0.15</td>
</tr>
<tr>
<td>130</td>
<td>11.77</td>
<td>0.14</td>
<td>130</td>
<td>10.14</td>
<td>0.18</td>
</tr>
<tr>
<td>140</td>
<td>11.26</td>
<td>0.13</td>
<td>140</td>
<td>8.91</td>
<td>0.17</td>
</tr>
<tr>
<td>150</td>
<td>11.16</td>
<td>0.13</td>
<td>150</td>
<td>10.45</td>
<td>0.19</td>
</tr>
</tbody>
</table>

**Table II**

**RBF FALL ACCELERATION, FALL (RANGE) 1 AND FALL (RANGE) 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 ($\alpha_{\text{max}}$), the acceleration behaviour during a RBF fall develops as follows:

1) from a value less than the half of $\alpha_{\text{max}}$, the acceleration value increases to obtain a value in the range: 
\[\alpha_{\text{max}}/2 - 0.5, \alpha_{\text{max}}/2 + 0.5\]

The person is rolling on the bed.

2) if the acceleration obtains a value in the previous range, its value decreases until the minimum value $\alpha_{\text{min}}$. The person is falling, a free fall.

3) the acceleration value goes from the minimum value $\alpha_{\text{min}}$ to the maximum value $\alpha_{\text{max}}$. The impact.

4) the acceleration value is established with values around the half of $\alpha_{\text{max}}$. The person is laying 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.

Taking into account the architecture of the prototype and the previous rules to define a RBF fall, in Example 5 the EPL of EsperTech pattern to the define a RBF fall is shown.

Example 5. RBF pattern

```xml
<variables>
<variable name="Roll" min="0.0" max="(Roll)/5"/>
<variable name="Fall" min="0.0" max="(Roll)/5"/>
</variables>
<rules>
<rule weight="0.25" min="$\text{Roll}/4"" max="$\text{Roll}/4" sequence="inc"/>
<rule weight="0.25" min="$\text{Roll}/4"" max="$\text{Fall}" sequence="dec"/>
<rule weight="1" value="$\text{Impact}"/>
<rule weight="0" min="$\text{Roll}/2-0.25" max="$\text{Roll}/2+0.25"/>
</rules>
</custom_conditions>
```

To define the acceleration behaviour during a RBF fall three variables are defined: Roll, Fall and Impact. Impact will be the maximum value $\alpha_{\text{max}}$, Fall will be the minimum value $\alpha_{\text{min}}$ 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.

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.

It has to be emphasised that to obtain these rules for the definition of the acceleration behaviour several tests have been done. Once we obtained the desired results, test events were generated as they were necessary. The Figure 8 shows the acceleration values of some of the generated RBF falls using IoT-TEG and the new functionality. These generated events can be used to test the fall-detection system.

![IoT-TEG generated RBF falls](image)

**VI. IMPROVED PROTOTYPE**

**A. Architecture**

Taking into consideration the architecture of the prototype described in Section IV significant improvements were done. The improved hardware architecture is also based on four sensor nodes, but with the following significant innovations:

- A new hardware platform which is based on Arduino Primo Core [30]. This microcontroller provides built-in sensors and a Bluetooth Low Energy (BLE) interface for wireless data transmission. This satisfies the requirement of patient compliance which includes unrestricted movement. Additionally, using BLE brings the advantage to
build up a communication infrastructure with the smartphone to automatically inform the emergency services without any additional hardware.

- A different power supply mode is used for the improved prototype. Instead of using the Lithium Polymer (LIPO) batteries, the Arduino Primo Core [30] is supplied by a coin cell. This is an important aspect due to the fact, that exposing the LIPO-batteries to permanent shocks will damage them and drastically shorten their lifespan. Additionally, we have reached a lifespan of 2 to 3 weeks without exchanging the battery.

- The dataset which is sent by the sensor nodes is composed as follows and includes the values of $\alpha_i$, $i \in x, y, z$ in unit $m/s^2$:

  SensorID, X-Acceleration, Y-Acceleration, Z-Acceleration

Compared to the dataset format of the previous prototype, the sensor identification number was added, and the significantly change is that only the single axis values of the accelerometer are sent. Based on the individual axis values of the accelerometer, the person’s orientation can also be determined.

### 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 on the knees and then on the chest: fall against wall (FAW). In this study we have used two healthy subjects, and we have recorded falls with all possible realism while also trying to avoid risks. They have been doing fall-tests for a period of 2 minutes, the FAW fall type. The analysed data and videos can be found in [27].

In this analysis the same steps that the ones described in Section [V] have been done:

1. **Study of the values:** Given that the sensor 1 is the one that suffers the impact, its acceleration values are the first to be analysed. The acceleration values are normalised ($N(m/s^2)$) and the impacts of the falls, peaks are 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.

2. **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 during FAW fall. The range of extracted values are a set of data that happen in less than a time window of 1 second, to meet the fall rule of [20] described in Formula [2]. The Table II shows the acceleration value during one FAW fall of person 1 and person 2.

<table>
<thead>
<tr>
<th>Time (s.ms)</th>
<th>Acceleration (m/s^2)</th>
<th>N. Acceleration (N(m/s^2))</th>
<th>Time (s.ms)</th>
<th>Acceleration (m/s^2)</th>
<th>N. Acceleration (N(m/s^2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>57.311</td>
<td>3.63</td>
<td>0.01</td>
<td>25.254</td>
<td>8.53</td>
<td>0.09</td>
</tr>
<tr>
<td>57.359</td>
<td>3.43</td>
<td>0.00</td>
<td>25.303</td>
<td>10.79</td>
<td>0.19</td>
</tr>
<tr>
<td>57.408</td>
<td>7.85</td>
<td>0.17</td>
<td>25.352</td>
<td>11.28</td>
<td>0.22</td>
</tr>
<tr>
<td>57.506</td>
<td>10.99</td>
<td>0.30</td>
<td>25.401</td>
<td>13.83</td>
<td>0.34</td>
</tr>
<tr>
<td>57.506</td>
<td>6.47</td>
<td>0.12</td>
<td>25.449</td>
<td>9.42</td>
<td>0.13</td>
</tr>
<tr>
<td>57.554</td>
<td>4.61</td>
<td>0.05</td>
<td>25.503</td>
<td>8.63</td>
<td>0.09</td>
</tr>
<tr>
<td>57.603</td>
<td>4.22</td>
<td>0.03</td>
<td>25.546</td>
<td>0.89</td>
<td>0.10</td>
</tr>
<tr>
<td>57.651</td>
<td>22.86</td>
<td>0.07</td>
<td>25.596</td>
<td>8.04</td>
<td>0.07</td>
</tr>
<tr>
<td>57.700</td>
<td>28.84</td>
<td>1.00</td>
<td>25.645</td>
<td>13.93</td>
<td>0.35</td>
</tr>
<tr>
<td>57.749</td>
<td>21.78</td>
<td>0.52</td>
<td>25.693</td>
<td>22.7</td>
<td>0.76</td>
</tr>
<tr>
<td>57.800</td>
<td>9.03</td>
<td>0.22</td>
<td>25.742</td>
<td>10.59</td>
<td>0.19</td>
</tr>
<tr>
<td>57.846</td>
<td>10.79</td>
<td>0.29</td>
<td>25.791</td>
<td>8.63</td>
<td>0.09</td>
</tr>
<tr>
<td>57.900</td>
<td>10.79</td>
<td>0.29</td>
<td>25.841</td>
<td>8.34</td>
<td>0.08</td>
</tr>
<tr>
<td>57.944</td>
<td>9.32</td>
<td>0.23</td>
<td>25.888</td>
<td>6.67</td>
<td>0.00</td>
</tr>
<tr>
<td>57.986</td>
<td>13.15</td>
<td>0.38</td>
<td>25.937</td>
<td>17.46</td>
<td>0.51</td>
</tr>
<tr>
<td>58.042</td>
<td>20.99</td>
<td>0.69</td>
<td>25.986</td>
<td>27.76</td>
<td>1.00</td>
</tr>
<tr>
<td>58.186</td>
<td>28.25</td>
<td>0.95</td>
<td>26.082</td>
<td>11.58</td>
<td>0.23</td>
</tr>
<tr>
<td>58.188</td>
<td>27.47</td>
<td>0.95</td>
<td>26.084</td>
<td>11.38</td>
<td>0.22</td>
</tr>
<tr>
<td>58.240</td>
<td>10</td>
<td>0.26</td>
<td>26.134</td>
<td>10.5</td>
<td>0.18</td>
</tr>
<tr>
<td>58.266</td>
<td>10.99</td>
<td>0.30</td>
<td>26.181</td>
<td>8.83</td>
<td>0.10</td>
</tr>
<tr>
<td>58.334</td>
<td>11.67</td>
<td>0.33</td>
<td>26.230</td>
<td>12.75</td>
<td>0.29</td>
</tr>
</tbody>
</table>

**TABLE II**

FAW FALL ACCELERATION, PERSON 1 AND 2

Fig. 9. Acceleration comparison during FAW fall.

For the FAW, we have decided to define the acceleration behaviour with normalised values; so the normalised acceleration behaviour during the FAW consists on:

1) the variation of its values while the person is walking. We have divided this rule in two rules:
   a) The normalised acceleration values increase in a range [0, 0.35].
   b) The normalised acceleration values decrease in a range [0, 0.35].

2) the normalised acceleration value while the impact of the person against a wall, has to be greater than 0.7.

3) the normalised acceleration values decreases to a range [0, 0.35].

The values of the acceleration are in the mentioned range depending on the size of the person; the larger person results in a longer range and if the person retains a position prior to a fall thanks to the wall. Moreover, a subtle peak could appear as a consequence of a rebound.

4) the normalised acceleration value when the person hit the ground, a second impact, has to be greater than 0.7.

5) finally, the person is laying on the ground and the normalised acceleration value decreases. The values of the acceleration are between [0.10, 0.35] and no subtle peaks 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.
Taking into account the architecture of the prototype and the previous rules to define a FAW fall, in Example 7 the EPL of EsperTech pattern to define a FAW fall is shown.

Example 7. FAW pattern

```
select a1.accelS1, a2.accelS1, a3.accelS1, a4.accelS1, a5.accelS1
from pattern(every(a1 = BodyEvent(a1.accelS1 >= 9.81) ->
a2 = BodyEvent(a2.accelS1 < 9.81) ->
a3 = BodyEvent(a3.accelS1 - a2.accelS1 >= 9.81) ->
a4 = BodyEvent(a4.accelS1 < 9.81) ->
a5 = BodyEvent(a5.accelS1 - a4.accelS1 >= 9.81))
where timer:within(1sec))
```

The EPL query checks if the starting acceleration \(a1.accelS1\) is \(\geq 9.81\) \(m/s^2\) which means that the person is walking. The next acceleration value \(a2.accelS1\) is used to detect the free fall phase to the wall. If \(a2.accelS1\) is \(< 9.81\) \(m/s^2\) it indicates that the person is falling against the wall. Additionally, if the difference between the current acceleration value \(a3.accelS1\) and the previous acceleration value \(a2.accelS1\) is \(\geq 9.81\) \(m/s^2\) it indicates the impact against the wall. After that, the person suffered the impact against a wall, the upcoming acceleration value \(a4.accelS1\) should be less than \(9.81\) \(m/s^2\). This indicates that the person is falling to the ground. If the difference between the successive value \(a5.accelS1\) and the previous value \(a4.accelS1\) is \(\geq 9.81\) \(m/s^2\) it means that the person has suffered an impact to the ground. To classify this event as a FAW fall this pattern flow should happen within a time window of 1 second.

3. To define the fall event: Once 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 [8], [9]. Given that the involved event attribute in this fall is the acceleration, the Example 2 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 8.

Example 8. Rules to define a FAW fall

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

<variables>
<variable name="Base" value="9.81"/>
<variable name="ImpactWall" min="$(Base)+$(Base)*0.7" max="$(Impact)*0.35" sequence="inc" />
<variable name="Impact" min="$(Base)+$(Base)*0.7" max="$(Impact)*0.35" sequence="dec" />
</variables>

<rules>
<rule weight="0.25" min="0" max="$(ImpactWall)" sequence="inc" />
<rule weight="0.25" min="0" max="$(ImpactWall)" sequence="dec" />
<rule weight="1" min="$(ImpactWall)" />
<rule weight="0.25" min="0" max="$(Impact)"/>
<rule weight="0" min="$(Impact)" />
<rule weight="0" min="$(Base)" max="$(Base)*0.10" />
</rules>
</custom_conditions>
```

To define the acceleration behaviour during a FAW fall three variables are defined: Base, ImpactWall and Impact. Given that the acceleration behaviour during a FAW fall has been according to the normalised values, the variables and rules have been defined according to that analysis. The acceleration value in a stationary position is a variable depending on the person, so we have considered the established value, \(1g \approx 9.81m/s^2\). That is the fixed value of the Base variable.

To determine the values of the impacts, we have taken into account that the normalised value has to be \(> 0.7\). So, to ensure that the impacts, ImpactWall and Impact, have a value meeting the mentioned condition the minimum value of the impact is \(Base+Base*0.7\), and the maximum value of the impacts is \(3g \approx 9.81 * 3 = Base+3\).

Given that we have considered to define the FAW fall behaviour rules according to the normalised values, the values to be generated depend on the maximum value. Due to this fact there are two values that can be the highest one, ImpactWall or Impact, the rules that depend on the maximum value contain the reference to the value, ImpactWall or Impact, according to the proximity of the rule. For instance, the first and second FAW fall rules contain a reference to ImpactWall, the impact in the wall (third rule), which happens after the person is falling, something described in the first and the second rules. The fourth and sixth rules contain a reference to Impact, the impact on the ground (fifth rule), which happens after the person is falling and the person is laying on the floor, fourth and sixth rules.

It is important to highlight that to obtain these rules to define the behaviour of the normalised acceleration several tests have been done. Once we obtained the desired results, test events were generated as they were necessary. The Figure 10 shows the acceleration values of some of the generated FAW falls using IoT-TEG and the new functionality.

Fig. 10. IoT-TEG generated FAW falls.

The improvements in the new prototype are encouraging. They affect not only in the way of obtaining the data, but also in the format and their values. The impact values from one analysis to the other are quite different, this is because of several reasons described in Section VI-C. The difference of values was not a problem for IoT-TEG to define the behaviour rules and to generate the test events. That means that the introduced functionality can be adapted to the analysed behaviour. Moreover, it has to be emphasised that the application of the new functionality covers any event attribute which follows a behaviour; so IoT-TEG is not limited.

C. Detected problems

After testing the current fall-detection prototype, some problems were found, and some considerations will be applied in future tests. First of all, we are going to explain the synchronisation problems of the prototype:

- Fall data from the beginning should always be discarded because the sensors start sending data at different times.
The Table III will be used to explain the problem. Let us say that in “fall 1”, there are more than 30 values that define the fall, and later in “fall 3”, there are 12 values that define the fall. 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 11; 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.

<table>
<thead>
<tr>
<th>TIME (s.m.s)</th>
<th>ACCEL. (m/s²)</th>
<th>N. ACCEL. (N (m/s²))</th>
<th>TIME (s.m.s)</th>
<th>ACCEL. (m/s²)</th>
<th>N. ACCEL. (N (m/s²))</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.561</td>
<td>9.81</td>
<td>0.11</td>
<td>25.546</td>
<td>8.73</td>
<td>0.1</td>
</tr>
<tr>
<td>25.561</td>
<td>9.81</td>
<td>0.11</td>
<td>25.596</td>
<td>8.04</td>
<td>0.07</td>
</tr>
<tr>
<td>25.562</td>
<td>11.38</td>
<td>0.17</td>
<td>25.645</td>
<td>13.93</td>
<td>0.35</td>
</tr>
<tr>
<td>25.564</td>
<td>11.38</td>
<td>0.17</td>
<td>25.693</td>
<td>22.76</td>
<td>0.76</td>
</tr>
<tr>
<td>25.609</td>
<td>15.11</td>
<td>0.33</td>
<td>25.742</td>
<td>10.59</td>
<td>0.19</td>
</tr>
<tr>
<td>25.610</td>
<td>15.11</td>
<td>0.33</td>
<td>25.791</td>
<td>8.63</td>
<td>0.09</td>
</tr>
<tr>
<td>25.610</td>
<td>15.11</td>
<td>0.33</td>
<td>25.841</td>
<td>8.34</td>
<td>0.08</td>
</tr>
<tr>
<td>25.611</td>
<td>15.11</td>
<td>0.33</td>
<td>25.888</td>
<td>6.67</td>
<td>0</td>
</tr>
<tr>
<td>25.612</td>
<td>31.4</td>
<td>1</td>
<td>25.937</td>
<td>17.46</td>
<td>0.51</td>
</tr>
<tr>
<td>25.612</td>
<td>23.84</td>
<td>0.69</td>
<td>25.986</td>
<td>27.76</td>
<td>1</td>
</tr>
<tr>
<td>25.657</td>
<td>23.84</td>
<td>0.69</td>
<td>26.082</td>
<td>11.58</td>
<td>0.23</td>
</tr>
<tr>
<td>25.657</td>
<td>23.84</td>
<td>0.69</td>
<td>26.084</td>
<td>11.38</td>
<td>0.22</td>
</tr>
</tbody>
</table>

**TABLE III**

**Synchronisation problem, Fall 1 and Fall 2**

![Fig. 11. Synchronisation problem; acceleration values during two FAW falls.](image)

The acceleration values show that in less than 250 milliseconds there are more than 30 values from “fall 1”, and in more than 350 milliseconds there are 12 values from “fall 2”. There is a lack of synchronisation not only in the amount of data, but also in the time.

- Some sensors transmit more data than the others, i.e. 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 did not transmit data in one moment of the simulation, see Figure 12.

![Fig. 12. Synchronisation problem; acceleration values from the four sensors during a FAW fall.](image)

If we focus our attention on the hardware, the battery life is also a factor that requires improvement. Nowadays the duration of the battery is around 2 or 3 weeks, it depends on its use. If we want to use this IoT system we have to increase the duration of the battery in order to avoid a permanent battery change, which is a drawback in terms of patient compliance.

The analysis has revealed that while we were studying the acceleration data tests, some of its values could be misinterpreted. This is because the test 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 test person should wait at least 2 seconds laying on the floor after the fall for a better fall simulation. In a real situation, if a person falls and stands up this 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 does not stand up this means that maybe the person is unconscious or is not able to move and call the emergency services. Therefore, waiting at least 2 seconds between falls in our test scenarios, it will help not only to understand better the behaviour of the acceleration values but also to do a better fall-simulation.

Comparing the measured values of the fall-events of the two prototypes, it can be determined that the measured impact values of the first prototype are considerably higher than those of the improved solution. This difference is caused by multiple reasons. The main reason is the different experimental set-up of the first prototype and the improved solution. The hardware design of the first prototype is bulky compared to the improved one. Due to the bulky construction, the sensor nodes are more exposed to vibrations and shocks, which results in higher impact values. Additionally, the sensor nodes shifted during the fall. The result was that the test person repositioned the sensor nodes which leads to influence the impact value. So, a smaller microcontroller platform, which was described in [VII] was used to solve this problem. Despite the design differences in the two prototypes, we are able to compare fall-types.
VII. CONCLUSIONS AND FUTURE WORK

We have illustrated part of the process of the development of an IoT system to detect falls. This process involves different types of testing, we have been using IoT-TEG [8, 9] to generate the test events in order to replicate the behaviour of the sample falls. The implemented functionality allows to generate events by defining rules which describe a desired behaviour. We can assign behaviour rules as many event attributes as the event type contains, and the values of each event attribute will follow the assigned behaviour.

According to the falls, RBF and FAW, we have detected the necessity to define the EPL query for each fall-type in the literature; this will help to identify a fall from a no-fall. Moreover, it will be interesting to add different types of sensors such as an air pressure sensor, and medical data such as ECG, in order to identify and characterize the falls.

The future work related to IoT-TEG [8, 9] is to improve it in order to generate faithful test events which simulate the relevant situations that need to be filtered and, sometimes, are very difficult to imitate: adverse environment conditions, rise or fall in blood pressure, heart attack, falls...

To generate test events, reliable data transmission of all sensor nodes must be guaranteed. The synchronisation problem can lead to data loss during data transmission and thus influences the analysis of falls. So, the use of a real-time system capable microcontroller platform is planned, which facilitates the synchronisation of the sensor nodes using a priority-controlled task scheduler.

Moreover, after studying the work [24], different post-fall postures will be analysed. Thus, new developments will include hazard analysis methods, e.g. STAMP/STPA, to satisfy safety standards, i.e. IEC61508, IEC60601, etc.

Finally, in our final product, a unique identification of the Human Body will be included, so a patient/user management could be added. Moreover, the final system will include security aspects inside the Body Area Network.

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 DARfOS TIN2015-65845-C3-3-R, and the Program of Development and Impulse of Research Activity of the University of Cadiz.

REFERENCES


