FP7-313161

A holistic, scenario-independent, situation-awareness and guidance system for sustaining the Active Evacuation Route for large crowds

RULE SET TOOL

Deliverable Identifier:	D7.5
Delivery Date:	Apr 30, 2015
Classification:	Public
Editor(s):	Jorge Rodríguez (Indra)
Document version:	1.0 – 2015
Contract Start Date:	April 1st, 2013
Duration:	48 months
Project coordinator:	EXODUS S.A. (Greece)
Partners:	EXO (GR), IT INNOVATION (UK), ICCS (GR), HKV (NL), TEL (GR), TEK (ES), AIA (GR), VITRO (IT), CDI (UK), INDRA (ES), KUL (BE), DXT (FR), POLITO (IT), STX-FR (FR), TUD (DE), TUC (DE), ASRS (ES), METB (ES), TIM (IT)

Project co-funded by the
European Commission under the
7th Framework Programme

Document Control Page

Title	Rule Set Tool	
Editors	Jorge J. Rodríguez	INDRA
Contributors	Pedro Garibi	INDRA
	Jorge J. Rodríguez	INDRA
	Estrella Ruiz	INDRA
	Sandro Viola	Vitrociset
Peer Reviewers	Jorge Berzosa	IK4-Tekniker
	Jose Miguel Landeta	IK4-Tekniker
	Dimitris Drakoulis	Telesto
	Athanassios Moralis	Telesto
Format	Text - Ms Word	
Language	en-UK	
Work-Package	WP7	
Deliverable number	D7.5	
Due Date of Delivery	30/04/2015	
Actual Date of Delivery	18/05/2015	
Dissemination Level	Public	
Rights	eVACUATE Consortium	
Audience	☒ public ☐ restricted ☐ internal	
Date	18/05/2015	
Revision	None	
Version	1.0	
Edited by	Jorge J. Rodríguez (INDRA)	
Status	☐ draft ☒ Consortium reviewed ☒ WP leader accepted ☒ Project coordinator accepted	

Revision History

Version	Date	Description and comments	Edited by
0.1	13/03/2015	First draft	J Rodríguez
0.2	13/04/2015	Vitrociset contribution	S Viola
0.3	14/04/2015	Typos, format	J Rodríguez
0.4	17/04/2015	Intro and last sections added	J Rodríguez
0.8	17/04/2015	First stable version	J Rodríguez
0.9	29/04/2015	Peer review included – IK4-Tekniker	J Rodríguez JM Landeta
1.0	13/05/2015	Peer review included – Telesto	J Rodríguez D Drakulis A Moralis
1.0	18/05/2015	Ready for submission	D. Petrantonakis

Table of Contents

List of Figures

Executive summary

The following document is part of one deliverable within the WP7 of the FP7 R&D project eVACUATE. Deliverable D7.5 Rule Set Tool includes both a software prototype and a report describing the technological achievement. The document in itself does not include the software as is, but it explains the prototype, including functionality, access to it, general guidelines for its use and screenshots. The reader will easily find references to follow and test the real, available implementation of the Rule Set Tool.

The Rule Set Tool implementation within the Smart Space resorts to Complex Event Processing technology. This may not be always straightforward intuitive, for this reason a section is included to describe the selected solution, the reasons leading to the selection, and the general context and architecture that in turn led us that way.

The reader needs to be familiar with a number of concepts, including for instance the SOFIA architecture, before delving in the specifics of the Rule Set Tool. Therefore, the explanations have been reduced to a minimum. Readers wanting more details can follow bibliographical references to access more comprehensive information.

The main focus of this deliverable is the software development, with the current document being but an explanatory addendum. This is why the document includes a detailed explanation of the application, including its use. The application was developed and adapted to fit eVACUATE's needs.

Chapter 5 of this document will deal with an alternate solution to the problem developed by the team. While the final implementation was the SOFIA-based CEP solution, this other option is considered interesting and well worthy of explaining.

The conclusions of the document will detail issues such as the impact and the next steps to be taken in the following months of the project.

1. Introduction

This document deals with the Rule Set Tool of the eVACUATE Project. The Rule Set Tool is software integrated in the Smart Space, which explains why it is within in the scope of **Work Package 7 Smart Spaces.**

The Rule Set Tool in itself will be used by the eVACUATE system to *allow the local control of the elements within the Smart Space based on the information collected from it.*[i] It is, and should be, a computer program.

Considering that the current implementation of the Smart Space as of Month 25 of the project has a strong influence from SOFIA2, the team decided that the best implementation of the Rule Set Tool should be based on a Complex Event Processing (CEP) compatible with SOFIA2.

An interface was also developed, adapted to the specifics of eVACUATE. It is the goal of the developers for the interface to be used by qualified, authorized people in the eVACUATE environment (e.g., security managers).

The availability is also achieved thanks to Vitrociset's contribution, which also provides flexibility to the project's technology.

The project's Description of Work initially stated that Vitrociset was the leader of the Task generating this deliverable. However, the leadership was changed to Indra, by Vitrociset's request and common agreement, during 2014.[ii]

1.1 Scope

D7.5 Rule Set Tool is not a report-type deliverable. D7.5 includes both a textual report and a software prototype. The current document does not include the full prototype because it is an Internet-based online application - already available as of Month 25 of the project. The document does include information on how to reach the prototype, screenshots and other proof of its delivery, and a general guideline to describe its use.

References to the online application are not the only allusions that will be made in the current document. For instance, the document is not a thorough user guide of the software prototype. It will refer to other external sources when needed.

The bulk of this deliverable is the software. The textual report just intends to state the achieved attainments. To do so, it will refer other related documents.

This document is not the place to delve in details that are not relevant for our purposes, including access protocol, subscription methods and specific commands. SOFIA has been described and explained several times within the eVACUATE project and there is specialized documentation detailing its workings. If the reader is particularly interested, he is referred to the reference for more detail.[iii]

2 Background

The following section of this document will provide important information that, per the authors opinion, is relevant to understand the rest of the text. Due to this, it is believed that this section must be included at the beginning of the text.

The intended goal to provide a deliverable as self-containing as possible, meaning that it does not necessarily require previous comprehension of other deliverables in the eVACUATE project. Having said this, most of the information in the background section is related to other deliverables of the project and to documents that project partners may already be familiar with - e.g. meeting minutes, user guides, and documentation.

The section will be nonetheless concise, providing the specific information that is believed to be relevant for this document.

2.1 The Smart Space

The eVACUATE project agreed a definition of Smart Space, which is the following one:[iv]

> *A smart space is a physical area, with spatial limits, and reasonably small to be passed through on foot by a common human adult; where there are several smart and/or intelligent devices, including sensors and/or devices that can interact with the environment; and where these devices interact with each other in an intelligent, synergistic way.*

For purposes of the project implementation, several smart spaces will be available in each of the venues (for instance, in Athens International Airport) and in any test environment that may be required for the implementation. A smart space made of simply a few input and output devices with connectivity to the Internet and an available server may be considered. A smart space was created as a demonstrator, with a hardware including only two input sensors (a light sensor and two buttons representing turnstiles) and one output device (a fan).[v] [vi] [vii]

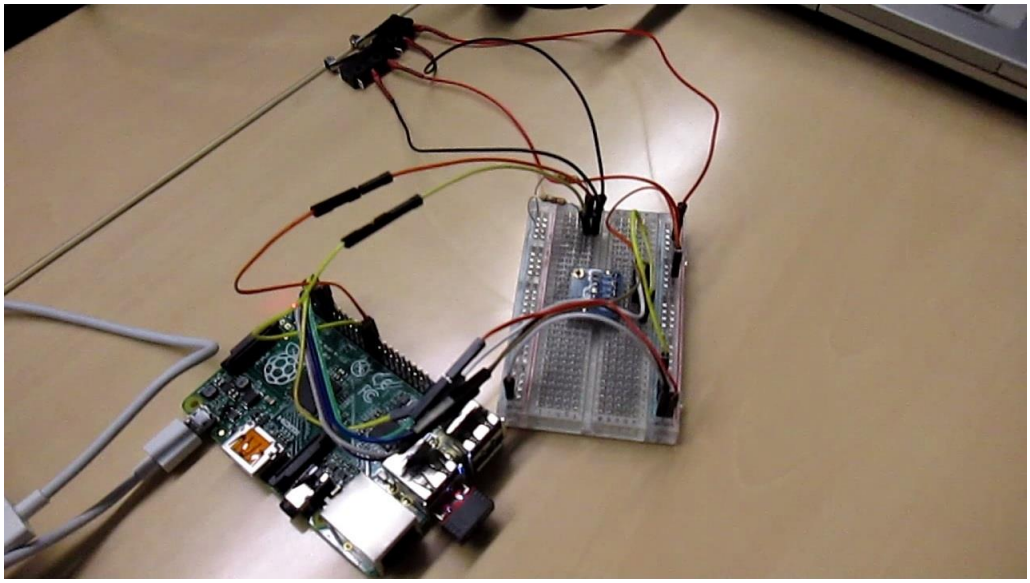

Figure 1: Demonstrator of Deliverable 9.1 The eVACUATE Integrated System release 1

From a more precise point of view, a bunch of hardware devices in a limited space do not necessarily correspond to a smart space because there is no intelligence ("smartness") to it: They do not interact with each other in any intelligent way. The implementation in eVACUATE includes software to allow for this to happen. The first step is providing each of the devices with an equivalent software representation, which is where the concept of agents is introduced.

An agent is an abstraction representing a real, physical entity that sends information to the system and/or receives information from it. The agent can communicate with the system's central brain, called SIB (see Section 2.3 SOFIA), commonly through the Internet.[viii] Thus, as can be seen, the intelligence of the system will not be necessarily in any of the visible devices.

As the agent is a software associated to a device, the device requires some processing capability, albeit limited. In practice, this is achieved by connecting the device to a computer that can control it and connect to the Internet for instructions. More specifically, Raspberry Pi was chosen to support the processing needs of the demonstrator.

2.2 How to Control the Smart Space

The first point to specify here is that the smart space is not a system to be externally controlled - due to its own nature (in this case, its design), the smart space can control itself and its devices in an intelligent way. It is our belief that an automatic control of the inputs, following a predefined set of rules, will be more efficient than a manual verification by human individuals - faster, less error-prone, etc. A critical system such as the product of eVACUATE relies on the quick validation of inputs to provide answers.

Obviously, a critical system also requires human interaction, where a person can perform validation checks.[ix] No-one intends to replace this part. This deliverable, however, will not deal with the human-machine interface because it focuses in the automatic rules.

The general idea of a rule, to be defined in more detail later, will be: The system has some rules in its memory, with each rule having a defined trigger function or trigger. This trigger is a specific combination of inputs and/or readings (at least one) that must happen simultaneously. The system listens to the devices to get their inputs and readings. Should any rule be triggered, then the system executes the events also included in the rule.

Notice that human beings can have their own agents. One agent can represent a security guard patrolling the smart space and manually sending an alert to the system; another agent can represent an inspector who is notified whenever there is a situation that requires specific answers.

From a technical point of view, the rules will not be enacted in any of the physical devices in the smart space, but in a server with more powerful processing resources - this way, the devices will only send a small amount of data, and the greatest computing is, logically speaking, centralized.

Consider the venues in real conditions and the potential connectivity outage, especially in situations where the system would be used - situation where, due to a disaster, the venue must be evacuated. Internet access is not always guaranteed in this case, or in situations such as a cruise ship in the open sea. This is one of the reasons why the server will not be completely remote - it will instead be a computer physically within the smart space, but functioning as a server in the architecture (There are other additional arguments for this design choice to do this, such as the need to avoid broadcasting any personal data through the public Internet thus avoiding potential security issues).

To do this, the eVACUATE system will use the platform known as SOFIA2 presented hereafter.

2.3 SOFIA

SOFIA (Smart Objects For Intelligent Applications) was a research & development project in the ARTEMIS programme. It involved 19 European partners working from 2009 to 2011.[x] The resulting technology was later evolved by industrial partners, including Indra Sistemas, S.A., to obtain SOFIA2, a more stable interoperability platform currently used in systems such as smart cities, home automation and NFC-based rental services.[xi]

SOFIA identifies a central processing part known as the Semantic Information Broker (SIB), which receives the data from the physical devices, and sends data to them. A lighter software functionality known as Knowledge Processor is incorporated in each Agent to provide communication with the SIB, adapted to the technological process.

More specifically, the KPs send semantic information (ontologies) about the situation of the smart space to the SIB. The SIB then processes this information and sends the output, including commands, to the KP's. The SIB can, on its own initiative, send information to a given KP; however it is more common that a KP subscribes to a specific kind of information and receives any event that fulfils its request.[xii] [xiii]

For instance, consider there is an event named "evacuation alarm". This event is created whenever any of several triggers happen (Trigger 1: A security guard presses a specific button. Trigger 2: There is a combination of inputs from sensors, such as low visibility and high temperature in a given area). Consider there is a LED sign that should be switched on whenever the event "evacuation alarm" happens, no matter how it was created. The KP associated to this LED sign device is *subscribed* to "evacuation alarm" events. The event will be generated in the SIB; and the Agent will also include other information, for instance what to do exactly whenever the event is received - notice that, for instance, a loudspeaker device can receive the same event, but the specific actions associated to that event reception will be different; see **Section 3.1 The Rules** for more.[xiv]

3 Description

As explained in **Section 1 Introduction** of this same document, most of the work in Task 7.5 had to do with the development of the tool to introduce rules in the Smart Space. The current document explains both the work, and the reasons that led to choose some functionality to develop. However it is not meant to be a user guide

We will see more detail in the following sub-sections.

3.1 The Rules

Following the Description of Work document, the eVACUATE smart space must have some kind of tool so that the smart space elements will be controlled based on the information produced by the smart space itself.[xv] Furthermore, again according to the DoW, this will be an asset for the Decision Support, but not necessarily a part of it, and will allow the control of the smart space's elements through the agents.

The tool that will be described in section 4 fits this description.

Let us delve a little bit in the definition of what is a rule. This document's **Section 2.2 How To Control The Smart Space** described the rule as having a trigger and an output. Later, in **Section 2.3 SOFIA**, we saw an example in which the same rule may have different effects in different devices of the smart space. Consider that a rule is triggered and it generates an output event "evacuation alert". This event is sent to two output devices in the smart space: A LED sign and a loudspeaker. The LED sign has a protocol whenever this event is received: The sign must switch on. The loudspeaker has a *different* protocol: It must play a specific sound file. This is one single event where two devices provide different outputs.

Now we will go to the bigger picture: The whole evacuation system. For a given scenario, the system should have a full protocol with some details, including something like "If there is low light and high temperature, then an evacuation alert is created. If the security guard presses the button, then an evacuation alert is created. If there is an evacuation alert, then the LED sign must be switched on and the loudspeaker must play file Evacuationalert.mp3". This protocol is correct and not all of it falls within the Rule Set Layer. It will be divided in several parts:

- **WP8 Decision Making and Optimal "Situation-aware" evacuation strategy** and **WP9 – SAES (Situation Awareness and Evacuation System): Framework design and system Integration** deal with this kind of sentence as a whole. They define how the system must react as a whole. Of course these WP's will be based on the output of **WP2 Scenarios Definition, User/System Requirements.**
- The technical details on how the SIB receives the inputs from the physical space (how the SIB knows that the button has been pressed) are already explained.
- The Rule itself, from this point of view, deals with how the alert is created - meaning it stops as soon as the SIB creates (and broadcasts) an alert.

- WP8 and WP9 define which output devices should be notified, and what they should do.
- For those devices to receive the specific alerts affecting their behaviour, WP9 deals with the implementation of their subscription to alerts.
- Lastly, the Agents in the Smart Space, WP7's scope, specifically in Task 7.2, implement the specific behaviour of a given device whenever it receives an event.

3.2 Implementation Approach

This task has specific goals, mainly the creation of a tool to include rules in the eVACUATE smart space. There are limits on the activity that will be performed here.

The venues (Athens International Airport, STX France SA, Real Sociedad de Fútbol S.A.D. and Metro Bilbao S.A.) are the experts in their own facilities. They will define the specific rules applying to each of them. Task 7.5 will not delve in rules specific to any of the venues that may not apply in the others. The definition of all the rules for a given scenario is out of the scope of Task 7.5

Indeed, Task 7.5 is interested in creating the tool and demonstrating its use, *not* in massively using the tool to insert a lot of rules. Only a few rules will be inserted to be used in the examples.

A technical detail related to SOFIA2 requires that each of the devices has an associated ontology that describes the item and all the associated information (for instance, a thermometer will have an ontology with an attribute for the temperature). This is not something that Task 7.5 describes in detail, but it will mostly assume that ontologies already exist - because this is a closed issue in the eVACUATE project, something already available. Please refer to the SOFIA2 web console use guide if interested in this.[xvi]

The current document is, as requested by the Description of Work, a report on the developments made of Task 7.5. It is not a use manual for the Rule Set Tool. The software has already available multi-language use manuals.[xvii]

3.3 Research and Innovation

The work in this task was aimed toward research and innovation.

More specifically the primary aim is the development of a specialized tool to insert (and indeed to CRUD[xviii]) rules intended to evolve an already innovative platform, going beyond the state of the art to maximize usability and adapt to the current environments.

Harmonizing with this, the innovation was directed specifically toward the needs of eVACUATE, so that it can be helpful in the concrete venues where the pilots will be made. Improvements were made to adapt to the expected functionalities and the informed requirements. By doing this, the team has also advanced the eVACUATE project's objectives.

One should bear in mind that this development will also be available for third parties, due to the open-source nature and philosophy of SOFIA2.

The implementations will also be used as a reference in the near future, improving the dissemination of the R&D European project. The partners involved in this task intend to flaunt this development in fora where it will participate.

4 The Rule Set Tool

The following section will describe the main Rule Set Tool, developed by Indra Sistemas and totally adapted to and integrated within the SOFIA infrastructure and interoperability platform.

The Rule Set Tool as described in the following pages is currently (as of March 2015) operative and in use in real conditions for purposes of developing eVACUATE pilots.

4.1 Philosophy

As previously explained (see e.g. **Section 3.1 The Rules**), the SOFIA platform is distributed. The KP's are physically in the smart space, with some of them sending the information they generate to the SIB. Once that information reaches the SIB, the SIB will verify if any rule is triggered.

In its full extent, the SOFIA platform has a number of logic modules:

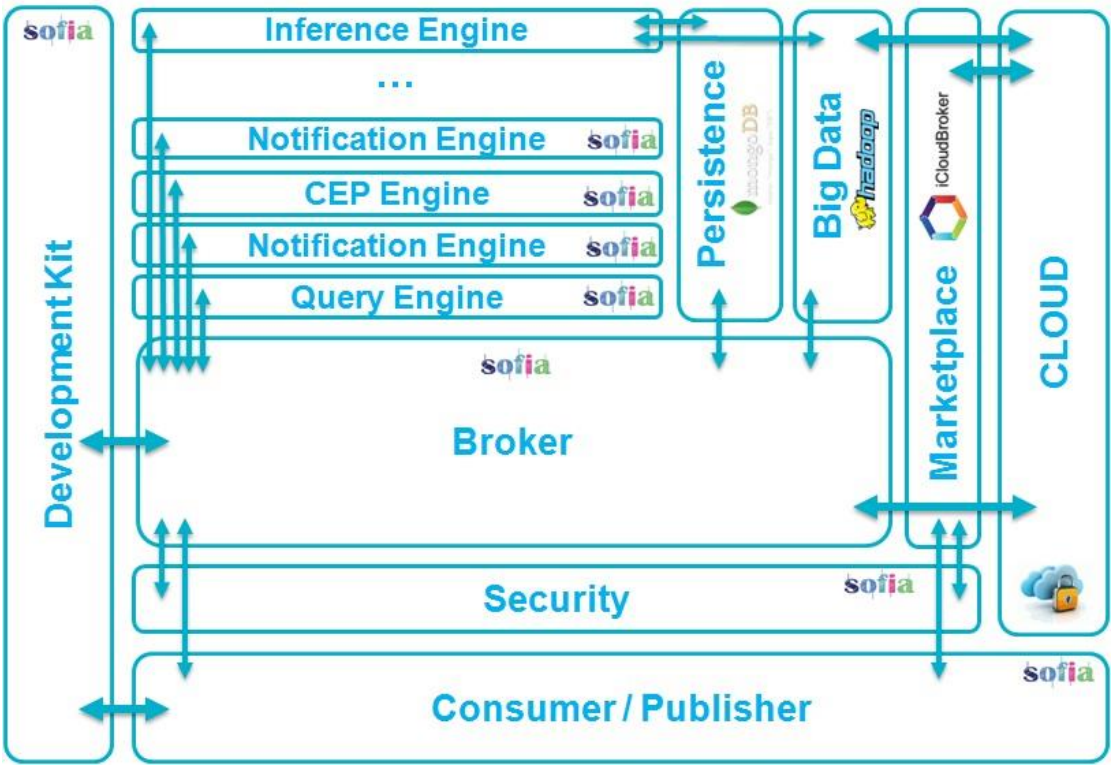

Figure 2: SOFIA platform's logic modules

One of these is the CEP Engine, which will be used for this purpose. To check the conditions of the rules, the CEP Engine will have to evaluate the information of *all* the instances of one or more ontologies, *and* their respective timestamps. One important advantage of the CEP Engine is that it can store previous information and include this in the comparison. For instance, a rule may be triggered if the temperature

reading goes beyond a threshold, say, four times within ten minutes, considering that there is a temperature reading every second. This kind of time-sensitive condition can be incorporated in a more complex rule: The rule can be triggered if the temperature has gone beyond a threshold T and the light has gone beyond a threshold L *within* the same ten minutes - even if at no point both conditions have happened at the same time.

To do this, the rules will be written by a human user. The user will first define an event working with input data and providing an output. This rule may have several parameters, but the inputs are expected to be instances of ontologies that are sent by the KP's or, theoretically, the output of another rule.

This kind of rule has an output whenever the input condition is fulfilled ("true") - but the Rule Set Tool (RST) allows the user to provide a different output whenever the condition is *not* fulfilled ("false"), that is to say, an "else" line for our "if" construct. The "else" line is optional, albeit the output (the "then" line) is mandatory.

Each rule must have a unique identifier. This does not only provide a technological control for the system, but it also helps the users to keep track of their own work.

4.2 Operation

The RST is available from a web-based console, so that a graphic interface is provided. To access the web console, start by going to **http://sofia2.com/console**

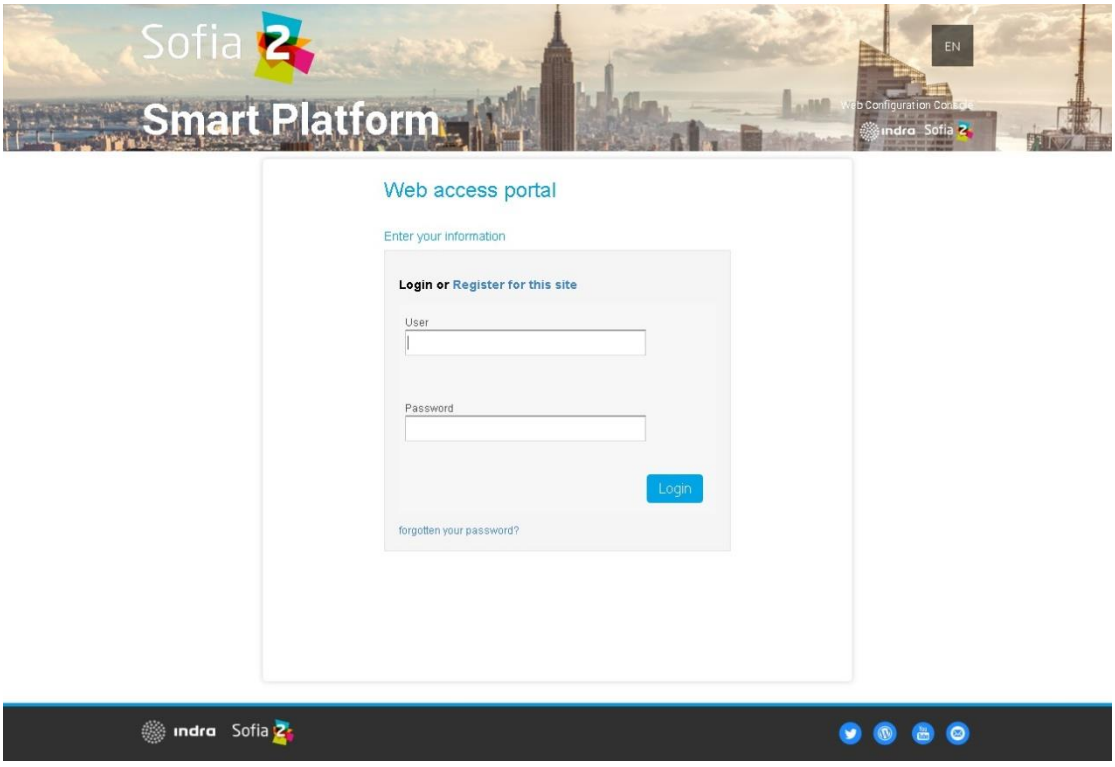

Figure 3: SOFIA2 console

As you can see, one of the first requirements will be having a valid user, particularly one with the role COLABORADOR.[xix]

The process includes several steps, which will be looping depending on how many devices are required for a given rule, and whether they have been defined beforehand.

Firstly, the user must define the ontology of the device she wants to use - with the data that the device sends and/or receives, such as the temperature and geographical position of a thermometer, etc. To do this, the user will have to write the ontology; the system will offer JSON templates that will make this process easier.

Then, the user must create a KP for each ontology. The KP will be the connection between the device and the SIB. This is not related to the Agent. The KP definition is done through a graphic interface, where the user can simply select available ontologies on boxes.

Those steps were a pre-requisite to create a rule. If a user wants to use the same devices for a different rule, he/she does not have to create ontologies and KP for those devices again.

The user must create an Event, using the graphic interface accessible through the button aptly named “New Event”. An event is not a rule; the difference will be further defined at the end of this section. The user must use an identifier to register the Event with a name, then select an ontology, and then access to the attributes of the ontology - a button “Load attributes” is included, because the development team did not feel that asynchronous web development technologies were really needed. Once the user can see the attributes of the ontology, she can select those attributes she wants to use.

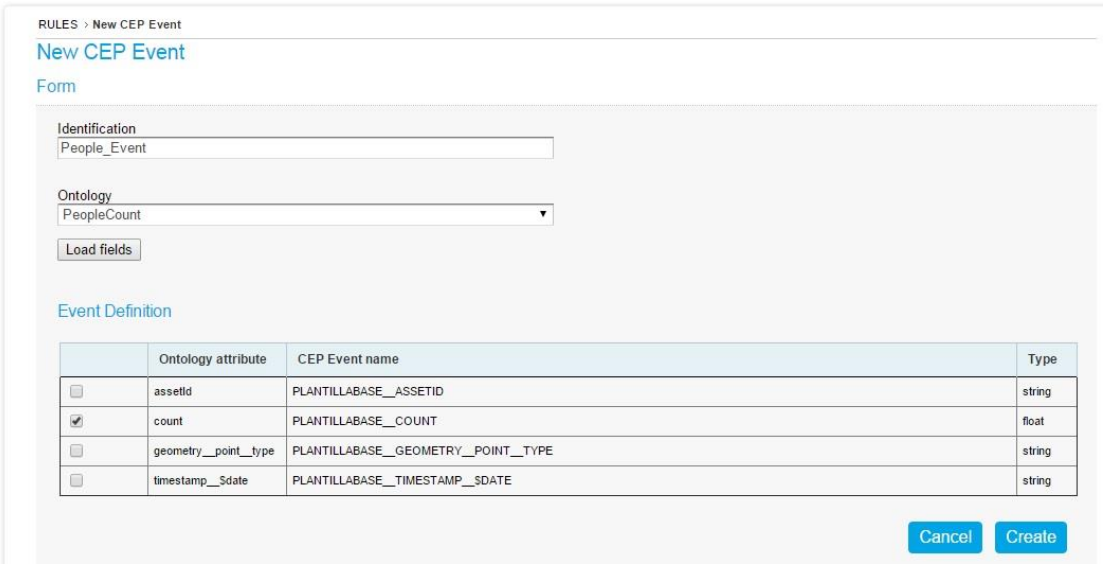

Figure 4: Creating a CEP Event; notice the “Load Fields” button

Now that we have an event, we can create a CEP Rule, again using graphic menus. The Rule has mostly three boxes with names that will be familiar to those users knowledgeable in SQL: Select, From, Insert Into. The text of the rule will be written,

but the system and user guide provides examples on how to do this so that the user, even if she is not an expert in computer sciences, can define the rule.

The RST has a number of other functionalities that can be useful, such as the Scripts, in easy Groovy language, a Test Scenario, a *visualizer* of Process Status so that the user can check if a rule has been triggered in some conditions, and a Length Window to specify that a rule must recover not only the last received event but the last N events that have been received, and use them in the check process.

4.3 Example

The CEP Guide provides a useful example of work flow to create a rule.[xx] We will now see it at a glance; refer to the said document for details on how to do so.

The rule is based on three devices: A light sensor, a fan and a counter of people. The light sensor provides a reading of lux; the counter of people specifies the number of people in a given area; and the fan is supposed to turn on if there are more than 10 people and less than 30 lux at the same time. This is the rule.

The user first logs in the system and verifies that she has a role allowing the creation of rules by checking her privileges. If that user cannot create rules, the procedure is followed to get a role with that privilege by contacting an administrator.

Then the ontology creation is described. The system offers JSON templates that make this process easier. One of the templates is selected and the user defines the desired attributes. The ontology is then automatically translated to a JSON Schema. This translation requires specific definitions, but the system provides useful help for this.


```
"PlantillaBase": {
    "timestamp": {
      "$date": "2014-09-17T09:15:04.423Z"
    },
    "assetId": "light_sensor",
    "luminosidad": {
      "ambient": 9075,
      "ir": 1166,
      "lux": 255.4330564642427,
      "low": false
    },
    "geometry": {
      "type": "Point",
      "coordinates": [
        40.291843,
        -3.74271
      ]
    }
  }
```



```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PlantillaBase Schema",
  "type": "object",
  "required": [
    "PlantillaBase"
  ],
  "properties": {
    "PlantillaBase": {
      "type": "string",
      "$ref": "#/datos"
    }
  },
  "datos": {
    "description": "Info Plantila Base",
    "type": "object",
    "required": [
      "timestamp",
      "assetId"
    ],
    "properties": {
      "timestamp": {
        "type": "object",
        "required": [
          "$date"
        ],
        "properties": {
          "$date": {
            "type": "string",
            "format": "date-time"
          }
        },
        "additionalProperties": false
      },
      "assetId": {
        "type": "string"
      },
      "luminosidad": {
        "type": "object",
        "properties": {
          "ambient": {
            "type": "number"
          },
          "ir": {
            "type": "number"
          },
          "lux": {
            "type": "number"
          },
          "low": {
            "type": "boolean"
          }
        }
      },
      "geometry": {
        "type": "object",
        "properties": {
          "point": {
            "type": "object",
            "required": [
              "coordinates",
              "type"
            ],
            "properties": {
              "coordinates": {
                "type": "array",
                "items": [
                  {
                    "type": "number",
                    "maximum": 180,
                    "minimum": -180
                  },
                  {
                    "type": "number",
                    "maximum": 90,
                    "minimum": -90
                  }
                ],
                "minItems": 2,
                "maxItems": 2
              },
              "type": {
                "type": "string",
                "enum": [
                  "Point"
                ]
              }
            },
            "additionalProperties": false
          }
        }
      }
    }
  }
}
```


Figure 5: Ontology and its JSON Schema

After registering the ontology, the system provides a summary of it:

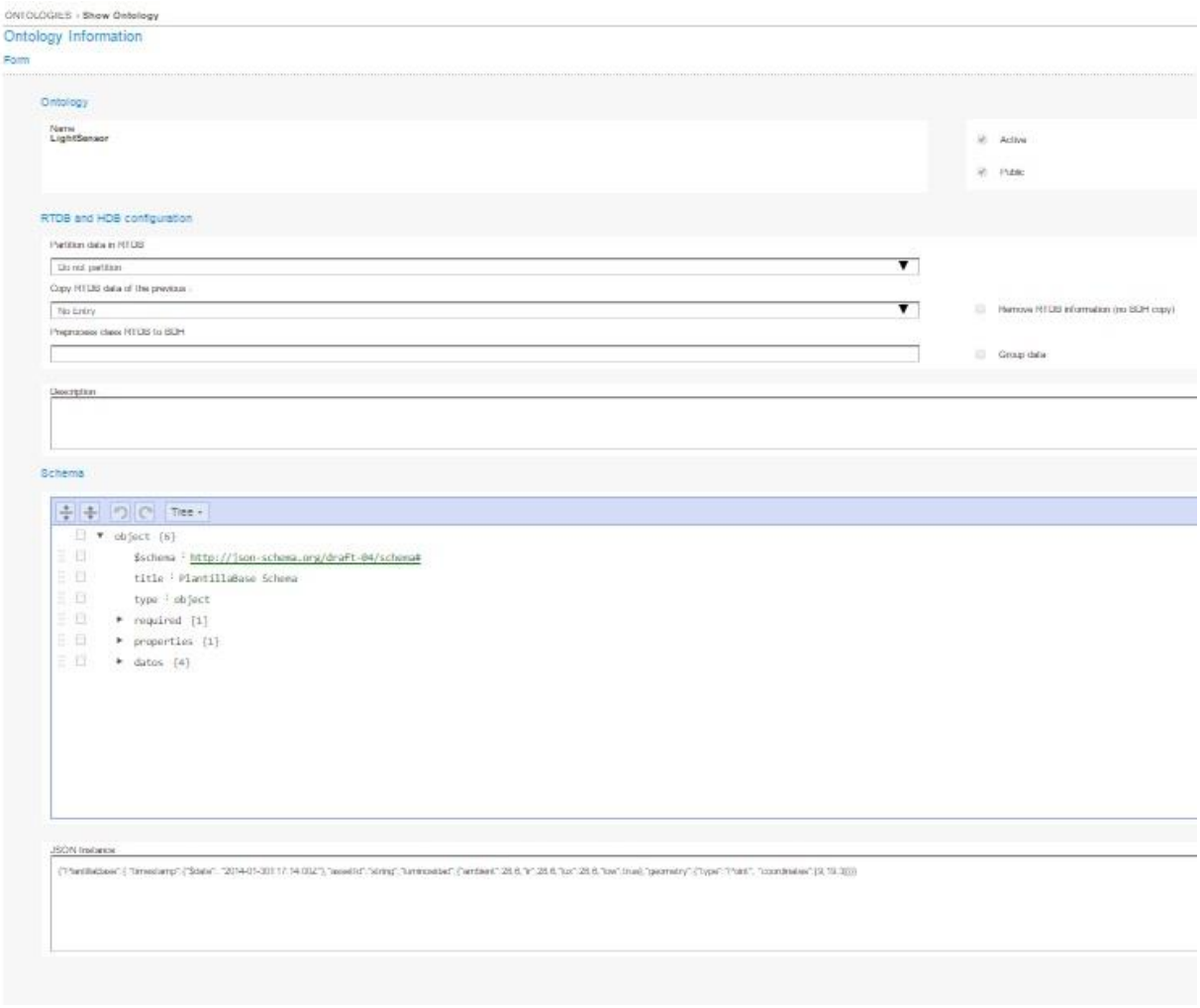

Figure 6: Ontology summary

This allows the User to create a KP associated to that ontology, which is achieved more productively with the graphic interface. As Figure 7 shows, the user selects ontology from the left box and it is added to the right box of selected ontologies.

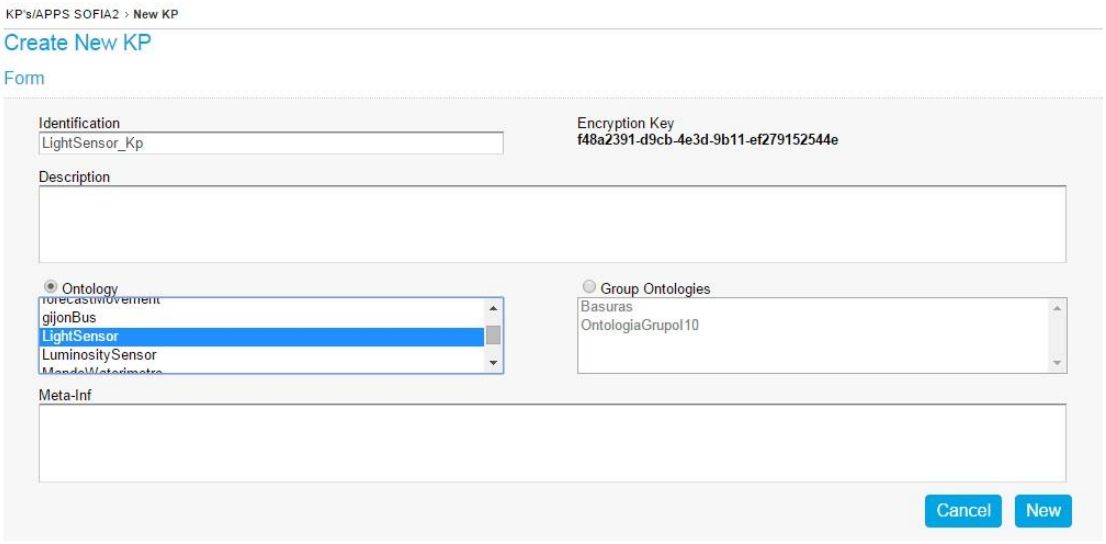

Figure 7: Creating a KP

The registration of Events requires, as said in Section 4.2 Operation, first deciding a unique identifier for the Event, seen in the box "Identification". The user then selects an Ontology (using a drop-down menu), presses the button "Load fields" and then gets the list of attributes. Now the user selects which attributes are required for this Event.

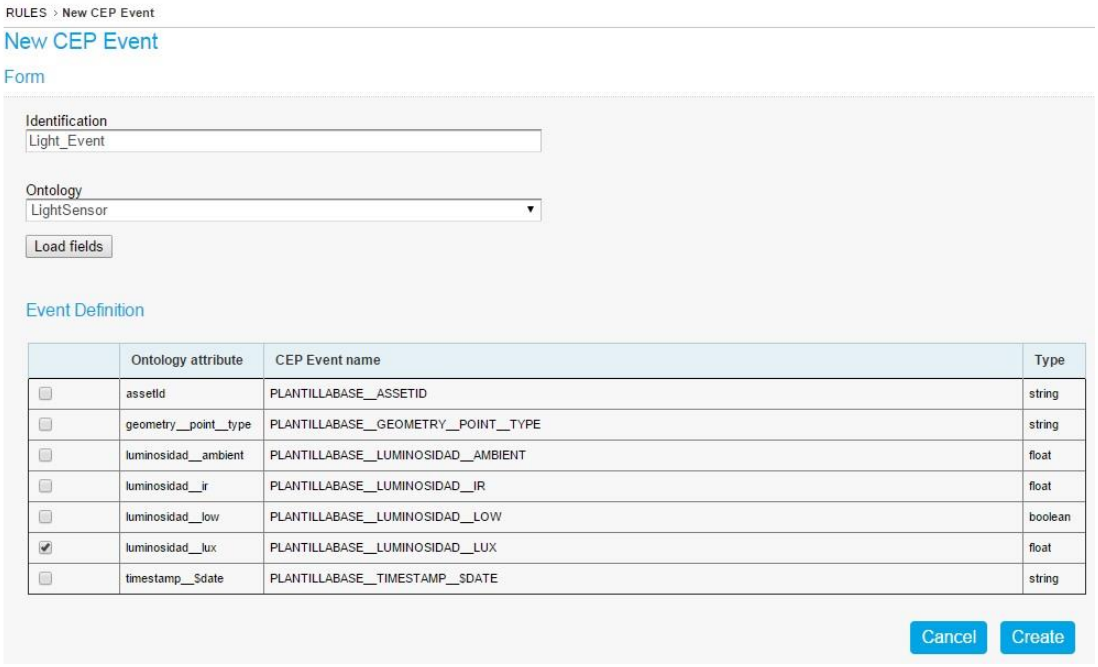

Figure 8: New CEP Event, with attributes

The creation of a rule requires writing the required lines in the boxes as shown in Figure 9 (for instance, the line "ALERTA_EJEMPLO" in the box "Insert Info"):

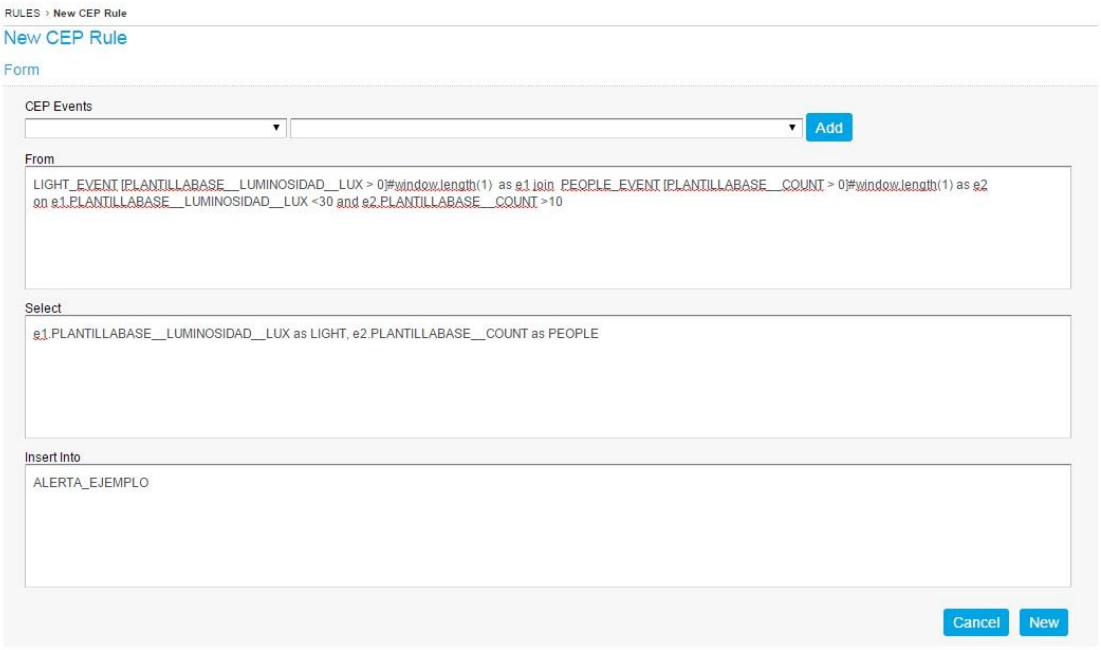

Figure 9: New CEP Rule

After creating a rule, it is strongly recommended that the user to make some tests in the Test Scenario and verify the rule's functioning in the Process Visualizer. This Visualizer is an advanced functionality, described in more detail in the user guide

5 Alternative Rule Set Tool

5.1 Introduction

In this chapter we attempt to analyse the world of rules into a Complex Event Processing in order to obtain a set of properties for an ideal tool for set rules.

Approach used for the argument will not be based on this specific project, but will be general purpose in some cases, while in others we will analyse different types of systems, but in any case, it will not focus on the specific concepts of the eVACUATE system.

5.2 Rules

A rule engine is the most important part of Complex Event Processing. If we want to give a weight to this part, philosophically, we can say that the words “Complex” and “Processing” of a CEP engine are due exclusively to the capability to have a rules engine.

We can also say that the quality of a CEP reflects the power and flexibility of the language(s) used for the development of the rules. Before thinking how we can develop an efficient tool for set rules, we should ask ourselves what properties it should have.

CEP engine is typically general purpose, so the use of the rules for complex processing and the Data Context will have some constraints depending on the target system purpose.

We're going to analyse two types of systems with goals and modalities of application of the rules completely different, in order to extrapolate the main properties and choose those of our interest.

Also, we’re going to examine the methods of composition of the rules, and the rule design tools currently used in order to provide some indications, understood as suggestions to use these tools.

5.2.1 Real Time Systems

The real time systems have unique characteristics of their kind, it needs to receive and process an event in the shortest time in which it is generated, and this is because the situation described by the event itself may not make more sense after a very short time.

Time in real time system is a fundamental base of elaboration, depending on the specific nature of the system, we can calculate that after a certain period of time in which the event occurred, if processing is not completed, or the action taken to manage it is delayed, the result is:

- The event is obsolete.
  - ✓ In case of cyclic events, another event is received with up-to-date parameters.
  - ✓ The action could not have sense or effect.
- The object, if geo-referenced, could no longer be located in the previous physical space.
- The status of the object could be evolved into a different state not compatible with the rule/action.

For example, if our system processes air tracks with high speed (250 meters per second), the processing time must not exceed the order of few milliseconds, if it came to take action after 1 minute, the air track that travels online straight line, it would be 15 Km away. In this case the action produced by the Event Processing could not have more sense, or effect.

Another example is a fire in a building reserved for flammable materials, if the action against an alarm from a smoke detector is late, a small electric short circuit can turn into a big and dangerous fire in a very short time, and the action “check if there is a real fire” is obsolete than “activate sprinklers”.

In these two examples we can see how an object tracked by the system can change its own properties in terms of physical space and/or internal status, and how the event processing in time is crucial.

What can we ask to a rules engine in order to support this type of requirements?

1. A time operator in order to analyse the right event in the right time period.
   a. Sliding Windows
   b. Time Windows
   c. Expiration/Persistence Event Time
   d. …
2. Create, Manage and Process a Context Data on memory
   a. Fast access to all up to date objects information related to the object treated (sometimes the actions to do about a threated object could depend from others objects properties).
   b. Apply the rules on real time flows of events.

5.3 Rules Engineering

The design of the rules is a necessary exercise to get guaranteed results on the coverage of the system specifications obtained from the requirements.

The rules cannot be improvised, the time of design cannot be chosen at random, it is not an activity that can be done without the prerequisites that it need or at the same time this prerequisites are being defined. It should be done beforehand.

Depending on the type of system, there are a series of preparatory activities that must provide a precise and stable output. Let's list them.

Data Model: Simply all objects treated, directly and indirectly.

- ✓ Database Logic Schema
- ✓ Class Diagram (Incoming Data)
- ✓ Class Diagram (Outgoing Data)

Data Context: All objects memory-resident, for which demand fast processing and the maintenance and dissemination of object properties (status, measure etc.)

- ✓ Class Diagram related to the Objects under Processing (typically is a subset of extended (Incoming + Outgoing Class Diagram)

Reasoning: All basic reasoning are obtained from system/functional specifications

- ✓ Set of Reasoning used to produce actions
- ✓ Classification of rules (used for grouping a series of rules for type, purpose, etc.)

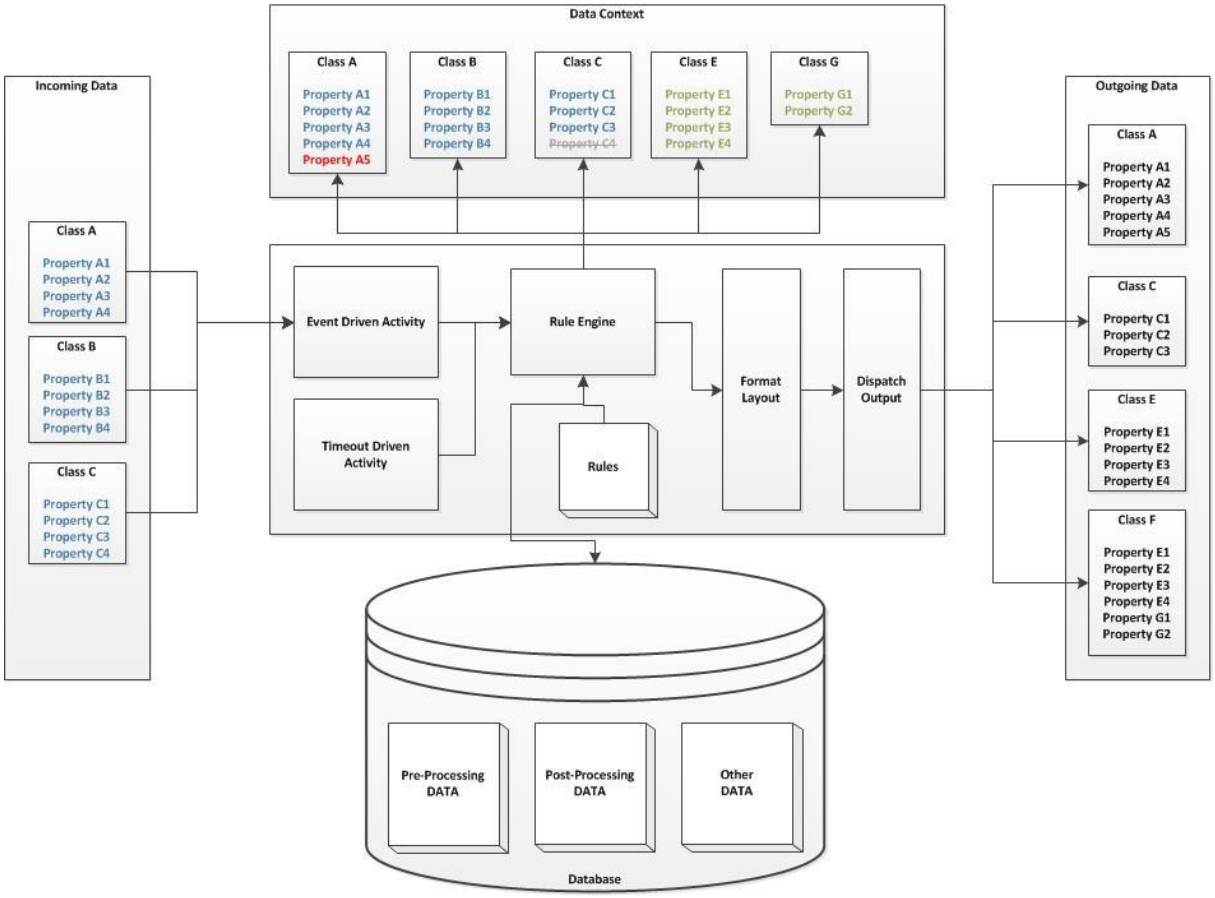

Figure 10: Data Processing example

In the next chapters we try to describe languages of rules, and the importance of having available languages intermediate so as to pass easily from the analysis to development phase.

We will use Drools CEP and its support tools as an example, but the basic principles can be applied to any CEP or similar, such as SOFIA Smart Spaces component.

5.3.1 Rule Languages

Rule language is a direct expression of the rule engine, not so much for the use of mathematical or logical operators (all rule engines use them), but for the previously described properties, the ability to execute rules of objects directly in the incoming flows, on databases, on a date-context in memory.

The most popular CEP implementations provide more different interfaces to implement rules, in order to support different programming language like C/C++, Java, or in order to support database language like SQL, so we could use Java-like or SQL-like language to implement rules, according to convenience.

Some of the popular CEP implementations have their own IDE to implement rules, and sometimes, being the most used, some popular open-source IDE make plug-ins to implement the rules of “most famous CEP”.

This situation opens the possibility to use intermediate languages, for which plug-ins can transform the meta-rules produced in compatible rules, freeing the originators of the plug-ins to operate on meta-language and tool.

In this way, a CEP can use several intermediate languages that did not depend from the CEP itself, but from a plug-in developed into a particular IDE.

In any way, Files, Wizard, Editor or Graphic tools, all depend by the produced reasoning and the data on which to apply the originated rules.

In addition to these tools for the composition of the rules, there are others for validation (syntax and semantic) and for the test, but these are not the subject of this report.

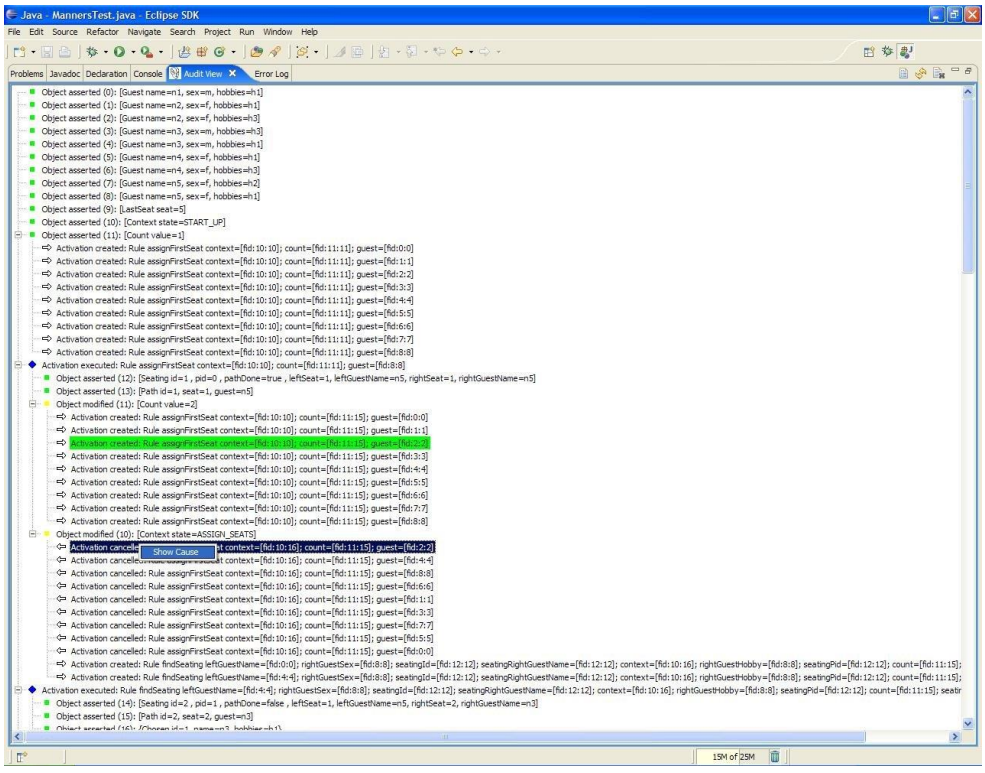

Figure 11: Event Audit of Rules

5.3.2 Design Rules using Translation Tool

Translation tool is a useful method for the creation of rules using a meta-language.

The advantage of this method is that the user can create reasoning using meta-language, and automatically generate the rules. The disadvantage is that the rule engine must be instrumented in order to learn meta-language for translation.

This language is part of domain specific languages and is used by business analyst. The rules look like as a business analyst would describe them into a reasoning document, also it provide an insulation level between domain objects and rules in order to facilitate an eventual refactoring of the objects.

The optimal use of this language is when you apply the same set of rules on different objects with a common set of properties.

The files produced by these tools have .dsl extension (Domain Specific Language).

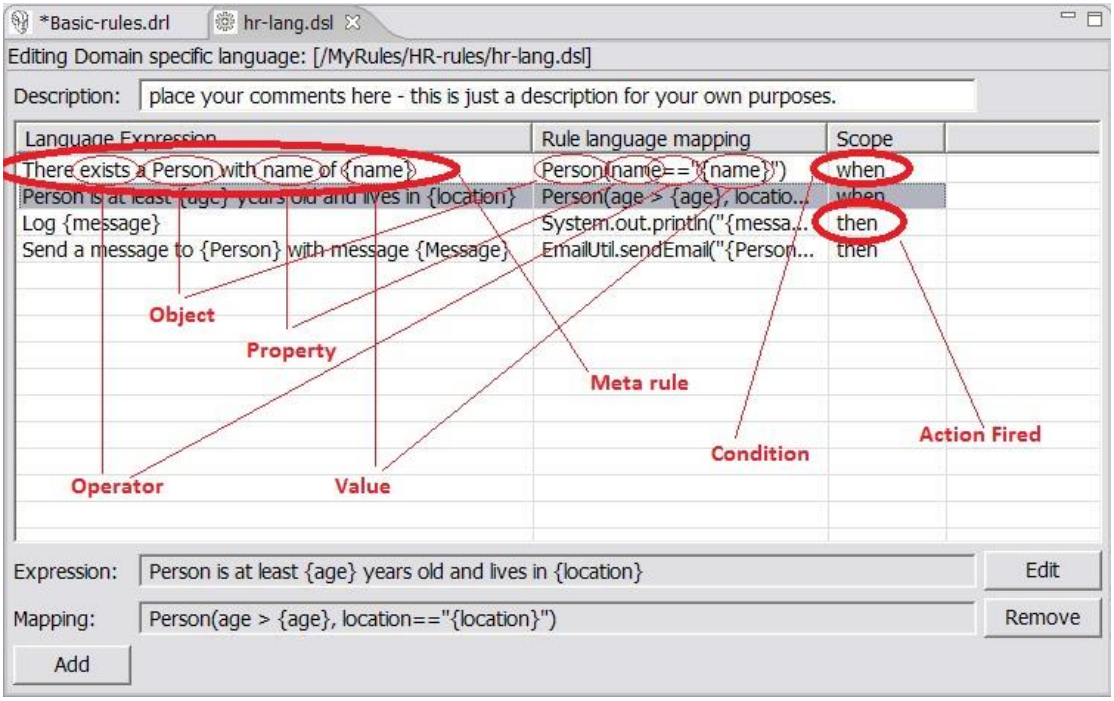

Figure 12: Rule, Actions translator tool

The configuration of meta-rules/rules will create a dictionary ready to use into a project extending the normal rule language by the “meta-rules configured as rules”, into the CEP project files (see Figure 13).

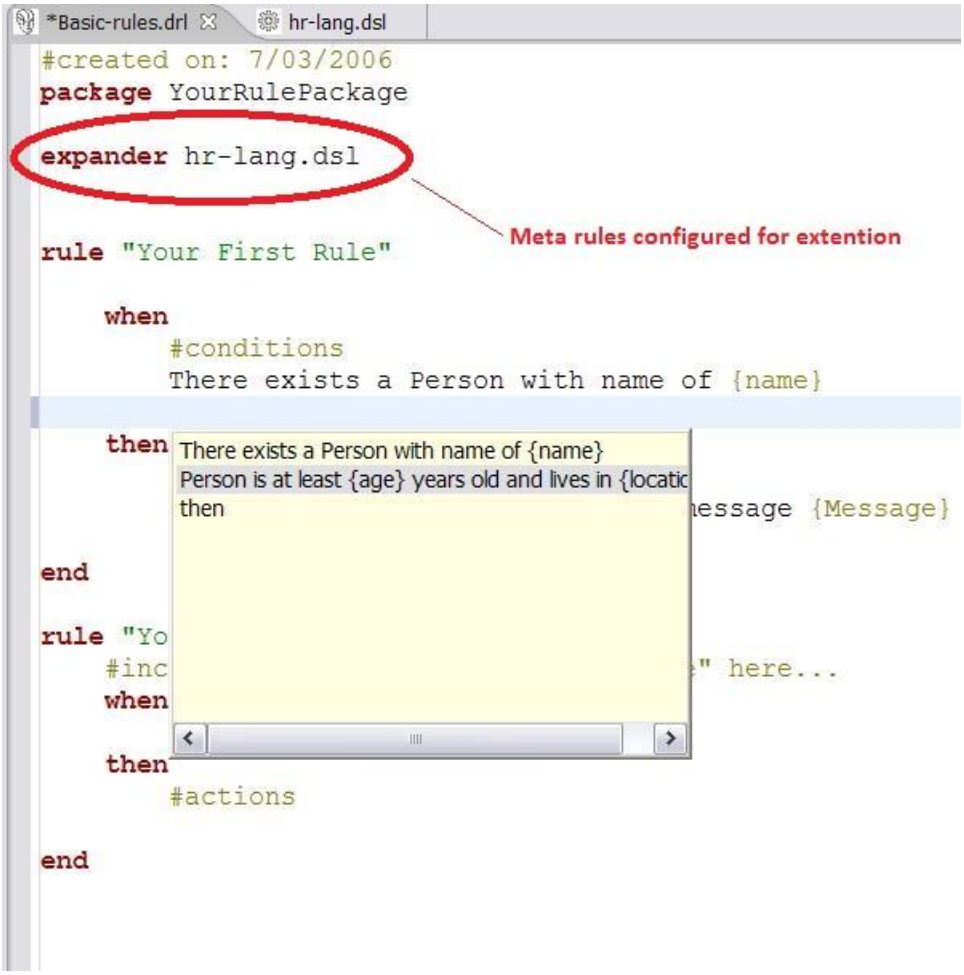

Figure 13: Edit rules in java language using meta-rules

This way of working is much appreciated when in the project plan there are employed a group of analysts in order to provide the reasoning, which, could provide directly the reasoning in the suitable format for the creation of the .dsl files, or even, provide .dsl files using the tool for translation.

Also it is easy work using a classification of set of rules.

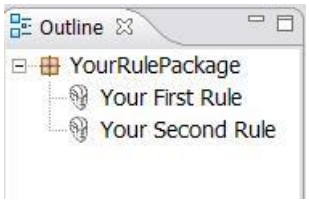

Figure 14: Manage Rules by Classification

5.3.3 Design Rules using Guided Editor

Guided rule editor is another useful tool for edit rules.

To use this type of tool the data model must be completed and in a stable form, objects and attributes will be presented as were modelled. The tool provides the rules applied directly to objects, and consequently, their attributes. This means that the tool knows what and how many objects are to be submitted to the rules, whether they are on the database or data context.

The use is particularly intuitive and immediate, operators are textual, and attributes are processed directly as a function of the type (numbers, strings, etc.).

This allows the tool to display the correct text and operators to apply them correctly, also the actions may already be contextualized, like sending alerts, or further enriched by inserting code in the language currently used (C / C ++, Java, JS etc.).

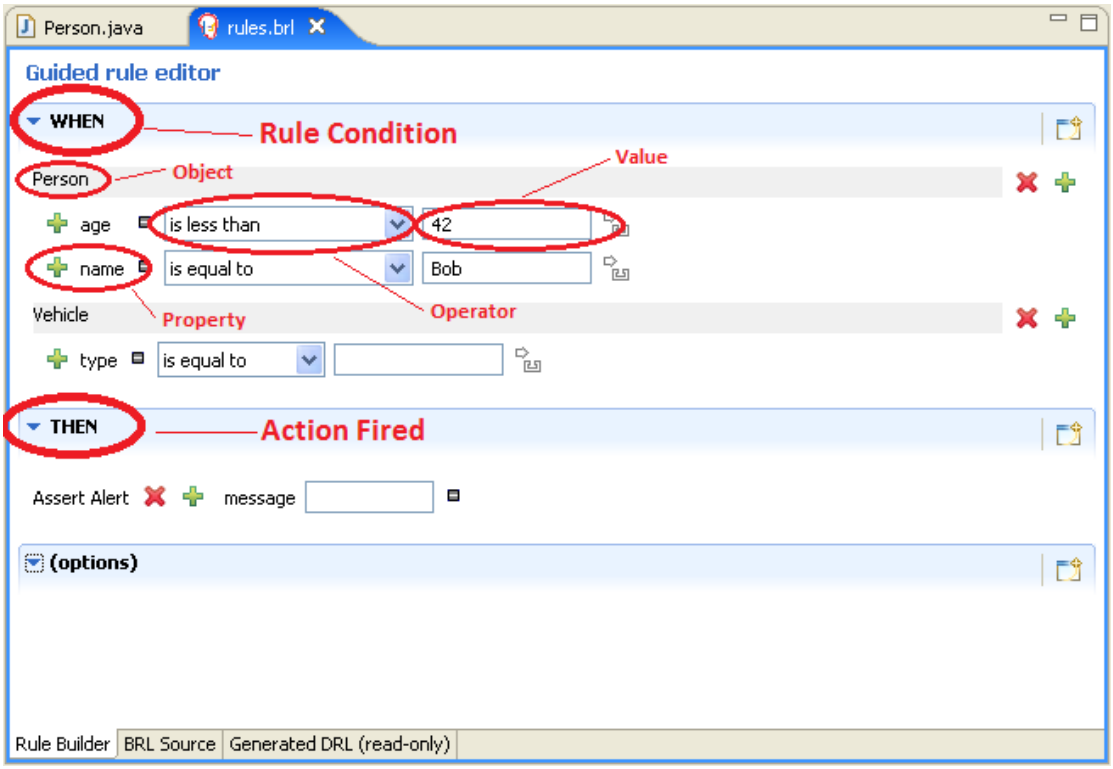

Figure 15: Guided Editor translator tool

To allow the tool to know the objects, these should be described in the configuration files, or the same source code to be instrumented with appropriate pragma, or can be extended classes from base classes provided by the tool itself.

This also applies if the objects are contained in the database.

5.3.4 Design Rules using Graphic Tool

The graphical tool is the most complete, but also the most costly to provide. Currently there are no plans to include it in the implementation of eVACUATE.

Each element has its own graphical objects (Java or C ++), database objects, operators of transition from one to another form (object + operator = action (new object generated), or simply a change of status).

Using a graphical editor you can access objects and their attributes, and to enforce the rules by configuring and or writing rules and source code.

This tool enriches the previous ones, but aside from the ease of composition of flows, and the subjective fact of visualization, does not provide additional and decisive advantages.

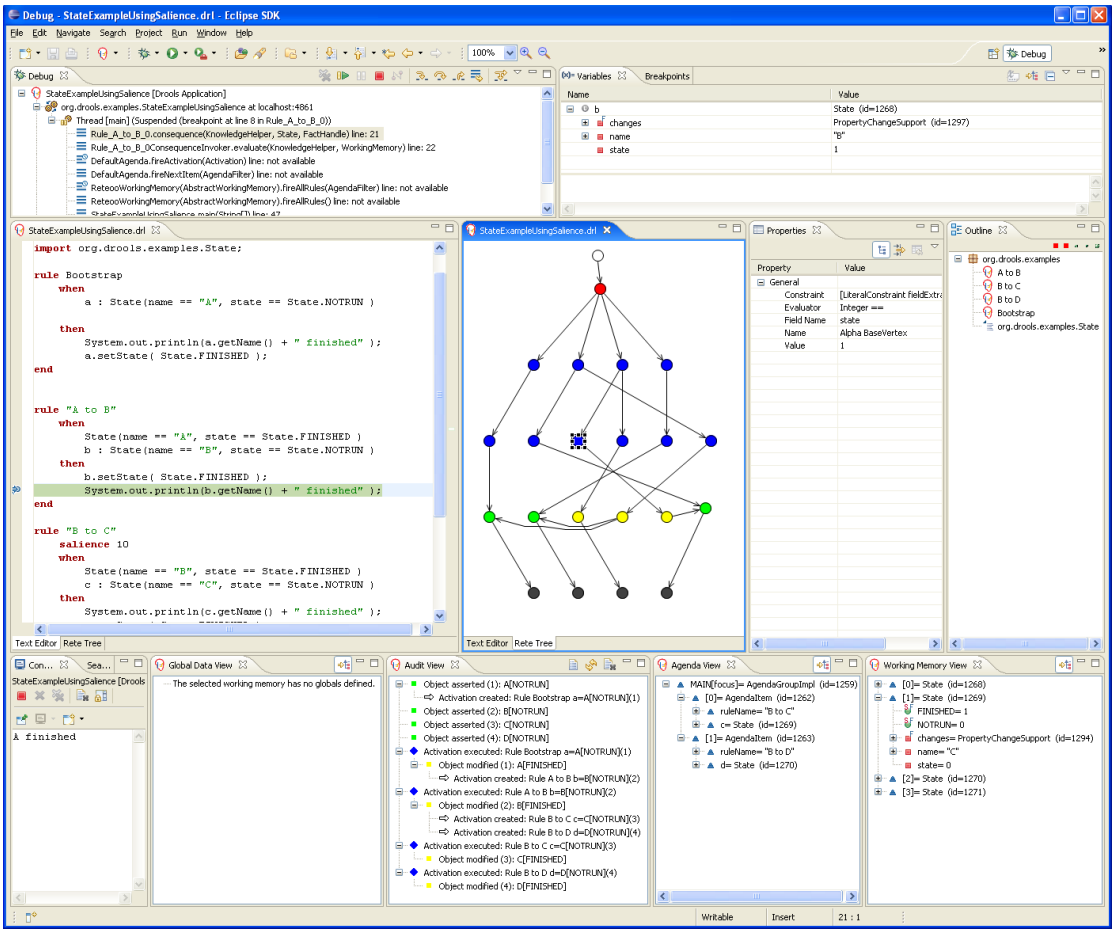

Figure 16: Rule, Actions translator tool

5.4 Conclusions of the Section

We have seen and analysed the rules and the types of tools used for the composition of rules in eVACUATE.

Some types favour the composition of a large number of rules applied to a finite number of objects, favouring the passage of reasoning from the analysis to the composition of the rules (Translation Tools), but, in fact is beneficial only if you have a lot of data and a lot of rules, typical of the projects related to business intelligence, we can say more “analysis friendly”.

Other types are immediate and intuitive to adopt, to be used for specific purposes where the subjects that produce the reasoning could be the same ones that make the rules (high knowledge of the developer), finite number of rules to be applied to a limited number of objects, typical of projects related to the management of real-time events (surveillance, identification, deductions or predictions very specific applied anyway on live events).

Other providing a greater visual support of the flows of information and processing (workflow like), but that do not give decisive advantages in terms of “ease of the composition”, and which must be necessarily provided together with an evolved IDE that allows a whole series of functions, important but not decisive (validation, debugging, testing, coverage etc.).

In all cases, as already stated, the data model must have already been produced, because it is directly used by the tool to apply the meta-rules / rules.

In the case of a tool insert which deals with rules to be applied to event streams / both live and historical data, we would like to suggest one of these two methods (depending on the target of use), or even both (in the case of target with large spectrum).

It is worth spending a few words for rules classification.

Classification is an exercise always convenient, although the tool does not provide many instruments to manage it.

As already mentioned, in the case of few events, a user tends to enclose all the rules applied to the object in a single file consists of one or more rules.

In the case that the inference is very complex, and the same object are applied many rules to obtain numerous inferences or predictions (via Data Fusion with other objects or with other deductions / predictions on other objects or on historical data), and then the classification becomes essential for managing current and future rules.

So we can suggest to insert mechanisms in order to provide an infrastructure inside the tool that has for aim rules management by category, the cost for the insertion of this management is not high, it can be achieved without adding complexity to the production process, simply manage set of rules by including them in a package-specific category by using different files (see Figure 14).

6 Conclusions

6.1 Introduction of Rules

The eVACUATE team has developed a Rule Set Tool, an online software application integrated in the Smart Space, to be used for the introduction of rules within the scope of the task. The RST is currently available in operative conditions.

The introduction of the rules themselves is considered beyond the scope of this task. The task included the development and testing of the software application, plus proofs of concepts, but not its widespread use which will be made at a later stage. The task and the deliverable also include the software and derivative documents that have been developed for the application, and that are not included in this report.

6.2 Impact

The task has helped reaching the objectives of the eVACUATE project by providing a tool to be used within the project, in the pilots and in further implementations of the eVACUATE system.

It has also helped evolving the current technology beyond the state of the art, providing even more reliability to the SOFIA platform, and helping it fit with the availability and methodology restrictions in the current environment.

7 Annexes

7.1 Annex A: List of Acronyms

CEP	Complex Event Processing
CRUD	Create, Retrieve, Update, Delete
DoW	Description of Work
dsl	Domain Specific Language
DWH	Data WareHouse
GUI	Graphic User Interface
IDE	Integrated Development Environment
JS	JavaScript
JSON	JavaScript Object Notation
Km	Kilometre
KP	Knowledge Processor
LED	Light-Emitting Diode
R&D	Research and Development
RST	Rule Set Tool
S.A.	Sociedad Anónima (Spanish; roughly equivalent to public limited company in common law jurisdictions.)
S.A.D.	Sociedad Anónima Deportiva (Same as S.A., applied to sports)
SAES	Situation Awareness and Evacuation System
SIB	Semantic Information Broker
SOFIA	Smart Object For Intelligent Applications
SQL	Structured Query Language
WP	Work Package

7.2 Annex B: End notes

[i] eVACUATE Description of Work, page 71
[ii] Minutes of the Rome face-to-face meeting.
8.8. Domain Specific Languages, Figure 8.9. The Domain Specific Language editor: https://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html/ch08.html.
8.11. Debugging Rules, Figure 8.11. Debugging: https://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html/ch08.html.
[iii] SOFIA2 Concepts.
[iv] D7.2 Architecture of the Smart Space
[v] Minutes of the Athens face-to-face meeting, November 2014
[vi] D9.1 The eVACUATE Integrated System release 1
[vii] Meetings of the Dresden face-to-face meeting, March 2015
[viii] D7.2 Architecture of the Smart Space
[ix] http://en.wikipedia.org/wiki/1983_Soviet_nuclear_false_alarm_incident
[x] http://sofia2.com/
[xi] SOFIA as a Platform to develop Smart Cities (July 2013)
[xii] SOFIA2 Concepts
[xiii] SOFIA as a Platform to develop Smart Cities (July 2013)
[xiv] D7.2 Architecture of Smart Spaces
[xv] eVACUATE Description of Work - eVACUATE - part B – Integration Project
[xvi] SOFIA2 web console use guide
[xvii] SOFIA2-CEP Guide Step by Step (EN) v1.1
[xviii] CRUD = Create, Retrieve, Update, Delete
[xix] SOFIA2 web console use guide
[xx] SOFIA2-CEP Guide Step by Step (EN) v1.1