# A Real-time Complex Event Discovery Platform for Cyber-Physical-Social Systems

Minh-Son Dao[*], Siripen Pongpaichet[+], Laleh Jalali[+], Kyoungsook Kim[*], Ramesh Jain[+], and Koiji Zettsu[*]

[*]National Institute of Information and Communications Technology

3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan

[+]University of California, Irvine, USA

dao.minhson@nict.go.jp, {spongpai,lalehj}@ics.uci.edu, ksookim@nict.go.jp, jain@ics.uci.edu, and zettsu@nict.go.jp

## ABSTRACT

We are living in the Internet of Things (IoT) era where all the (smart) objects around us are connected and communicated with each other to serve our life better without the need of explicit instruction. Soon we have to cope with trillions of heterogeneous data streams coming from IoT. Since data is not information, methods for discovering useful and correlative information from data and utilising them for the better life, in real-time mode, are the utmost requirements.

In order to tackle this problem, we introduce an Event Information Management platform (EvIM) that can be used to develop applications run as cyber-physical-social systems. EvIM includes two components (1) EventWarehouse: is built for harvesting, storing, and analysing data coming from large scale heterogeneous sensors, and (2) EventShop: plays as a real-time complex spatio-temporal event processing system.

Differ from conventional systems that use data-driven or pre-defined event-based approaches, the proposed platform can alleviate the burden of human intervention meanwhile increase the scalability, robustness, feasibility, and applicability by offering series of services that not only automatically discover correlations among sensors' data to extract useful information but also can help users design and monitor situations visually, efficiently and effectively, under on-fly mode.

## Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.3 [**Information Storage and Retrieval**]: Content Analysis and Indexing,Information Search and Retrieval

## General Terms

Theory, Algorithms

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
*ICMR* '14 April 01 - 04 2014, Glasgow, United Kingdom
Copyright 2014 ACM 978-1-4503-2782-4/14/04 ...$15.00.

## Keywords

Internet of Things, Cyber-Physical Cloud Computing, Sensors Networks, Complex Event Processing, Data Stream Management System, Anomaly Analysis

## 1. INTRODUCTION

We are living in the era where all objects around us are connected and communicated via Internet. They are expected, and designed, to gradually become smarter to know and react in the best manner the human requirements without explicit instructions. That leads to the concept of Internet of Things that *"allows people and things to be connected Anytime, Anyplace, with Anything and Anyone, ideally using Any path/network and Any service"* [18]. IoT comprises sensors, processors, and actuators to collect data, discover useful information from these data, and perform the decided action made by the processors, respectively. The *Whitepaper NISTIR 7951* [19] , written by NICT and NIST, is discussed about *"a system environment that can rapidly building, modify and provision auto-scale cyber-physical systems composed of a set of cloud computing based sensor, processing, control, and data services"*. Such a system environment called Cyber-Physical Clouding Computing (CPCC) somehow setups a standard for designing and deploying IoT systems. Nevertheless, there is still lack of, according to [13], suitable IoT middle-ware that can provide context-awareness functionality and easily being deployed in cyber-physical systems.

In the light of these discussions, we are proposing an IoT middle-ware solution that runs on CPCC platform to harvest and analyse data coming from trillions of sensors, to generate useful information that then are used to process and make a decision to lead the act of actuators. Figure 1 illustrates a vision of CPCC and ability of top-up IoT middleware solution focusing on gathering, analysis, and services components. The proposed solution called *Event Information Management* (EvIM) that is designed under context-awareness perspective, and integrates two components (1) EventWarehouse: is in charge of harvesting and analysing data streams coming from heterogeneous sensors networks. This component can homogenise data format into unique Information Object (IO). Moreover, it can discover correlations among information objects to automatically generate situation models based on a given context. (2) EventShop: a spatio-temporal-theme complex event processing system

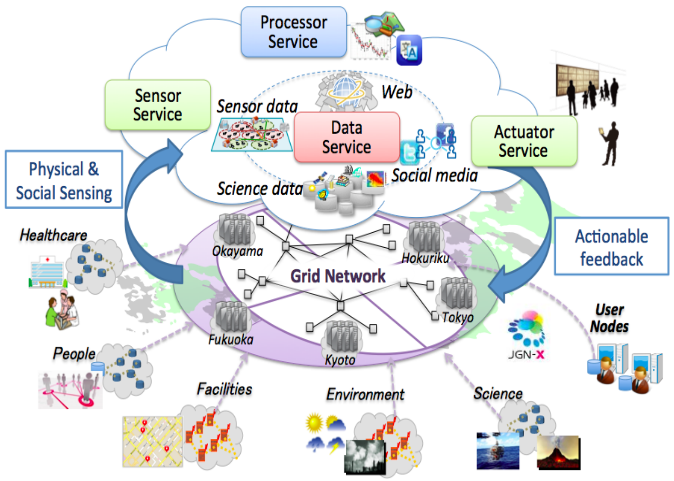


**Figure 1: Vision of CPCC**

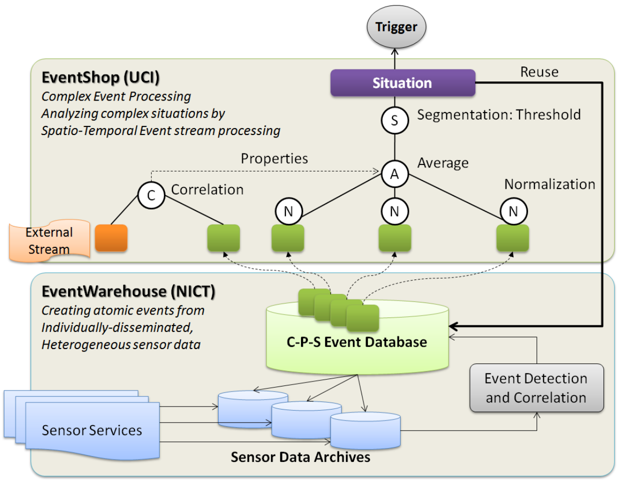


**Figure 2: Event Information Management: Architecture Overview**

that can detect and predict situations and send actionable information to actuators. Two components have a mutual support by which the number of information objects is increased gradually along the cycle of life of the solution. Figure 2 describes the overview of EvIM.

In our best knowledge, the proposed solution is the first end-to-end effort which tries to discover hidden pattern from large amount of observed data from both physical and human sensors, process on real-time heterogeneous data streams in order to recognise and predict real-world situation, and support personalised control based on the situations and end-user needs. To build situation awareness applications across multiple domains, the generic toolkit has been presented for ease application developers.

## 2. RELATED WORK

In fact, our work is a mix of several research areas such as Data Stream Management System (DSMS), Complex Event Processing (CEP), Cyber Physical Cloud Computing (CPCC), Data Mining and etc. DSMS can perform detection, analysis, manipulation, and store of complex events. The standing queries are executed and run continuously to update their answers when new data arrive. Several DSMSs [1][2][6][3] provide a declarative language like SQL, but complex event recognition is more naturally expressed using the operational flow of an imperative language. Thus, CEP systems provided more promising approach and gained a lot of interest. Commercial vendors, such as IBM [4], Oracle [1], Tibco [2], Coral8 [12] and StreamBase [3], have built event processing systems with functionalities of both DSMS and CEP systems. However, none or little works has attempted the integration of data stream in textual, visual, audio, and other rich multimedia formats. The traditional CEP systems still lack the capability to semantically interpret and characterise data to recognise real-world situations since their goal is to detect events, not situations. Spatial attributes of events have been analysed as symbolic data instead of geographical data which is the connection between cyber-physical systems. While geo-location and time semantics are the main citizens in our work.

In another research area, Geographic Information System (GIS) has strong foundation on developing operators and tools for understanding and analysing geographic data such as satellite image, field sensors and etc. This area has led to robust progress in data modelling, geographical analysis operators, and analysis and visualisation tools. ArcGIS [11] is one the most popular tool available. However, most of researches in this field have focused on disk-based archived data sets. The temporal analysis operators and real-time stream processing aspect are just beginning to get more interest into research. Besides, human and social sensors are not received enough attention yet. Our framework does take inspiration from GIS in designing the representation of spatial region under consideration.

Comparing to our previous work [15] [14] which mainly focused on "live" heterogeneous streams of data, we delivered additional components to retrieve and process on both "live" and "achieved" data streams. In fact, previous works process data with a traditional complex event processing schema where data are stored temporally in cache memories and event or situation models are pre-built with support of experts. With this approach, we cannot take into account historical data which can provide a lot of hints to discover "hidden knowledge" forwards to build situation models automatically. Based on these observations, we develop a "special processor" (described in section 5.1) that works not only on archive but also live data to discover hidden patterns and use them to build situation models. This processor has an ability to adapt itself to situation instances, including user profiles. Therefore, the new architecture can cope with ever-evolving situations. Another advanced point of the proposed architecture is a scalability: the system can smartly select an optimal subset of correlative sensors to build situation models. Since at a given point in time and location, not all the data streams are available, to make decision how to aggregate suitable sensors to detect situations does not a trivial issue. Thus, the proposed system is designed to solve these problems: (1) be able to automatically select the best

[1]http://www.oracle.com/technetwork/middleware/complex-event-processing/overview/index.html
[2]http://www.tibco.com/products/event-processing/default.jsp
[3]http://www.streambase.com

candidates and provide good quality results, (2) be able to store situation models along with their spatio-temporal context and the semantic of the application, and (3) be able to on-fly interfere by using metadata that are used for adjusting the model and helping application developers during the model design phase.

The works, in [9][10] discussed the need of mining information from raw data, especially streaming data where challenges of accurate and fast information processing are utmost requirements when dealing with large-scale streaming data coming from heterogeneous domains sensors. The utmost question is how to build a suitable "information patterns" from raw data coming from large-scale heterogeneous sensors in order to detect, predict, and make a suitable decision in real-time. In [9][10][13], a very interesting survey of how people can deal with this challenges. Most of existing methods take a benefit of data mining, machine learning and pattern recognition to detect or discover (hidden) patterns of information. Nevertheless, there is still a long way to get to the ideal solution.

## 3. CHALLENGES AND DESIGN GOALS

The world is dynamic. It is always changing. Most new data is collected to capture this dynamic nature of the world. The dynamic data from the past is important because one can model the world using that data. Those models are used to understand the world and predict future events. Human thirst for knowledge is closely related to human desire to predict and control the future. Until a few years ago, computational facilities limited data collection and processing to very narrow aspects of the world. Around the same time one saw beginnings of stream processing and CEP that dealt with a number of data streams in an organisation. In less than two decades, the situation has changed dramatically. Now we are creating a planetary data warehouse that involves massive number of heterogeneous data streams that must be analysed in real time for predicting evolving situations and managing them.

### 3.1 Challenges and Solutions

As mentioned in previous sections, there are three main challenges the proposed solution needs to overcome: (1) sensors: how to discover useful information from heterogeneous and large-scale sensors?, (2) processors: how to detect or predict situations that can make the human life better?, (3) actuators: how to communicate successfully and instantly with potential actuators?

#### *3.1.1 Heterogeneity*

It seems that human being now has lost in a universe of data created and maintained by themselves. These data are totally influenced by characteristics or purposes of those people who create them. Therefore, they exist in heterogeneous data formats, reflect knowledge from diverse domains, and exchange and propagate in different prototypes. Therefore, we have to deal with following situations in order to extract information from sensors data accurately, completely and dependably:

- several sensors can report the same data for one information,
- one information can only be achieved by a set of sensors not only by one sensor

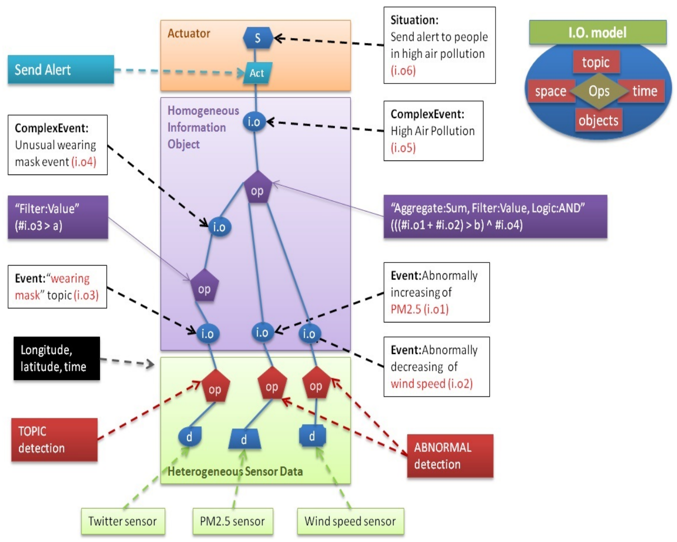

Actuator
S
Act
Send Alert
Situation: Send alert to people in high air pollution (i.o6)
I.O. model
topic
space
Ops
time
objects
Homogeneous Information Object
i.o
ComplexEvent: High Air Pollution (i.o5)
ComplexEvent: Unusual wearing mask event (i.o4)
op
"Filter:Value" (#i.o3 > a)
"Aggregate:Sum, Filter:Value, Logic:AND" (((#i.o1 + #i.o2) > b) ^ #i.o4)
Event:"wearing mask" topic (i.o3)
Event:Abnormally increasing of PM2.5 (i.o1)
Longitude, latitude, time
Event:Abnormally decreasing of wind speed (i.o2)
TOPIC detection
ABNORMAL detection
d
Heterogeneous Sensor Data
Twitter sensor
PM2.5 sensor
Wind speed sensor

**Figure 3: Situation Model: Structure and Manipulation**

- data streams may be of different media types (e.g. free text, numeric value, categorical, or rich media types like images, audio and videos),
- the format of observation streams can be different such as structure(records, XML, array), semi-structure, and unstructured.

In order to cope with these challenges, we propose a new concept for manipulate information, called *Information Object* (IO). Such an IO is designed as follow:

- General case: (data, operators) = Information Object Model
- Specific case:
  (sensors' data, operators) = Event Model (aka atomic event model)
  (events, operators) = Complex Event Model
  (events/complex events, actions) = Situation Model

In other words, Information Object can model information coming from sensor data, event, complex event, situation, and (optional) actuator, and transfer to an unique structure. This structure can encapsulate major characteristics of information/event: what, where, when, who. Moreover, this structure also offers ability to trace-back to original sensor's data source, if required. Therefore, Information Object Model can be used with most of state-of-the-arts methods working in the same area. Besides, Information Object also reflects exactly the spirit of IoT. Figure 3 illustrates one example of situation model based on Information Object Model.

#### *3.1.2 Situation Abstraction and Situation Model*

- **Situation Abstraction**: At UCI, we started developing a general framework and approaches for situation recognition using massive volumes of spatio-temporal heterogeneous data available on the WWW or in the cloud. We define situation as "*an actionable abstraction of observed spatio-temporal descriptors*" [17]. Using this definition, different aspects of situation recognition from different data streams are considered and a framework to define and detect situations in different applications is developed. The framework, called EventShop is implemented as an open source situation modelling and recognition platform. EventShop framework decouples complex event processing and application logic and hides the event processing implementation in the background. This allows application designers to simply create situation-awareness application by focusing solely on their application logic and the semantics of their systems.

- **Situation Recognition Model**: To create situation awareness application in the past, application developers must first know what kinds of information are available, and how to communicate with the source providers. The situation recognition model is designed with the limitation of the small set of data sources and relies purely on the knowledge of application domain experts. Now, there are plenty of data warehouse available. This is an inroad to a new challenge of how to select appropriate data source and provide effective situation model with the combination of knowledge from domain expert and hidden pattern insight the data. Similar situation may reuse the same situation recognition model. The event ontology may be used to obtain a common understanding about events that occur in a given domain.

## 3.2 Design Goals

This work is designed to aid any application designers and developers in any expert domains who want to create situation-aware applications in the large scale and dynamic fashion. Based on the survey in [16], all applications follow a general pattern where they input one or more sets of data, do some value addition, and provide the output in different formats. The input data can be sensor or human report driven, real-time or archived, and homogeneous or heterogeneous format. The processing of data streams can be shallow or deep mash-up integration. Several common operators have been used such as arithmetic logic, value projection, nearest neighbours, and spatio-temporal properties. The output information can be provided to the end users or analysts using some visualisation, action and alerts, and query method.

However, these applications have been so difficult to build and mostly done in an ad-hoc or per-application manner. The main problem is the lack of uniform support for designing, building and executing these types of applications. The designers have to spend lots of effort on the details that are common across all applications especially on data acquisition process. Thus, they are forced to designing the situation recognition model from bottom-up (focus on the data sources available, and think what they can recognise from them). The effectiveness of these applications is also relied on the quality of the model which is usually limited to the knowledge of the domain experts. Nowadays, the characteristic of data has changed tremendously. Abundant of data in variety formats including rich media are available both in real-time streams and archived stores. This can led to a new way of discovering hidden events patterns and generated semi-automated situation recognition models. Another problem is there is no easy and user friendly toolkit for application builders to leverage off when designing their application.

There are several ultimate goals in designing our framework. First, this work has to be versatile to support the diversity of applications. This means it should start with the commodity aspects. The last mile specific issues can be left to the individual application designers where required. Second, the applications should be easy to build and require less technical or computer science expertise. The user input needs to be a declarative specification of "what". The procedural detail of "how" need to be abstracted away whenever possible. The final goal is the ability to improve the quality of the defined situation model and support dynamic and personalised actions based on evolving situations. With these goals in mind, we present the integrated platform combining global event data warehouse with stream processing engines to continuously update the ware-house and compute relevant models for computations and actions.

## 4. ENRICH SITUATION AWARENESS APPLICATION: FROM TRILLIONS OF INFORMATION OBJECTS TO FOCUSED SITUATIONS

The EventWarehouse developed by NICT, aims to gather large scale, heterogeneous sensor data from different domains of anonymous sensing sources like physical sensor networks, SNS and Web, and create metadata for annotating the occurrences of specific events. The EventShop [15] developed by UCI, conducts complex spatio-temporal event processing in real-time for understanding situations. Once a situation is detected by the system, an appropriate action can be taken. This system is similar to event processing concept here [8], but mainly focuses on the global and real-world events. The integrated framework aims at event information management in a collective sensing paradigm. It discovers a set of heterogeneous sensor data stored in EventWarehouse for representing occurrences of a semantic situation declared by the EventShop. This can aid thousand of application developers in building situation awareness application to help million of people.

The process of building any situation awareness applications using our platform can be divided into four main phases including (1) application requirement and specification, (2) model retrieval and discovery, (3) model alteration and design, and (4) application execution and monitoring. An overview of this platform is shown in 4.

### 4.1 Application Requirement and Specification

First of all, application designers must analyse situation behaviours and specify the application's requirement. The designers should able to answer these two questions: what the situation an application needs to recognise, and what action to take when the situation is recognised. This step is the crucial point of building situation awareness applications, and typically impacts the rest of the application design. The content in the application specification consists of four main categories including the purpose of the application in text format, addition information that related to what designers want to monitor, context (space and time aspect for this application), and actions to perform when

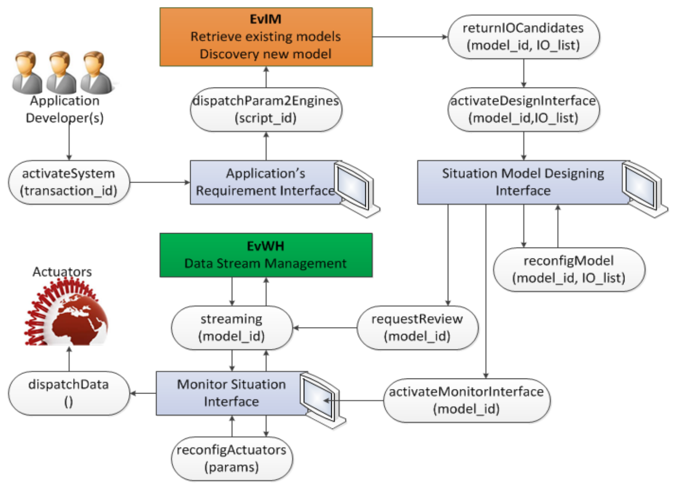


**Figure 4: Integrated Platform Overview**

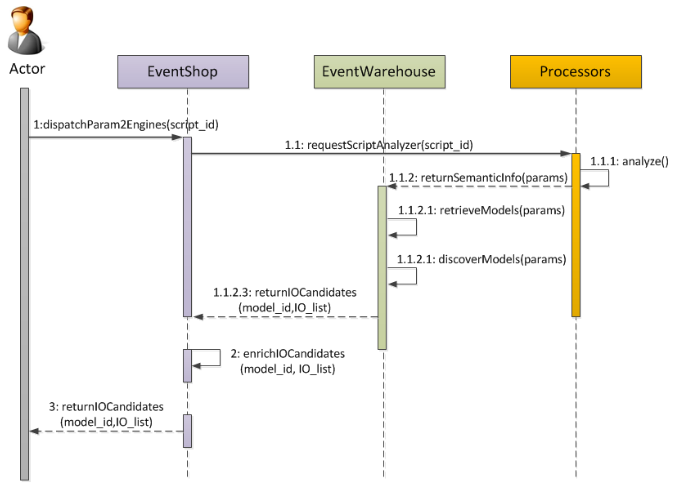

Actor
EventShop
EventWarehouse
Processors
1:dispatchParam2Engines(script_id)
1.1: requestScriptAnalyzer(script_id)
1.1.1: analyze()
1.1.2: returnSemanticInfo(params)
1.1.2.1: retrieveModels(params)
1.1.2.1: discoverModels(params)
1.1.2.3: returnIOCandidates
(model_id,IO_list)
2: enrichIOCandidates
(model_id, IO_list)
3: returnIOCandidates
(model_id,IO_list)

**Figure 6: UML Process**

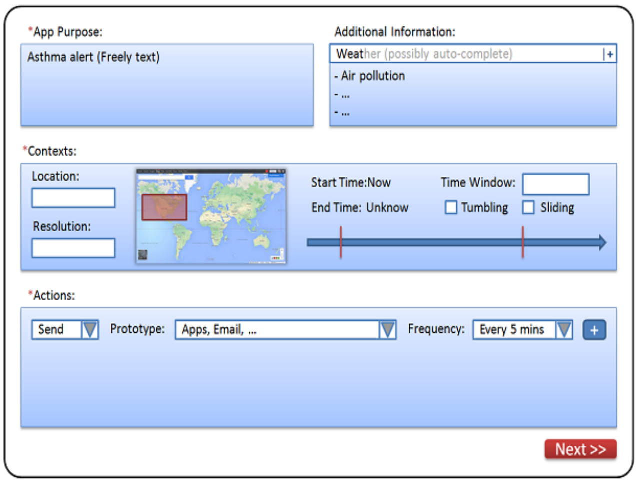

*App Purpose:
Asthma alert (Freely text)
Additional Information:
Weather (possibly auto-complete)
- Air pollution
- ...
- ...
*Contexts:
Location:
Resolution:
Start Time:Now
End Time: Unknow
Time Window:
Tumbling
Sliding
*Actions:
Send
Prototype: Apps, Email, ...
Frequency: Every 5 mins
+
Next >>

**Figure 5: Situation Specification: a Proposed GUI**

situations are detected.

Instead of asking designers to determine what sensors, hardware or software are available to provide the related data source and information object according to the specified situation, the system automatically analyses the application requirement and provides the list of situation models and information object candidates. This allows designers to go top-down approach and focus on the goal of application and do not need to explicitly or prior know where specific information object are. Figure 5 illustrates an example of GUI for the asthma alert application specification stage.

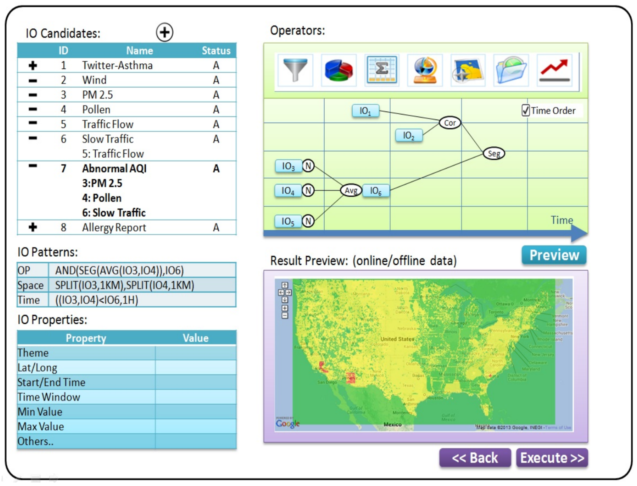
## 4.2 Model Retrieval and Discovery

The system extracts the important semantic information from the application specification script and sends to EventWarehouse to retrieve situation model candidates. This step can be fast or slow, depending on whether the required situation already exists in the system environment or not. There are two modes in this step: **retrieveModels** and **discoveryModels**. The former aims to filter list of existing situation models, stored in EventWarehouse that *similar* to the given request. Jaccard index on topic of situation models and a set of keywords is used, as one good solution, to find the similarity between models. Then the list of models and information objects candidates are sent directly to the next phase. If the similar model cannot be found, the later mode is used to discover a new situation model. The system will process on historical data to realise set of correlative sensors within a context and automatically discover new candidate models to detect given situations. Figure 6 describes underneath process of this stage.

## 4.3 Model Alteration and Design

This is the process of conceptually describing situation of interest using generic building blocks provided by the system. They yield explicit, computable blue-prints of data sources, operators, and the logic required to recognise any situations. Building a system to handle the "*Flu epidemic in USA*" might appear to be too vague and intimidating for a new user. Nevertheless, with some abundances from existing model, the users can break their problem into modular, explicit, and computable chunks. As shown in the figure 7, the list of information object and model candidates are loaded to the interface from EventShop.

This phase can be essential or accidental, depending on the situation model chosen. If the existing model can satisfy the user need, this phase can be accidental. Usually this is not the case. Therefore, users should be able to modify and reconfigure the model as they wish. Although, the model candidates are not exactly what designers want, they still more or less give some hints and directions in designing more appropriate and effective model. In addition, users are allowed to create a new model and add new information objects they want to process at this point. This requires little bit more sophisticated step but it is possible. The new model and information objects are stored backed in the EventShop and EventWarehouse where they can be reused again by other applications. The more detail about EventShop building box, data-sources and operators can be found in [15]. As inspired by PhotoShop which users can preview the result after applying a filter, EventShop allows the designers to preview the result of their model in realtime. The final situation model is sent to the next step to start the application.



IO Candidates:

| | ID | Name | Status |
|---|---|---|---|
| + | 1 | Twitter-Asthma | A |
| − | 2 | Wind | A |
| − | 3 | PM 2.5 | A |
| − | 4 | Pollen | A |
| − | 5 | Traffic Flow | A |
| − | 6 | Slow Traffic<br>5: Traffic Flow | A |
| − | **7** | **Abnormal AQI<br>3:PM 2.5<br>4: Pollen<br>6: Slow Traffic** | **A** |
| + | 8 | Allergy Report | A |

IO Patterns:

| | |
|---|---|
| OP | AND(SEG(AVG(IO3,IO4)),IO6) |
| Space | SPLIT(IO3,1KM),SPLIT(IO4,1KM) |
| Time | ((IO3,IO4)<IO6,1H) |

IO Properties:

| Property | Value |
|---|---|
| Theme | |
| Lat/Long | |
| Start/End Time | |
| Time Window | |
| Min Value | |
| Max Value | |
| Others.. | |

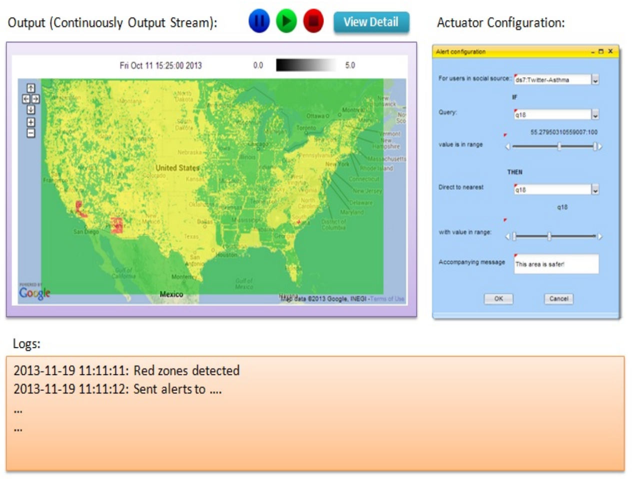
**Figure 7: Situation Model Designing: a Proposed GUI (a) left-top (IO candidates): shows a tree-based structure of situation or complex event model where users can click on + or - to reach children IO or parent IO. This component denotes a data correlation, (b) left-middle (IO patterns): shows Theme-Spatio-Temporal operators used to construct the current IO, (c) left-bottom (IO properties): shows the properties (i.e. contained information) of each IO when selected, (d) right-top (Operators): illustrates (a) and (b) in visual form, (e) right-bottom (Result Preview): illustrates an instance of situation model for previewing**

Even though, the designers are not required to know where the information objects are stored and how to access them, they need to understand the types of information and context provided in order to determine what event and situation can be derived from these objects. For example, a GPS on mobile phone receiver and check-in mobile application like Foursquare may both appear to provide location context. The GPS receiver provides current and continuous latitude and longitude location, while Foursquare only indicates the presence of an individual at a particular location for a particular instant in time. Foursquare may give additional personal context such as home, office, or restaurant name. Both of these objects can be seen as providing location information, but they provide quite distinct information.

## 4.4 Application Execution and Monitoring

The final situation model is sent to the EventShop which continuously executes it and updates the new answer when new data arrive.

Application designers can configure the actuator part. The simple Event Condition Action (ECA) rules can be used to generate personalised alert to the sensors and/or end-users. For example, the designer can set a rule to automatically sending a warning notification to asthma patients when they are approaching to the area where the situation is not safe for them. Depends on the detected situation, designers can simply adjust the content of the messages, the frequency of sending alerts, the people receiving the notification, and the mechanism to transmit them. During the disaster situation such as hurricane or flood, we might need to inform people



Alert configuration

For users in social source:: ds7:Twitter-Asthma

IF

Query: q18

value is in range 55.27950310559007:100

THEN

Direct to nearest q18

q18

with value in range:

Accompanying message: This area is safer!

Logs:

```
2013-11-19 11:11:11: Red zones detected
2013-11-19 11:11:12: Sent alerts to ....
...
...
```

**Figure 8: Situation Monitoring: a Proposed GUI**

more frequently that normal situation. Figure illustrates a proposed GUI for this stage.

In addition to sending alert, the detected situations can be send back and store in EventWarehouse. They can be used to detect evolving situations by the same application or reused as new information object to detect new situations by other applications. Note that the situation detected and produced by one particular application can be consumed and acted on by completely different applications. With our framework, there is no need for the applications to be aware of each others' existence.

## 5. SCENARIO

Actually, for the past couple of years, several applications have been designed and developed using the EventShop framework to recognise real world situations. For example, hurricane detection and migration, demand hot-spots of business products identification, flu outbreak, wildfires detection, flood migration, and allergy risk and recommendation [15].

Here, we demonstrate asthma relief application built on the proposed solution, the most motivating one. The goal is to suggest safer areas to people who are in asthma risk zones. From asthma study, the severeness of an environment to an asthma patient is related to the pollen count and air quality in that area. From crowd sourcing aspect, if many people from a specific area discuss about asthma, it usually suggests that the situation in that area is not very friendly to asthma patients. We combine data from these three data sources, and then segment the aggregated data over entire US into three danger zones based on the values in aggregated E-mages as shown in Figure 8. Then, a situation action rule is created to broadcast a safe area to an individual.

Clearly, we have to cope with heterogeneous data coming from various domain. Specially, pollen count, data of PM2.5, rain, temperature, wind, etc. are coming from physical sensors; Data of people talking about asthma is crawled from Twitter; and data of safe-place or emergency support centre are achieved from web-pages/websites.

Here, we design a "special processor", plug-in to EventWarehouse, that can automatically discover correlation among potential sensors data to discover useful information and encapsulate them into "information object". Such an information object will be a candidate situation model further fed to EventShop for designing.

The idea of building such an processor based on "anomaly detection". In fact, people usually pay attention on anomaly things, and most of sensor devices are designed to report "abnormal behaviour" of nature and human as well [5]. As mentioned in section 3.1, using single sensor cannot gather an accurate and complete information. Thus, sensors data fusion must be paid attention in our design. In the light of these discussions, we propose a new algorithm to automatically discover correlation among sensors data to build situation models based on anomaly detection running on time-series data.

### 5.1 Situation Model Discovery from Anomaly Phenomena

We design a situation model discovery from anomaly phenomena as a processor component plug-in to Eventwarehouse (see figure 9), it has four components: (1) semantic decoupling, (2) spatial-correlative sensors clustering, (3) anomaly detecting, and (4) patterns discovering. The first one uses NLP techniques to extract as much as possible "context information" that can leverage the mapping between problem requirements and sensor sources. This component can generate the set of candidate sensors that may give a relative information to the current requirement. The second one runs as water-shed clustering - start from the set of candidate sensors, within a certain distance, gather all of sensors that are located at or report data from the same area with candidate sensors. The third one using STL method [7] to detect outliers from time-series sensors data. All of "outlier" peaks are marked and symbolised by special symbols. These symbols are then arranged by time and aligned to find the most "frequent pattern" by using multiple sequential alignment techniques, run by the last component. Figure 10 shows one example of "information patterns" detected from the proposed algorithm.

The output of the proposed algorithm implies the correlation among sensors, and these correlations are discovered to build candidate situation model as the input of EventShop. The algorithm theoretically can run with heterogeneous data format varying from text, numeric, audio, image, to video, to name a few.

## 6. CONCLUSIONS

In this paper, we introduce a proposed solution running on CPCC platform, called EvIM, to harvest and analyse data streams coming from heterogeneous sensors. EvIM is designed as the integrated platform between the global event warehouse and real-time processing. This system provides several vital benefits in building situation awareness applications across domains. We also introduce a unique Information Object (IO) concept to harmonise data streams from heterogeneous sensors networks.

Our user interface is simply to use and require non special software skill. It facilitates the separation of application semantics from low-level event acquisition process. For example, an application logic to detect the asthma risk and take action does not have to be modified if the model to detect anomaly PM2.5 phenomena in the underneath information object is changed. The related information object and similar situation models are automatically discovered by the system. Users can easily browse through them and create their own situation model. This also allows information objects to be easily reusable and shared by multiple applications. Besides, they can preview the result of their models before actually execute them.

The EventWarehouse is designed for easily plug-in new functions of "Processor" component (see figure 6). This component is in charge of discovery "information patterns" that are then used by EventShop to design situation models. Therefore, at the designing stage, application developers only pay attention to look-up a list of detected or stored situation models, and modify the models to serve their own purposes regardless the knowledge of sensors networks, data structures, and data sources. This makes our solution differ from conventional CEP or ESP systems where designers have to take care all of these things and must have strong domain knowledge to design "information pattern" from scratch.

## 7. REFERENCES

[1] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, et al. The design of the borealis stream processing engine. In *CIDR*, volume 5, pages 277–289, 2005.

[2] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: a new model and architecture for data stream management. *The VLDB Journalï£¡The International Journal on Very Large Data Bases*, 12(2):120–139, 2003.

[3] T. Bernhardt and A. Vasseur. Esper: Event stream processing and correlation. *ONJava, in http://www. onjava. com/lpt/a/6955, O'Reilly*, 2007.

[4] A. Biem, E. Bouillet, H. Feng, A. Ranganathan, A. Riabov, O. Verscheure, H. Koutsopoulos, and C. Moran. Ibm infosphere streams for scalable, real-time, intelligent transportation services. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 1093–1104. ACM, 2010.

[5] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

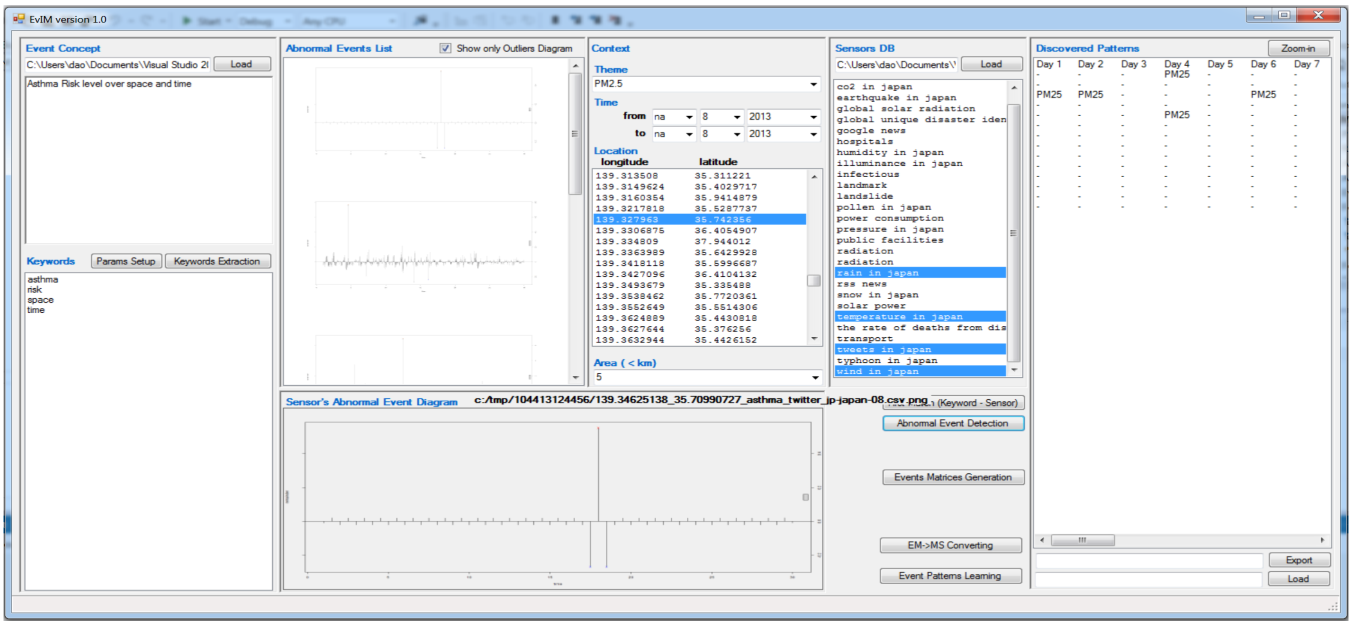
[6] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. R. Madden, F. Reiss, and M. A. Shah. Telegraphcq: continuous dataflow processing. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 668–668. ACM, 2003.

[7] R. Cleveland, W. Cleveland, J. Mcrae, and I. Terpenning. Stl: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1):3–73, 1990.

[8] O. Etzion and P. Niblett. *Event processing in action.* Manning Publications Co., 2010.

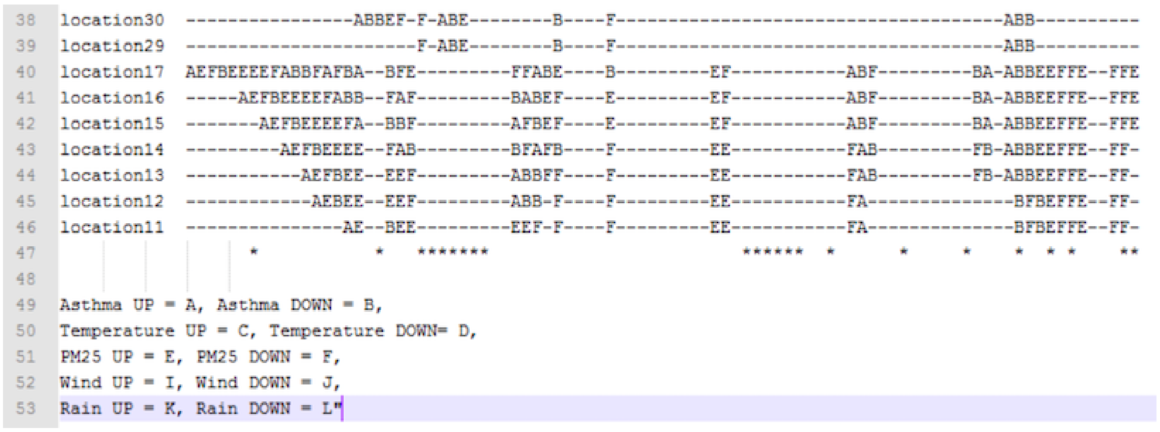
[9] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: A review. *SIGMOD Rec.*, 34(2):18–26, June 2005.

[10] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent Pattern Mining: Current Status and Future



| longitude | latitude |
|---|---|
| 139.313508 | 35.311221 |
| 139.3149624 | 35.4029717 |
| 139.3160354 | 35.9414879 |
| 139.3217818 | 35.5287737 |
| 139.327963 | 35.742356 |
| 139.3306875 | 36.4054907 |
| 139.334809 | 37.944012 |
| 139.3363989 | 35.6429928 |
| 139.3418118 | 35.5996687 |
| 139.3427096 | 36.4104132 |
| 139.3493679 | 35.335488 |
| 139.3538462 | 35.7720361 |
| 139.3552649 | 35.5514306 |
| 139.3624889 | 35.4430818 |
| 139.3627644 | 35.376256 |
| 139.3632944 | 35.4426152 |

Figure 9: Situation Model Discovery from Anomaly Phenomena: GUI tools



```
location30  ----------------ABBEF-F-ABE--------B----F--------------------------------------ABB----------
location29  ----------------------F-ABE--------B----F--------------------------------------ABB----------
location17  AEFBEEEEFABBFAFBA--BFE---------FFABE----B---------EF-----------ABF---------BA-ABBEEFFE--FFE
location16  -----AEFBEEEEFABB--FAF---------BABEF----E---------EF-----------ABF---------BA-ABBEEFFE--FFE
location15  -------AEFBEEEEFA--BBF---------AFBEF----E---------EF-----------ABF---------BA-ABBEEFFE--FFE
location14  ---------AEFBEEEE--FAB---------BFAFB----F---------EE-----------FAB---------FB-ABBEEFFE--FF-
location13  -----------AEFBEE--EEF---------ABBFF----F---------EE-----------FAB---------FB-ABBEEFFE--FF-
location12  ------------AEBEE--EEF---------ABB-F----F---------EE-----------FA--------------BFBEFFE--FF-
location11  ---------------AE--BEE---------EEF-F----F---------EE-----------FA--------------BFBEFFE--FF-
                  *           *   *******                               ******  *      *     *    *  * *    **

Asthma UP = A, Asthma DOWN = B,
Temperature UP = C, Temperature DOWN= D,
PM25 UP = E, PM25 DOWN = F,
Wind UP = I, Wind DOWN = J,
Rain UP = K, Rain DOWN = L"
```

Figure 10: Example of Discovered Information Patterns

Directions. *Data Mining and Knowledge Discovery*, 14(1), 2007.

[11] M. D. Kennedy. *Introducing Geographic Information Systems with ArcGIS: A Workbook Approach to Learning GIS*. Wiley. com, 2013.

[12] J. Morrell and S. Vidich. Complex event processing with coral8. white paper (2007).

[13] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the internet of things: A survey. *Communications Surveys Tutorials, IEEE*, PP(99):1–41, 2013.

[14] A. Gupta,, and R. Jain. *Social life networks: a multimedia problem?*. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 203-212. ACM, 2013.

[15] S. Pongpaichet, V. K. Singh, M. Gao, and R. Jain. Eventshop: recognizing situations in web data streams. In *WWW '13 Companion*, pages 1359–1368, 2013.

[16] V. K. Singh and R. Adviser-Jain. *Personalized situation recognition*. Dissertation of Doctor of Philosophy, UC Irvine, 2012.

[17] V. K. Singh, M. Gao, and R. Jain. Situation recognition: an evolving problem for heterogeneous dynamic big multimedia data. In *ACM Multimedia*, pages 1209–1218, 2012.

[18] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelffle. *Vision and Challenges for Realizing the Internet of Things*. ISBN:978-92-79-15088, 2010.

[19] E. Simmon, et. al.: *A Vision of Cyber-Physical Cloud Computing for Smart Networked Systems, NIST Interagency/Internal Report (NISTIR) 7951* http://www2.nict.go.jp/univ-com/isp/doc/NIST.IR.7951.pdf, 2013