Towards an Open Smart City Notification Service

Luiz Cajueiro, Silvino Neto, Felipe Ferraz, Ana Cavalvanti
Recife Center for Advanced Studies and Systems (CESAR)
Recife – PE, Brazil
e-mail: {luiz.cajueiro, silvino.neto}@gmail.com, fsf@cesar.org.br, anapaula.cavalvanti@gmail.com

Abstract— **Heterogeneity is a critical barrier to event processing in Smart Cities. Over the past few years many techniques have been proposed for message exchange between specific domains systems. However, identifying the context for all kinds of events is hampered due to the large number of protocols and formats that exist. Under these circumstances, this paper proposes a middleware for analyzing multiple events and to infer their respective contexts, thereby providing a flexible way of communication between distributed systems in different domains of a Smart City. Our studies have lead us to believe in the feasibility of the proposed approach, for it is an approach which reduces the burden of implementing/mapping all of the existing formats which can be activated in a given situation.**

Keywords:Middleware; CEP; Pub/Sub; Heterogeneous Systems; Smart City; Events; Google Cloud Platform.

I. INTRODUCTION

In one of our previous works [1] we presented the Platform for Real-Time Verification of Epidemic Notification (PREVENT), a cloud-based message-oriented middleware platform for real-time disease surveillance which uses the HL7-FHIR specification. FHIR is HL7's new specification that comprises of a set of international standards to exchange applications. This work originated in the need to allow PREVENT to receive messages which were structured in any given non-FHIR healthcare protocol. Through this experiment we observed that the heterogeneity problem faced by PREVENT is common to other systems in different domains. This fact has motivated our research to seek a common solution or framework to address this problem.

Cities are the main centers of human and economic activities. They have the potential to create and maintain means which generate development opportunities for their inhabitants. However, as cities grow they lead themselves into a wide range of complex problems [2][3]. Through technological innovation, the smart city concept emerges as an approach which promises to work in favor of more efficient and sustainable cities. Since its inception, the concept designed to enhance the potential for smart cities evolved from the specific projects implementation level to global strategies aimed at addressing the broader challenges of cities [2].

Each city is a complex *ecosystem* which consists of several subsystems working together to ensure the different services being provided (e.g., energy and water supply). Due to the growth of each of these ecosystems, the amount of information for decisions to be made becomes overwhelming. As a consequence, there are neither standard courses of action nor well-established ways to handle such a massive amount of data.

The intelligence used for this control is, in general, digital or analog - and almost inevitably human - pointing towards the utility of an automated process. In the context of smart cities, automation is a vital component for the connections between systems [4].

Most solutions applied in cities behave monolithically and are not interoperable [5]. Through communication between different systems, it is possible to achieve drastic changes in applications and services offered to citizens, thus enabling the concept of smart cities to become a reality [6]. Each one of these heterogeneous systems works and deals with specific functions which operate in each context. However, such an individualized approach is usually part of a bigger and more complex scenario, with many other applications involved. Thus, contextualized event sharing can provide the necessary triggers to generate a standalone flow in the treatment of occurrences arising in cities [5][7].

Different specific domain entities usually adopt a common domain communication protocol; but such a consensus can remain unknown to other areas or system domains. In a smart city there are various standards for communication as well as an extremely high and continuous flow of generated events. Protocols such as HL7-FHIR [1] bring standards to a specific domain where applications can be adapted to embrace solutions. On the other hand, trying to adapt legacy systems to communicate with these protocols can be too expensive in terms of cost and effort in a way that could lead the project to impracticability.

This paper proposes a cloud middleware for analyzing the different events arising in heterogeneous systems. It also provides a mechanism for events to be notified to the interested parties, considering topics of interest previously informed through the Publish/Subscribe (Pub / Sub) design pattern. A middleware with notification-based services (ASN), which applies discrimination filters based on context, may provide the required tools in order to enable Smart City mechanisms to ensure automation and efficiency in the treatment of events which may arise [8].

The proposed middleware adopts complex event processing (CEP) techniques, and a set of adaptive rules is used in order to establish the correlation between the information content of incoming events [9].

The remainder of this paper is organized as follows: Section 2 presents background concepts. Then, Section 3 describes adopted platform flows, processes and other architectural aspects of the middleware. Finally, Section 4 presents conclusions and discusses possible future works.

II. BACKGROUND

This section presents the concepts required for a better understanding of this paper.

A. Smart Cities

Smart cities have their origin in two main factors: (i) the increase in world population, which has caused a rural exodus and (ii) ecological awareness related to scarce natural resources [10].

This concept represents an innovation in the management of cities, as well as their services and infrastructure. It is based on the use of information technology to support the management of problems which occur in modern cities, making them more efficient and sustainable [2].

The smart city concept aims to provide an integrated system in which it is possible to achieve specific goals related to the improvement of cities management and in bettering inhabitants' lives, maximizing efficiency in the implementation of these activities.

1) Features of a Smart City

Implementations for smart cities may differ significantly depending on their focus. The Regional Science Center at Vienna's University of Technology [11] enumerates six aspects that define a smart city. These aspects represent the main challenges related to core areas:

a) SmartEconomy (Competitiveness): Based on the economic structure of cities, it focuses on promoting multiple sectors of the economy in order to maintain the stability of the whole economy if any of its sectors crashes.

b) SmartGovernance (Citizen Participation): Focuses on promoting the current governance model of co-existence (top-down) with informal initiatives, incorporating the format (bottom-up) for its functioning.

c) SmartEnviroment (Healthcare): Applied by reducing land consumption in the city's expanse and aiming for a more efficient use of the environment already used. It highlights the concern with natural conditions, pollution, environmental protection and resources for sustainable management.

d) Smart People (Social and Human Capital): Promotes initiatives in order to solve high unemployment levels, taking into account all citizens, regardless of age, gender, culture or social status.

*e) SmartMobility (Transport and ICT):*Aims at reducing pollution and congestion through alternative transport for cars and the provision of sustainable public transport, available to all citizens.

f) Smart Living (Quality of life): Promotes social cohesion, better health and a decrease in crime rates [10][12][13].

2) Projects:

Over recent years, many initiatives (public and private) have been proposed around the world in order to adapt urban areas to smart city issues. New initiatives propose the creation of new cities, projecting their smart infrastructure from the beginning of their construction. Some initiatives to achieve smart cities include:

- **Amsterdam**, Europe: Created by the merging of many organizations and companies. Named Amsterdam Smart City (ASC), it focuses on the application of smart city concepts for achieving three objectives: economic growth, a change in inhabitants' perception and a focus on improving quality of life [10].
- **Shanghai**, China: It is one of the first pilot cities in China to apply smart cities concepts. It aims to achieve gains in areas such as security and the development of information technology in city systems [12].
- **Rio de Janeiro**, Brazil [13]: A partnership with IBM has resulted in the construction of the Rio De Janeiro city center of operations. It aims to improve the monitoring and control of events in the city through cameras and a better integration between the city's service providers [10][11].

B. Publish/Subscribe

Publish/subscribe design pattern - or pub/sub - is widely used by services which work with interest-based notification. Due to its asynchronous decoupling property, it provides an elegant technique for the implementation of an infrastructure capable of providing a significant level of many-to-many communication. Such a feature enables independence between the entities involved. The pattern is widely used as a main component in projects from various fields, e.g., medical applications [14], air traffic management and industrial production [15].

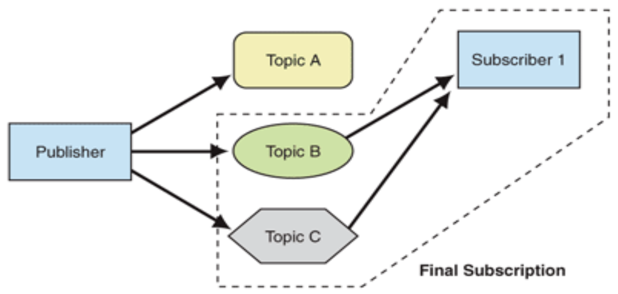


Figure 1. Publish/Subscribe Diagram

1) Pub/Sub Based Services

Pub/sub services are composed of three main components: (i) the publisher, (ii) the subscriber and (iii) a notification service (named broker) which will make the interconnection between subscribers and publishers. This architecture is based on the object-oriented design pattern Subject-Observer.

A pub/sub service starts when the publisher sends a message to the notification service, containing specific information, called an event. This event is evaluated using some fixed condition as a basis within the service and it is extracted as information and/or topic of interest. The service maintains a list of subscribers, which may be linked to one or more points of interest. Thus, the service retrieves subscribers that are interested in the event content and sends a notification to each one through the endpoint informed during the subscription process. Fig. 1 presents the topic-based subscription flow. A system may perform both publisher and subscriber roles, and the distinction between these roles is determined by whether or not it has sent an event, or just subscribed to a topic of interest [15][16].

Notification Services can define different behaviors depending on subscription models. They are:

- Channels-based: The notification service has a number of communication channels. Events are sent and received only by its subscribers, regardless of the content.
- Topics-based: Publishers assign to the sent event an identification tag with some default information. Also, subscribers report which tags of interest should generate notification for them, e.g., accidents, muggings, etc. [14].
- Type-based: Subscribers report interest in events belonging to any area of particular interest, such as public safety or health.
- Content-based: Some rules are defined, rules which the event content must satisfy. Events only generate notification if the defined rules are positively validated [17][18][19].

Such model efficiency is already well established and proven. Its use can be noted in many frameworks, as well as through the OMG Data Distribution Service (DDS) and Java Messaging Service (JMS) [15]. It is also available as a service on cloud platforms like Google Cloud Platform (Cloud Pub / Sub) [20] and Amazon Simple Notification Service (SNS) [21].

C. *CEP (Complex Event Processing)*

Complex event processing (CEP) has become important technology for big data processing because it enables the consumption of a lot of events in a relatively low period of time. CEP is part of an architecture that relies on the detection, use, analysis and generation of events. It is an efficient way to use rules to detect correlations between events within a certain scope of processing [9]. It has been used by many applications ranging from medical systems to real-time financial analysis, fault detection, and so on.

A CEP system consists of relationships between event generators, processing server and other systems called event consumers. Event generators can be sensors which monitor environments or other systems which notify when a specific scenario occurs (e.g., security vulnerabilities or price changes in stocks). Event consumers are typically decision-making systems that perform some action based on notification received by the CEP server. The existence of a server intermediating such communication is one major advantage of using a CEP server, since it eliminates the need for decentralized treatment in end systems for processing events generated by different channels. Fig. 2 shows the structure of a CEP system, as well as its relationship with its components [22].

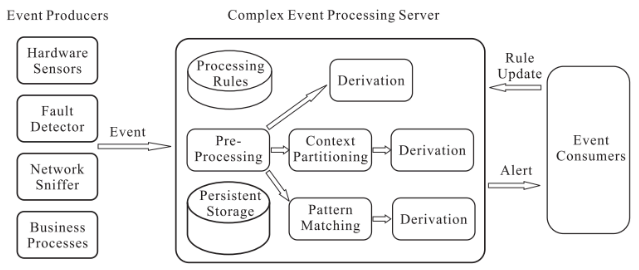


Figure 2. CEP System Structure

Two categories stand out among the existing processing models: Detection-oriented monitoring and aggregation-oriented monitoring [23].

- Detection-oriented model: The data are analyzed from a base, which consists of past events information. In this model, the goal is to find patterns from the processed content.
- Aggregation-oriented model: The data are analyzed in real time and each event received is processed individually [24].

Events are analyzed using a specific set of criteria. Among the most common are:

- Time: The event timestamp is used. Events are linked when they are in a specific range of creation.
- Location: The correlation is performed through the place where the events occurred.
- ID: Events can embed some default identifier for its context.

The unified analysis of temporal and geospatial data allow the identification of heterogeneous events in a Smart City, enabling information sharing between domain-specific systems for processing events.

After the analysis step, a further evaluation is performed in order to determine whether any action will be taken. This kind of trigger is widely used in applications that require to run a procedure in response to external events.

Unlike other client/server approaches (in which the client typically sends a request to the server and expects its completion before sending a new one), in CEP systems communication is made in a single direction. The event generator sends requests continuously and does not wait for an answer. When a consumer system wants to change or create specific criteria, it must send a message to the server, and then it will be notified whenever that new rule is identified [20]. This flow can be seen in Fig. 2 [22].

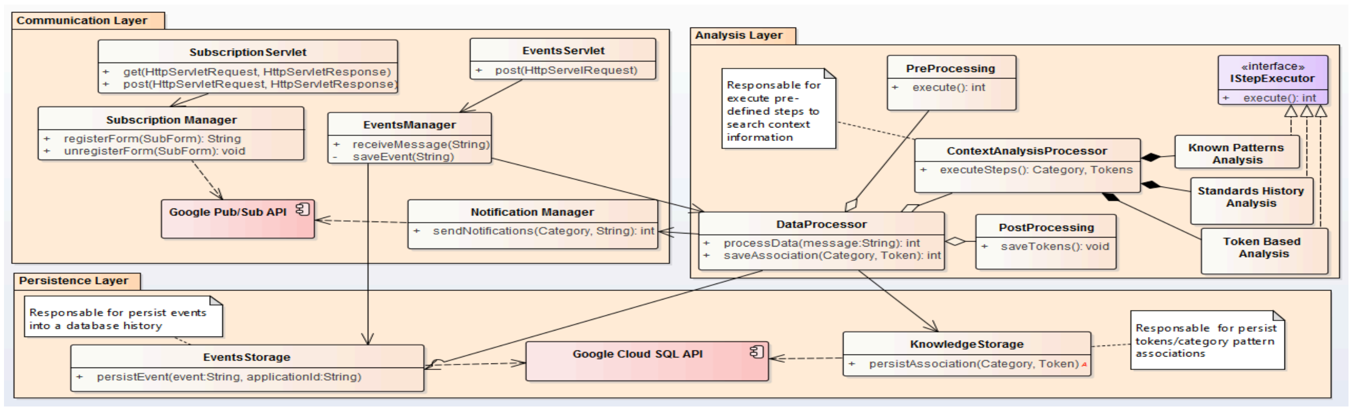

Communication Layer
SubscriptionServlet
+ get(HttpServletRequest, HttpServletResponse)
+ post(HttpServletRequest, HttpServletResponse)
EventsServlet
+ post(HttpServelRequest)
Subscription Manager
+ registerForm(SubForm): String
+ unregisterForm(SubForm): void
EventsManager
+ receiveMessage(String)
- saveEvent(String)
Google Pub/Sub API
Notification Manager
+ sendNotifications(Category, String): int
Analysis Layer
Responsable for execute pre-defined steps to search context information
PreProcessing
+ execute(): int
«interface» IStepExecutor
+ execute(): int
ContextAnalysisProcessor
+ executeSteps(): Category, Tokens
Known Patterns Analysis
Standards History Analysis
Token Based Analysis
DataProcessor
+ processData(message:String): int
+ saveAssociation(Category, Token): int
PostProcessing
+ saveTokens(): void
Persistence Layer
Responsable for persist events into a database history
EventsStorage
+ persistEvent(event:String, applicationId:String)
Google Cloud SQL API
KnowledgeStorage
+ persistAssociation(Category, Token)
Responsable for persist tokens/category pattern associations

Figure 3. Interaction Class Diagram of the Layers

III. MIDDLEWARE

Heterogeneity is a major barrier for processing events generated in smart cities. There are several standards on the market, such as FHIR [1] (specified for medical use), FIX [25] and ISO 8583 [26] (specified for the financial industry). However, due to the amount of existing protocols and formats, the ability to recognize, in a simple way, the context for all kinds of events generated in a smart city is compromised [13].

Aiming to mitigate the impact of mapping and implementing all the possibilities of existing formats, this middleware is designed for analyzing and inferring in which category a received event will be categorized.

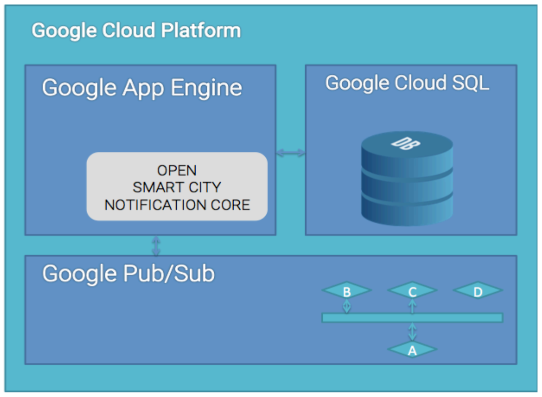


Figure 4 - Middleware Deployment Architecture

This section discusses the flows, processes and other architectural aspects of the middleware. The middleware is designed with a division into three layers. The communication layer is responsible for managing information exchanged with stakeholders. The processing layer performs context analysis and, finally, a layer for persistent data control (Fig. 3). The middleware composition is conducted by independent modules, which can be replicated on multiple servers, thus promoting scalability to meet a lot of requests. The proposed structure is based on events, and, in this model, other systems and entities will provide inputs for the middleware.

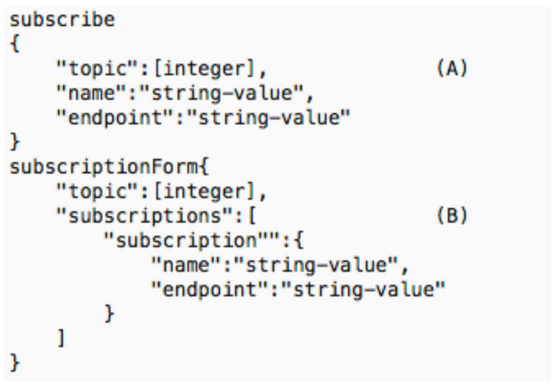
A. Composition

This work adopts the Google Cloud Platform (GCP), a set of cloud-based scalable services. This platform does not directly address the heterogeneity problem, but it provides the necessary infrastructure and, in addition, facilitates the creation of similar instances. Among the services provided by the platform, Google App Engine (GAP) has been chosen to deploy the application, since it provides automatic resizing as the number of requests increases. GAP also provides automatic resources management if the number of requests increases. Moreover, the application is held in a container which can be replicated and run on multiple servers, providing high scalability for the middleware.

For data storage, the proposed middleware adopts Google Cloud SQL for events information maintenance and other system persistence requirements. Message-oriented modeling has been adopted in order to promote communication with subscribers, through the use of the Google Pub/Sub platform, which provides mechanisms for sending and receiving messages asynchronously. Fig. 4 illustrates the middleware structure in the cloud platform. System processing flow consists of three general steps, which are: (i) subscription step, in which systems of interest subscribe to the topics of interest. Next, there is (ii), the analysis and processing step, in which the events are caught and handled, and, finally, (iii) sending notification to subscribers. These steps are further detailed as follows:

B. Subscription Step

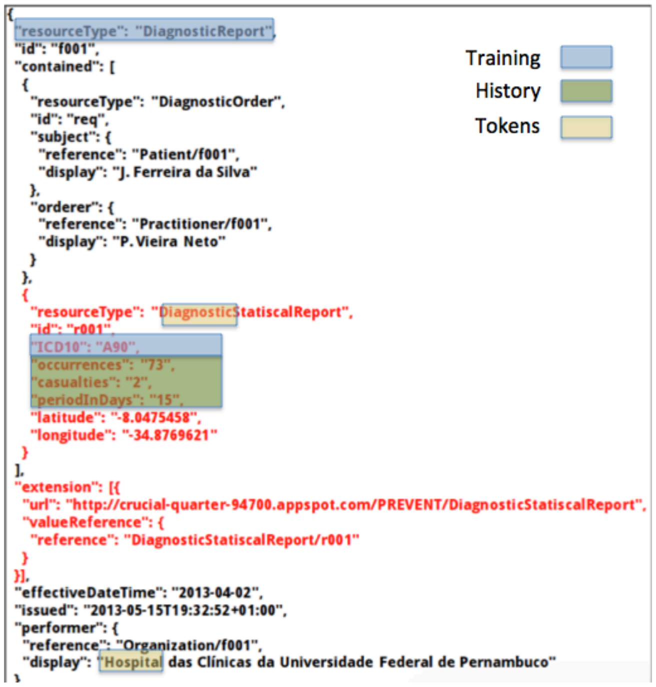
The proposed middleware implements the design pattern publish/subscribe, adapting underwriting issues by topic, type and content-based analysis (Section 2). In order to perform the signature process, the system provides a REST service and a GET method called "subscribe", through which the subscribers must specify a list of topics (or areas) of interest, as well as an endpoint to be triggered as to when a notification should be sent, due to its processing result (Fig. 5 - A).



```
subscribe
{
    "topic":[integer],              (A)
    "name":"string-value",
    "endpoint":"string-value"
}
subscriptionForm{
    "topic":[integer],
    "subscriptions":[               (B)
        "subscription"":{
            "name":"string-value",
            "endpoint":"string-value"
        }
    ]
}
```

Figure 5 - Subscriptions JSON Format

Furthermore, the application also provides a protocol for batch registration. Thus, governance agencies can manage entity groups to be notified whenever a given event type occurs. Therefore, the application provides a method called "subscribeForm", which should contain a list of endpoints return as well as information for filtering interests (Fig. 5 - B). The system returns an ID to subscribers in application methods, and such an ID can be used by authorities updating or canceling its subscription. Such an identifier can also be used if the source system wants to inform any event to the system.



```
{
 "resourceType": "DiagnosticReport",
 "id": "f001",
 "contained": [
  {
   "resourceType": "DiagnosticOrder",
   "id": "req",
   "subject": {
    "reference": "Patient/f001",
    "display": "J. Ferreira da Silva"
   },
   "orderer": {
    "reference": "Practitioner/f001",
    "display": "P. Vieira Neto"
   }
  },
  {
   "resourceType": "DiagnosticStatiscalReport",
   "id": "r001",
   "ICD10": "A90",
   "occurrences": "73",
   "casualties": "2",
   "periodInDays": "15",
   "latitude": "-8.0475458",
   "longitude": "-34.8769621"
  }
 ],
 "extension": [{
  "url": "http://crucial-quarter-94700.appspot.com/PREVENT/DiagnosticStatiscalReport",
  "valueReference": {
   "reference": "DiagnosticStatiscalReport/r001"
  }
 }],
 "effectiveDateTime": "2013-04-02",
 "issued": "2013-05-15T19:32:52+01:00",
 "performer": {
  "reference": "Organization/f001",
  "display": "Hospital das Clínicas da Universidade Federal de Pernambuco"
 }
```

Figure 6. FHIR-Based Message

C. Event Arrival and Analysis

Whenever a new event arrives it is saved in the database and attached to the source system ID. Also, the event content is forwarded to the analysis module.

The review process attempts to discover in the event message one or more key tokens which are content and syntax identifiers. Therefore, all the received and interpreted messages are stored in database records. This database contains references to the messages source systems, as well as finding tokens which made the interpretation possible, so that data can be used for future message interpretation. The middleware performs a three-step process in order to make a structural analysis of the received text message.

Each step performs a different type of search, and if any step is successful, no other needs to be performed. The middleware must be trained before it can run the analysis. The precondition of training reflects the need to enter a set of rules and known message formats into the database, as well as a token list attached to specific contexts. Such information is taken in the following steps:

1) *Known Patterns Analysis:* The system searches for a known format identifier in the message. In Fig. 6, for example, search terms are "ICD10", "ResourceType" and "Diagnostics Report". Identifiers recognition is achieved by comparing message structure with the patterns previously registered in the middleware training phase. Whenever the number of combinations meets a specific threshold, the system sets the message topics of interest.

2) *History Recovery:* Whenever a message does not contain a known pattern identifier the system verifies if it fulfills a set of tokens for historically interpreted messages. Thus, it is possible to assess whether a pattern has already been used and accepted previously, thereby ensuring greater reliability in the results. In the example of Fig. 7 the terms "occurrences", "casualites" and "periodoInDays" have been classified as tokens for the context of a previous analysis.

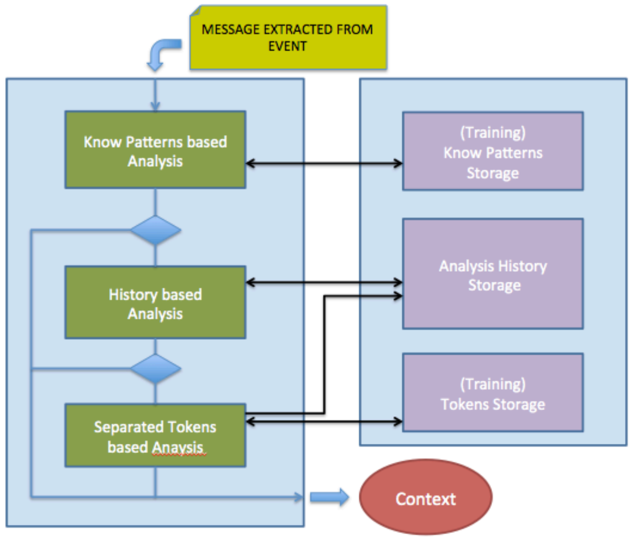

MESSAGE EXTRACTED FROM EVENT
Know Patterns based Analysis
History based Analysis
Separated Tokens based Anaysis
(Training) Know Patterns Storage
Analysis History Storage
(Training) Tokens Storage
Context

Figure 7. Message Analysis Flow

3) *Sundry tokens-based analysis:* In the case that no pattern has been identified in a message, the system uses a set of predefined tokens which are already linked to some

context and which have been stored in the training process. Whenever the amount of combinations meets a defined threshold, the system will consider the message and token group as a new pattern, storing them in the database. In Fig. 6, the terms: "Diagnostic" and "Hospital" have been added to the training stage and linked to the health context. All data saved at that step is used for future history recovery. Fig. 7 illustrates the flow described in the preceding steps.

D. Notification dispatch to subscribers

Through topics of interest information derived in the previous step, the system searches for signatures callback information in the database. Such information is submitted to the pub/sub communication module. Due to the pub/sub adopted engine, all notifications are sent through multiple asynchronous threads. Notification messages are targeted to endpoints reported in the registration step, either for individual subscribers or forms submitted by third parties, to receive a notification message. The sent message's body incorporates in textual representation the original received message in the processed event.

IV. CONCLUSION AND FUTURE WORK

This paper has presented and discussed aspects related to the requirements for information sharing over multiple domains. The information is derived from events of heterogeneous systems that are spread across different systems which comprise a Smart City.

Besides these aspects, many obstacles and difficulties have been raised, resulting from the existence of the numerous communication protocols and formats which are used by each one of these systems. It is believed that the development of the proposed middleware can lead to a complete solution in order to perform the identification and dissemination of events/occurrences in smart city environments.

It is important to mention that this work is currently in progress. In order to clarify the next objectives, a list of targeted milestones has been introduced, as follows:

1) Implementation: Developing the proposed architecture, promoting the adoption of the chosen Cloud Platform framework.

2) Case Study: The simulation of a heterogeneous environment uses the PREVENT framework as a subscriber to receive reports through our middleware. To validate the capacity of analysis and conversion, messages in a non-FHIR format will be delivered to our middleware. As the main expected output our middleware should be able to convert, process and send the messages received(using the knowledge it has been trained for) as events which are understood by the PREVENT plataform.

3) Training Data: As discussed in Section 3, the proposed middleware requires an initial loading of data in order to identify the default messages in the event. This step consists of collecting required data for training the pattern recognition engine adopted by the middleware.

As the output for the listed items, in addition to the success rate obtained for context inference this technique also aims to extract metrics related to middleware performance; for example, the response time for processing and delivering notifications for processed events.

REFERENCES

[1] S. Neto, M. Valéria, P. Manoel, and F. Ferraz, "Publish/subscribe cloud middleware for real-time disease surveillance," *ICSEA 2015 Tenth Int. Conf. Softw. Eng. Adv.*, no. c, pp. 150–157, 2015.

[2] M. Andres, "Smart Cities Concept and Challenges: Bases for the Assessment of Smart City Projects," *Transp. Res. Centre, Univ. Politécnica Madrid, Prof. Aranguren s/n, Madrid, Spain*, pp. 11–21.

[3] S. P. Pawar, "Smart City with Internet of Things (Sensor networks) and Big Data," *Academia.edu*, no. 9860027825, p. 10.

[4] M. Jung, J. Weidinger, W. Kastner, and A. Olivieri, "Building Automation and Smart Cities: An Integration Approach Based on a Service-Oriented Architecture," *Adv. Inf. Netw. Appl. Work.*, pp. 1361–1367, 2013.

[5] S. Wahle, T. Magedanz, and F. Schulze, "The OpenMTC framework - M2M solutions for smart cities and the internet of things," *2012 IEEE Int. Symp. a World Wireless, Mob. Multimed. Networks, WoWMoM 2012 - Digit. Proc.*, pp. 2–4, 2012.

[6] J. Wan, D. Li, C. Zou, and K. Zhou, "M2M communications for smart city: An event-based architecture," *Proc. - 2012 IEEE 12th Int. Conf. Comput. Inf. Technol. CIT 2012*, pp. 895–900, 2012.

[7] A. Elmangoush, H. Coskun, S. Wahle, and T. Magedanz, "Design aspects for a reference M2M communication platform for Smart Cities," *2013 9th Int. Conf. Innov. Inf. Technol. IIT 2013*, no. MARCH 2013, pp. 204–209, 2013.

[8] A. Chanda, K. Elmeleegy, A. L. Cox, and W. Zwaenepoel, "Composite Subscriptions in Content-Based Publish/Subscribe Systems," vol. 3790, no. December, pp. 42–59, 2005.

[9] B. Schilling, B. Koldehofe, and K. Rothermel, "Efficient and distributed rule placement in heavy constraint-driven event systems," *Proc.- 2011 IEEE Int. Conf. HPCC 2011 - 2011 IEEE Int. Work. FTDCS 2011 -Workshops 2011 Int. Conf. UIC 2011- Work. 2011 Int. Conf. ATC 2011*, pp. 355–364, 2011.

[10] S. Pellicer, G. Santa, A. L. Bleda, R. Maestre, A. J. Jara, and A. G. Skarmeta, "A global perspective of smart cities: A survey," *Proc. - 7th Int. Conf. Innov. Mob. Internet Serv. Ubiquitous Comput. IMIS 2013*, pp. 439–444, 2013.

[11] R. Giffinger, "Smart cities Ranking of European medium-sized cities," *October*, 2007. [Online]. Available from: http://linkinghub.elsevier.com/retrieve/pii/S02642751980005 0X 2016.07.11

[12] X. Lin, H. Quan, H. Zhang, and Y. Huang, "The 5I Model of Smart City: A Case of Shanghai, china," *2015 IEEE First Int. Conf. Big Data Comput. Serv. Appl.*, pp. 329–332, 2015.

[13] Y. Fujiwara, K. Yamada, K. Tabata, M. Oda, K. Hashimoto, T. Suganuma, A. Rahim, P. Vlacheas, V. Stavroulaki, D. Kelaidonis, and A. Georgakopoulos, "Context Aware Services: A Novel Trend in IoT Based Research in Smart City Project," *Comput. Softw. Appl. Conf. (COMPSAC), 2015 IEEE 39th Annu.*, vol. 3, pp. 479–480, 2015.

[14] J. Singh, D. M. Eyers, and J. Bacon, "Disclosure Control in Multi-Domain Publish / Subscribe Systems," *ACM Int. Conf.*

Distrib. event-based Syst., no. Ic, pp. 159–170, 2011.

[15] C. Esposito and M. Ciampi, “On security in publish/subscribe services: A survey,” *IEEE Commun. Surv. Tutorials*, vol. 17, no. 2, pp. 966–997, 2015.

[16] M. Corporation, “Publish/Subscribe,” 2004. [Online]. Available from: https://msdn.microsoft.com/en-us/library/ff649664.aspx 2016.07.11

[17] A. Carzaniga, M. Papalini, and A. L. Wolf, “Content-based publish/subscribe networking and information-centric networking,” *Proc. ACM SIGCOMM Work. Information-centric Netw. - ICN ’11*, no. 1, p. 56, 2011.

[18] R. Baldoni, L. Querzoni, and A. Virgillito, “Distributed Event Routing in Publish / Subscribe Communication Systems : a Survey,” *Tech. Rep.*, pp. 1–27, 2005.

[19] J. L. Martins and S. Member, “Routing Algorithms for Content-Based Publish / Subscribe Systems,” *Communications*, vol. 12, no. 1, pp. 39–58, 2010.

[20] Google, “Google Cloud Platform,” 2015. [Online]. Available from: https://cloud.google.com/appengine/docs/whatisgoogleappengine 2016.07.11

[21] Amazon, “Amazon Simple Notification Service (SNS),” 2016. [Online]. Available from: https://aws.amazon.com/sns/ 2016.07.11

[22] H. Chai and W. Zhao, “Towards Trustworthy Complex Event Processing,” pp. 1–4.

[23] D. B. Robins, “Complex Event Processing,” *2010 Second Int. Work. Educ. Technol. Comput. Sci.*, p. 10, 2010.

[24] M. Antunes, D. Gomes, and R. Aguiar, “Semantic-Based Publish/Subscribe for M2M,” *2014 Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov.*, pp. 256–263, 2014.

[25] P. B. P. Tongkamonwat, “IFIX: A new information exchange framework for financial organizations,” *Adv. Informatics Concepts, Theory Appl. (ICAICTA), 2015 2nd Int. Conf.*, vol. 16, pp. 1 – 5, 2015.

[26] ISO, “ISO 8583-1:2003,” 2003. [Online]. Available from: https://www.iso.org/obp/ui/#iso:std:iso:8583:-1:ed-1:v1:en 2016.07.11