A Generic and Scalable IoT Data Fusion Infrastructure

Vangelis Nomikos, Ioannis Priggouris, George Bismpikis, Stathes Hadjiefthymiades and Odysseas Sekkas

Abstract Applications in the IoT domain are in need of information coming from many different sources. To mitigate the processing overhead for increased volumes of raw data (seen at the application layer) on vast pervasive networks (a significant pillar of 5G networks), we introduce a customizable middleware platform. Such platform allows the treatment of incoming data flows through complex, yet fully specifiable/controllable data processing workflows. The derivation of new information through this high level processing task is termed data fusion, hence, the presented architecture is named "Fusion Box". The middleware platform is customizable through a Domain Specific Language that allows the domain (and not the IT) expert to easily specify the needed processing and automatically transform such specification to executable workflows. The coupling between the Domain Specific Language and the Fusion Box is based on the concept of contextors, a versatile processing unit that can be instantiated and managed in many different ways as needs dictate.

Keywords Internet of Things · Data processing · IoT middleware · Software abstractions · Domain specific language · Ubiquitous computing

V. Nomikos (✉) · I. Priggouris · G. Bismpikis · S. Hadjiefthymiades
Department of Informatics and Telecommunications, Pervasive Computing Research Group, National and Kapodistrian University of Athens, Athens, Greece
e-mail: vnomikos@di.uoa.gr

I. Priggouris
e-mail: iprigg@di.uoa.gr

G. Bismpikis
e-mail: gbismpikis@di.uoa.gr

S. Hadjiefthymiades
e-mail: shadj@di.uoa.gr

O. Sekkas
Research Department, Mobics, Athens, Greece
e-mail: sekkas@mobics.gr

C.X. Mavromoustakis et al. (eds.), *Internet of Things (IoT) in 5G Mobile Technologies*, Modeling and Optimization in Science and Technologies 8,
DOI 10.1007/978-3-319-30913-2_13

1 Introduction

The vision of an Internet of Things (IoT) has attracted the interest of researchers and practitioners aiming to deliver innovative applications that improve aspects of our daily lives. Although, over the past decade, advances in hardware development, sensing capabilities and IoT software architectures have triggered important technical and commercial successes, challenges and opportunities persist as we move towards generic and scalable approaches for composing and interoperating existing IoT functionality [1, 5, 9].

By 2020, industry analysts estimate that 25 billion devices will be connected to mobile networks worldwide.[1] To cope with the explosion of connected devices that will be part of the IoT, a new level of wireless internet connectivity will be required. 5G is the name being given to the next generation of wireless networks and it is envisioned to make things go smoothly [15]. We are still in an early stage of the 5G revolution but the key objectives that will unlock the potential of IoT have been set. The higher transmission data rates will enable mobile devices to send more bits of information to other devices, gateways, or cloud-based infrastructures. Also, the latency (network response times) of 5G is expected to be just a single millisecond (i.e., 50 times faster than 4G) and the increased reliability factor is something particularly important for industrial and mission critical IoT applications (e.g., fire detection).

Our work focuses on prototyping a generic and scalable architecture that will enable processing and consolidating data from heterogeneous sources. The Fusion Box (FB), as we call our prototype, facilitates the integration and interpretation of different types of data sources, through the definition and execution of multiple algorithmic flows triggered by these sources. The main concept behind this is to derive new indices and metrics based on a combination of data flows. The use of multiple types of data sources increases the accuracy with which a quantity or phenomenon can be observed, interpreted and used for event matching/recognition, while redundancy can provide an improved estimate of a physical measurement. The most fundamental mechanism of the FB involves (i) the detection of pre-defined events, (ii) the decision or inference regarding the characteristics of an observed entity and (iii) an interpretation of the observed entity in the context of its surroundings and relationships to other entities. The FB context detection capability is among the key characteristics of IoT middleware [2, 13].

Our intentions in building the FB are multi-fold. Specifically, we aimed to:

- shift the complexity and computational load from the application to a versatile middleware platform,
- perform complex operations efficiently even under increased workload,
- allow the integration of heterogeneous information sources, a key aspect in IoT,

[1] OpenStreetMap (2015). Retrieved June 1st from http://www.openstreetmap.org.

- enable the customization of the middleware processing according to the specific requirements of a wide spectrum of application scenarios, and,
- enable the universal modeling of data processing workflows and their effortless specification by application domain experts.

To cope with the listed needs we have introduced a workflow processing engine built around basic blocks termed contextors that handle autonomous algorithmic steps, structured in a way that satisfies high scalability requirements and comes along a Domain Specific Language.

The FB architecture is far more versatile than existing IoT platforms [4, 8, 16] as it allows full customization and support of the application domain requirements. Existing IoT platforms impose a certain (static) processing in the collected data on their route from the data collection layer to the application or focus on different aspects of IoT (e.g., network support). Even the Complex Event Processing (CEP) middleware that exists nowadays has quite limiting semantics that are largely dependent upon event occurrence [7, 17].

2 Basic Principles: Contextors and Algorithms

The architecture of the Fusion Box draws its principles from the theory of contextors [6] and leverages the Open Geospatial Consortium (OGC) Sensor Observation Service (SOS) standard for the interoperable integration of data sources [11]; an important catalyst of IoT solution proliferation. In its initial conception, as described in [6], the contextor is introduced as an abstract functional unit, which defines some inbound data flows and a single outbound flow, generated by the contextor's functional core. This core is what, actually, differentiates one contextor from another, defining its behavior; that is what its outbound flow will be based on the received input. In addition each contextor defines an inbound and an outbound control flow; the former is used for controlling its operation and the latter for producing control directives towards other contextors. These control flows allow for the creation of complex structures composed from a taxonomy of a limited number of elementary contextors (i.e., Elementary, History, Threshold, Translator, Fusionor, Abstractor).

Our conception of a contextor is that of an abstract software entity which, like in principal theory, features one or more inputs, has a single output and defines a functional core consisting of a software implemented algorithm used for transforming the inbound flows to a specific outbound flow. The term flow is intentionally used here since our contextor has been designed to handle time-aligned data flows and produce outputs which are both context sensitive and time specific. The conceptual design of the FB contextor is depicted in Fig. 1.

The basic deviation from the principal contextor is that contextors inside the FB do not define any discrete control flows. This modification was based on the assumption that control commands can be considered as special data flows and as

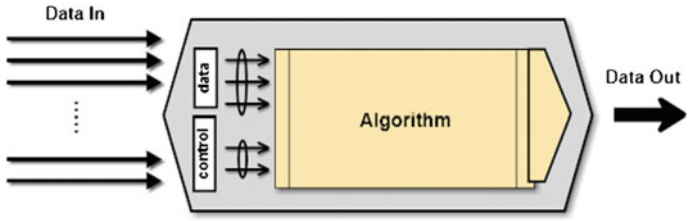

Fig. 1 Conceptual design of the FB contextor

such they can be integrated with the data flows, without discount in the level of achieved control functionality. For example, an on/off control functionality in our consideration is nothing more than a 0/1 data flow, stemming out from a contextor's outbound flow towards another contextor's inbound flow. Moreover, the FB contextor does not use the taxonomy of contextors defined in [6]. We took a more generic approach by defining an abstract contextor, which encapsulates a single algorithm or operator.

3 The Fusion Box Architecture

The core principle of the Fusion Box design is the provisioning of the necessary middleware services in order to handle the full lifecycle of a contextor. In this perspective, the FB acts as a sandbox for deploying and running complex data processing workflows (DPWs) composed of multiple contextors. It provides services both on the microscopic level of the contextor as well as on the macroscopic level of the data processing workflow, thus allowing a contextor to:

- Retrieve data from different data sources
- Execute a dynamically selected algorithm from a predefined set in order to produce a specific output
- Disseminate the produced output

Moreover, the FB provides the necessary infrastructure services to enable:

- The exchange of information between different contextors through an enterprise messaging framework
- The dynamic deployment and provisioning of DPWs consisting of multiple contextors
- The dispatching of the output produced by DPWs either within the same FB or outside the FB

Taking a bird's eye view of the FB architecture (Fig. 2), we see its southbound interface towards data sources. Heterogeneous sources are one of the most important characteristics of the IoT. These sources can be any IoT device or anything else

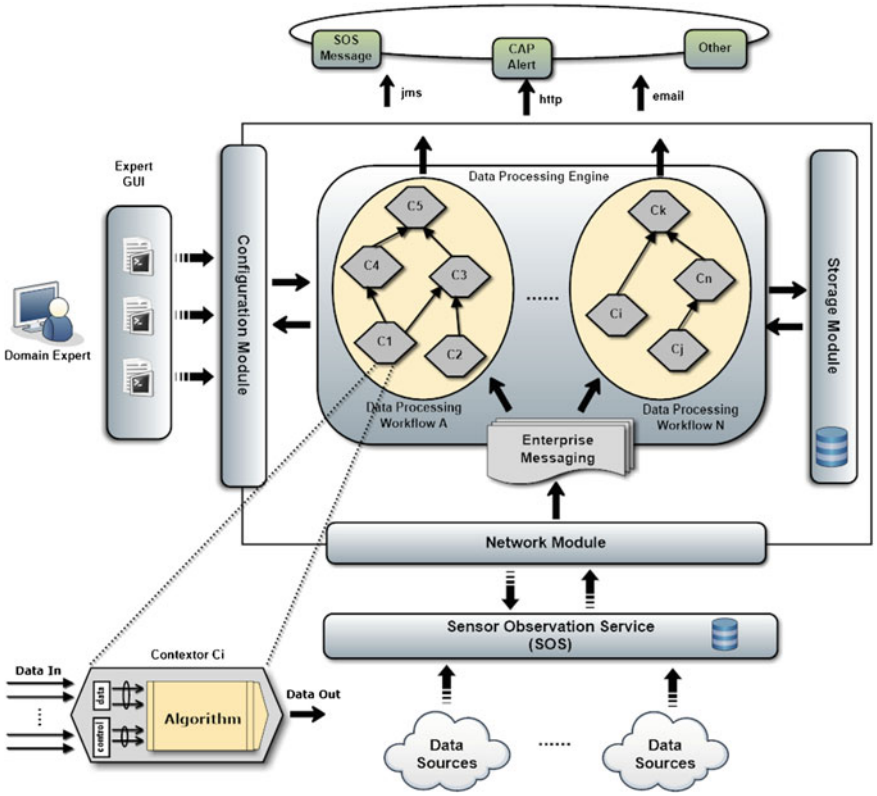

Fig. 2 The high level architecture of the FB

capable to provide a data flow. We opted to use an SOS compliant interface in this boundary in order to achieve interoperability between sources and provide an open and layered architecture [13].

However, SOS proved highly extensible and capable to accommodate data flows coming from other sources as well. It has also inherent support for relaying and storing the position and the timestamp of a data element which is a highly desirable feature. Surely, many real world data sources do not currently support SOS, which means that additional work needs to be done in order to transform raw data flows to SOS compliant flows for our system. However, its adoption is highly advantageous as it conceals the complexities of different underlying communication protocols and data formats, allowing for a seamless interoperable integration of external data sources. The current SOS implementation is based on 52o North's reference implementation and uses PostgreSQL database with spatial extensions enabled [18].

In order to achieve its role as a data processing workflow middleware sandbox, the FB features a modular approach. From an implementation perspective, the FB modules are software entities that have been developed using enterprise software development abstractions and technologies (e.g., JBoss Application Server, JavaEE).

3.1 The Network Module

The Network Module (NM) handles information flows from external data sources that have been selected as inputs by a specific data processing workflow. In this perspective it provides the appropriate services for enabling a DPW to query and select the appropriate sources that will feed with data its contextors. This selection is performed upon the deployment of a new DPW application and results to the establishment of a DPW specific data source context, composed by the set of selected data sources (Fig. 3). This context is kept alive, inside the network module, throughout the lifetime of the workflow acting as a peer entity to the DPW that monitors the activity of the selected sources and dispatches their incoming data to the DPW application. The module provides a rich interface towards the DPW application allowing selection of sources based on multiple criteria from the selection of specific sources based on their identifiers to spatial queries (e.g., an area of interest) or queries based on specific characteristics of the source (e.g., the data type of the flow). Such selection criteria are defined inside the DPW script where specific directives of a domain specific language (i.e., DPWDL) are used for this purpose.

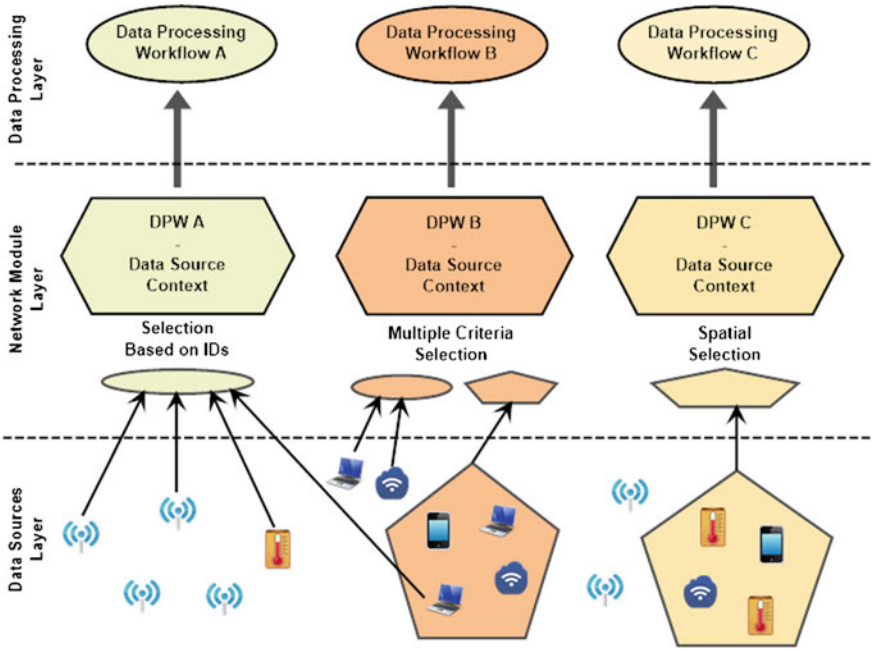

Fig. 3 Network module: the concept of the DPW data source context

3.2 The Configuration Module

The Configurator Module (CM) of the FB is a component that receives the DPW script, parses and validates the provided information which is described through the DPWDL and creates the binary/executable entity of the corresponding DPW application. A DPW application comes in the form of a java Enterprise ARchive (EAR), which is automatically generated by combining the DPWDL directives, with a template DPW application archive and injected into the data processing core engine (Fig. 4). The Configuration Module provides a very simple API which allows for three main operations:

- Deployment of a new DPW application
- Undeployment of an existing DPW application
- Retrieval of DPW application metadata

Retrieval of application meta-information is also possible through HTTP, with the information returned in JSON like format.

3.3 The Data Processing Engine

The Data Processing Engine (DPE) comprises the application layer of the FB architecture, which provides the appropriate runtime environment for all DPW applications. As already mentioned, a DPW application consists of several contextors interconnected such as one's output serves as input for another, so that all together to form a directed acyclic graph (DAG). The structural elements of this graph are instances of the FB contextor, where a specific algorithm is encapsulated and runs within each instance. A list of the algorithms/operators that have been implemented and are currently available for use by the FB contextor is displayed in Table 1. Surely, this pool of algorithms is not static; new algorithms conforming to the directives that

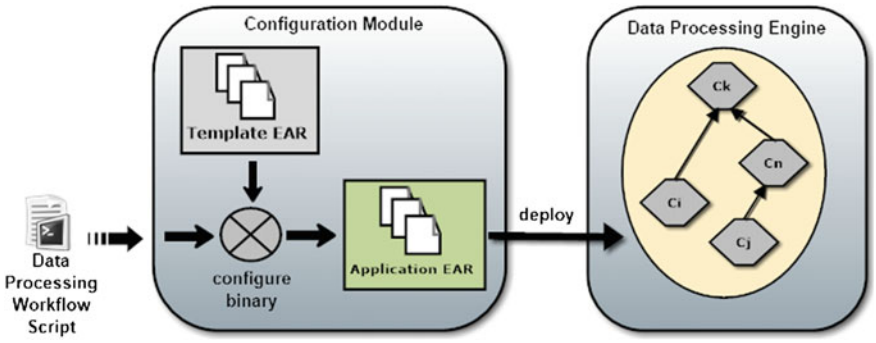

Fig. 4 Creation and deployment of a new DPW application

Table 1 Pool of algorithms and operators that is currently available in the DPE

Algorithm/Operator	Description
Arithmetic aggregation operators	e.g., minimum, maximum, average, median
Belief aggregation operators	e.g., probabilistic, Lukasiewicz
Linear and symmetric opinion pool	Combines the probabilities of different sources to produce a social probability
Dempster-Shafer rule of combination	Part of the Dempster-Shafer theory. Combines evidence from two or more sources to form inferences
Voting algorithms	e.g., threshold voting algorithms [12]
Missing value substitution algorithms	e.g., current mean value, polynomial extrapolation
Cumulative sum detection algorithm	Detects a change on a distribution of a time series $x_t \in R$ w.r.t. a target value
Shewhart control chart	A variable x_t is detected to deviate at time k from its normality denoted by two control limits: the Upper Control Limit (UCL) and Lower Control Limit (LCL) [3]
Crisp value Bayesian network	A probabilistic graphical model that represents a set of random variables (with crisp values) and their conditional dependencies [14]

the FB contextor architecture dictates can be easily implemented and added to this pool, thus, enriching the set of available algorithms.

From an operational perspective, the execution of each contextor is completely event based, triggered by the reception of a new data element in any of its data-in channels. This data element comes either from the data out channel of another contextor or from the DPW data source context that has been configured, inside the network module. All data dispatching inside the core is handled by the FB enterprise service middleware which has been implemented on top of the HornetQ Message-oriented Middleware (MoM).

Another important aspect of a DPW application running inside the FB is the way it dispatches data outside its infrastructure. Here the FB adopts a rather abstract approach. There is no particular built in support for relaying data to external entities. What FB does instead is to integrate the data dispatch functionality inside the DPW application per se. This is achieved through a specialization of the FB contextor, which we call "Output Contextor". In its conception, the output contextor is completely alike the FB contextor featuring a similar structure. The only exception is that the single algorithmic part in its core has been replaced by a pipeline of two functional entities (Fig. 5) namely: (a) the formatter and (b) the adapter.

The latter handles the actual dispatching of data to external entities using a specific transmission protocol, while the former formats this data prior its relay outside the FB. Again, there is a pool of specific formatters and adapters that has been implemented and is available for immediate use in DPW applications. As applies in the case of the algorithms also, this list can be easily enriched by new implementations. Currently, we opted to support all major internet-related delivery

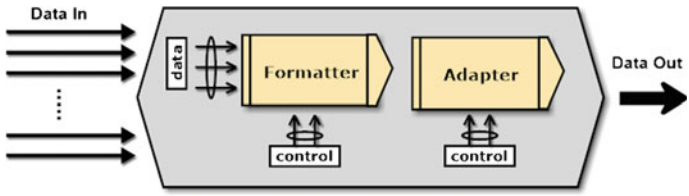

Fig. 5 Conceptual design of the output contextor

Table 2 List of output formatters supported by the FB

Formatter	Description
CAP	Formats the application output to produce a Common Alerting Protocol compliant message (a.k.a., an alert)
SOS	Formats the application output to produce a Sensor Observation Service compliant message

Table 3 List of output adapters supported by the FB

Adapter	Description
CAP	Dispatches CAP formatted messages. The need for this specific adapter stems from the fact that the CAP protocol needs to maintain some kind of session information for the CAP alerts that are dispatched
Email	Dispatches formatted message of any type to specific email recipients
HTTP	Relays the formatted output message through HTTP to a specific URL
MQ	Dispatches the formatted message to a specific JMS endpoint

methods (e.g., http, email) along with some more sophisticated such as java message service (jms) and short message service (sms). Information elements can be delivered both in SOS format, mainly for the case of application-to-application communication, as well as more user-friendly formats such as plain text or domain specific formats such as the Common Alerting Protocol (CAP) [10]. Tables 2 and 3 below provide a comprehensive list of available FB implementations for both formatters and adapters.

3.4 The Fusion Box Expert User Interface

The Fusion Box Expert (Graphical) User Interface is an easy to handle management tool targeting the FB user. The tool is web based and provides functionalities for:

- Monitoring of the input data sources
- Deploying/undeploying of DPW scripts
- Monitoring of deployed DPW applications

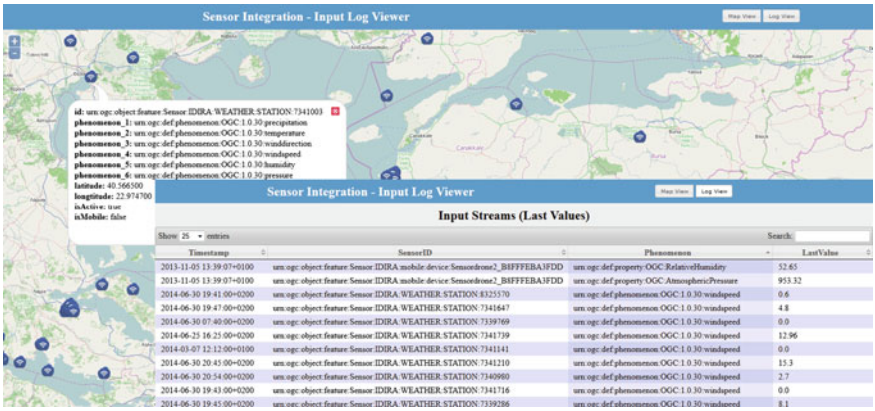

Fig. 6 Functionalities of the Fusion Box Expert GUI: **a** map view and **b** last reported measurements

The definition of a DPW script dictates the need to know the metadata (e.g., type of data, location) of the available data sources. Only if this information is provided, the domain expert is able to successfully select the input data streams and compose the data processing workflow. Currently, this functionality is provided by the map view which depicts the already integrated data sources in the SOS part of the FB. For that purpose, a base map (e.g., OpenStreetMap[2]) with clickable descriptive icons representing the available sources is used and enables the domain expert to spatially identify the necessary data sources.

However, in order to demonstrate a more dynamic view of the underlying data sources, a monitoring mechanism for the last reported measurements is also provided. The information is visualized in a tabular format and is related to:

- the unique identifier of the data source that reported the value,
- the timestamp that the measurement has been taken place,
- the phenomenon that this measurement is related with and
- the reported value

In a case where a lot of data streams appear on the table, a searching mechanism along with a sorting functionality assists the user to obtain the needed information. Figure 6 illustrates snapshots of a real world deployment of the FB Expert GUI.

For the definition and management of DPW scripts, the Expert GUI provides all necessary functionalities which enable the domain expert to load DPW scripts and initiate their deployment to the FB Configuration Module. The Expert GUI provides three options for deploying DPW applications:

[2]OpenStreetMap (2015). Retrieved June 1st from http://www.openstreetmap.org.

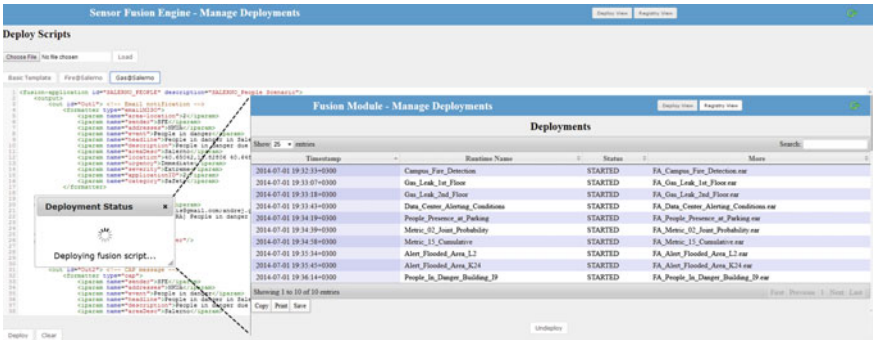

Fig. 7 Deployment, monitoring and undeployment functionalities of the FBox Expert GUI

- Upload them from the local file system
- Upload pre-defined/built-in DPW scripts
- Write them from scratch in the provided text editor

The GUI provides also a monitoring view for the DPW applications already deployed. This view displays a list of all deployed DPW applications, along with their current status (i.e., deployed, started, stopped) as well as means for undeploying these applications. Figure 7 depicts a snapshot for the aforementioned capabilities of the FBox Expert GUI.

4 The Data Processing Workflow Description Language

A data processing workflow comprises from a directed acyclic graph of contextors, with defined external inputs and outputs. Each input is a data flow coming from the network module, while each output is another data flow dispatched in specific format using a specific protocol adapter. For the purpose of defining data processing workflows, inside the FB, an XML-based language has been devised. The Data Processing Workflow Description Language (DPWDL) provides structures that allow defining:

- The input data flows
- The set of contextors along with their configuration parameters
- The connections between the defined contextors
- The format of the desired output, along with the transmission protocol/adapter which will be used for the dissemination of data produced

Domain experts can use this language to create scripts, which can be subsequently injected in the FB. Each DPW application is represented by a unique identifier accompanied by a short description as well.

From a structural perspective, a DPW script is divided into three distinct mandatory sections. The first section is defined through the “streamers” tag. In this

section, the input data sources are specified. The selection of the input streams can be based on spatial criteria and/or through direct definition of specific characteristics of the data sources or their unique identifiers. The streamers section constitutes the bottom layer of every FB application. They, actually, define which information sources are relevant to our application.

The second section is represented by the "contextors" tag where all the contextors that participate in the data processing workflow are declared. For the proper definition of a contextor, the algorithm's name (i.e., algorithm tag), the parameter(s) (i.e., iparam tag) and the input(s) (i.e., src id tag) must be provided. If a contextor encapsulates a single-input single-output algorithm and there are more than one inputs, the configurator module of the FB will instantiate so many instances of this contextor as the cardinality of input channels and a different input data flow will be assigned to each one. This section is correlated to the middle layer of a FB application. It is where all processing takes place. The application developer simply defines contextors to use, parameterizes their underlying algorithms and defines how they are interconnected, aiming to process, shape and channel the appropriate information towards the top layer. Algorithms to use are selected from the existing pool presented in Table 1 but the extensibility of the system allows him to build a new one, inject it to the existing pool specifying a unique name for it, and then use it.

The last section is defined through the "output" tag and contains all the information pertaining to the formatting and dispatching of the FB output result to every interested entity. Hence, for every output block, a formatter and an adapter must be defined as a pair through the corresponding DPWDL tags. The configuration of the selected formatter and adapter is realized through the generic "iparam" tag. This section of the script builds the top layer of the application which handles the dissemination of the produced information towards external entities/systems, other FB applications, or other FB instances.

Table 4 presents the mapping between the main logical concepts of a DPW script and their counterparts in DPWDL terms.

4.1 An Example DPW Script: Detection of a Fire Event

In order to make clearer how the language is used, we present a specific scenario, which was developed to assess the functionality of the system during the field-trials of the IDIRA project. The scenario implements a fire event detection case; it is fairly simple but quite representative of how DPW applications can be built and executed inside the FB, using the DPWDL.

Let us consider the case where an FB user is interested in the detection/monitoring of an event such as forest fires, a quite common threat in the wider area of Mediterranean basin countries due to their climate and geographic morphology. We assume that in the monitored region a variety of sensors sources has already been deployed such as water level, pluviometer, humidity, temperature, vision sensors and a number of weather stations that can monitor multiple phenomena.

Table 4 Structural blocks of a DPW script

Logical concepts	DPWDL structural blocks

Output formatting and dispatching	`<output>`
	`  <out id="...">`
	`  <formatter type="...">`
	`  <iparam name="...">...</iparam>`
	`  </formatter>`
	`  <adapter type="...">`
	`  <iparam name="...">...</iparam>`
	`  </adapter>`
	`  <sources>`
	`  <src id="..."/>`
	`  <src id="..."/>`
	`  </sources>`
	`  </out>`
	`</output>`

Data processing workflow of contextors	`<contextors>`
	`  <contextor id="...">`
	`  <algorithm name="...">`
	`  <iparam name="...">...</iparam>`
	`  </algorithm>`
	`  <sources>`
	`  <src selector="..."/>`
	`  <src selector="..."/>`
	`  </sources>`
	`  </contextor>`
	`</contextors>`

Selection of data sources	`<streamers>`
	`  <streamSelector id="...">`
	`  <select type="spatial">...</select>`
	`  <select type="phenomenon">...</select>`
	`  </streamSelector>`
	`</streamers>`

Although this sensory infrastructure is already integrated into the FB through the SOS, the simple interpretation of these data is not enough to provide the necessary information for the detection of the hazardous event of interest. The collection and the combination of data that stem from heterogeneous sources though, could provide a more comprehensible figure to the domain expert. For example, measurements from water level and pluviometer sensors are inappropriate for producing estimates related to the fire incident while weather stations, temperature, humidity and vision sensors that reside near the area of interest are of paramount importance. Data sources that are far from the observed area do not provide any added value to the data processing process and as consequence they are ignored during the definition of the necessary DPW script.

Having the aforementioned in mind, the expert concludes to an information flow where the incoming data are combined in a phenomenon based approach. There is no need for a priori knowledge on the available sensor identifiers in the region, just the spatial determination of the area. On the one hand, all the temperature streams are monitored for anomalies in their distributions by applying change detection

algorithms in order to trace possible increase in the received values while the humidity streams must be observed for respective decreases. From each category, the percentage of sensors that deviates from the expected phenomenon value is kept as intermediate information. These percentages are forwarded to the next level of the workflow where the heterogeneous and also independent values are combined by applying the appropriate weight for their reliability based on the experience of the domain expert on the factors that imply a forest fire break out. On the other hand, the received values from the vision sensors are also accounted as a different category/group since these are preprocessed estimations regarding the possibility of a fire incident in their current field of view. The average of the reported values is taken into account and is combined with the resulted value from the association of the temperature and humidity outcomes. The weights in this occasion are equally distributed. The output of the final level from this fusion procedure provides a belief of the occurrence probability for the fire phenomenon. Should the probability is greater than 60 % the domain expert will be notified through an email alert in his personal account for the incident's detection. Figure 8 depicts the graphical representation of the analyzed information flow.

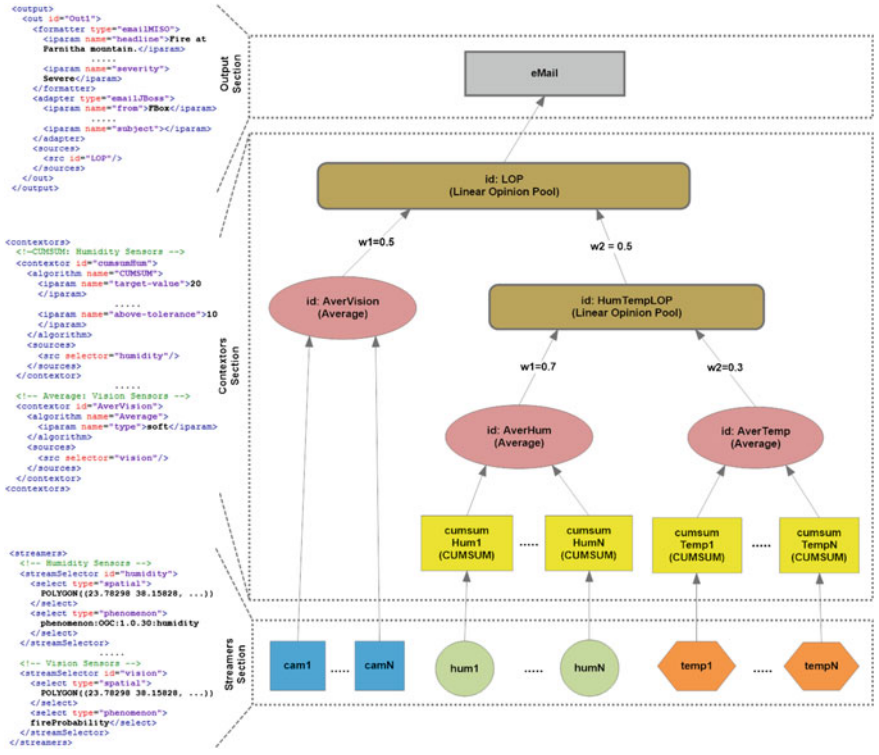

Fig. 8 Graphical representation of the example DPW script. The original script is available at http://satia.di.uoa.gr/iot45g/Example_DPW_Script.xml

5 The Fusion Box Ecosystem

The generic and versatile architecture of Fusion Box along with the high degree of expressiveness of the Data Processing Workflows Language allow its operation based on a variety of patterns of use. At the most common operation scenario a single instance of the FB is running on a single machine and the external data sources are integrated through the appropriate interface. Multiple DPW scripts are injected into the FB and new applications are instantiated over raw data in order to produce higher level information elements. These elements can next be formatted in a set of supported formats and transmitted to external entities (e.g., a decision support system) using a set of common communication protocols. In a more advanced operation pattern (Fig. 9) a single instance of the FB hosts a set of different DPW applications and enables the inter-application communication leveraging the interoperable way to integrate data into the FB. The output of an application is encoded in SOS format and becomes input for other DPW applications running on the same FB instance.

In order to handle a large number of data flows and derive more complex information elements a federated multi-layer FB architecture can be used (Fig. 10). Different FB instances hosts blocs of DPW applications and they are distributed on different dedicated machines or on cloud-based infrastructures. Hence, the potential of creating such a multi-layer architecture allows for increased performance and scalability by distributing workload among FB instances participating in the federation.

Such versatility is largely connected to the 5G ecosystem. The various patterns of FB usage can be realized by means of machine-to-machine communications

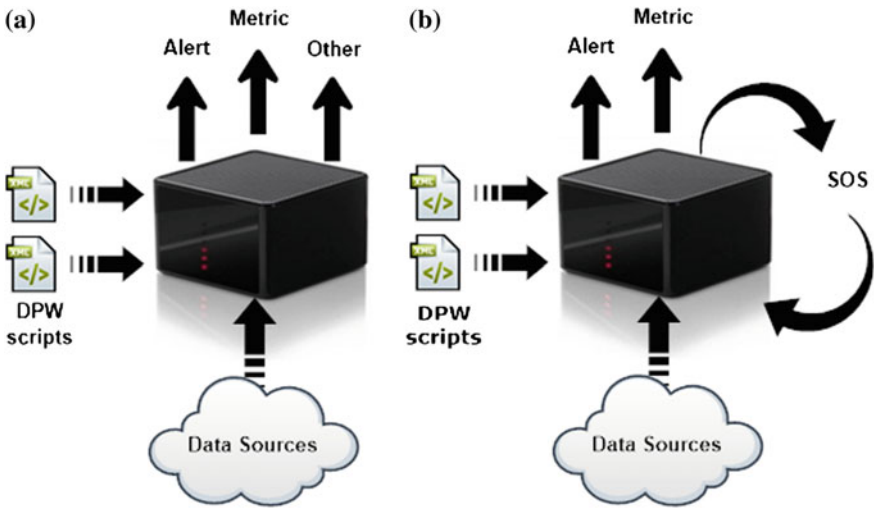

Fig. 9 **a** Single instance of FB, **b** Single instance of FB with inter-application communication

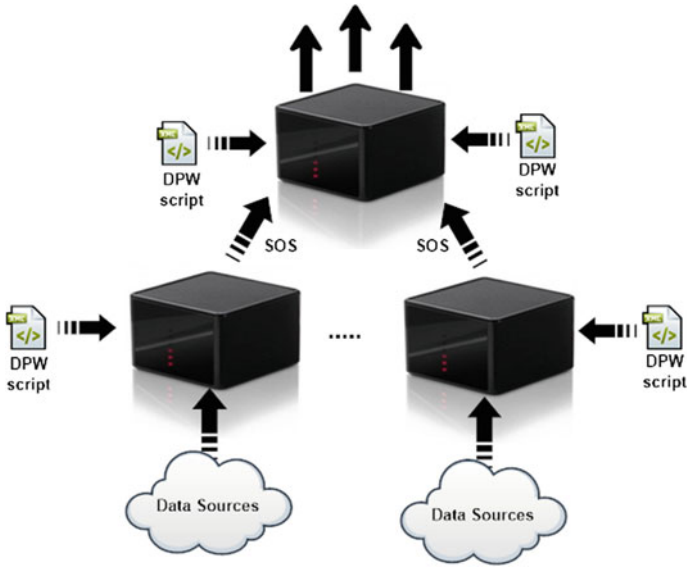

Fig. 10 Multiple instances of the FB in a federated multi-layer architecture

which in the 5G landscape can support a vast number of devices and associated streams. The FB capability to filter and fuse volumes of data contributes to the scale vision of 5G.

6 Conclusions and Platform Extensions

Fusion Box is more than just a platform for developing IoT applications based on the interpretation of raw data. It is a generic and scalable architecture based on well-defined modeling and programming approaches. The DPWDL simplifies the creation of new DPW applications allowing for the realization of a dynamic and scalable model which enables algorithmic workflows to be defined and run over raw data, in order to produce higher level information elements. These elements can next be used either by external entities or become input for other DPW applications, running on the same FB instance or external FB instances (i.e., in a federated FB architecture), in the perspective of producing even higher level information elements and at the same time enhancing the scalability of the system. The possible patterns of FB use are well harmonized with the needs set forth by the 5G technology. The FB greatly facilitates the operation of pervasive networks but also capitalizes on the M2 M communication infrastructure that 5G offers.

Our plans for further enhancing the basic FB architecture presented above include the enrichment of the algorithmic pool that is currently available. Hence, the

FB will be able to cope with a much wider spectrum of data processing workflows and deliver more advanced IoT solutions. Additionally, we aim to address the requirement of setting up and running data processing workflows in a fully dynamic manner. For this need we are developing a meta-contextor framework which will receive external stimuli and dynamically organize and instantiate contextors as the application scenario dictates.

Acknowledgments This work has been partially supported by European Commission within the 7th Framework Programme through project IDIRA (Interoperability of Data and procedures in large-scale multinational Disaster Response Actions), contract FP7-SEC-2010-1. An application of the presented platform in the fire detection domain (environmental risk) has been pursued in the context of Research Funding Programme Thales through project SWeFS (Sensor Web Fire Shield), contract THALES-180, funded by the Greek Ministry of Education (Operational Program "Education and Lifelong Learning").

References

1. Atzori, L., Iera, A., Morabito, G.: The Internet of Things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)
2. Bandyopadhyay, S., Sengupta, M., Maiti, S., Dutta, S.: Role of middleware for Internet of Things: a study. Int. J. Comput. Sci. Eng. Surv. **2**(3), 94–105 (2011)
3. Basseville, M., Nikiforov, I.V.: Detection of Abrupt Changes: Theory and Application, vol. 104. Prentice Hall, Englewood Cliffs (1993)
4. Blackstock, M., Kaviani, N., Lea, R., Friday, A.: MAGICBroker 2: an open and extensible platform for the Internet of Things. In: Internet of Things (IOT), pp. 1–8. IEEE (2010)
5. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013)
6. Coutaz, J., Rey, G.: Foundations for a theory of contextors. In: Computer-Aided Design of User Interfaces III, pp. 13–33. Springer (2002)
7. Liang, D., Wang, D., Sheng, H.: Design of RFID middleware based on complex event processing. In: Cybernetics and Intelligent Systems, pp. 1–6. IEEE (2006)
8. Kovatsch, M., Lanter, M., Duquennoy, S.: Actinium: a RESTful runtime container for scriptable Internet of Things applications. In: Proceedings of the 3rd International Conference on the Internet of Things, pp. 135–142. IEEE (2012)
9. Miorandi, D., Sicari, S., De Pellegrini, F., Chlamtac, I.: Internet of things: vision, applications and research challenges. Ad Hoc Netw. **10**(7), 1497–1516 (2012)
10. OASIS: Common Alerting Protocol (CAP) Version 1.2. http://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.html (2014). Accessed 10 June 2014
11. OGC: Open Geospatial Consortium, Sensor Observation Service (SOS). http://www.opengeospatial.org/standards/sos (2014). Accessed 15 May 2014
12. Parhami, B.: Voting Algorithms. IEEE Trans. Reliab. **43**(4), 617–629 (1994)
13. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the Internet of Things: a survey. Commun. Surv. Tut. IEEE **16**(1), 414–454 (2014)
14. Phoha, S., La Porta, T.F., Griffin, C. (eds.): Sensor Network Operations. John Wiley & Sons (2006)
15. Rodriguez, J.: Fundamentals of 5G Mobile Networks. John Wiley & Sons (2015)
16. Wang, F., Zhou, J., Zhao, K.: A data processing middleware based on SOA for the Internet of Things. J. Sens. (2015)

17. Wu, E., Diao, Y., Rizvi, S.: High-performance complex event processing over streams. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, pp. 407–418. ACM (2006)
18. 52o North: Reference implementation of Sensor Observation Service (SOS). http://52north.org/downloads/sensor-web/sos (2014). Accessed 22 June 2014