functionality using mobile computing, rather than a VAN. None of the above works take into consideration the specific requirements of medical waste and are therefore not directly applicable to the problem at hand.

The rest of the paper is structured as follows: Section **??** discusses the medical waste life-cycle and Section **??** the System architecture in terms of the functionality of each of the distributed components. Section **??** discusses in detail the layered approach that is adopted in this work, separating concerns between the data acquisition (communications) layer, the business logic (event-based middleware) layer and the application layer. Included is also a brief discussion of the implementation setup (**??**). Next, Section **??** discusses system evaluation, presenting the main benefits through simple KPI analysis while Section **??** discusses related work from the literature. Finally, Section **??** concludes.

II. Hospital Waste Life-cycle

Waste is collected in special cardboard containers, lined with a hardened plastic bag. The first step in the lifecycle is the preparation of these containers that are delivered to the contractor in pallets. As pallets that contain flat containers are opened, the pallet barcode is scanned first; next, each container is assembled and it is tagged with an RFID tag that has been activated by an RFID printer. The tag is both electronic and printed, i.e., it can be scanned by an RFID reader and read by a human and it contains the tag id and the name of the healthcare unit that has requested it. After all RFID tags that are included in a pallet have been tagged and activated, the pallet barcode is scanned again indicating the end of the load. This process enhances the trace-ability of the system; in case a fault occurs with a tag, the system can detect the pallet which the faulty tag originated in, in order to check the rest of the tags and report the fault to the pallet provider, for example. The second step in the lifecycle involves loading containers in a van in order to ship them to the healthcare units. During this step, they are scanned by the warehouse fixed RFID reader and a high level loading event
$container\_left\_warehouse\_dockdoor(id, ts)$
is generated, notifying all interested parties. When containers arrive at the healthcare unit, they are scanned again with a handheld RFID reader, generating the high-level event
$container\_delivered\_to\_hospital(id, ts, < long, lat >)$
The healthcare unit, places the containers in the premises and uses them to dispose of the waste. When the containers are due to be picked up (i.e., *expired*), a notification is generated at the contractor site, e.g., by the calendar application, and the truck is sent on a round to collect the expired containers. At each stop, full containers are collected and loaded on the truck, while at the same time, they are being read by the fixed RFID reader at the truck's dock-door and weighted at the electronic scales platform on-board. The CMB system that resides in the truck's mobile terminal, saves the data and forwards it to the MMB component, with the pressing of a button by the driver. As a result, the event
$container\_collected(id, ts, < long, lat >)$
is generated and shown in the MMB-Web Dashboard. A mobile, 3G-printer prints the automatically generated *Certificate of Delivery and Acceptance*, containing a summary of RFID readings and weights for each container as well as an invoice. Both the contractor (e.g., driver) and the healthcare unit sign the documents. When the truck reaches its destination, the incineration unit, containers are offloaded from the truck and delivered to the person responsible for having them destroyed. A second weighing takes place during delivery, to ensure that the contents are intact, but also in order to calculate the charge and print an invoice. The high-level event
$container\_delivered\_to\_incinerator(id, ts, < long, lat >)$
is generated similarly to the previous step. An updated certificate is printed by the mobile terminal: since the billing scheme of using the incinerator depends on the total weight, the report also contains new measurements of weights of the containers that are being delivered. If the readings have not changed since the loading events, then the old measurements are copied in the report, which is generated automatically. Else, an updated report needs to be generated from scratch. Figure **??**

Fig. 1. Hospital Waste Life-cycle

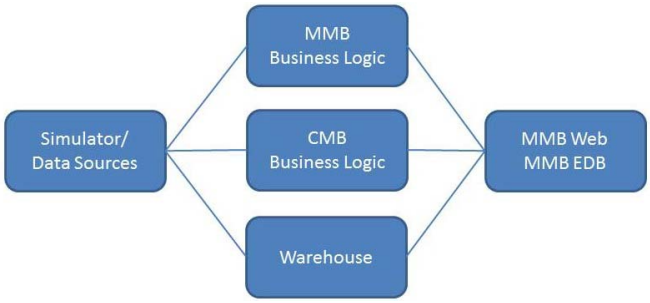


Fig. 2. Component architecture

III. System Architecture

The Greenactions information system is distributed and consists of three core components, *MMB, CMB and Warehouse*, each corresponding to one of the system's *logical locations/zones*: the *Headquarters,* the *Truck* and the *Warehouse*, respectively (Figure **??**). The *Main Monitoring Bay (MMB)*

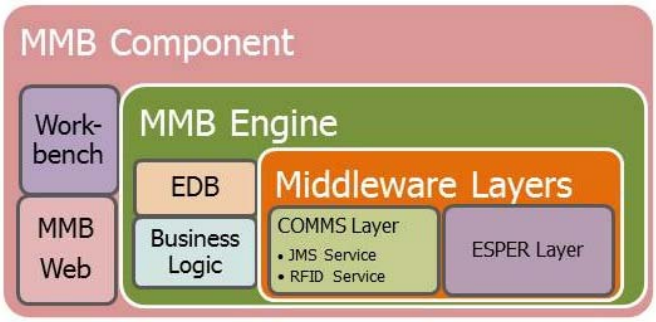


Fig. 3. Main Monitoring Bay (MMB)

(Figure **??**) is a server-side information system that resides in the company's headquarters. It receives low-level events, e.g., that a tag was read by a reader, from sensors, RFID readers or other system components that are either connected to it locally or remotely and it generates high-level events that correspond to distinct steps in the lifecycle, e.g., that a container with a specific weight was collected from a hospital unit. It also generates alerts that indicate that something out of the ordinary has happened, e.g., that a container stayed too long inside the truck.

Once an event has been received and processed in the above manner, two things happen:

- MMB stores the events, along with the raw data in the MMB database where it can be queried using SQL and used in order to build an interface with the ERP system Atlantis [**?**]. Note that the MMB database is accessible from anywhere where there is an internet connection,through a secure vpn client.
- MMB also sends the high-level events that have been generated using the Java Messaging Service (JMS) [**?**] messaging middleware to a web-service [**?**], *MMB Web*, that displays it. MMB Web is akin to a dashboard: it shows in a simple graphical mode what is happening *now*, i.e., at the current instant, at the core logical locations in the system (warehouse, truck, handheld reader). It also displays a log of historic events (timestamped, high-level events and alerts and associated positions) so that the status of the system can be visually checked from any location, using a browser application.

The *in-Car Monitoring Bay (CMB)* is a light-weigh version of MMB that is responsible for monitoring events generated at the truck from the on-board hardware system, i.e., the fixed RFID reader and electronic scales placed at the area of the dock door and communicating with the ruggedized mobile terminal. Note that the CS101 [**?**] handheld reader and barcode scanner can also be used for flexibility. Since it is equipped with the CS501 [**?**] 3G communication module, it can function independently of the mobile terminal that is used to collect only the weight readings. In this case, weight readings are correlated by the MMB with RFID readings of the same container, by means of the respective timestamps. Similarly to the MMB, CMB generates high-level events from the low level raw data that originate in the on-board hardware modules. As long as communication is possible, e.g., via the 3G card embedded at the mobile terminal, CMB forwards the events to MMB using SOAP [**?**] messages. Both raw data and events are also stored locally, in the CMB database so that they can be used as backup copy and to refresh the MMB database, when the truck is next at the headquarters. A mobile, 3G-printer is connected to CMB in order to print the automatically generated *Certificate of Delivery and Acceptance*, containing a summary of RFID readings and weights for each container as well as an invoice. Both the contractor (e.g., driver) and the healthcare unit sign the documents. An updated certificate is generated by CMB during delivery of the containers to the incineration unit. Since the billing scheme of using the incinerator depends on the total weigh, the report also contains new measurements of weights of the containers that are being delivered.

Finally, the *Warehouse module* is a remote client site that is equipped with an Alien 9000 fixed RFID reader [**?**], a CS101 handheld RFID reader and barcode scanner and a Zebra [**?**] RFID printer. Here containers are tagged using printed RFID labels that also contain printed information on the hospital or healthcare unit they are destined to. When leaving the warehouse, tagged containers are usually read in batch by the fixed reader monitoring the dock-door. The corresponding high-level events denoting that a container left the warehouse are generated and forwarded to MMB.

IV. Reference Architecture

The reference architecture is shown in Figure **??**. It is broken up into three conceptual layers. The Communication Layer (aka Sensor Abstraction Layer) provides a common API to integrate with sensors to collect various kinds of data from them. The Event-based Middleware Layer performs custom business processing rules on the data. The Application Layer provides a means to integrate the business events collected in the Application layer with other systems (such as databases, ERP systems, and web applications) that consume the events.

A. The communication middleware layer (COMMS).

The COMMS layer is responsible for the local and remote connection of heterogeneous data sources. It allows the MMB server to connect to various kinds of sensors and collect data from them. In many scenarios, this consists of connecting to a Gen2 fixed RFID reader (such as Alien 9800 [**?**], Motorola LLRP [**?**], etc), and collecting EPC [**?**] information. However, the MMB server is designed in a way so that it can collect many kinds of data (active, passive, etc.) from many kinds of devices. It can be connected to any kind of sensor with a digital output that communicates over TCP/IP. (e.g., network-enabled RFID readers, Barcode scanners, mobile devices, temperature sensors, etc.) For mobile devices such as handheld RFID and barcode readers, additional data-level middleware components needs to be written by the user in order to convert the (proprietary) device data format to that expected by this layer. Additional effort is also needed for sensors that communicate over different protocols, such as 802.15.4, Zigbee, 6LowPan etc, where the user has to write the connection logic. This

layer allows users to connect to devices in a sensor-agnostic way to collect the kind of data required for the application.

The kinds of sensors can be wide ranging, from network-enabled Gen2 RFID readers to barcode scanners to JMS queues of events. In addition, the events can be of various types. The most common events are Gen2 RFID, but other event types can be collected as well, such as barcodes and GPIO events. In addition to collecting events, the sensor layer allows some level of sensor configuration that depends on the capabilities of the sensor and on how much work has been put into the adapter. The sensor is then either configured to *push* events back to the server or the server can *poll* the sensor. Therefore this layer has three main roles:

- *Connection Logic*. The sensor session contains the logic for connecting to and maintaining a connection with a sensor. It should normally detect when a sensor has been disconnected and attempt to reconnect if possible.
- *Command Execution*. It ensures that commands sent out to sensors are carried out in a thread-safe way.
- *Protocol Parsing*. It has the responsibility to parse incoming messages according to the protocol that the sensor uses.

Several communication protocols are being used at this layer. For example, because TCP/IP is a typical protocol used for connecting to readers in the same LAN (and the same sub-network), the API supplies an abstract class that handles TCP/IP connections robustly (it can detect if the socket is closed and attempts to reconnect). Several other classes are provided that handle various kinds of connections, such as TCP sockets, SOAP messages and JMS messages.

Some sensors require a command to tell sensor to start sending back tag reads. For others, it is necessary to poll the sensor to ask if it has seen any new data. It is the job of the sensor session to ensure that the correct command is issued at at time and to allow commands to be scheduled for remote execution as necessary.

A significant part of the effort at this layer goes into protocol parsing, since this is what varies widely from sensor to sensor. For example, the LLRP sensor sends back byte messages that are encoded according the LLRP specification. The Alien reader sends back clear-text strings. Most barcode readers will send back the barcode that they read. The very first step was to define an algorithm for generating Electronic Product Codes (EPC) to be used as unique identifiers for tagged containers for the domain of hospital waste management. The end goal of this layer is to parse events that come back from a sensor and put them into the Esper [**?**] event engine (see next session). From there, the Event-based middleware can process the events.

1) MMB - CMB communication: Typically, the CMB module provides *streams* of raw data such as low level RFID readings and weigh readings while the warehouse module provides mainly RFID readings. Each element in the data stream is tagged with temporal information (timestamp), other data such as RSSI strength, the id of the antenna that read the tag and in the case of the CMB module, spatial information (GPS position).

The communication between the MMB and remote modules (such as the CMB module or a remote warehouse) implements the *connection logic* abstraction by applying the WEB 2.0 [**?**] standard (webservices) complemented with TCP (HTTP) sockets. In terms of *command execution*, the CMB is continually polled by the server forwarding data streams consisting of RFID and weight readings to a proxy web-service that is part of the MMB, listening at a specific port. The proxy web-service implements *protocol parsing*: first, it invokes business logic in java, that converts the format of the CMB data streams generated by the mobile terminal or the handheld reader to that required by the MMB module; next, it inserts the converted data stream into the MMB, which stores the raw data and at the same time forwards it to the ESPER layer for generating the appropriate business logic (see next section). The same model applies for the MMB - RFID handheld reader communication.

2) MMB - Warehouse module communication: Here we distinguish two alternative modes. If the warehouse site is on the same LAN as the MMB, then communication is TCP based. This is the default mode for current fixed RFID readers, such as the Alien 9000 series. On top of the connection logic using the TCP session, this layer provides session management information. It can issue commands to start and stop the reader, push or pull the data and execute these commands repeatedly at various intervals. The Automatic Link Establishment (ALE) protocol is also supported. Otherwise, the communication mode is similar to that of the MMB-CMB.

B. The event-based middleware layer(ESPER)

One common requirement for RFID applications is to define logical read zones for applications. In essence these are *spatial abstractions* that represent a polygon around an RFID reader where tags are read and outside of which they are ”invisible”. These zones can be used to group logically events that originate in the same zone. For example, in order to generate alerts that refer to things that can go wrong when dispatching the containers from the warehouse, the business logic responsible for generating this alert will only be interested in tags from the Warehouse reader. The same applies to the part of the MMB Web application that displays alerts from the Warehouse Dock Door, rather than the Weigh Station (truck). This layer provides also the following RFID services for generating high-level events that are one level above the raw data, using the read zones.

- *Read Zone Monitoring Service*, which notifies subscribers when a tag enters a particular read zone and when it leaves a read zone. Implementing this service, gives two methods:
  tagArrived(tagRedEvent tag),
  tagDeparted(tagReadEvent tag)

- *Unique Tag Interval Service*, which notifies subscribers the first time a unique tags is seen at a read zone and periodically after that if the tag is still in the read zone. The following method provides an implementation of this service:
  tagBatchSeen(TagReadEvent tag)
- *Stable Set Service*, which notifies subscribers when a given amount of time has passed without any new tags

having arrived. This is useful if you have a forklift that carries a batch of containers from one zone to another. The following method provides an implementation of this service:

$stableSetReached(Set < TagReadEvent > stableSet)$

The data sources, their respective zones and first level events, are shown in Table **??**. For the electronic scales, none of the above services could be applied directly, so the event $Weighted$ was specified manually, using the conceptual read zone SCALES that was associated with the particular device.

1) Espertech Engine: Espers query syntax is much like SQL. Applications can add Esper "statements" to the Esper runtime. In addition, they can define listeners to certain statements. Because it is important to keep up with which statements have been added and to make sure statements are removed when the application starts up, the Esper layer provides two methods that should be used when adding statements and listeners:

- *addStatement(String)* is used to add a single statement to the Esper runtime
- *addStatement(String,StatementAwareUpdateListener)* is used to add a statement and a listener to that statement. The listener will allow the user to handle any events which trigger the statement.

Furthermore, the Esper runtime must know ahead of time what kind of events will be put into it. Several kinds of events are already defined in the Esper runtime that every application can make use of, such as *ReadCycle, TagReadEvent, GPIEvent*, and *GPOEvent*. If an application needs to add its own custom event type, it can do so using any of the *addEventType()* methods.

2) Implementation: First, the read zones of Table **??** were specified in the system. Next the corresponding *level-one* (one level above the raw data) event types were written and added to the Esper Engine using the provided API. Next, *level-two* (two levels above the raw data) events types were grouped into two categories: Events and Notifications, as shown in Table **??**, that also links event types to the lifecycle step they correspond to.

After creating the event types, for each read zone, several Esper statements were developed, linking each level-two event type with level-one events or even raw data. Next, the statements were added to the Esper Engine, while a listener was added to the statement. An example of such a statement is shown in Figure **??**.

C. The application layer (WEB).

Applications are at the heart of the Application Layer; these classes are used to add custom Esper statements to look for events, subscribe to services, and integrate with existing infrastructure such as databases and JMS queues but also with legacy systems such as ERP and Servlets that present results. The latter is possible using the OSGI standard for building java services, in combination with the communication layer, for the systematic interaction between the various services. Therefore this layer provides a base set of services to applications including:

TABLE I
READ ZONES AND LEVEL-ONE EVENTS

Data Sources	Zone	RFID Service Events
Warehouse RFID Reader	DOCK_DOOR	DockDoorArrived DockDoorDeparted
Truck RFID Reader	WEIGH_STATION	WeighStationArrived WeightStationDeparted
Electronic Scales	SCALES	Weighted
Handheld RFID Reader	HAND_HELD	HandheldSeen

TABLE II
LEVEL-TWO EVENTS AND ALERTS

Event Type	Event Group	Life cycle step
container left warehouse dockdoor	Event	Step 2
container too long on dock door	Notification	Step 2
container moved backwards	Notification	Step 3/ Step 2
container delivered to hospital	Event	Step 3
container not at hospital in time	Notification	Step 3
container loaded and weighted on weigh station	Event	Step 4
container collected	Event	Step4
container too long on weigh station	Notification	Step 4
container overweight	Notification	Step 4/ Step 5
container offloaded and weighted from weigh station	Event	Step 5
container delivered to incinerator	Event	Step 5
container not offloaded in time	Notification	Step 5
container changed weight	Notification	Step 5
Container reloaded on weigh station	Notification	After Step 5
Container not offloaded at correct location	Notification	Step 5

- Life cycle management (starting and stopping the application)
- Configuration management (using property files to provide input parameters to the application)
- Esper management (ensuring that Esper is used correctly)
- OSGI Implementation (allowing the application to be structured as an OSGI service)
- Design Patterns and Communication Interfaces (allowing the integration of the applications according to accepted Design Patterns, such as the Model View Controller (MVC) and JMS messages).

For example, Figure **??** shows how the RFID services described in the previous section can be integrated with the MMB Application.

1) MMB Web: MMB Web is a Servlet application that resides within the OSGI container and that shows the high-level events and alerts that are generated by MMB, in a Dashboard view. Together with MMB and EDB, they implement the Model-View-Controller (MVC) design pattern: MMB Web is the View, EDB is the Model and MMB implements the business logic of Controller.

The Model consists of four entities *Warehouse Dock-Door*,

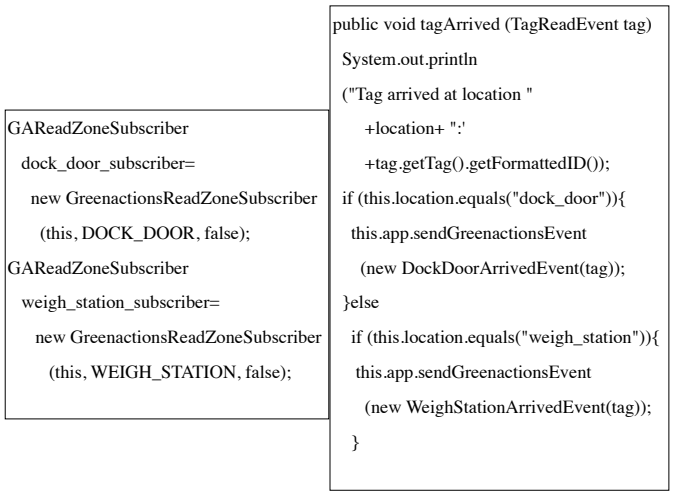


Fig. 4. Examples 1 and 2

CMB Weigh Station, *Handheld reader* and*Alerts*. The first three entities are read zones, i.e., logical zones while Alerts is a logical grouping of all alerts. For each model entity, a specific view is written that displays any current events that represent activity about a certain entity, e.g., $container\_left\_warehouse\_dockdoor$ in a cell in the GUI that corresponds to that entity (e.g., Warehouse Dock-Door, Figure **??**).

The controller is responsible for mapping the high level events and alerts generated at the Esper layer, to the View component to display at the right zone. To achieve this it uses the timing information included in their timestamp to determine whether they are current or whether they are historic and need to be sent to the Alerts entity. The Java Messaging Protocol (JMS), provided by the COMMS layer, connects the MMB and MMB Web through a dedicated shared message bus. Although it is a higher-level communication protocol, it follows a similar principle to low-level communication: generated business logic events and alerts are wrapped in JMS messages and they are forwarded through the JMS bus to the MMB-Web Web Service. MMB-Web unwraps the data, extracting the various message fields and shows them visually using he View part of the Java (MVC) framework. Figure **??** shows how read zones can be used to generate high-level events, wrapped in JMS messages and forwarded to the MMB Web by means of the*sendGreenactionsEvent(event e)* method.

2) Interface with Atlantis ERP: A basic integration step was carried out, in the first instance, in order to enable the integration of MMB and Atlantis. This is necessary in order to link enterprise information relating to materials (pallets of cardboards and plastic bags, RFID tags, paper) as well as company's assets (several desktops, servers and laptops, the RFID printer, two fixed RFID readers, one handheld RFID reader, one barcode reader and the electronic scales) with the business process of medical waste collection. This integration is done by means of the EDB ER diagram, that links the life cycle of tagged containers with pallets and assets, through the various high-level events. For example the process of scanning the barcode of the pallet that contains the flat containers before and after their assembly and activation, as described in Section **??** links tagged containers to the pallets and therefore the supplier, information stored typically in the ERP system. Similarly, activation events can be used to estimate the use of the the RFID printer and consumed tags, something that can be of interest to economic tasks associated with the ERP system.

3) Implementation Details: A prototype implementation (proof of concept) was carried out using the Rifidi Edge Server v1.2 [3], an open source RFID middleware. Rifidi provides basic RFID communication services with the fixed readers, the Espertech Engine [2] on top of which the business logic discussed in Section IV-B were developed and a simulator that was used to simulate the Warehouse, Weigh Station and Handheld reader components, thus achieving the end-to-end functionality. In addition, a low-level middleware component, a proxy-web service was developed to communicate with the handheld reader and the electronic scales. This service first, converts the format of data streams generated by the mobile terminal or the handheld reader to that required by the MMB module; next, it inserts the converted data stream into the MMB, for generating the appropriate business logic. Both the MMB and CMB components were implemented as Java OSGI bundles while MMB Web was developed as a Java Servlet. The EDB was implemented using MySQL Database v.5.5 and the MySQL benchmark tool.

V. Evaluation and Key Performance Indicators (KPIs)

In order to evaluate the benefits from the Greenactions system, we introduce a number of Key Performance Indicators (KPIs) that make sense in the scenarios that we investigate. This includes the construction of a data set, the simulation of a one-year-operation scenario, the design of suitable KPIs, their implementations by means of efficient queries and finally the visualization of results. KPI implementation was based on a free web data analyzer known as Splunk [**?**]. Splunk has its own expressive, real-time query language which was used for KPI visualisation. With the dropdown menu and the multiselect options Splunk is a friendly-user web data analyzer.

A. Simulation

The design of the data set was created with the assumption that a company use this system and has in its position two trucks, a warehouse located in Xanthi, Greece and several medical waste containers (boxes) of different capacities, i.e., (10 lt, 20 lt, 38 lt and 57 lt). Furthermore, it was assumed that it provides its services in some of the departments of three existing hospitals in Northern Greece. The client hospitals are located in the cities: Thessaloniki, Xanthi and Alexandroupoli. Medical waste is collected from each hospital by one of the trucks and is transported to an incinerator unit located outside Thessaloniki. The truck's schedule is Xanthi-Alexandroupoli-Incinerator for the first truck and Xanthi-Thessaloniki-Incinerator for the second and they make the round trip once or twice every week. Table **??** represents the

Fig. 5. Main Monitoring Bay Servlet (MMB-Web)

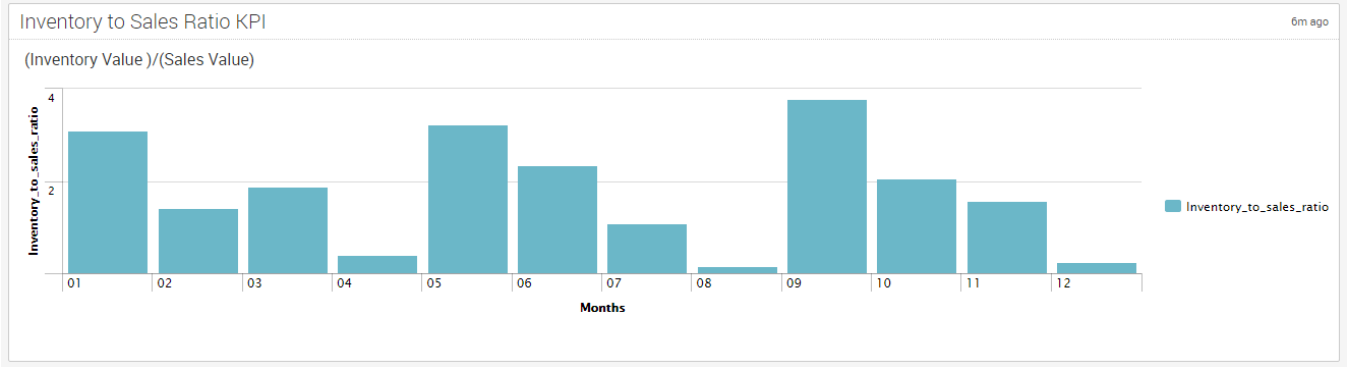


Fig. 6. Inventory to Sales Ratio KPI

month of December, in respect to the simulated scenario; it illustrates the number of boxes collected from the hospitals and delivered to incinerator. The activation of RFID tags on the empty containers, the departure from the providers warehouse and the delivery of the boxes to hospitals all take place on the same day while the collection and delivery of the boxes to the incinerator unit also take place in one day (one of the following days).

The calculation of the amount of waste produced from each hospital is based on the assumption that a hospital produces 5 kg [**?**] of medical waste per bed per day, out of which, 2 kg [**?**] are solid waste and the rest are liquid. It was decided to adopt a conservative approach and halve the estimated total waste for our scenarios, hence total solid waste is calculated as:

$$(\#Beds\ per\ department * 2kg)/2 \quad (1)$$

and for the rest:

$$(\#Beds\ per\ department * 3kg)/2 \quad (2)$$

Next, the amount of boxes needed to collect this quantity of waste was calculated leading to the construction of a simulation scenario for the operation of this provider that spans 12 months.An indicative such month is shown in Table **??**. Finally, this scenario was implemented in Splunk and visualized using test (dummy) data. It should be noted that in certain weeks (e.g., 28/12), some containers were deliberately missed, e.g., between their collection and incineration, to generate a more realistic scenario. Furthermore, wherever necessary the cost of operations was calculated based on realistic estimation of container cost, as shown in Table **??**. The results are discussed in Section VI-B

TABLE III
A MONTH FROM THE SCENARIO

Date of delivery	Date of incineration	Delivered	Incinerated
07/12	11/12	153	153
14/12	18/12	114	113
21/12	25/12	153	153
28/12	31/12	127	125

TABLE IV
PRICES OF THE USED BOXES IN EUROS

Capacity of box	Price bought	Price Sold
10 liter	1,5	3
20 kilogram	1,2	10
38 liter	2,8	8
57 liter	4,2	12

B. KPI Results and Visualisation

1) Inventory to Sales Ratio: The *Inventory to Sales Ratio KPI* measures the amount of inventory stored in the warehouse. It is calculated as:

$$(Inventory\ Value)/(Cost\ of\ Goods\ Sold) \quad (3)$$

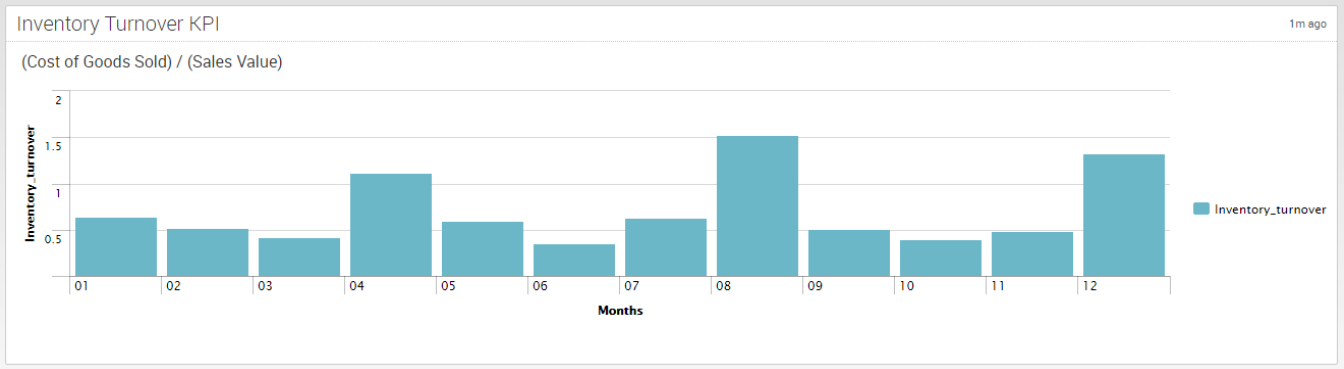


Fig. 7. Inventory Turnover KPI

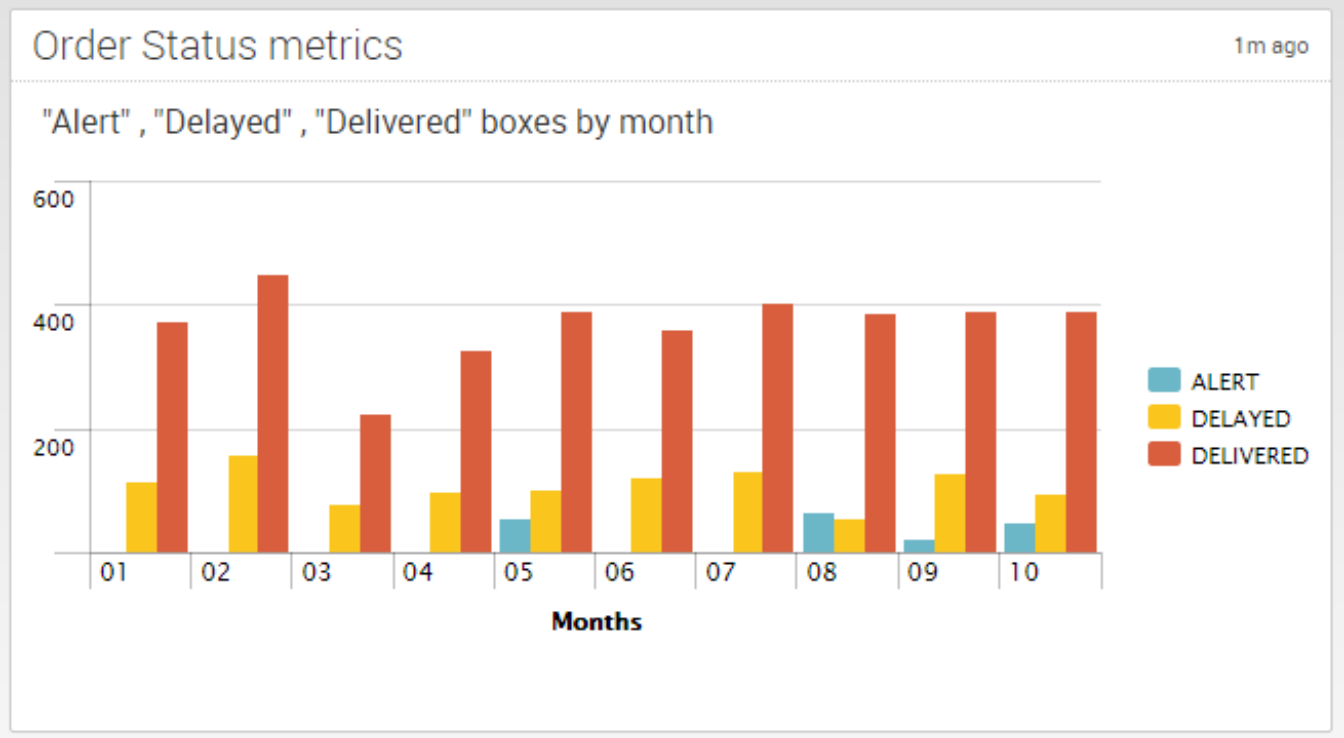


Fig. 8. Order Status metrics KPI

where $Cost\ of\ Goods\ Sold$ is calculated as:

$$\begin{aligned}&(beginning\ inventory)\\ &+(inventory\ purchases)\\ &-(ending\ inventory)\end{aligned} \tag{4}$$

for each month and *Inventory Value* is the cost of boxes calculated in the warehouse at the end of each month. The target values is a low or dropping inventory to sales ratio (Figure **??**). Table **??** has the prices that were used in the calculation. Also we assume that we buy new boxes every four months and the validation method that we use is FIFO [**?**]. This means that the Cost of Sales is determined by the cost of the items that were purchased at the earliest while Inventory Value comprises the cost of the items that were purchased at the latest.

2) Inventory Turnover: The *Inventory Turnover KPI* measures how many times a year the company is able to sell its entire inventory. It is calculated by the formula:

$$(Cost\ of\ Goods\ Sold)/(Average\ Inventory) \tag{5}$$

where *Average Inventory* is calculated as:

$$[(beginning\ inventory) + (ending\ inventory)]/(2) \tag{6}$$

With respect to Figure **??**, it is shown that months with a KPI value above one indicate that the inventory is then coming to an end.

3) Order Status: Order Status metrics (Figure **??**) tracks the status of all orders that have been categorized as Delayed Alert, or Delivered as follows: The time between collection from hospital and delivery to incinerator must not overcome a threshold specified by the administrator of the system.

This threshold is placed with basic criterion the approximate time the truck-driver needs to collect the boxes from the hospital and deliver them to incinerator. Boxes which have not delivered in time (overcome the threshold), take either the Delayed status or the Alert one. The difference between them is that Delayed boxes overcome the threshold for a short time-period (less than one hour). Figure **??** shows the amount of Delivered , Delayed and Alert boxes per month, for the one-year simulation period.As well as monitoring the date and status of orders, this metric can also include information such as order accuracy.

4) Order Status 2: The *Order Status metrics 2* measures the number of boxes which are delivered to incinerator "Damaged" or "Intact". This classification is feasible by means of a simple comparison of the total box weight at the time of collection from a hospital and at the time of delivery to the incinerator. If there is a difference between those values, it is assumed that some of the boxes have been delivered to incinerator damaged (Figure **??**). Like the previous one, this KPI was constructed not only for financial but also for security reasons, that is to flag damaged or missing containers and it can be combined with Order Status to aid further investigation.

5) Units per Transaction: The *Units per Transaction KPI* (Figure **??**) measures the average number of waste boxes handled per hospital served over a period of time and compares that value to target values. This KPI is calculated using the following formula:

$$(\#of\ boxes\ handled)/(\#of\ hospitals\ served) \quad (7)$$

This KPI provides important data about customer waste producing trends and has been created to observe if the sales are increase or decrease and if the customers still prefer this company over a potential competitor. Transaction is taken to mean the complete lifecycle of a waste container.

6) Rate of delivery: The *Rate of delivery KPI* measures the rate at which empty waste boxes sold to hospitals are collected for disposal or collected waste items are delivered to the final disposal site. The key to these metrics is providing an insight in the fate of medical waste and reducing its inappropriate disposal, i.e., to municipal refuse sites. The target is a value as high as possible. It is a comparison between the number of boxes that were collected from hospital and those delivered to incinerator. In Splunk with the option of the dropdown menu, the comparison could be made for each hospital (Figure **??**).

7) Perfect Order Rate: The *Perfect Order Rate KPI* measures how many empty waste boxes are sold without incident, where incidents are missed boxes, inaccurate orders, or late shipments. This is very similar to the previous KPI and a single implementation could be used for both KPIS. The target is a value as high as possible. The company can detect in which month the missed boxes are observed (Figure **??**).

8) Order Status 3: The *Perfect Order Rate KPI* targets a single month. The *Order Status 3 KPI* extends this by providing tools for querying not only per month but also per day, hospital, department, container id, thus eventually helping administrators discover not only the which hospital department potential missing boxes originated from but also their unique identifiers, taking appropriate mitigation measures (Figure **??**).

VI. Literature Review

Till recently GPS based vehicle tracking systems, tracking clearance of secondary collection points, GIS for selection of suitable landfill/dump sites, and optimization of collection routes and containers using a Decision Support System (DSS) have contributed well to efficient solid waste systems. However, without the precise information provided by the RFID technology, only rough data can be collected, leading to limited monitoring and decision making regarding the waste lifecycle. RFID is now somewhat being used by municipalities and private operators to monitor waste pickup. However, such systems are proprietory and very costly. They are not standardised which makes it difficult to compare them in terms of their functionality. They differ in terms of communication capability, real-time data management, sensor technology; they do not separate concerns between data acquisition (sensor abstraction), business logic (event correlation and processing) and applications. Furthermore, they tend to focus on route optimisation and spatial abstractions rather than rich-data, container management. When dealing with hazardous waste disposal, a high degree of visibility and trace-ability in disposing of hazardous or otherwise sensitive waste material is required, that is not currently provided for by existing solutions. Summarising, several industrial components exist in the literature:

Air-Track [**?**], a fleet-tracking services provider based in San Diego, California, has developed an RFID-enabled Waste-Connect solution that tracks the locations and activities of its trash-pickup trucks, but also monitors which bins are dumped during rounds of waste and recycling collection. This component uses the Intermec IV7 vehicle-mounted RFID reader [**?**] together with the Intermec CV60 [**?**] or CV30 vehicle-mounted computer, which does not however seem to have capability for remote, high-power communication (e.g., GPRS/3G) and therefore stores information locally. Not being part of an integrated solution, it does not directly satisfy the requirements discussed in this work.

Smartrac [**?**] and MPI Label systems [**?**] offer a broad range of ticket and label inlays for tagging waste containers. BOS [**?**] provides a waste cart module with an RFID tag, a dispatch management module containing an RFID reader for the waste collection vehicle that relays data on bins read back to the BOS server, generating reports. ArtRFID is a full-service supplier of end-to-end RF-based asset tracking and monitoring solutions based upon proprietary low-cost active-RFID systems operating in the UHF bands. IonWaste [**?**] has similar technology and goals. Cascade Engineerings RFID system [**?**] for the waste management consists of recycling and trash containers mounted with RFID xtreme tags from Xtreme RFID, which are powered by UPM ShortDipole ultra-high frequency inlays guaranteeing performance in extreme temperatures and weather conditions. This information can be used to automate data collection and billing. AgoraBee [**?**]

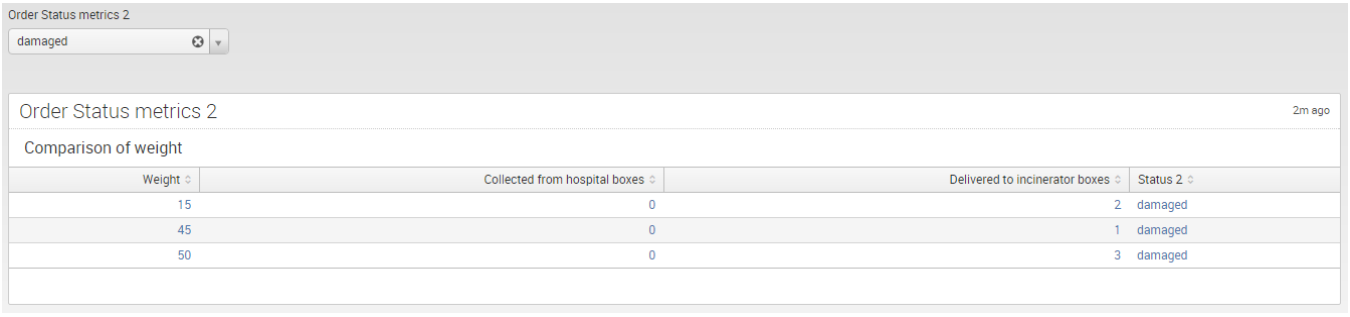


Fig. 9. Order Status metrics 2 KPI

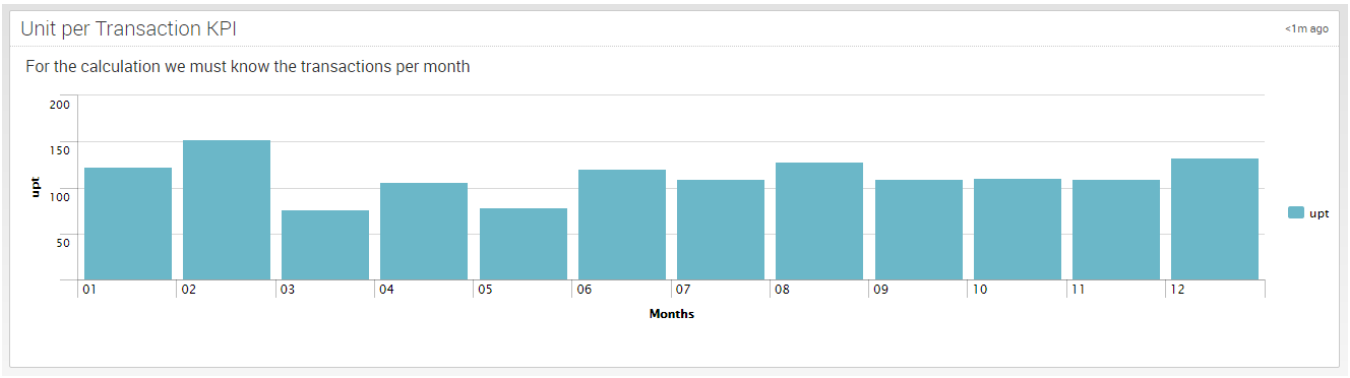


Fig. 10. Units per Transaction KPI

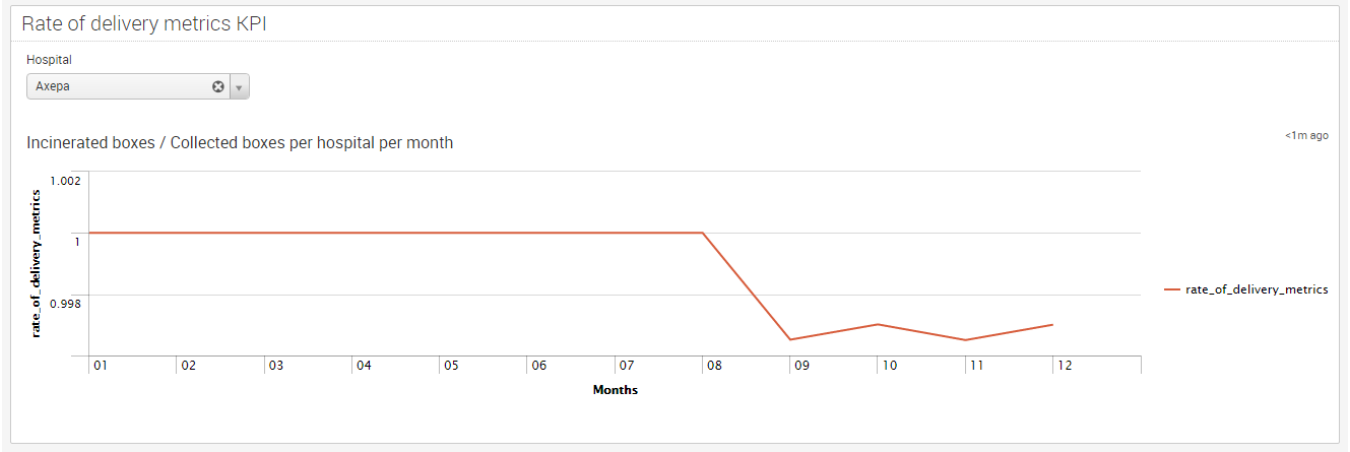


Fig. 11. Rate of Delivery metrics KPI

developed *ChisFleet*,a solution for tracking the locations of its containers and waste bins as they are deposited at construction sites and other locations, using Krypton RFID tags on the bins, as well as their own reader technology on trucks to transmit GPS location data, along with each tag's unique ID number. Their goals are different to ours. In the UK and Ireland, Amcgroup [**?**] designs on-vehicle solutions, such as a vehicle data-hub with integrated GPRS, GPS & WIFI RFID reading and load cell measurement and the ability to handle I/O signals and CANBUS data as well as a wearable RFID reader in the form of a bracelet (aRM reader). This system is quite sophisticated. It also provides a proprietary software, ELEMOS, for monitoring municipal waste and optimizing truck routes. MAWIS EM [**?**] is a software for the waste disposal industry that uses a waste identification and /or weighing system to assign transponders and emptyings to containers. The software is both proprietary and modular, providing services from container management to route planning and billing.

Moving from passive to a more active waste container is the Opti [**?**] System from Plastic Omnium. With this technology, each bring-bank is fitted with ultrasound sensors and remote messaging systems that send an SMS text to a local authority database when nearly full. The cBin [**?**] solution consists of a remote sensor that sends fill level and asset status information via wireless communications to a Web portal that provides an "at a glance" view of all containers in a community for

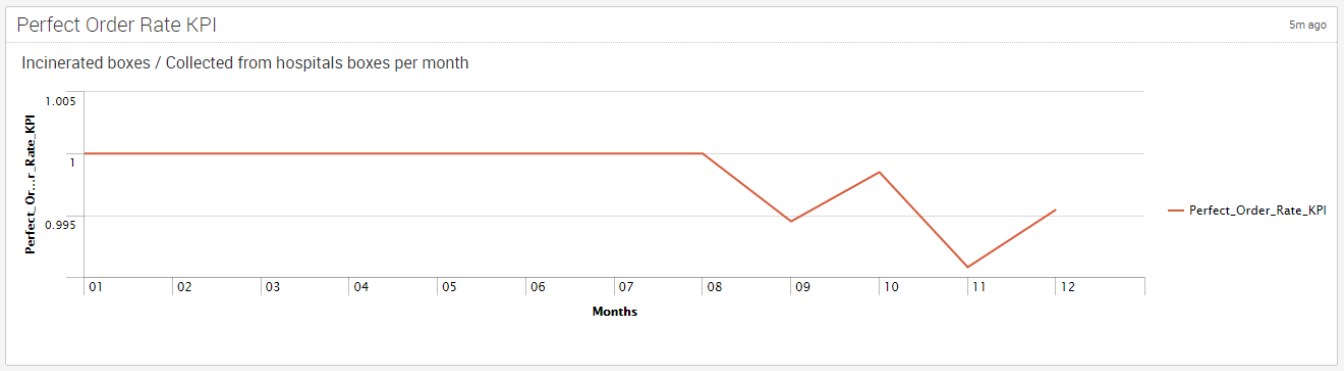


Fig. 12. Perfect Order Rate KPI

rapid evaluation of container status. Immediate updates are sent if fill levels exceed action levels. Both works shift the management focus from the waste itself to the container and could complement the work presented here.

The work of [**?**] proposes an automated solid waste collection monitoring system using RFID technology integrated with GPS, GIS, GSM and GIS technologies for managing solid municipal waste. Delhi Waste Management also uses a similar system. However, different research goals than our work: It does not accommodate the special health and safety requirements of hospital waste, for example, does not seem to use a weighing platform for calculating real-time weight-based charges for waste disposal. It is also not concerned with any of the middleware abstractions and services described in this paper.

Sensor-driven information such as that derived from RFID systems requires specialised processing models that can cater for the real-time requirements of this information, i.e., very short response-times, at most a few seconds and a high degree of scalability that spans thousands of events. Typical such systems include event processing mechanisms [**?**], [**?**], [**?**], such as finite-state-machines and petri nets. This type of system is known as middleware. Previous work by the first author presents a middleware structured as a a service-oriented-architecture that can be used for the development of pervasive computing applications [**?**].

VII. Conclusions

This paper discusses Greenactions, a novel Pervasive Computing system for the remote management of the full lifecycle of hospital waste. The contribution of this work is two- fold: first, it essentially IOT-enables medical waste containers: by tagging each container separately with RFID technology the system gives to each container a unique id that includes the hospital and department, where the waste came from. By modeling the life-cycle as a state-machine, it is able to know what state each container is in at each instant. By integrating a web Servlet, MMB-Web, the system achieves an integration of each container with the Internet of Things. Second, by integrating multiple sites (e.g. Warehouse, Headquarters) as well as both fixed and mobile (handheld) sensors and RFID readers, it can monitor continually a large range of medical waste management scenarios.

The full cycle of medical waste is managed, from the dispatching of empty containers to the healthcare unit through to the their collection and subsequent disposal to the incineration or sterilization unit where they are destroyed. The above lifecycle is monitored using both RFID and sensor technology. An event-based middleware component, included in the system, generates, in real-time, high-level, business intelligence events and notifications, from low-level RFID and sensor readings. The system consists of several distributed components: the *Main Monitoring Bay (MMB)* that resides in the headquarters of the contractor company and collects, correlates and stores event streams; the *in-Car Monitoring Bay (CMB)*, a mobile version of MMB, that is embedded in a ruggedized mobile terminal kept in the truck that collects the containers and transports them to the incineration unit. CMB monitors an RFID reader that controls the truck door and electronic scales for weighing containers while they are loaded/offloaded from the truck and it forwards the data to MMB; a *Warehouse* module that monitors the warehouse dock door from where empty containers, tagged appropriately, are shipped to healthcare units to be filled in with waste. The orchestrated system can deal effectively with a large variety of heterogeneous sensor technologies, by providing:

- a *communication layer* that is sensor-agnostic, i.e., provides abstractions on *Connection Logic* (TCP/IP, TCP/HTTP sockets, JMS, SOAP, etc) *Command Execution* (pull, push, iterate, etc.) and *Protocol Parsing* thus supporting the integration of EPC Gen1 and Gen 2, LLRP RFID readers, barcode readers, sensors etc.
- an *event-based middleware layer* that provides spatial abstractions in the form of *read zones*, core RFID monitoring services, a simple language for specifying high-level events as regular expressions involving raw data as well as health and safety policies, a reasoning engine for detecting these events and enforcing these policies.
- an *application layer* that allows for applications to be written in a service-oriented way, structured as OSGI services (bundles), help integrate these applications with communication and event-based layer services as well as external services and systems such as ERP, databases and

Fig. 13. Order Status metrics 3

web Servlets.

ACKNOWLEDGMENT

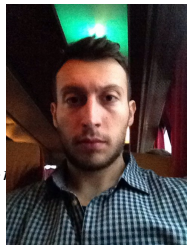
The authors would like to thank the Hellenic General Secretariat for Research and Technology (GSRT) - Hellenic

Ministry of Education and Religious Affairs, for funding this work.

REFERENCES

[1] H. Kopka and P. W. Daly, "A Guide to LaTeX", 3rd ed. Harlow, England: Addison-Wesley, 1999.
[2] EsperTech - Event Series Intelligence, *www.espertech.com*
[3] Rifidi Edge Server, *www.rifidi.org*
[4] OSGi Alliance, *http://www.osgi.org/Main/HomePage*
[5] Atlantis E.R.P.,*http://www.altecsw.gr/pages/15.Atlantis-E-R-P-.html*
[6] Java Message Service 2.0 (JSR 343), *https://jcp.org/en/jsr/detail?id=343*
[7] Simple Object Access Protocol (SOAP) 1.1., *http://www.w3.org/TR/2000/NOTE-SOAP-20000508/*
[8] World Wide Web Consortium (W3C), *http://www.w3.org/*
[9] Web or Services, *http://www.w3.org/standards/webofservices/*
[10] Convergence Systems CS501 Module, *http://store.immediasys.com/convergence-systems-cs501-gsm-gprs-gps-module-for-cs101-rfid-reader/*
[11] CS101 Handheld RFID Reader, *http://www.ttsys.com/pdf/CS101.pdf*
[12] Alien Readers, *http://www.alientechnology.com/readers/*
[13] RFID Printers, *http://www.zebra.com/us/en/products-services/printers/printer-type/rfid.html*
[14] Low Level Reader Protocol (LLRP) Standard, *http://www.gs1.org/gsmp/kc/epcglobal/llrp*
[15] Electronic Product Code (EPC): An overview, *http://www.gs1.org/docs/epcglobal/an_overview_of_EPC.pdf*
[16] Claire Swedberg, *Air-Trak Brings Visibility to Waste Management*, RFID Journal, Feb 2014
[17] IV7 Vehicle Mounted Reader, *http://www.intermec.com/products/rfid2_iv7/index.aspx*,
[18] CV61 Fixed Vehicle Mount Computer, *http://www.intermec.com/products/cmptrcv61/index.aspx*,
[19] Smartrac, *http://www.smartrac-group.com/en/products.php*
[20] RFID for Waste Management, *http://www.mpilabels.com/mpi-products/rfid-for-waste-management*
[21] Real-time visibility for supply chain management, *http://www.boscom.com/?CategoryID=238&ArticleID=81*
[22] IonWaste Systems, *http://www.art-rfid.com/wastemanagement.html*
[23] RFID Tags Help with Recycling and Waste Collection in Cincinnati and Grand Rapids, *http://www.rfidworld.ca/rfid-tags-help-with-recycling-and-waste-collection-in-cincinnati-and-grand-816*, RFID World Canada
[24] ChisFleet. The professional extended geolocation solution, *http://www.agorabee.com/jl25n/images/AgoraBee_2013/PDF2013/AB_FactSheet_CHISFLEET.pdf*
[25] Integrated Environmental Software and Solutions, *http://www.amcsgroup.com/uk/products/waste-truck-vehicle-technology/*
[26] Weighing and Identification System for communal waste MAWIS, *http://www.moba.de/en/products/waste-and-logistics/mawis-identification-system-for-communal-waste.html*
[27] Tagged with intelligence: Applying IT in household waste collection, *http://www.waste-management-world.com/articles/print/volume-8/issue-1/features/tagged-with-intelligence-applying-it-in-household-waste-collection.html*, Waste Management World
[28] RFID & GPS for waste management, *http://rfid.thingmagic.com/rfid-blog/bid/88365/RFID-GPS-for-Waste-Management*
[29] S. Purohit, V. Bothale, *RFID-Based Solid Waste Collection*, Information Technology in Developing Countries, 22(2), July 2012
[30] Healthcare waste management, *http://www.who.int/mediacentre/factsheets/fs281/en/*
[31] Operational Intelligence, Log Management, Application Management, Enterprise Security and Compliance Splunk, *http://www.splunk.com*
[32] Medical Waste Management, *https://www.icrc.org/eng/assets/files/publications/icrc-002-4032.pdf*
[33] Biomedical waste management in a Large teaching hospital, *http://medind.nic.in/jab/t07/i1/jabt07i1p57.pdf*
[34] How to Value Your Inventory, *http://www.entrepreneur.com/article/55578*
[35] Katsiri E., Bacon J., Mycroft A. An Extended Publish/Subscribe Protocol for Transparent Subscriptions to Distributed Abstract State in Sensor-Driven Systems Using Abstract Events 3rd International Workshop on Distributed Event-Based Systems (DEBS'04) 2004
[36] Katsiri E., Moschou K. A pervasive computing system for the remote management of hospital waste. Submitted to Journal of Communications and Software Systems, 2015
[37] Mansouri-Samani M., Sloman M. GEM - A Generalised Event Monitoring Language for Distributed Systems,IEE/IOP/BCS Distributed Systems Engineering Journal. 1997;4:96–108

Eli Katsiri is Assistant Professor at the Department of Electrical and Computer Engineering at the Democritus University of Thrace. She is also Scientific Collaborator at the Institute of Management Information Systems (IMIS) at Research Center "Athena". From October 2007 to October 2010 she was a Lecturer in Computer Science at the Department of Computer Science and Information Systems, at Birkbeck College, University of London, where she supervised several MSc and BSc students, including one PhD student. From 2005 to 2007 she worked as a postdoctoral researcher at Imperial College London in distributed Software Engineering. Eli has a PhD from Cambridge University's Engineering Department (2005) and she has done research in the Cambridge University's Computer Laboratory working on Pervasive Computing (2000-2004). Dr Katsiri has an MEng in Computer Engineering and Informatics from the University of Patras, Greece (1998). From 1998 to 2000 she worked as a Systems Analyst in Procter&Gamble, in Brussels. She has also worked as a developer for Velti S.A. in London and Cetrus Inc in California, as well as a consultant in e-Government for the Hellenic Deputy Minister of Education and Religious Affairs, Prof. Panaretos. Her research is broadly concerned with programming abstractions and models for real-time, efficient management of sensor data.



Konstantinos Moschou is a recent graduate (2015) of the Department of Electrical and Computer Engineering at the Democritus University of Thrace (DUTH). His Diploma thesis involved the design and implementation of the KPIs described in this work.