DataCare: Big Data Analytics Solution for Intelligent Healthcare Management

Alejandro Baldominos[1]*, Fernando De Rada[2], Yago Saez[1]

[1] Computer Science Department, Universidad Carlos III de Madrid, Leganés (Spain)
[2] Camilo José Cela University, Madrid (Spain)

Received 15 July 2016 | Accepted 3 February 2017 | Published 27 March 2017

Abstract

This paper presents DataCare, a solution for intelligent healthcare management. This product is able not only to retrieve and aggregate data from different key performance indicators in healthcare centers, but also to estimate future values for these key performance indicators and, as a result, fire early alerts when undesirable values are about to occur or provide recommendations to improve the quality of service. DataCare's core processes are built over a free and open-source cross-platform document-oriented database (MongoDB), and Apache Spark, an open-source cluster-computing framework. This architecture ensures high scalability capable of processing very high data volumes coming at fast speed from a large set of sources. This article describes the architecture designed for this project and the results obtained after conducting a pilot in a healthcare center. Useful conclusions have been drawn regarding how key performance indicators change based on different situations, and how they affect patients' satisfaction.

Keywords

Architecture, Artificial Intelligence, Big Data, Healthcare, Management.

DOI: 10.9781/ijimai.2017.03.002

I. Introduction

When managing a healthcare center, there are many key performance indicators (KPIs) that can be measured, such as the number of events, the waiting time, the number of planned tours, etc. Often, keeping these KPIs within the expected limits is key to achieve high users' satisfaction.

In this paper we present DataCare, a solution for intelligent healthcare management. DataCare provides a complete architecture to retrieve data from sensors installed in the healthcare center, process and analyze it, and finally obtaining relevant information which is displayed in a user-friendly dashboard.

The advantages of DataCare are twofold: first, it is intelligent. Besides retrieving and aggregating data, the system is able to predict future behavior based on past events. This means that the system can fire early alerts when a KPI in the future is expected to have a value that falls outside the expected boundaries, and to provide recommendations for improving the behavior and the metrics, or in order to prevent future problems attending events.

Second, the core system module is built over a Big Data Platform. Processing and analysis are run over Apache Spark, and data are stored in MongoDB, thus enabling a highly scalable system that can process very big volumes of data coming at very high speeds.

This article is structured as follows: section II will present a context of this research by analyzing the state of the art and related work. Section III will present an overview of DataCare's architecture, including the three main modules responsible for retrieving data, processing and analyzing it, and displaying the resulting valuable information.

* Corresponding author.

E-mail address: abaldomi@inf.uc3m.es

Sections IV, V and VI will describe the preprocessing, processing and analytics engines in further detail. The design of these systems is crucial to provide a scalable solution with an intelligent behavior. Section VII describes the visual analytics engine, and the different dashboards that are presented to users.

Finally, section VIII describes how the solution has been validated, and section IX provides some conclusive remarks along with potential future work.

II. State of the Art

Because healthcare services are very complex and life-critical, many works have tackled the design of healthcare management systems, aimed at monitoring metrics in order to detect undesirable behaviors that decrease their satisfaction or even threaten their safety.

The design and implementation of healthcare management system is not new. Already in the 2000s, Curtright et al. [4] describe a system to monitor KPIs summarizing them in a dashboard report, with a real-world application in the Mayo Clinic. Also, Griffith and King [7] proposed to establish a "championship" where those healthcare systems with consistently good metrics will help improve decision processes.

Some of these works explore the sensing technology that enable proposals. For instance, Ngai et al. [11] focus on how RFID technology can be applied for building a healthcare management system, yet it is only implemented in a quasi-real world setting. Ting et al. [13] also focus on the application of RFID technology to such a project, from the perspective of its preparation, implementation and maintenance.

Some previous works have also tackled the design of intelligent healthcare management systems. Recently Jalal et al. [8] have proposed an intelligent depth video-based human activity recognition system to track elderly patients that could be used as a part of a healthcare management and monitoring system. However, the paper does not

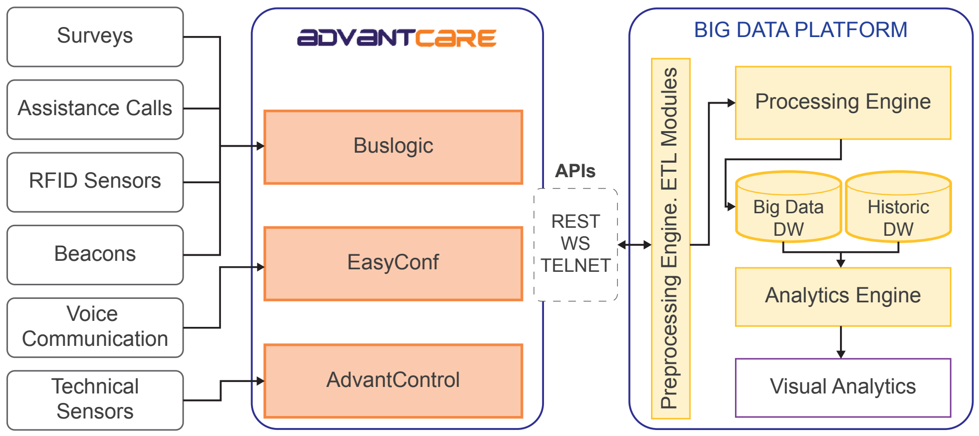


Fig. 1. DataCare's architecture. The first column lists the data sources, which are retrieved and aggregated by AdvantCare software (second column). The last column shows the Big Data platform, which contains engines for the data processing and analytics module (yellow) and the data visualization module (purple).

explore this integration. Also, Ghamdi et al. [6] have proposed an ontology-based system for prediction patients' readmission within 30 days so that these readmissions can be prevented.

Regarding the impact of data in healthcare management system, the important of data-drive approaches have been addressed by Bossen et al. [3]. Roberts et al. [12] have explored how to design healthcare management systems using a design thinking framework. Basole et al. [2] propose a web-based game using organizational simulation for healthcare management. Zeng et al. [16] have proposed an enhanced VIKOR method that can be used as a decision support tool in healthcare management contexts. A relevant work from Mohapatra [10] explores how a hospital information system is used for healthcare management, improving the KPIs; and a pilot has been conducted in Kalinga hospital (India), turning out to be beneficial for all stakeholders.

Some works have also explored how to increase patients' satisfaction. For example, Fortenberry and McGoldrick [5] suggest improving the patient experience via internal marketing efforts; while Minniti et al. [9] propose a model in which patient's feedback is processed in real-time and drives rapid cycle improvement.

To place this work into its context, what we have developed is a data-driven intelligent healthcare management system. Because of the Big Data volume and fast speed, we have used a Big Data architecture based on the one proposed in Baldominos et al. [1], but updating the tools to use Apache Spark for the sake of efficiency. Also, a pilot has been conducted to evaluate the performance of the proposed system.

III. Overview of the Architecture

DataCare's architecture comprises three main modules: the first oversees retrieving and aggregating the information generated in the health center or hospital, the second will process and analyze the data, and the third displays the valuable information in a dashboard, allowing the integration with external information systems.

Figure 1 depicts a broad overview of this architecture, while this section describes each of the modules in further detail.

A. Data Retrieval and Aggregation Module

Data retrieval is carried out by AdvantCare software, developed by Itas Solutions S.L. AdvantCare is the set of hardware and software tools designed to manage communications between patients and healthcare staff. Its core comprises three main systems: 1) Buslogic manages and aggregates the information of actions carried out by non-doctor personnel (nurses and nursing assistants), 2) AdvantControl monitors and controls the infrastructure, and 3) EasyConf manages voice communication.

In the hospital rooms, different data acquisition systems are placed, which often consist on hardware devices connected to an IP network and include one of the following elements:

- Sensors measuring some current value or status either in a continuous or periodic fashion and sending it to Buslogic or AdvantControl servers; such as thermometers or noise or light sensors.
- Assistance devices such as buttons or pull handlers that are actioned by the patients and transmit the assistance call to the Buslogic server.
- Voice and video communication systems that send and receive information from other devices or from Jitsi (SIP Communicator), which are handled by EasyConf.
- Data acquisition systems operated by means of graphical user interfaces in devices such as tablets; e.g., surveys or other information systems.

In general terms, the information retrieved by AdvantCare belongs to one of the following:

- Planned tours: healthcare personnel will periodically visit certain rooms or patients as a part of a pre-established plan. Data about how shifts are carried out is essential to evaluate assistance quality and the efficiency of nurses and nursing assistants.
- Assistance tasks: nurses and nursing assistants must perform certain tasks as a response to an assistance call. It would be great to know in advance these tasks, so they can be monitored properly.
- Patients' satisfaction: the most important service quality subjective metric is the patients' satisfaction, which is obtained by mean of surveys.

As said before, AdvantCare software comprises three systems, as well as communication/integration interfaces.

1) Buslogic

This software oversees communication with the assistance calls system. It also handles GestCare and MediaCare, which are the systems used for tasks planning, personnel work schedules, patient information, satisfaction surveys, and entertainment. Buslogic will retrieve core business information about the assistance process: alerts, waiting times to assist patients, and achieved assistance objectives.

2) AdvantControl

This software controls and monitors the infrastructure and automation functionalities, including the status of lights, doors or the DataCare infrastructure itself. It will provide real-time alerts about possible quality of service issues.

3) EasyConf

This software manages SIP Communicator and provides data about calls such as the origin, the destination and the total call duration.

4) Communication/Integration APIs

Data can be retrieved from AdvantCare servers by means of SOAP web services, which will be used in those requests that require high processing capacity, and are stateless. Also, the information can be accessed via a REST API, where the calls are performed through HTTP requests, and data is exchanged in JSON-serialized format. REST servers are placed in the software servers themselves (either Buslogic, AdvantControl or EasyConf), thus allowing real-time queries; as well as parameters modifications. Finally, a TELNET channel will allow asynchronous communication to broadcast events from the servers to the connected clients.

B. Data Processing and Analysis Module

The Data Processing and Analysis Module is part of a Big Data platform based on Apache Spark [14], which allows an integrated environment for the development and exploitation of real time massive data analysis, outperforming other solutions such as Hadoop MapReduce or Storm, scaling out up to 10,000 nodes, providing fault tolerance [15] and allowing queries using a SQL-like language.

As shown in Figure 1, this module comprises four different systems: Preprocessing Engine, Processing Engine, Big Data and Historic Data Warehouses and Analytics Engine.

1) Preprocessing Engine

This system performs the ETL (*Extract-Transform-Load)* processes for the AdvantCare data. It will first communicate with AdvantCare using the available APIs to retrieve the data, which will be later transformed into a suitable format to be introduced to the Processing Engine. Because of the metadata provided by AdvantCare, the information can be classified to ease its analysis. Normalized and consolidated data will be stored in MongoDB, the leading free and open-source document-oriented database, where collections will store both data for real time analysis as well as historic data to support batch analysis to compute the evolution of different metrics in time.

2) Processing Engine

This system runs over the Spark computing cluster, and oversees data consolidation processes for periodically aggregating data, as well as to support the alert and recommendation subsystems.

3) Data Warehouses

Data filtered by the Preprocessed Engine and enriched by the Processing Engine will be stored in the Big Data Warehouse, that will store real-time information. Additionally, the Historic Data Warehouse stores aggregated historic data, which will be used by the Analytics Engine to identify new trends or trend shifts for the different quality metrics.

4) Analytics Engine

This system runs the batch processes that will apply the statistical analysis methods, as well as machine learning algorithms over real-time Big Data. Along with the historic data, time series and ARIMA (autoregressive integrated moving average) techniques provides diagnosis of the temporal behavior of the model. This engine also implements a Bayes-based early alerts system (EAS) able to detect and predict a decrease in the service quality or efficiency metrics under a preset threshold, which will be notified via push or email notifications.

C. Data Visualization Module

This module provides a reporting dashboard that will receive information from the Big Data platform in real time and will display two panels. The first panel will show the main quality and efficiency metrics in real time, along with its evolution over time and the quality thresholds. The second panel will provide the diagnoses computed by the Analytics Engine, as well as intelligent recommendations to prevent reaching undesired situations, such as metrics falling below acceptable thresholds.

The dashboard is implemented using the D3.js library, providing nice and intuitive visualizations.

```
{
“_id”: ObjectId(“565c234f152aee26874d7a18”),
“full_event”: true,
“presence”: {
      “ev”: “EV PRES”,
      “ts”: ISODate(“2015-10-02T01:35:36.384Z”)
},
“area”: “Madrid”,
“notification” : {
      “ev”: “EV NOTIF”,
      “ts”: ISODate(“2015-10-02T01:32:21.984Z”)
},
“room_number”: “126”,
“location”: “PERA”,
“activation” : {
      “week”: 40,
      “weekday”: 5,
      “user”: “Anonimo”,
      “hour”: 1,
      “minute”: 31,
      “year”: 2015,
      “month”: 10,
      “day”: 2,
      “ev”: “EV PERA”,
      “ts”: ISODate(“2015-10-02T01:31:45.696Z”)
},
“room_letter”: “-”,
“center”: “Aravaca”,
“day_properties”: {
      “holiday_or_sunday”: true,
      “social_events”: true,
      “rain”: true,
      “extreme_heat”: true,
      “summer_vacation”: true,
      “holiday”: true,
      “weekend”: true,
      “friday_or_eve”: true
},
“floor”: “1”,
“times”: {
      “cancellation_notification”: 195,
      “used”: 194,
      “idle”: 36,
      “cancellation_activation”: 231,
      “total”: 230,
      “cancellation_presence”: 1
},
“hour_properties”: {
      “shift_change”: true,
      “shift”: “TARDE”,
      “sleeptime”: true,
      “nurse_count”: “8”,
      “dinnertime”: true,
      “lunchtime”: true
},
“cancellation”: {
      “ev”: “EV CPRES”,
      “remote”: true,
      “ts”: ISODate(“2015-10-02T01:35:37.248Z”)
}
}
```

Fig. 2. Sample JSON document representing an assistance task event in the MongoDB events collection.

```
{
        “_id”: ObjectId(“569e50b1aa40450a027eb4ec”),
        “floor”: 3,
        “room”: 326,
        “date”: “1/10/15”,
        “hour”: “9:00:45”,
        “center_name”: “Aravaca”,
        “ts”: ISODate(“2015-10-01T09:00:45.000Z”),
        “shift_type”: “MAÑANA”
}
```

Fig. 3. Sample JSON document representing a shift in the MongoDB *shifts* collection.

```
{
        “_id”    :    ObjectId(“569e483daa404509a9796754”),
        “care_punctuation”: 2,
        “center”: “Aravaca”,
        “area”: “Madrid”,
        “floor”: 2,
        “night_punctuation”: 5,
        “morning_punctuation”: 4,
        “speed_punctuation”: 2,
        “price_quality_punctuation”: 2,
        “afternoon_punctuation”: 4,
        “year”: 2015,
        “month”: 11,
        “day”: 27,
        “date”: ISODate(“2015-11-27T00:00:00.000Z”),
        “global_punctuation”: 2,
        “id”: “Anonimo”,
        “room”: 221
}
```

Fig. 4. Sample JSON document representing a satisfaction survey in the MongoDB *surveys* collection.

IV. Preprocessing Engine

The Preprocessing Engine performs the ETL process over the data, and this section will describe how different data are extracted from the various sources, transformed and loaded as a part of this process.

A. Extraction

This engine extracts the assistance calls data by polling the AdvantCare module every five minutes, retrieving all data generated by all the rooms. Data from planned tours are retrieved daily also by polling the REST API, while patients' satisfaction surveys are loaded as CSV files.

B. Transformation

The Preprocessing Engine performs several transformation tasks so that data is in a suitable format to be handled by the Processing Engine and the Analytics Engine.

1) Assistance Tasks Events

Assistance tasks events will be transformed into MongoDB documents, where each event will be stored in a different document, and all of them will belong to the *events* collection. When one event status changes (e.g., from “activated” to “notified”), the document is updated to reflect these changes.

Figure 2 shows a sample document representing an event.

2) Planned Tours

Data from planned tours are retrieved daily from AdvantCare using the REST API, and are transformed to a MongoDB document in the *shifts* collection. A sample document is shown in Figure 3.

3) Satisfaction Surveys

As stated before, satisfaction data are loaded as CSV files. The Preprocessing Engine transforms it into a MongoDB document, which will be stored into the *surveys* collection. Figure 4 shows the structure of a sample document representing a satisfaction survey.

```
{
“_id”: ObjectId(“5665a51f0b1d4cf6f9728ae4”),
“center”: “Aravaca”,
“date”: {
   “week”: 40,
   “weekday”: 4,
   “hour”: 4,
   “ts”: ISODate(“2015-10-01T04:00:00.000Z”),
   “year”: 2015,
   “month”: 10,
   “day”: 1
},
“idle_time”: 67,
“wait_time”: {
   “floors”: {
      “1”: 0.6363636363636364,
      “2”: 29.5,
      “3”: 120,
      “4”: 0.5
   },
   “shifts”: {
      “NOCHE”: 23.7222222222222
   },
   “total”: 427,
   “types”: {
      “EV HABA”: 4,
      “EV PERA”: 359
   }
},
“used_time”: 344,
“activity”: {
   “floors”: {
      “1”: 11,
      “2”: 2,
      “3”: 3,
      “4”: 2
   },
   “shifts”: {
      “NOCHE”: 18
   },
   “total”: 18,
   “types”: {
      “EV HABA”: 17,
      “EV PERA”: 1
   }
}
}
```

Fig. 5. Sample JSON document representing consolidated data in the hourly collection.

C. Load

Once data is transformed into MongoDB documents (BSON format), they are loaded into the corresponding MongoDB collection.

V. Processing Engine

The Processing Engine will run batch processes to consolidate data previously transformed by the Preprocessing Engine. This consolidation will aggregate data to be handled by the Analytics Engine.

A. Periodic Data Consolidation

As the Processing Engine consolidates data periodically; two new collections are created, namely *hourly* and *daily*, depending on the periodicity of the aggregated data. A sample document in the *hourly* collection is shown in Figure 5. This aggregation enables fast visualization of aggregated data, and it is key for the Analytics Engine

```
{
        "_id"    :    ObjectId("56850cb00b1d4cf6f9b4f2da"),
        "center": "Aravaca", "activity": {
        "total": [
        {"type": "yesterday", "hour": 0, "value": 106},
        {"type": "lastweek", "hour": 0, "value": 58},
        {"type": "lastmonth", "hour": 0, "value": 52},
        {"type": "alltime", "hour": 0, "value": 51.1489},
        {"type": "yesterday", "hour": 1, "value": 20},
        {"type": "lastweek", "hour": 1, "value": 33.571},
        ...
}
```

Fig. 6. Sample JSON document representing a fragment of the real-time information for the KPI "activity" in the *realtime* collection.

to detect strange behaviors, fire alerts, or make recommendations.

Both the *hourly* and *daily* collections are indexed by timestamp, to enable fast filtering on consolidated data based on temporal queries.

B. Real-time Data Processing

To support the real-time dashboard, a process will take the data from the *hourly* collection and compute the average value for each KPI for different time periods: last day, last week, last month, and since the beginning. This allows comparing the current value for a KPI with the average of past periods of time. A small fragment of a sample document in the *realtime* collection showing the aggregated data for the "activity" (number of events) KPI is shown in Figure 6.

VI. Analytics Engine

The Analytics Engine is responsible of performing an intelligent analysis of the data to compute daily prediction, firing alerts when an undesired condition is detected (e.g., a certain metric falls under a specified threshold) and suggesting recommendations. This section describes these processes.

A. Prediction System

The prediction system takes the data contained in the *events* collection along with contextual data (weather, holydays or labor dates, etc.) and predicts the estimated value for each KPI for every hour in the next day. This batch process is executed daily. The predicted values are stored in a document per each KPI, in the *predictions* collection in MongoDB. A sample document is shown in Figure 7.

The prediction algorithm will analyze behavioral patterns in the events data and will apply these patterns to simulate future behavior. The algorithm proceeds as follows for each KPI:

Given N clusters, the algorithm computes a matrix M where each row is a cluster and each column is an hour, thus resulting in a $Nx24$ matrix. The value in the position $M_{i,j}$ will contain the average value of the KPI for events happening in the cluster i and in the jth hour of the day:

$$M = \begin{bmatrix} M_{1,0} & \cdots & M_{1,23} \\ \vdots & \ddots & \vdots \\ M_{N,0} & \cdots & M_{N,23} \end{bmatrix}$$

Also, vector DA will contain the hourly averages from the previous day:

$$DA = (DA_0, DA_1, \dots, DA_{23})$$

Then a vector of weights $w = (w_1, \dots, w_N)$ is computed, where each element is obtained as given in (1):

```
{
        "_id": ObjectId("5683f978e4b0d671e427e1db"),
        "center": "Aravaca",
        "name": "wait_time.total",
        "date": "1/10/15",
        "predictions": {
        "0": 5637,
        "1": 28557,
        "2": 15711,
        "3": 4133,
        ...
}
```

Fig. 7. Sample JSON document representing a fragment of the predictions for the "wait time" KPI in the *predictions* collection.

```
{
        "_id": ObjectId("5697b55d0b1d4cf6f9b59a63"),
        "center_name": "Vistalegre",
        "date": ISODate("2016-01-14T15:00:00.000Z"),
        "type": "activity.types.EV HABA",
        "status": "unseen",
        "group": "anticipated",
        "description": "WARNING: It has been detected
           a decrease in the activity of the type EV HABA
           between 15:00 and 16:00 (14/01/16), falling below
           the acceptable threshold.",
        "shift": "noon",
        "subject": "Early alert: activity of type EV HABA"
}
```

Fig. 8. Sample JSON document representing an alert in the *alerts* collection

$$w_i = \frac{1}{1 + \sqrt{\sum_{j=0}^{j=23} \left(DA_j - M_{i,j}\right)^2}} \tag{1}$$

Every day at 12 AM the vector containing the estimation for the following day (DE) is computed as in (2):

$$DE = (DE_0, DE_1, \dots, DE_{23}) = \frac{w * M}{\sum_{j=1}^{j=N} w_j} \tag{2}$$

As the day goes by, we will be discovering information of the current days' vector (DP):

$$DP = (DP_0, DP_1, \dots)$$

At 8 AM and 4 PM, we will re-estimate the DE vector as in (3):

$$DE_j = DE_j + DE_j * \frac{\sum_{j=A}^{j=B} \left(\frac{DP_j}{DE_j} - 1\right)}{8} \tag{3}$$

In the previous equation, A will be 0 at 8 AM and 8 at 4 PM, while B will be 7 at 8 AM and 15 at 4 PM.

The N clusters are determined based on contextual information, such as whether the day was weekday, it was rainy, it was extremely hot (over 35 ºC) or it was an important day because any other reason.

B. Alert System

The Analytics Engine is able to provide two kinds of alerts: real-time or early alerts. The former alerts are thrown as the data is stored in real time. To check whether an alert is to be fired, a KPI's average value over the last hour is compared with its average historic value. An anomaly is considered when the current average value falls above or below a threshold determined by the historic average plus/minus its historic standard deviation, and if the anomaly occurs, then the alert is fired. The four metrics or KPIs considered for real-time alerts are the average number of events, the average waiting time, the average time required by the healthcare personnel, and the average time required by other processes (neither waiting time or time required by healthcare personnel).

The latter kind of alerts are computed hourly over the forecast provided by the prediction system, and these are thrown when these predictions estimate that certain KPIs will fall above or below the specified thresholds with high probability.

Once an alert is fired, a document (see Figure 8) is stored in the alerts collection, so that the alert information can be shown in the dashboard.

C. Recommendations System

The recommendation system consists of a set of rules closely related to the alerts, whose purpose is to optimize the service when some KPI can be improved. Some of these KPIs are the number of events, the waiting time, the satisfaction levels, etc.

The recommendation process runs weekly, as we have identified that it is the least amount of time required to find evidence of metrics that can be improved.

The rule database comprises 52 rules which have been designed by experts based on their domain knowledge. Besides the metrics themselves, some rules can also be based on contextual information such as weather. Also, if the system keeps firing the same alarm over time, the recommendation can be stated in more serious terms.

An example of rule stated in natural language is as follows: *If the current number of events is higher than the average number of events of the previous month plus half the standard deviation, and this excess has happened more than three times in the last month, then the recommendation is: "The activity is much higher than expected. At this moment, the center does not have enough healthcare personnel to attend all these events. It is urgent that the cause of the activity rise be identified or new personnel should be hired."*

When a recommendation is created, it will be stored in the *recommendations* collection, in a document formatted as shown in Figure 9. These documents will be processed and displayed by the dashboard.

```
{
"_id": ObjectId("56962a560b1d4cf6f9b5911e"),
"center_name": "Aravaca",
"date": ISODate("2016-01-14T00:00:00.000Z"),
"status": "unseen",
"group": "anticipated",
"text": "The activity is within the expected limits.
      No modification of the service is required.",
"status": "unseen",
"subject": "Recommendation about activity"}
```

Fig. 9. Sample JSON document representing a recommendation in the recommendations collection.

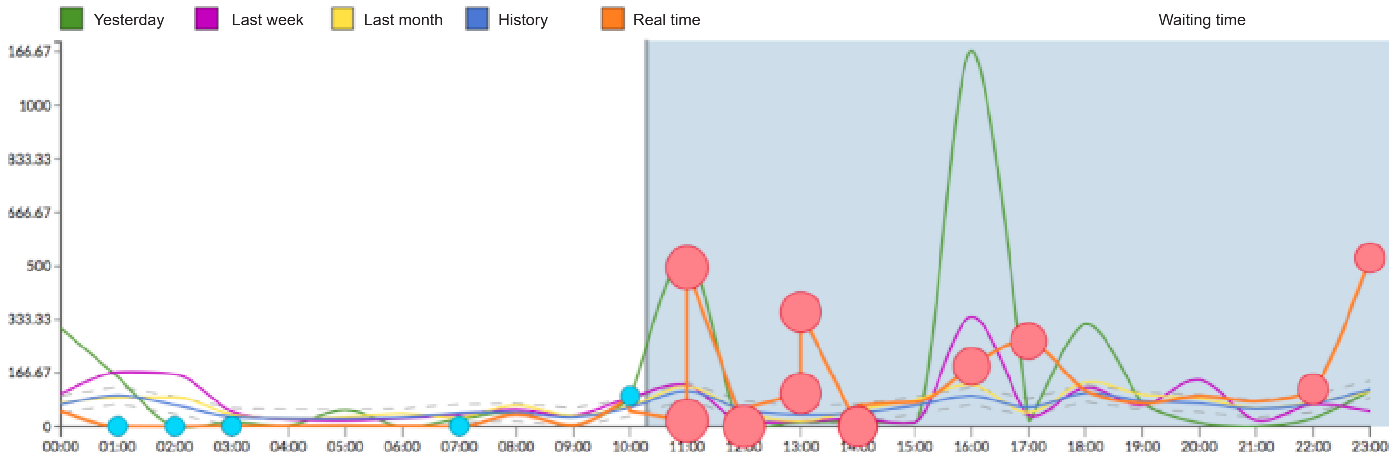


Fig. 10. Real-time dashboard displaying the average waiting times. The orange time series over the light blue background shows the predicted value for the rest of the day. Blue dots show real-time alerts, while red dots show early alerts. Different time series are shown, so that current and historic values can be compared.

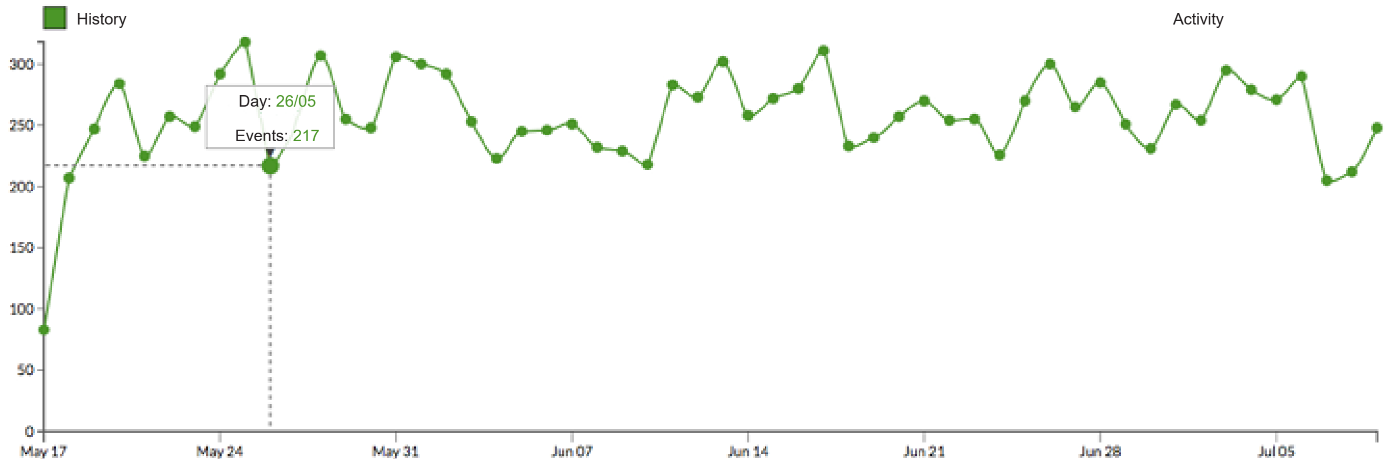


Fig. 11. History dashboard, showing the evolution in the activity (number of events) during two months in the past.

VII. Visual Analytics

The Visual Analytics engine allows visualization to easily see and understand the data gathered, processed and analyzed by the system. This engine provides six different dashboards, which are described in this section.

A. Home

The home dashboard displays tables with some basic information about the current status compared with historic values. For instance, we can see the value of each KPI today, compared with its value the previous day and the historic average.

B. Real-time

The real-time dashboard plots the evolution of the chosen KPI along the day, as shown in Figure 10 (in this case, the chosen KPI was "waiting time"). The orange line is the value for today, while other colors refer to historic values (green: yesterday, purple: last week, yellow: last month and blue: historic average). The light-blue section refers to the part of the day that belongs to the future, and thus the orange line in there is the forecast provided by the prediction system. Two dashed gray lines show the computed thresholds which determine the expected values for the KPI, and values outside that threshold are either shown with blue dots (real-time alerts) or big red dots (early alerts).

In this dashboard, not only the KPI can be chosen, but different filters can be applied: center, shift, type of event, etc.

C. Alerts

The alerts dashboard lists the alerts provided by the system, both real-time and early alerts. Also, information about the alerts can be obtained by clicking in the dots in the real-time dashboard.

D. History

The history dashboard shows the historic time series for the chosen KPI. Unlike the real-time dashboard, the history dashboard shows the evolution of the time series within a specified range of time. This dashboard is shown in Figure 11, which shows the evolution of the number of events during two months in the past.

E. Recommendations

Similar to the alerts dashboard, the recommendations panel lists the recommendations provided by the system, and the user can click on one of them to read further information about it.

F. Surveys

If the center has gathered information from satisfaction surveys, a summary of the results of these surveys is shown in this dashboard. It also shows the trend (whether positive or negative) using a color code, so that users can easily identified whether patient perception has improved regarding a certain KPI.

VIII. Evaluation

The system has been evaluated over the residential center of Aravaca (Madrid, Spain), gathering a total of 7,473 events. The KPIs that have been identified as essential are the number of hourly events (avg.: 15.37), the average waiting time (351.15 secs), the average time required by the healthcare personnel (35.47 secs), the average time required by other processes (315.68 secs), the daily number of remote cancellations (avg.: 46.36) and the average number of available nurses (6.79).

During the pilot, we have observed that the average waiting time during the night is much smaller (184.54 secs) than in other shifts, and most of the events take place in the evening shift (16.14 vs. 7.76 in the morning and 8.19 at night). Also, we conclude that there is a positive correlation between the number of events and the waiting time.

TABLE I
Pearson R^2 Correlation Coefficient over Waiting Time or Activity with Patients' Satisfaction, Grouped by Shift and Floor

Shift	Floor	R^2 (Waiting Time)	R^2 (Activity)
Morning	1	-0.791	-0.320
	2	-0.574	0.176
	3	0.058	-0.767
	4	-0.456	0.147
Evening	1	-0.631	-0.174
	2	-0.611	-0.754
	3	-0.720	0.070
	4	-0.928	-0.404
Night	1	-0.733	-0.524
	2	-0.910	-0.163
	3	-0.841	-0.266
	4	0.032	-0.539

Also, regarding the floor number, we have seen that lower floors have more events, and higher waiting times; and the trend shows that as the floor number grows (from 1 to 4), the activity decreases.

The timeframe between 8 PM and 1 AM is the busiest, showing that more personnel is required to attend the center's demand.

In addition, we have considered satisfaction surveys as an additional validation mechanism. To ensure that the quality metrics match the surveys' results, we have computed the Pearson R^2 correlation between the satisfaction levels and the number of events and waiting times (see Table I). As we expected, in almost every case, there is a strong inverse correlation, showing that more activity higher waiting times lead to less satisfied patients.

IX. Conclusions and Future Work

In this paper we have presented DataCare, an intelligent and scalable healthcare management system. DataCare is able to retrieve data from AdvantCare through sensors which are installed in the healthcare center rooms and from contextual information.

The Data Processing and Analysis Module is able to preprocess, process and analyze data in a scalable fashion. The system processes are implemented over Apache Spark, thus are able to work over Big Data, and all data (both historic, real-time and consolidated and aggregated values) are stored in MongoDB.

The Analytics Engine, which is part of the aforementioned module, implements a three-fold intelligent behavior. First, it provides a prediction system which is able to estimate the values of the KPIs for the rest of the day. This system runs as a daily batch process and the forecast is updated twice, at 8 AM and at 4 PM, to provide more accurate results. Second, it can provide both real-time alerts and early alerts, the latter ones are fired when some future prediction of a KPI falls outside the expected boundaries. Third, a recommendation system is able to provide weekly recommendations to improve the overall center performance and metrics, thus impacting in a positive manner in patients' satisfaction. Recommendations are based on alerts and a pre-defined rules set consisting of 52 rules, which has been designed by experts.

For the users to be able to see and understand the valuable information provided by DataCare, the Visual Analytics Module provides six different dashboards which displays a summary of the current status, real-time KPIs along with predictions and expected thresholds, historic values, alerts, recommendations and patients' surveys results.

DataCare has been implemented and tested in a real pilot in the residential center of Aravaca (Madrid, Spain). To validate the software, patients' satisfaction and KPIs correlation was explored, obtaining the expected results. The software also lead to some interesting conclusions regarding how KPIs vary depending on the context, such as the shift or the floor.

After the pilot, we have identified some improvements which are left for future work. First, healthcare personnel attending patients are not identified by the system, even though the sensors used allow this identification with the use of RFID tags. By identifying personnel, the center could trace the efficiency of each employee individually. Also, information about planned tours is very limited as it only observes the visited rooms and the visit times, but no other metrics.

So far, DataCare polls the AdvantCare API REST to retrieve data, but in the shortcoming future we will update the platform so that the communication is asynchronous.

To evaluate the prediction system, we also propose to develop a self-monitoring system which evaluates the deviation between the predicted and the real series, firing an alert if this deviation goes above a threshold, as it would mean that the prediction system is failing to accurately forecast the KPI.

Acknowledgment

Special acknowledgements to WildBit Studios for the development and pilots of DataCare. This project is partially funded by the Spanish Ministry of Industry, Energy and Tourism in the "Economy and Digital Society Strategic Action" program, under reference number TSI-100105-2014-62.

References

[1] A. Baldominos, E. Albacete, Y. Saez and P. Isasi, "A scalable machine learning online service for Big Data real-time analysis," in *Proc. 2014 IEEE Symp. Comput. Intell. Big Data*, Orlando, FL, 2014.

[2] R. C. Basole, D. A. Bodner, and W. B. Rouse, "Healthcare management through organizational simulation," *Decision Support Systems*, vol. 55(2), pp. 552–563, May 2013.

[3] C. Bossen, P. Danholt, M. B. Ubbesen, "Challenges of data-driven healthcare management: new skills and work," *Danish National Research Database*, 2016.

[4] J. W. Curtright, S. C. Stolp-Smith, and E. S. Edell, "Strategic performance management: development of a performance measurement system at the Mayo Clinic," *Journal of Healthcare Management*, vol. 45(1), pp. 58–68, Feb. 2000.

[5] J. L. Fortenberry Jr., and P. J. McGoldrick, "Internal marketing: a pathway for healthcare facilities to improve the patient experience," *International Journal of Healthcare Management*, vol. 9(1), pp. 28–33, Jun. 2015.

[6] H. A. Ghamdi, R. Alshammari, and M. I. Razzak, "An ontology-based system to predict hospital readmission within 30 days," *International Journal of Healthcare Management*, vol. 9(4), pp. 236–244, Apr. 2016.

[7] J. R. Griffith, and J. G. King, "Championship management for healthcare organizations," *Journal of Healthcare Management*, vol. 45(1), pp. 17–30, Feb. 2000.

[8] A. Jalal, S. Kamal, and D. Kim, "A depth video-based human detection and activity recognition using multi-features and embedded hidden Markov models for health care monitoring systems," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 4(4), pp. 54–62, Jun. 2017.

[9] M. J. Minniti, T. R. Blue, D. Freed, and S. Ballen, "Patient-interactive healthcare management, a model for achieving patient experience excellence," *Healthcare Information Management Systems: Cases, Strategies and Solutions,* Health Informatics, part II, pp. 257–281, Springer International Publishing, 2016.

[10] S. Mohapatra, "Using integrated information system for patient benefits: a case study in India," *International Journal of Healthcare Management*, vol. 8(4), pp. 262–271, Mar 2015.

[11] E. W. T. Ngai, J. K. L. Poon, F. F. C. Suk, and C. C. Ng, "Design of an RFID-based healthcare management system using an information system design theory," *Information Systems Frontiers*, vol. 11(4), pp. 405–417, Sep. 2009.

[12] J. P. Roberts, T. R. Fisher, M. J. Trowbridge, and C. Bent, "A design thinking framework for healthcare management and innovation," *Healthcare*, vol. 4(1), pp. 11–14, Mar. 2016.

[13] S. L. Ting, S. K. Kwok, A. H. C. Tsang, and W. B. Lee, "Critical Elements and lessons learnt from the implementation of an RFID-enabled healthcare management system in a medical organization," *Journal of Medical Systems*, vol. 35(4), pp. 657–669, Aug. 2011.

[14] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, Boston, MA, 2010.

[15] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing," in *Proc. 9th USENIX Conf. Netw. Syst. Design Impl.*, San Jose, CA, 2012.

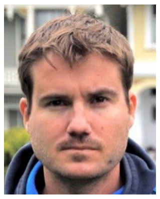
[16] Q. L Zeng, D. D. Li and Y. B. Yang, "VIKOR method with enhanced accuracy for multiple criteria decision making in healthcare management," *Journal of Medical Systems* (first online), Apr. 2013.

Alejandro Baldominos

Alejandro Baldominos is Computer Scientist and Engineer since 2012 from Universidad Carlos III de Madrid, and got his Master degree in 2013 from the same university. He is currently working as a researcher in the Evolutionary Computation, Neural Networks and Artificial Intelligence research group (EVANNAI) of the Computer Science Department at Universidad Carlos III de Madrid, where he is currently working in his Ph. D. thesis with a studentship granted by the Spanish Ministry of Education, Culture and Sport. He also works as Professor of the Master in Visual Analytics and Big Data at Universidad Internacional de la Rioja. He has published several conference and journal papers in the fields of artificial intelligence, Big Data and healthcare; and has been involved in several national and European research projects.

Fernando De Rada

Fernando De Rada is CEO at Wildbit Studios, and Associate Professor in Video Game Production at the University Camilo José Cela (UCJC) of Madrid. He received the degree in Physical Sciences from the Autonomous University of Madrid in 2003, and got his postgraduate Master in Image, Publicity and Corporate Identity from the UCJC in 2014. He is currently working in his Ph. D. thesis at the Faculty of Economic and Business Sciences of the National University of Distance Education (UNED). Over 25 years of experience as entrepreneur and senior executive in mobile, gaming and digital media, since the late 1980s has founded and managed different companies leading multidisciplinary teams with a verifiable track record of remarkable results. He has been managing several national research projects, in the fields of Big Data, healthcare, mobile and visual technologies. He is also member of the Spanish Interactive Arts and Sciences Academy, and has been awarded in 2014 with the Prize for a Professional Career by Retro Madrid and AUIC.



Yago Saez

Yago Saez received the degree in computer engineering in 1999. He got his Ph.D. in Computer Science (Software Engineering) from the Universidad Politécnica de Madrid, Spain, in 2005. Since 2007 till 2015 he was vice-head of the Computer Science Department from the Carlos III University of Madrid, where he got a tenure and is nowadays associate professor. He belongs to the Evolutionary Computation, Neural Networks and Artificial Intelligence research group (EVANNAI) and member of the IEEE Computational Finance and Economics Technical committee.