Project Acronym:	**LeanBigData**
Project Title:	**Ultra-Scalable and Ultra-Efficient Integrated and Visual Big Data Analytics**
Project Number:	**619606**
Instrument:	**STREP**
Call Identifier:	**ICT-2013-11**

D7.1 Use cases requirement analysis

Work Package:	*WP7 – User-Driven Research*	
Due Date:		30/11/2014
Submission Date:		31/11/2014
Start Date of Project:		01/02/2014
Duration of Project:		36 Months
Organisation Responsible for Deliverable:		ATOS
Version:		1.0
Status:		final
Author(s):	Jose Maria Fuentes Nines Sanguino Vrettos Moulos Pedro Torres Marc Sole Luigi Romano Victor Muntés-Mulero	ATOS ATOS ICCS PT CA SyncLab CA
Reviewer(s):	Ricardo Jimenez	UPM
Nature:	☒ R – Report ☐ P – Prototype ☐ D – Demonstrator ☐ O - Other	
Dissemination level:	☒ PU - Public ☐ CO - Confidential, only for members of the consortium (including the Commission) ☐ RE - Restricted to a group specified by the consortium (including the Commission Services)	
Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		

Revision history			
Version	**Date**	**Modified by**	**Comments**
0.1	21/05/2014	Jose Maria Fuentes López (ATOS)	Initial ToC
0.2	11/06/2014	Jose Maria Fuentes López y Nines Sanguino Gonzalez(ATOS)	Early draft requirement description of social network analytics use cases
0.3	16/06/2014	Jose Maria Fuentes López	Integrated Synclab contribution
0.4	03/07/2014	Jose Maria Fuentes López y Nines Sanguino Gonzalez(ATOS). Luigi Romano (Synclab)	Use case description for social network analisys case study (ATOS) and final version of use case description for financial banking (Synclab)
0.5	04/07/2014	Jose Maria Fuentes López y Nines Sanguino Gonzalez(ATOS)	Architecture description for social network analysis case study. Introduction.
0.6	11/07/2014	Jose Maria Fuentes López y Nines Sanguino Gonzalez(ATOS)	Case studies integration, Conclusions.
0.7	11/07/2014	Serge Mankovski Steve Greenspan Maria Velez-Rojas Victor Muntés-Mulero (CA)	Early draft of requirements for Cloud Data Center Monitoring
0.8	17/07/2014	Jose Maria Fuentes López y Nines Sanguino Gonzalez(ATOS)	Adding missing mockup
0.9	28/08/2014	Jose Maria Fuentes Lopez y Nines Sanguino (ATOS)	Analitycal query use case draft
0.10	26/09/2014	Jose Maria Fuentes Lopez y Nines Sanguino (ATOS)	Added analitycal query examples in use case description
0.11	13/10/2014	Marc Sole (CA)	Update of CA use case
0.12	19/11/2014	Pedro Torres (PT)	Added Targeted Advertisement use case
1.0	19/11/2014	Jose Maria Fuentes Lopez (ATOS)	First complete draft for internal review

---

Executive Summary

This document is the first deliverable of the Lean Big Data work package 7 (WP7). The main goal of the package 7 is to provide the use cases applications that will be used to validate the Lean Big Data platform. To this end, an analysis of requirement of each use case will be provided in the scope of task 7.1. This analysis will be used as basis for the description of the evaluation, benchmarking and validation of the Lean Big Data platform enclosed in the task 7.2.

Therefore, and in the context of task 7.1, this deliverable comprises the analysis of requirements for the following case of study provided in the context of Lean Big Data: Data Centre monitoring Case Study, Electronic Alignment of Direct Debit transactions Case Study, Social Network-based Area surveillance Case Study and Targeted Advertisement Case Study. Each case of study is presented by providing a general description as well as analysis of the requirements from functional and not functional point of view. A set of use cases and users are also provided to complete the definition of the scenarios.

The output of this deliverable will be used to describe the evaluation process in the task 7.2, as well as to guide the design process of the scenarios addressed in the work package 8 (WP8).

Table of Contents

Index of Figures

Index of Tables

Abbreviations and acronyms

DoW	Description of Work
WP	Workpackage
tbd	To be defined
SOA	Service Oriented Architecture
PEC	Portable Energy Consumption
TPP	Total Processing Power
HCI	Human Computer Interaction
SDLC	Software Development LifeCycle

1. Introduction

1.1. Purpose of the document

The present document has been produced by the consortium of LeanBigData in the context of work package 7 and corresponds to the deliverable D7.1 – Use cases requirement analysis, it is therefore in essence a report deliverable, describing the almost-final set of functional and not functional software requirements provided by the four main case of study in LeanBigData: Data Centre monitoring Case Study, Electronic Alignment of Direct Debit transactions Case Study, Social Network-based Area surveillance Case Study and Targeted Advertisement Case Study.

The goal of the document is to present the first output of the work package, a general description and analysis of the software requirements for each case of study. As part of the case of study definition a general overview, particular use cases for a scenario (set of possible sequences of interactions between systems and users in a particular use case), the actors involve in the use case and finally and main block a set of functional, not functional and context requirements is provided.

The resulting software requirements analysis will be used to proceed with the design and implementation of the use case applications, development process that will be addressed in the context of the work package 8. Once the scenarios will be defined, designed and implemented the resulting applications will be used for the validation process addresses as second and last output of the work package. The functional requirements also might include visualization requirements that could be useful for the visualization supported in work package 6.

Due to the main goal of the case of study is serving as a base for the evaluation process, and therefore benchmark the performance of the use case applications by executing them in uses-case tools and in high-scalable tools provided by LeanBigData, a common requirement for all the use case is obtained: all the scenarios should be centred in provide applications that deal somehow with large amount of data from the storage, analysed, visualized point of view or all of them. Taking into account the before-commented indication, following case of study will be implemented and therefore defined in the deliverable:

- Data Centre monitoring Case Study, a system model for monitoring and managing large IT infrastructures with the aim of predict energy consumption in a real life cloud environment
- Electronic Alignment of Direct Debit transactions Case Study, a system to avoid fraud on banking/financial transaction
- Social Network-based Area surveillance Case Study, is dealing with the challenging issue of using social media as complementary input for city officers and other actors in the prevention and detection of troubles in potentially problematic city areas
- Targeted Advertisement Case Study

1.2. Intended audience

The target readers are work package and task leaders involved with the LeanBigData developments, and the European Commission.

1.3. Structure of the document

The document is structured as follows:

Section 1 presents a brief overview of the document.

Section 2 presents and describes the methodology used to define the case of study requirements. The methodology helps to get a better understanding of the software requirement gathering process.

Section 3, Section 4, Section 5 and Section 6 present in detail the analysis of requirements respectively for Data Centre monitoring Case Study, Electronic Alignment of Direct Debit transactions Case Study, Social Network-based Area surveillance Case Study and Targeted Advertisement Case Study. For each section a case study overview, uses cases, and case study requirements sections are included.

Section 7 contains the conclusions and introduces the future work.

2. Methodological approach

2.1. Adopted methodology

This section describes the methodology, techniques and templates agreed and used to define the use cases requirements.

The use cases requirements methodology is following a classical approach followed by many ICT projects (e.g., (Wikipedia), (Nuseibeh, 2000)). It is a lightweight methodological approach taking into account several requirements categories and actors.

The deliverable describes the use cases both in a “business language”, and interpreted as simulated situations in which actors interact with the LeanBigData system to evaluate its viability and improvements with respect to the case study goals.

In the present document the domain knowledge acquired from the actors of the different use cases is described and, to some extent, formalised, deriving from this process the requirements that LeanBigData must satisfy from an end-user point of view.

2.2. Categorization of requirements

This section describes how requirements are organized across the document. IE: each case study defines several use cases. Each use case defines functional and non-functional requirements.

The LeanBigData case studies are specified through the following categories of requirements:

- List of actors;
- Context requirements – i.e., generic “environment” requirements;
- End-user requirements;
- Functional requirements;
- Non-functional requirements.

Finally, the non-functional requirements, especially targeting LeanBigData specific issues, are further subdivided in the following categories:

- Performance
- Integration
- Stability
- Maintainability
- Usability
- Scalability

It is worth noticing that the user-centric requirements specifically targeting user interfaces will be further elaborated in the scope of WP6 deliverable D6.1.

3. Cloud Data Centres: Data Centre Monitoring Case Study

3.1. Case Study Overview

Modern IT management systems employ variety of models for monitoring and managing large IT infrastructures. These models range from relatively simple event-condition-action rules to sophisticated simulation and multivariate regression. Models may also represent the relationship between different elements in the system. These models reflect normal, or expected, behaviour of managed environment. They are employed for detecting anomalies or changes, when behaviour of managed environment departs significantly from the expected behaviour given by a model, or for prediction in order to anticipate behaviour of the managed environment under changing load conditions and over time. The combination of graph models with other types of models is considered to be a powerful tool for root cause analysis.

Often these models are created using domain knowledge of experts. Another method of model creation involves process of training using monitoring or experiment data collected over a period of time. Once a model is created, the environment that produced training data might change significantly, and that would require model revision. For example, it might happen that the managed environment was patched to a new version of software. It is also possible that there was a change in a number of components within the managed environment, or there was a significant change in the way users use the managed environment. In all these cases a revision of a model is necessary.

In our case study, the produced model will be used for **predicting energy consumption in a real life cloud environment**, instrumented with CA products. Given a specific workload and a specific server, we may be able to predict the energy consumption of that particular server when running this workload.

However, conditions may change in dynamic environments and the models may require to be reviewed. The need for a model review can be derived from the monitoring and configuration data. For example, in a case where the environment was patched to a new version of software, monitoring data would reflect a change in behaviour that may not return to the previously observed pattern that was reflected in the training data used to create the model.

We will distinguish between two types of events:

- **Sporadic anomalies**: these are unexpected sporadic changes in the behaviour of the system that are not durable in time. Sporadic anomalies should be ignored in our case study, since models do not need to be recalibrated because the system behaviour has actually not changed. It is important to remark that, sporadic anomalies do not alter the model of what is normal, but they need to be identified and their cause and consequence need to be understood. Sporadic Anomalies can disrupt a system. In our case study we will need to explore the system when an anomaly is detected.
- **Durable changes**: are those changes that are reflected through durable alterations in behaviour of the system. Durable changes require model recalibration.

Therefore, in this case study we consider the use of both anomaly and change detection systems. These systems will identify time of change and set of affected components so that a new set of data could be created after a durable change is detected, for the purpose of training a new version of the model. We will also be able to analyse previous changes and anomalies in the past and try to learn patterns out of them that can improve our predictions in the future. Finally, when a new anomaly or change is detected we will be able to perform root cause analysis operations to help the analyst understanding the cause of those anomalies.

Business Motivation

CA Technologies provides different products that continuously monitor data and provide tools to analyse this data. Some of these products also provide mechanisms to model the data arriving to the system and even some tools for prediction. Following, we provide a short description of some of these products:

- **CA Application Performance Management** (CA APM): CA APM with application behaviour analytics provides application management of complex applications across physical, virtual, cloud and mainframe environments. It provides visibility across multi-tiers to help IT staff to better understand the impact of the analysed network and underlying infrastructure.
- **CA Lisa Data Mining**: CA LISA Data Mining enables customers to rapidly generate virtual services, automate the creation of test suites, use production performance data from CA Application Performance Management to create "live-like" test scenarios and increase collaboration between development and operations teams (DevOps).
- **CA Nimsoft Monitor**: CA Nimsoft Monitor is a scalable IT monitoring solution that provides 360-degree visibility into systems and infrastructure performance. It creates a single, unified architecture for both traditional and cloud environments, enabling system administrators to proactively monitor performance and availability. CA Nimsoft Monitor balances the simplicity associated with IT point solutions with enterprise scalability and multi-tenancy.
- **CA Data Center Infrastructure Management** (CA DCIM): CA DCIM provides real-time monitoring of power distribution and resource usage across data centers, enabling system administrators to measure, trend, alert and take action. It can also help visualizing and managing space and capacity, so that companies can better plan and optimize their data center operations. With CA DCIM, we can prevent downtime with continuous monitoring, intelligent alerting and visibility into data centers and IT infrastructures.
- **CA Capacity management:** CA Capacity Management is architected to help customers gain a deeper understanding of their near and long-term capacity needs, optimize asset utilization and cut costs. It provides end-to-end, cross-enterprise support to create an accurate picture of an infrastructure, suggest configuration changes and helps customers make more informed business decisions. It allows maintaining higher service levels across high-volume physical, virtual and cloud environments while helping to reduce capital expenditures and operating expenditures associated with hardware, software, power and cooling. In addition, it can assist in managing constant IT change with tools that predict the impact of workload increases, architectural changes or new application deployments. With this, it is possible to identify opportunities for consolidation and plan for capacity more accurately.

Figure 3.1, shows a brief overview about the relevant functionalities of these products with respect to our case study.

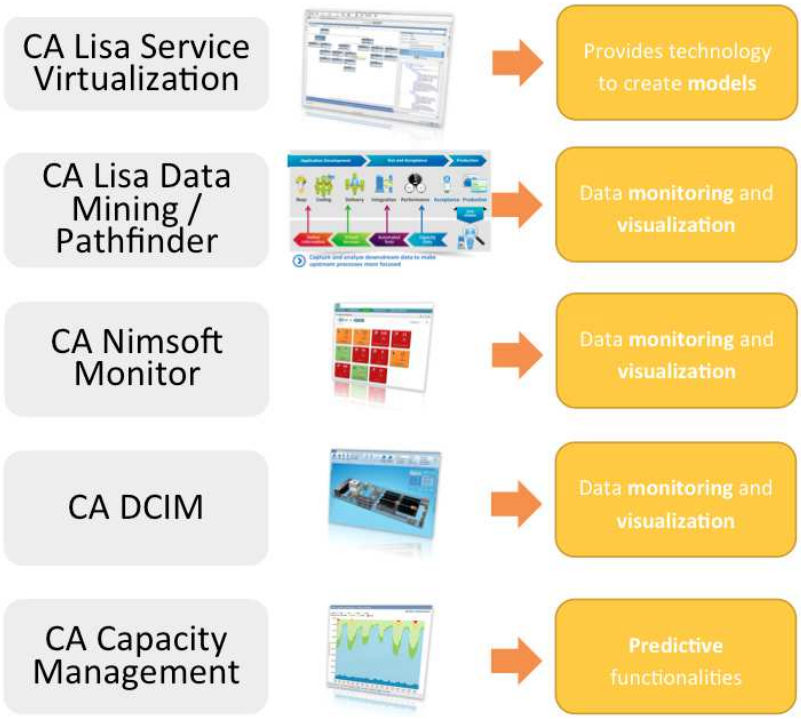


Figure 3.1. CA Products and functionalities related to this case study.

Just as an example CA Nimsoft ecoMeter allows running analytical models to predict future resource demands. Each model provides trend line forecasts based on existing usage data. Models calculations can employ **average**, **double exponential**, and **linear regression** algorithms. *Data extrapolation* allows a model to generate forecasts for variable calendar periods. CA Nimsoft ecoMeter allows creating and running models. To create a model, you define the model metadata. The metadata includes current usage capacity, expected growth rate, prediction algorithms, data source, and the number of forecast periods. Running the model applies the prediction algorithms to the data and generates the results. CA Nimsoft ecoMeter also allows adding events. This makes it possible to incorporate information about actual or "what if" changes in capacity, usage, or growth. Figure 3.2 shows an example of a screenshot showing part of the predictive functionality of this product.

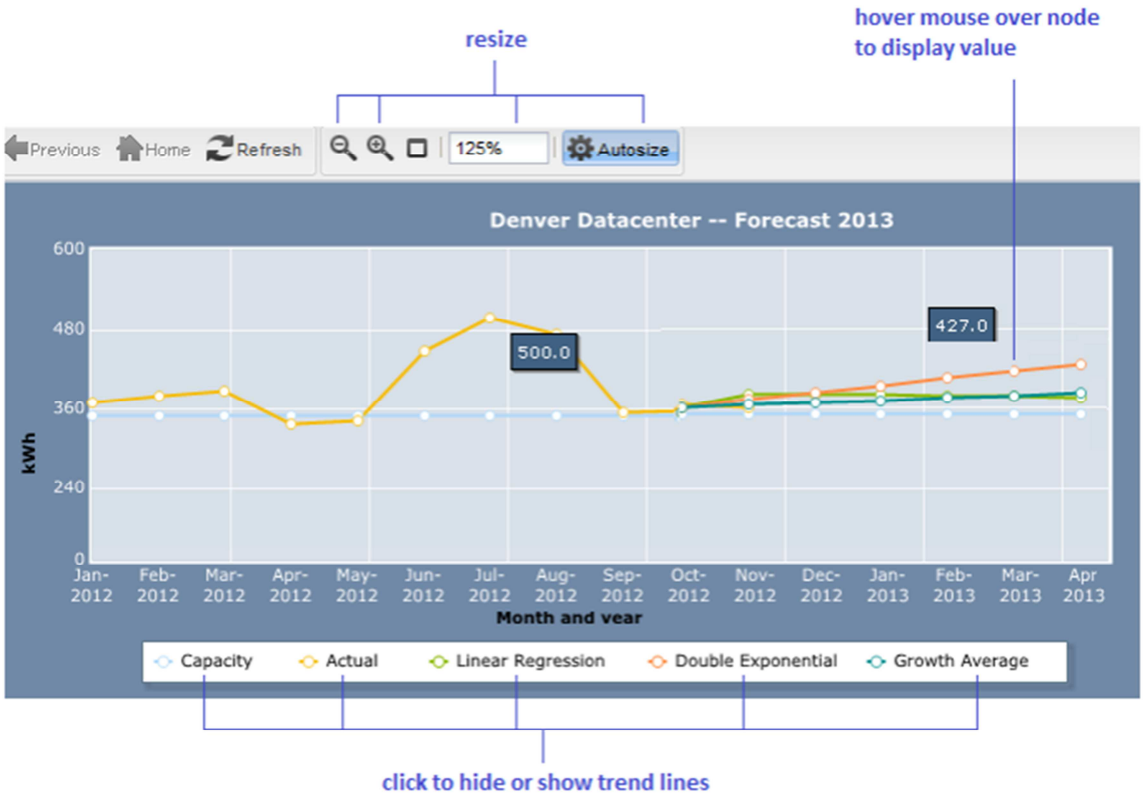


Figure 3.2. Screenshot of CA Nimsoft ecoMeter showing a forecast of energy consumption prediction in a data center, using different prediction models.

Visualization is also an important aspect in order to analyze the infrastructure of a data center. Following we provide some examples of visualizations currently offered by CA products. Figure 3.3 is a screenshot of the UI showing a facility summary view. In Figure 3.4 we show an example of a visualization layout for alarms overview. Figure 3.5 corresponds to an example of visualization for predictive analytic set up. An example of report can be seen in Figure 3.2, and an example of visualization of the infrastructure in a data center is shown in Figure 3.6.

CA Technologies also works with external partners to extend the functionalities of CA Nimsoft ecoMeter with new visualization tools. In particular, CA Nimsoft ecoMeter has extensions to present 3D visualizations of our data centers. Figures 3.7 and 3.8 show an example of a data center represented with this extension of our product.

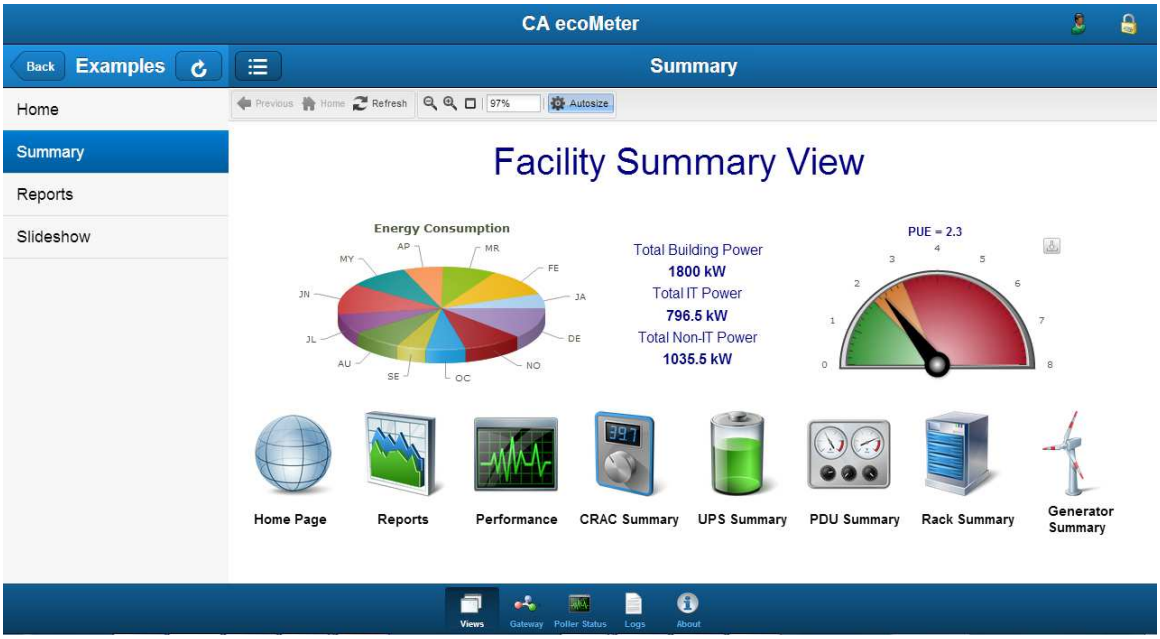


Figure 3.3. Visualization for data center overview in CA Nimsoft ecoMeter.

Figure 3.4. Visualization for alarm overview in CA Nimsoft ecoMeter.

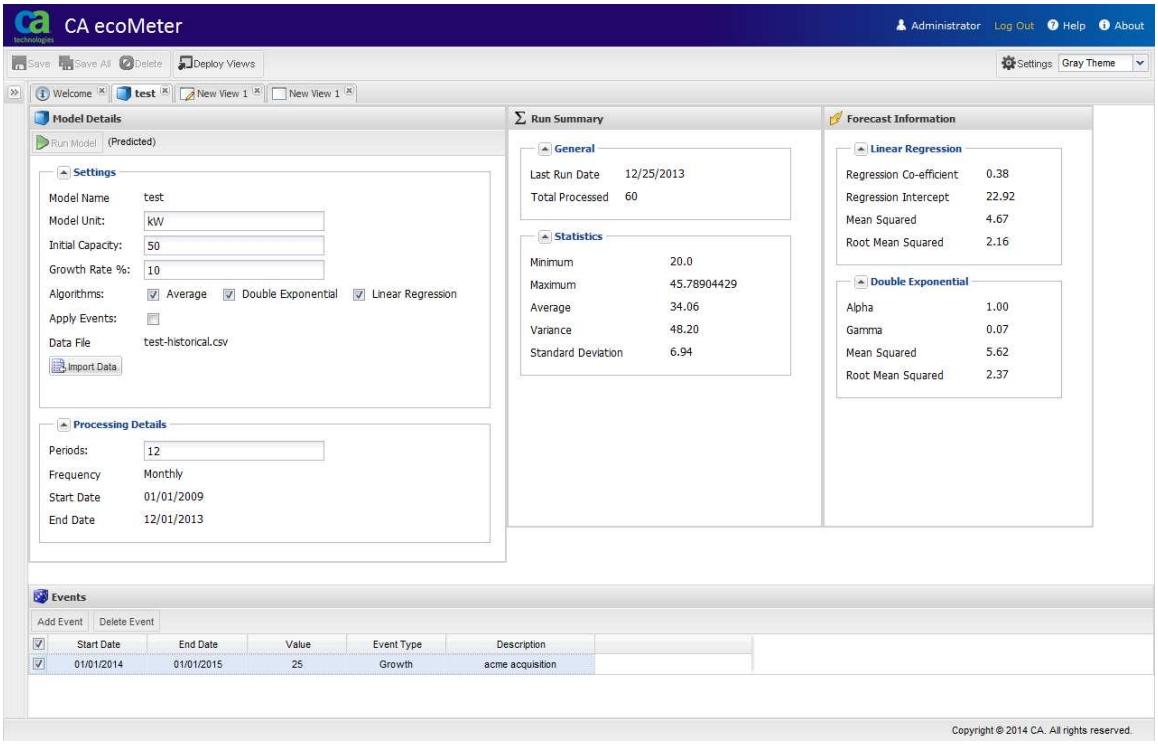


Figure 3.5. Predictive analytics setup in CA Nimsoft ecoMeter.

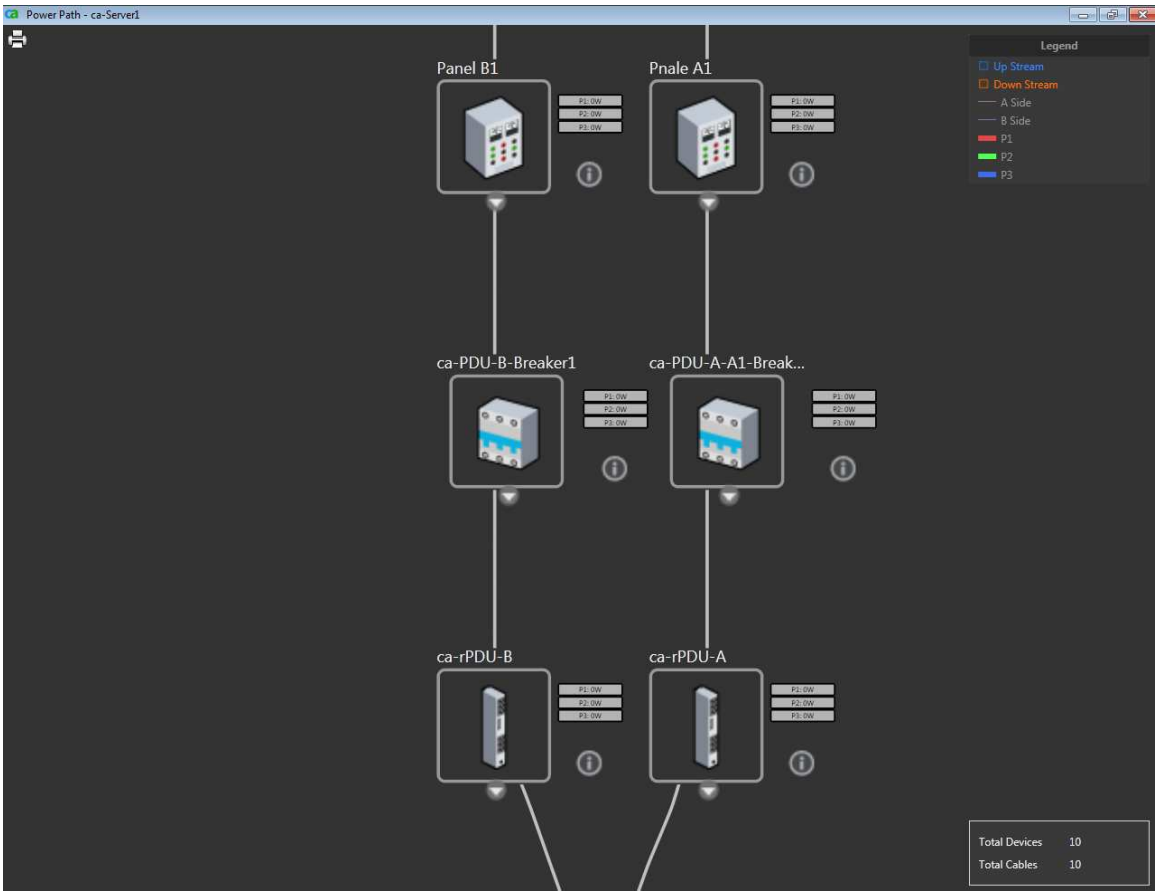


Figure 3.6. Example of infrastructure visualization.

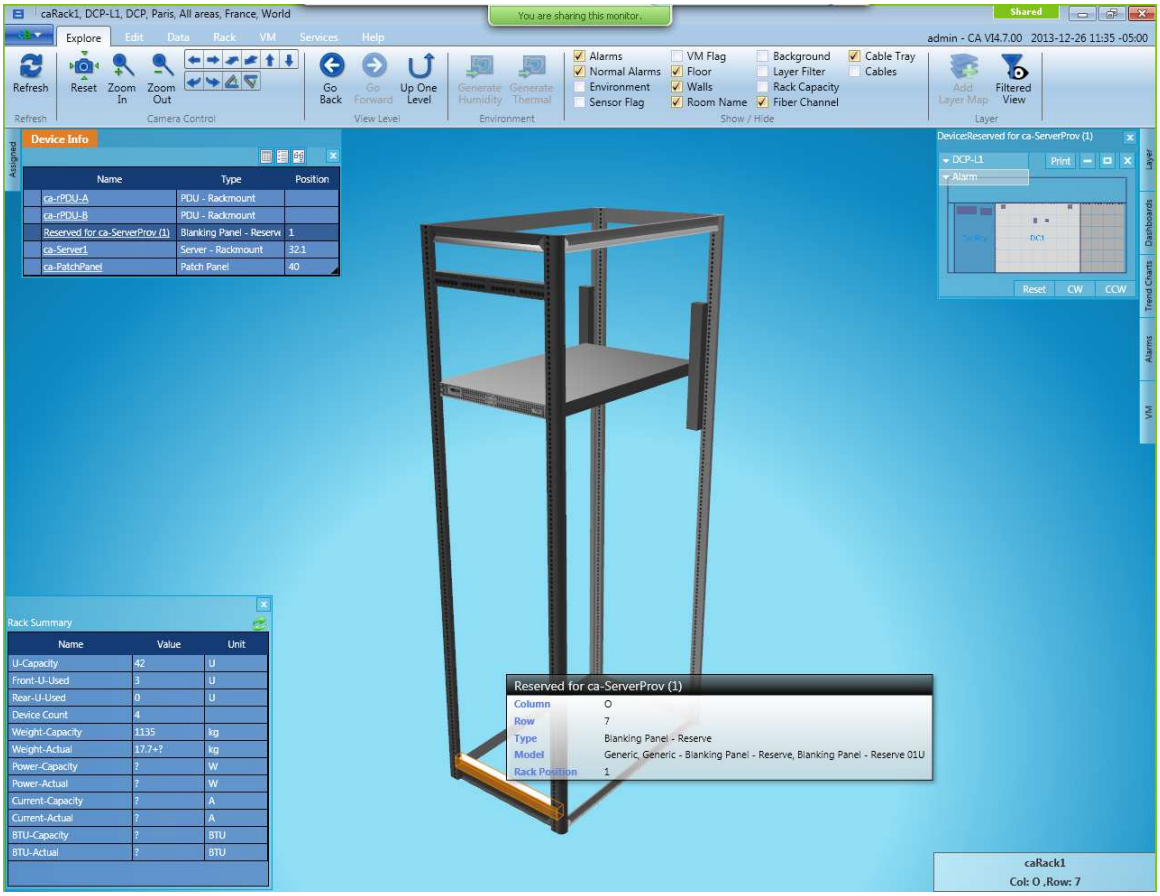


Figure 3.7. Single Rack 3D View.

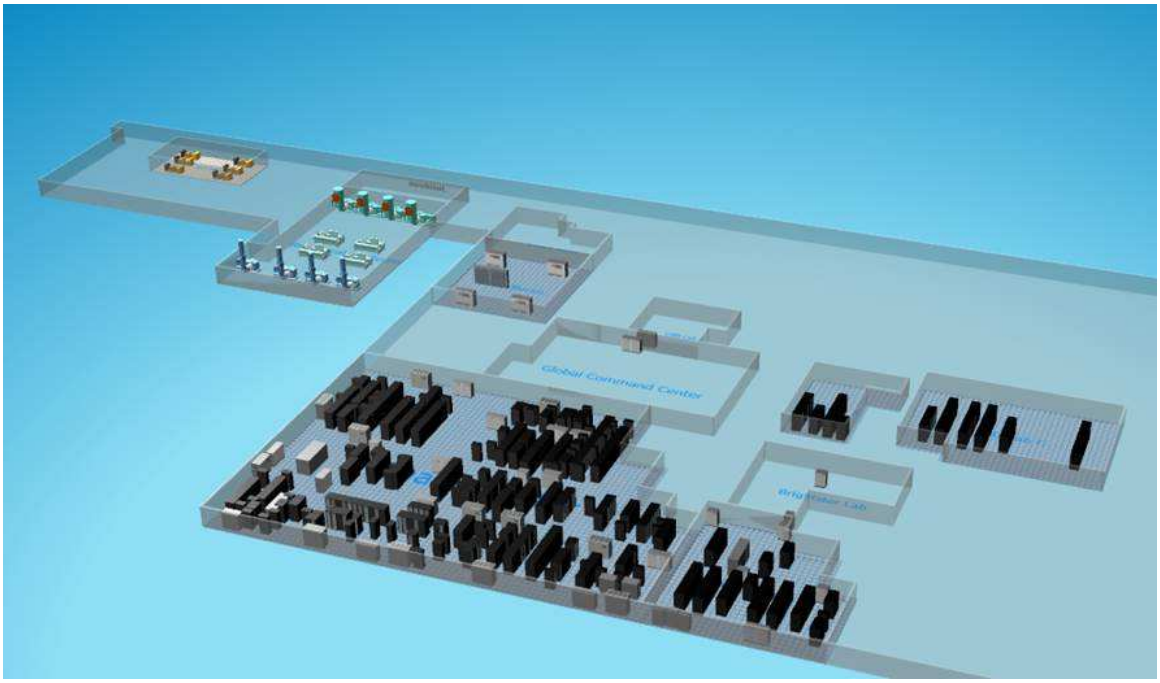


Figure 3.8. 3D Model of data center floor.

The purpose of our work in LeanBigData is to explore novel extensions of the functionality of these tools by using efficient big data storage and analysis functionalities. Specifically, we plan to use LeanBigData technology to improve our capabilities to recalibrate our models depending on the data collected from real systems, by using anomaly and change detection techniques and root cause analysis.

Since some of the decisions involved in this process might need human intervention (for instance to decide if an observed anomaly corresponds to a seasonal effect), we also plan to create new visualizations and to explore new HCI to improve the management of data centers to assist the users of these models.

Introduction to our case study

We divide our case study in two big blocks:

- Model recalibration and root cause analysis (Case Study – block 1)
- Anomaly pattern visualization and analysis from real-time and historic data for different media types, including novel HCI (Case Study – block 2)

Model recalibration and root cause analysis

One important part of our case study will enact the monitoring of a data center using CA tools to detect discrepancies with respect to an energy model of that facility. Detected changes will be automatically classified (when possible) as sporadic or durable and a root-cause analysis will be conducted. In some cases, human intervention would be required to make a decision, linking this block with the next one (Case Study – block 2). In case of a durable change or a seasonal effect that should be integrated into the model, the model will be recalibrated.

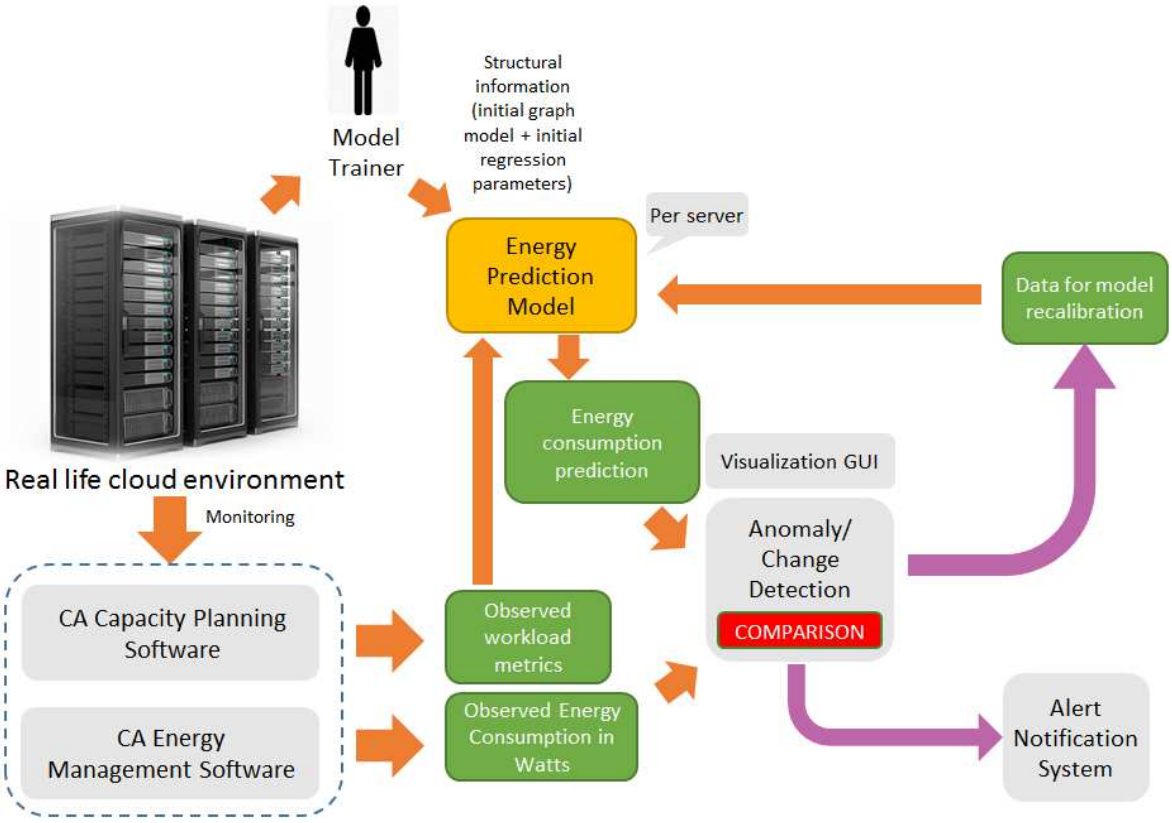


Figure 3.10. Overview of the regression model creation and recalibration in the Data Centre Monitoring Case Study

Figure 3.10 provides a general description of the part of the case study related to model creation and recalibration. There are several steps involved in this process: (i) model creation, (ii) monitoring, (iii) anomaly / durable change detection and (iv) model recalibration. We describe each of these steps as it follows:

- **Step 1: Model Creation**

The Model Trainer user will provide an initial graph model of the data center into the system, as well as the required structural parameters for the regression model. The initial monitored data coming from the data center will be used to create an initial regression model (together with the structural parameters fed by the Model Trainer) for the power consumption.

- **Step 2: Monitoring**

Energy consumed by computation, CPU utilization, memory utilization, network IO, and disk IO will be measured by some of the CA monitoring tools described in previous section. These products perform monitoring operations and process part of the data collected to produce new information or aggregates of raw data.

Monitoring data will be exported to LeanBigData store in form of a stream (see Section 3.2.2 for the details), where it will be integrated into a single homogeneous dataset. In this process, TPPs and energy consumption in watts will be computed.

- **Step 3: Durable change detection**

Based on the model computed in Step 1 and the data monitored in Step 2, we will compare the energy consumption predicted by the model and the energy consumption measured by the monitoring system. This step will allow assessing quality of the predictive models.

Anomaly and change detection software developed in LeanBigData would detect durable changes in the amount of energy consumed. This will require data mining algorithms for anomaly and change detection devised in T6.4 (WP6), as well as the root-cause analysis algorithms. All anomalies are stored into the system, and for those for which the automatic analysis was not conclusive or the type of event might involve human response (e.g., root-cause analysis determined that there is a possible malfunction of the cooling system in one of the racks), human intervention will be required, linking with the Case Study - block 2 functionalities.

- **Step 4: Model recalibration**

When a durable change is detected, the system would identify the moment in time when this change took place along with a list of servers affected by the change. This data will be used to produce a data query or ETL script that would extract data set from the LeanBigData store that could be used by system administrators to re-calibrate the models. The data set needs to be large enough to produce high quality regression.

Anomaly pattern visualization and analysis from real-time and historic data for different media types, including novel HCI

Detected deviations from model in (Case Study – block1) may be expected or not. For instance, when we modify the infrastructure of the data center by adding or removing elements, or we update software, we expect in general to detect changes in the behaviour of the system. However, in other situations unexpected durable changes or anomalies may occur. When this happens, it is important for system administrators to understand what the source of that particular change or anomaly is in order to decide if its consequence is significant and whether extra actions are required.

There exist many mechanisms for root cause analysis. Typically, these mechanisms provide indicators or hypothesis of potential root problems. Depending on the criticality of the problem, a system may take automatic actions based on automatic analysis (in our case, model recalibration as seen in Case Study – block 1). However, in other cases, the interpretation of these indicators may not be straightforward and human beings are required to validate predicted root causes.

For instance, if a server is moved from the top of the rack to the bottom, we should be able to detect a durable change in the temperature readings on the server through monitoring its temperature in idle state. Then, this would require changes not only in the temperature metric model, but also in the graph model. In particular, if we detect persistent discrepancy in temperature, this may be an indicator that a change in the ordering of *heat source* edges among servers in the graph model is required.

Another way in which human capabilities can be leveraged is to use visualization strategies that can profit from the fact that humans have an advanced visual system capable of identifying patterns. This can be of special utility to detect seasonal effects in the data center.

For instance, imagine a data center of a university that tries to reduce costs during August by relaxing the humidity control of the data center during that month. Looking at the power consumption models of August, the system administrator can appreciate an increment of 10% in the energy consumption, durable for several days. The model is recalibrated to accommodate the new change, but once August is finished, the consumption decreases 10%, and the model needs to be recalibrated again.

This kind of seasonal effects need that a human can visualize what happened in the data center in arbitrary time windows (last week, last month, same day of previous years, etc.). We plan to use the technology designed and developed in T6.2 and T6.3 (WP6) to make it easier for a

human being to detect and evaluate anomalies as well as the output quality of the algorithms performing automatic processes.

Since nowadays system administrators might have to react to anomalies in the shortest possible time, this means that they must be able to visualize and navigate through that information from a variety of devices. So visualizations have to consider:

- **Responsive multi-touch interaction**

Current mobile technologies open a whole new world of opportunities for Data Center management interaction tools. Being able to control any aspect of the Data Center from any device, without place limitations, would shorten a lot the response time from the system administrator. Emergencies in the Data Center could be managed from their own phone with just a few touches, without the need to go to the Data Center or to a physical desktop.

The idea behind this novel Human-Computer Interaction Data Center management technique is to adapt the display to the device the system administrator is using to monitor and control the Data Center, simplifying the data display and enabling controls using the touch screen that new smartphones and tablets have. The same display would be used for Desktops, but adapted to the mouse interaction and larger screens.

- **Gesture interaction**

The newest human interaction technologies can also fit when talking about Data Center management. They can provide innovative ways of controlling and exploring the Data Center, with easy and fast gestures that enhance the management process.

Contrary to the mobile environment where Data Center management is taken everywhere, in-place management can benefit from larger interactive displays that do not require the use of traditional input devices. For instance an operator could explore where an additional intervention is needed in the data center after changing a server (thus no longer having clean hands to properly use a mouse or a keyboard), without having to lose any time while keeping the control room clean, as only gestures would be enough to explore and diagnose the problem.

There are several steps involved in this Use Case block: (i) user visualizing and navigating historical data to find patterns and (ii) model modification. We describe each of these steps as follows:

- **Step 1: User visualizing and navigating historical data to find patterns**

When a system administrator has been notified that there are discrepancies to review, she connects to the LeanBigData system to visualize the type of discrepancy. From that point on, she can request to visualize and compare models or the data coming from the streams on arbitrary time frames.

- **Step 2: Model modification**

From the information extracted, the system administrator might:

- improve the anomaly automatic diagnosis (if not correct or incomplete)
- take actions on the data center,
- modify the graph model,
- add seasonal effects to the linear model (e.g., create a specific energy model for the Chinese new year)

3.2. Relevant models and data

In this section we present the models used in our case study and the data needed to work with them (to create, recalibrate, diagnose or store the models). The queries on that data are summarized after each Use Case (Section 3.3).

3.2.1 Models

In this case study, we will use two types of models: regression models predicting performance characteristics of IT elements; and graph models of relationships among elements in the IT infrastructure.

Regression models
Regression models can be represented as a function $E(v_1, \ldots, v_n, t)$. So given a set of input variables based on the data described in the previous section, it generates a function of t that we can use to explain how energy consumption will evolve in the next period T.

Graph models
In order to understand the potential effects that one element in the system may have with respect to other elements in the system, we need to take into account their relationship. For this, we create graph models that represent the relationship between these elements. Figure 3.9 shows an example of one of these graphs.
Let us assume a graph model G=(V, E). Two elements in the graph are connected when there is a relationship between them in data center. We have several types of nodes (see Data section), but let us concentrate on the two most important ones: racks and servers (V={R, S}). The relationships between nodes may be: contains, power source and heat source (E={C, P, H}). Following we describe these relationships in more detail:

- A rack r in R can relate to one or more servers in S. We have nodes in the graph representing both servers and racks. We call the relationship between a rack and one of the servers in the rack a *contains* relationship. This would be modeled as a Rack node connected to a number of server nodes by edges with *contains* label.
- In turn, a rack node r might also be the power source of several servers, hence there will be also edges between the server and the rack labeled *power source*.
- Air flows in the rack from the bottom to the top, so the servers below heat up servers above in the rack. Hence the servers themselves can have directed edges among them with a label *heat source*.

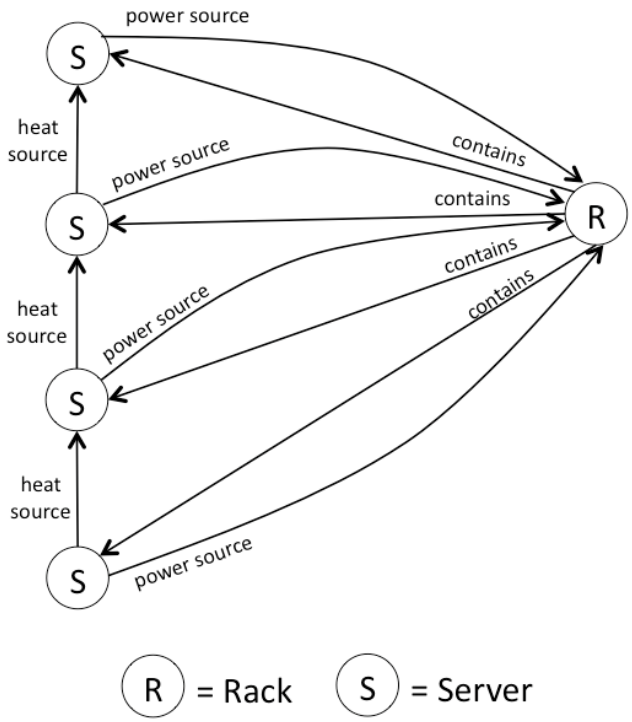


Figure 3.9. Example of graph model in the Cloud Data Center Monitoring case study.

Note though that even if you detect a persistent discrepancy in the temperature, you may not be able to infer the change in the position of servers, since these may be caused by other factors. In other words, if we detect a change in the metrics, we may recalculate the model, since we have non-debatable change in the metric values. However, if we detect a change in metrics, we cannot automatically infer changes in the graph. In this case, root cause analysis is required: based on the change detected in metrics, plus the previous graph model, we may recommend the system administrator to check if the position of the racks was changed. If the administrator confirms, then we change the graph model. So this would be a mechanism for human-guided RCA with the support of our system. In other words, regression models can be recalibrated (almost) automatically, while graph models need humans to validate hypothesis.

Following we describe the data that we plan to obtain from our data centers to build the models.

3.2.2 Data

Information from a data center will enter the CEP part of the LeanBigData data store as a stream. The stream (denoted as ***DataCenter Stream***) will contain tuples, separated with newlines, and each tuple will be a comma separated record (UTF8 encoded) with the following information:

Timestamp, Id, CPUload, BytesIn, BytesOut, DiskIO, Temperature, Power

Where

- **Timestamp** is the number of seconds since 1970 (Long)
- **Id** is an identifier string (String) for the server to whom this measure belongs to
- **CPUload** (Float) is the CPU load of that server, either the average load in the period from previous measure until current timestamp or the current value at the timestamp

- **BytesIn** (Long) are the bytes received by the server since the last measure
- **BytesOut** (Long) are the bytes sent from the server since the last measure
- **DiskIO** (Long) is a number correlated to the disk operations performed in the server (either bytes written/read, number of IO operations, etc.)
- **Temperature** (Float) the server's temperature (the exact measuring location might vary depending on data center and/or server type)
- **Power** (Float) the power consumed by the server (either an average since last measure or the instant one)

This stream must be stored and also handed over to the Energy Prediction Module, which will compute a prediction for the power consumption of each server. These predictions will be produced in form of a stream and stored in the system. This stream (denoted as ***Prediction Stream***) will be formed by tuples, separated with newlines, and each tuple will be a comma separated record (UTF8 encoded) with the following information:

Timestamp, Id, PredictedPower

Where

- **Timestamp** is the number of seconds since 1970 (Long)
- **Id** is an identifier string (String) for the server to whom this prediction belongs to
- **PredictedPower** (Float) the predicted power consumption of the server by the model

The Energy Prediction Module uses a regression model (see *Models* section) to compute the previous stream. This model might change over time (when it is recalibrated due to excessive discrepancies with current behaviour) and the system must be able to store the history of all such models. For this reason the regression models will be stored in the SQL storage of the LeanBigData data store in a table (*Models* table) containing the following fields (we do not include internally generated identifiers):

Name (String), Id (String), Date (Date)

Where

- **Name** is the model name
- **Id** is the server identifier for which this model applies
- **Date** is the creation date of the model, so that the history of models can be kept in the database (when a model is recalibrated, a copy of it is actually created and then modified)

The values of the regression model will be stored in table *Parameters*, with the following fields:

ModelId (Foreign key to *Models* table), Parameter (String), Value (Float)

Where

- **ModelId** identifies the model to whom this parameter belongs
- **Parameter** is the name of the parameter of the model
- **Value** is the actual value of the parameter

For root cause analysis a graph model is used instead (see *Models* section). In this case the graph will be stored in three tables, *Graphs*, *Nodes* and *Edges*, with the following fields

Graphs table: Name (String), Date (Date)

Nodes table: GraphId (Foreign Key to Graphs table), Room (String), X (int), Y(int), Z(int), Height(int), Type (String), Subtype (String), NodeId (String)

Edges table: Source (Nodes FK), Target (Nodes FK), Relationship (String), Value (Float)

Where

- **Name** is the name of the graph model
- **Date** is the creation date of the model, so that the history of models can be kept in the database
- **GraphId** identifies the graph model to whom the node belongs
- **Room** is the room identifier where the object is located
- **X, Y, Z** correspond to the Cartesian coordinates (Integer) of the server in the data center. Z is in U-units
- **Height** is the height of the component in U-units
- **Type** describes the type of the node (Rack, Server, Switch, KVM, Appliance, Router, Other)
- **Subtype** is the particular type of server, rack, etc. Might not only include the model, but also configuration information if necessary from the prediction model perspective.
- **NodeId** is the external id used to identify the rack or the server in the logs/streams (for instance for servers this should match the value Id in the *DataCenter Stream* and *Prediction Stream*)
- **Source** is the source node of the edge
- **Target** is the target node of the edge
- **Relationship** is the type of edge (e.g., heat source, contains, etc.)
- **Value** is an optional numerical value that allows quantifying the importance of that relationship

Discrepancies detected analysing the DataCenter Stream will be stored into the *Discrepancies* table, which has the following fields:

- **ServerId** (String) the identifier of the server where the anomaly was detected
- **EventId** (String) Id that allows grouping several discrepancies into a single logical event (should be a foreign key to a record in a datacenter event table)
- **Start time** (Long) timestamp of the anomaly start in the DataCenter Stream
- **End time** (Long) timestamp of the anomaly end in the DataCenter Stream
- **Type** (String) anomaly type (null, sporadic, durable, etc.)
- **Cause** (String) description of the root-cause of the anomaly (possibly including the Id of the root-cause element) [cause might be multi-string for several causes]
- **ProcessedByHuman** (Date) Timestamp at which a human processed this discrepancy (either by accepting the cause proposed automatically or by modifying it manually).

Discrepancies that correspond to durable changes might indicate the need to recalibrate a model or generate a new one. To decide so, the system administrator may browse historical data (either of the DataCenter Stream or the contents of the Model table), and indicate several time windows of the DataCenter stream that must be used to adjust/create the model (these windows might be initially suggested by the automatic root-cause analysis and later on manually

adjusted by the user). These time windows are kept in the *Training dataset* table, which contains the following fields:

- **DiscrepancyId** (Foreign Key to the Discrepancies table)
- **Start time** (Long) timestamp marking the beginning of a training data window in the DataCenter Stream
- **End time** (Long) timestamp marking the ending of a training data window in the DataCenter Stream

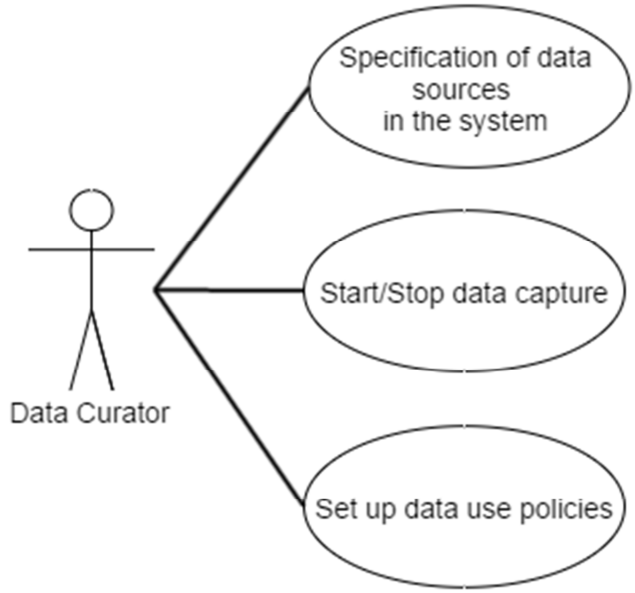
3.3. Use Cases

3.3.1 Use Case 1: Setup extraction of data from CA proprietary data stores and storing it in LeanBigData Store (CS1-UC1)

Use Case 1, consists in using LeanBigData technology to load, manage and combine the data produced by different CA products (like CA APM, CA Lisa, CA DCIM, CA Nimsoft, etc). This is an essential part of an end-to-end big data analytics process and we expect LeanBigData technology to help us speeding it up. In Use Case 1, we assume that a model has already been created to predict the behaviour of a system or subcomponents in that system. A <Data Curator> will be responsible for collecting data from different data sources and combine it together for later analysis.

Use case template	Description
Use case name	Extraction of data from CA proprietary data stores and storing it in LeanBigData Store
Priority of Accomplishment	Must Have

Use case description	Description

Use case diagram	
Goal	To prepare and load data extracted from proprietary data stores to LeanBigData.
Main Actors	<Data Curator>

Use case scenarios	**Description**
Main success scenarios	1. The <Data Curator> indicates the source of the data streams (either a remote source or a local file). 2. The <Data Curator> is able to start and stop the streams capture.
Preconditions	-
Postconditions	1. The stream data set is stored in the LeanBigData store

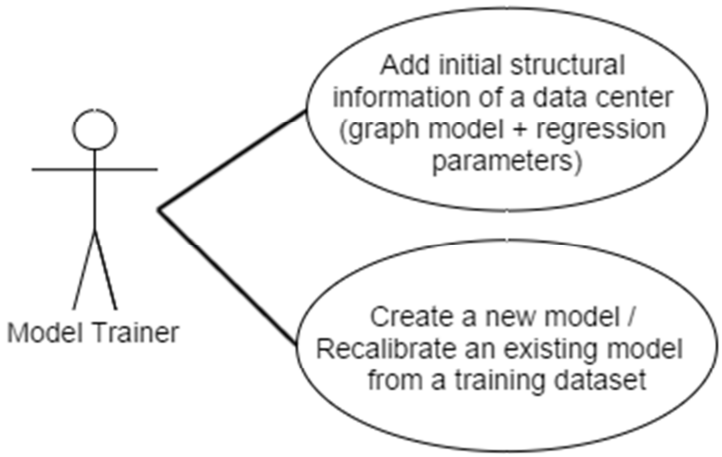
Other requirements

See Table 2, Table 3 and Table 4 in Section 3.4 for further detailed requirement descriptions.

3.3.2 Use Case 2: Model creation and recalibration (CS1-UC2)

There are two moments that require the manipulation of models. The first one is when the structural information of the data center is fed into the LeanBigData platform. The second is when we compare the current model predictions with the actual data collected and we realize that the model is not predicting correctly and needs to be adjusted. This use case allows the <Model Trainer> to create new models or recalibrate existing ones from training datasets that have been created in Use Case 3 and use Case 4.

Use case template	Description
Use case name	Model creation and recalibration
Priority of Accomplishment	Must Have

Use case description	Description
Use case diagram	
Goal	To create a model based on a set of data or to recalibrate an existing model.
Main Actors	<Model Trainer>

Use case scenarios	Description
Main success scenarios	1. The <Model Trainer> creates a new model that characterizes the data center (e.g., the graph model of the data center or linear regression constants). 2. The <Model Trainer> creates a new model or recalibrates an existing model that became obsolete because of a durable change in the system from a training dataset.
Preconditions	1. Data to create or recalibrate the model is stored in the system (structural information + training dataset)
Postconditions	1. The new or recalibrated model is stored

Other requirements

See Table 2, Table 3 and Table 4 in Section 3.4 for further detailed requirement descriptions.

Queries related to this Use Case

The creation/calibration of a model requires access to the *DataCenter Stream* stored data, using the time windows described by the training dataset (stored in the *Training dataset* table).

3.3.3 Use Case 3: Automatic discrepancy analysis (CS1-UC3)

CA Technologies provides software and solutions to simulate the behaviour of complex systems. This is done through the creation of models that try to predict the future behaviour of these systems. For example, CA Lisa Service Virtualization provides the technology necessary to create prediction models based on the data provided by other CA products, such as those presented in Use Case 1.

An important challenge faced by CA Technologies is to monitor the accuracy of the prediction of these models. It is important to detect when durable changes happen that make a previously generated model to be obsolete. A relevant research topic is the semi-automatic (or even automatic) evaluation of unexpected events in the monitored system, and the distinction between occasional anomalies and durable changes in the system that require recomputation of predictive models.

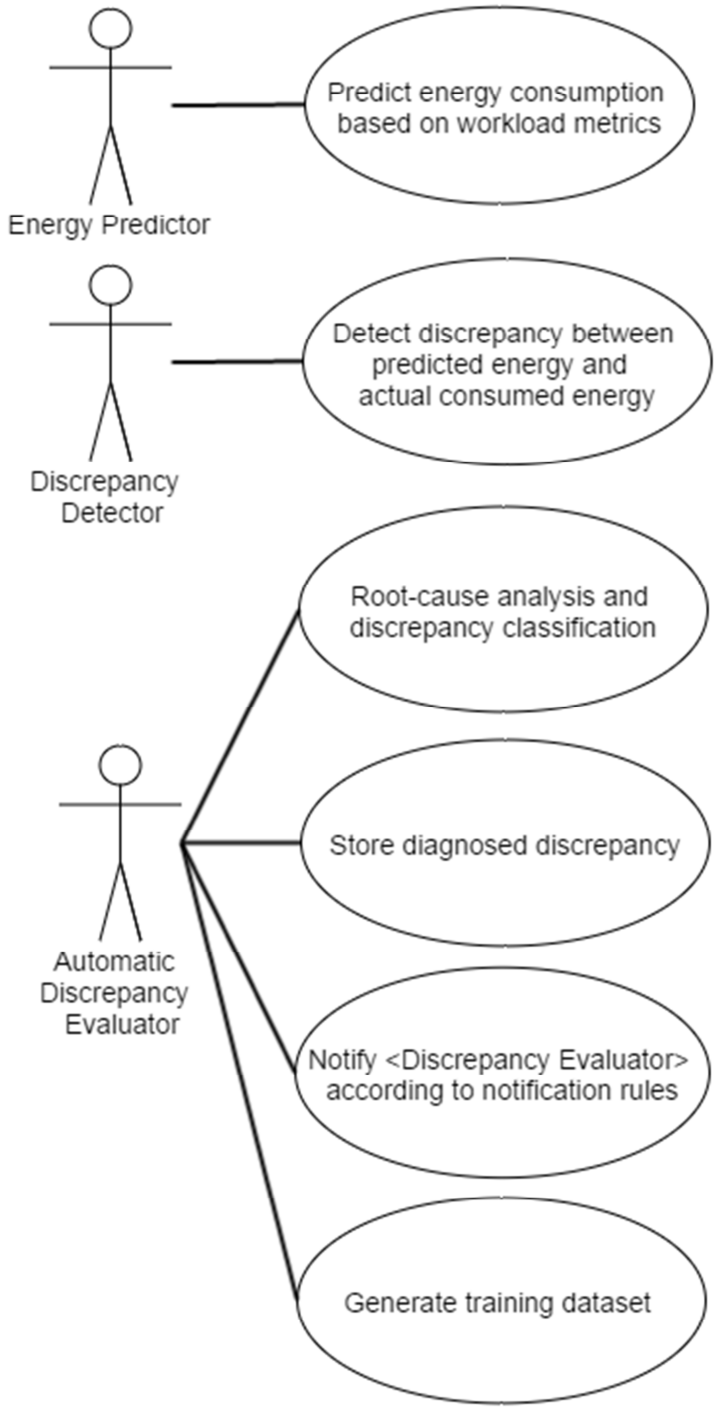
This Use Case deals with the detection of discrepancies between the predicted and real data, using root-cause analysis to diagnose the source of the discrepancy, and the automatic classification of those discrepancies as sporadic anomalies or durable changes. The output of such analysis is stored in the *Discrepancies* table.

In some cases the automatic analysis might be conclusive and the appropriate action would be to recalibrate the existing model. In such a case, then a training dataset would be created in the *Training dataset* table, thus linking this Use Case with Use Case 2.

Since the automatic analysis might not always be successful or conclusive, further investigation and data analysis by a human can be performed on such cases, linking this Use Case with Use Case 4.

Use case template	**Description**
Use case name	Automatic Discrepancy Analysis
Priority of Accomplishment	Must Have

Use case description	**Description**

Use case diagram	
Goal	To monitor the actual system and compare the indicators and metrics obtained with the predictions in the model, to detect discrepancies. Automatically diagnose and classify those discrepancies, storing them in the *Discrepancies* table and marking those that require human intervention. Generate training dataset if model needs to be recalibrated
Main Actors	<Energy Predictor>, <Discrepancy Detector>, <Automatic Discrepancy Evaluator>

Use case scenarios	Description
Main success scenarios	1. The <Energy Predictor> is able to produce a prediction (a tuple in the *Prediction Stream*) from the data in the *DataCenter Stream*. 2. The <Discrepancy Detector> is able to detect anomalies in the data monitored from the system compared to the model predictions. 3. The <Automatic Discrepancy Evaluator> finds the cause of the anomaly and classifies it as a sporadic anomaly or as a durable change.
Preconditions	1. The *DataCenter Stream* is receiving data.
Postconditions	1. A prediction for the consumed energy at the datacentre is produced. 2. The anomaly detector has detected anomalies in the energy prediction when these occur. 3. The anomaly is automatically diagnosed and recorded in the LeanBigData store. Depending on notification rules, a notification is sent to the <Discrepancy Evaluator> actor. 4. If automatic analysis was conclusive and model recalibration is needed, then appropriate training dataset is recorded in the LeanBigData store.

Other requirements

See Table 2, Table 3 and Table 4 in Section 3.4 for further detailed requirement descriptions.

Queries related to this Use Case

The energy prediction requires access to the *DataCenter Stream* stored data (typically the last measures of each server or a window of the last measures).

The anomaly detection needs to access the latest unprocessed tuple of each server in the *Prediction Stream* as well as the energy lecture in the *DataCenter Stream* corresponding to that prediction.

The root-cause analysis needs arbitrary access to the *DataCenter Stream* as well as the graph model (*Graph*s, *Nodes* and *Edges* tables) for particular servers.

This use case will write records on the *Discrepancies* table and the *Training dataset* table.

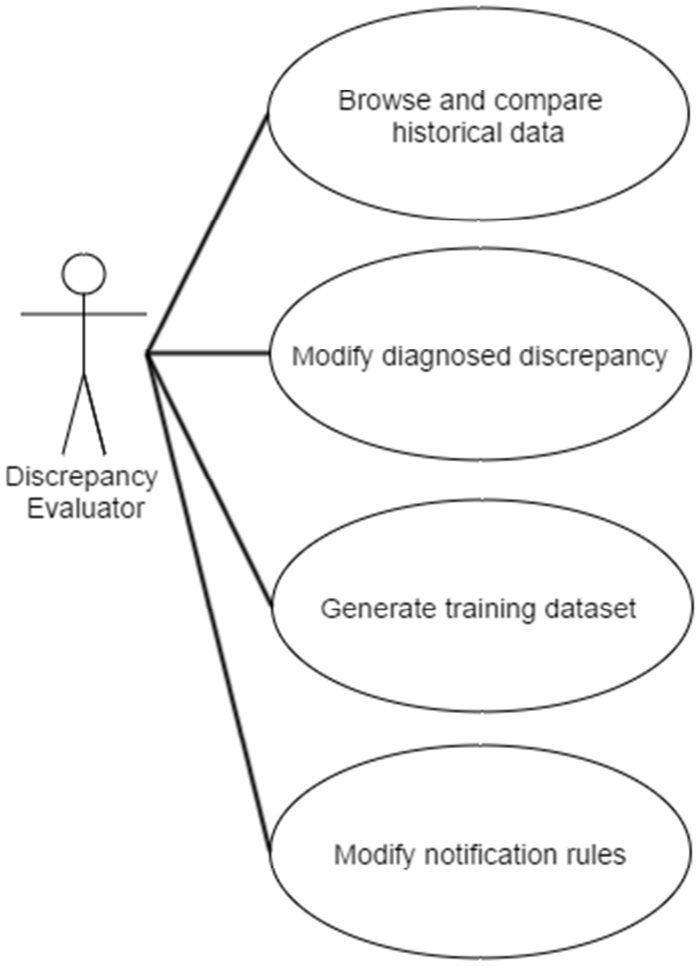
3.3.4 Use Case 4: Human discrepancy analysis (CS1-UC4)

The automatic analysis performed in Use Case 3 is inherently limited since not all necessary knowledge might be available to the algorithms to determine the type and cause of an anomaly. Thus to improve the anomaly evaluation, human intervention is required.

This Use Case deals with the human interaction needed: allowing the Discrepancy Evaluator actor to visualize and compare with historical data using a wide range of devices: from mobile phones to novel HIC interfaces.

The output of such an evaluation might be that a new model must be created (for instance to account for seasonal effects) or that the current one must be retrained. To do so, a Training dataset will be created in the *Training dataset* table, thus linking this Use Case with Use Case 2.

Use case template	**Description**
Use case name	Human anomaly analysis
Priority of Accomplishment	Must Have

Use case description	**Description**
Use case diagram	
Goal	To improve automatic anomaly diagnosis with human expertise. To extract a dataset from the LeanBigData system that can be used to recalibrate obsolete models.
Main Actors	<Discrepancy Evaluator>

Use case scenarios	Description
Main success scenarios	1. The <Discrepancy Evaluator> can visually compare behaviour of the discrepancy under analysis with historical data (either from the *DataCenter Stream*, the *Prediction Stream* or the models, i.e. the graph model or the parameters of the linear model). 2. If required, the <Discrepancy Evaluator> modifies the information of the discrepancy under analysis according to her findings. 3. If a new model needs to be created or the current model needs to be recalibrated, the <Discrepancy Evaluator> is able to store a training dataset in the LeanBigData store. 4. Notification rules are successfully modified.
Preconditions	An anomaly has been detected in the system.
Postconditions	A data set is created that can be used on Use Case 2 - Model creation and Recalibration by the <Model Composer> for recalibration

Other requirements

See Table 2, Table 3 and Table 4 in Section 3.4 for further detailed requirement descriptions.

Queries related to this Use Case

The comparison with historical data requires arbitrary access to the *DataCenter Stream, Prediction Stream* and model data stored, including temporal range queries.

Visualizations related to this Use Case

Show graphically the values for a certain metric or set of metrics in a Data Center representation for a particular timestamp. Allow zooming in, zooming out of elements in that representation. Metrics can be: data gathered, predictions, the models themselves or the difference between predicted and real data.

Show the same information but for a time window

Show a comparison between two or more timestamps.

Show a comparison between two or more time windows.

3.4. Case study requirements

3.4.1 Actors

ID	Actor name	Involvement	Notes
CS1-A1	Data Curator	Preparation and re-facturing of data for import/export from the LeanBigData store.	This is a human actor appears in Use case 1. It is in charge of loading data from CA proprietary stores into the LeanBigData store.

ID	Actor name	Involvement	Notes
CS1-A2	Model Trainer	Model Trainer actor creates the model. It feeds initial structural data about the Data Center under analysis (graph model and constants in the initial prediction models). This actor can also intervene in the model recalibration process using the training dataset generated on Use Case 3 or Use Case 4.	This human or machine actor is part of Use Case 2. It can assist in the model recalibration process.
CS1-A3	Energy Predictor	Automatic actor that selects, from the computed energy models available, one of them (e.g. different models might exist that apply to different periods of time) and uses it to compute a power consumption prediction for a server, using the stream of data coming from the monitored data center.	
CS1-A4	Discrepancy Detector	Automatic actor that compares the prediction stream with the actual power consumption data, searching for significant differences between them.	Triggers the Automatic Discrepancy Evaluator actor once a relevant discrepancy is detected.
CS1-A5	Automatic Discrepancy Evaluator	Automatic actor that classifies anomalies as sporadic or durable changes, and diagnoses the anomalies performing root-cause analysis using the models and the historic data. It can also initiate the generation for the training process of some of the affected models, if needed. If analysis is not conclusive, or needs human review, it may handle over the anomaly evaluation to a human (the <Discrepancy Evaluator> actor).	
CS1-A6	Discrepancy Evaluator	Human actor that supervises the process of anomaly detection. It evaluates the anomalies that could not be automatically fully diagnosed by the <Automatic Discrepancy Evaluator>. It can initiate the training data generation process for model recalibration.	The Anomaly Detector is a human actor using a visualization interface for visual anomaly detection

Table 1. Case Study 1 Actors

3.4.2 Context Requirements

The specific “environment” requirements generated are listed in Table 2

ID	Name	Description	UC	Notes
CS1-CR1	Data Import from Legacy Sources	Process of extraction of data from CA proprietary	CS1-UC1	As bare minimum, the system should

ID	Name	Description	UC	Notes
		products requires a number of data importing capabilities.		be able to import and export data from/to SQL sources and from/to CSV text files.
CS1-CR2	The system should be able to protect privacy of data	Data fields that contain personally identifiable (personal and corporate) information should be obfuscated or encrypted with appropriate access control restrictions.	CS1-UC1	
CS1-CR3	The system should be able to consider different model types.	In order to model the behavior of certain parameters of the system, the system should be able to allow for different types of models.	CS1-UC2	Different types of models may include for instance time series regressions, event-condition-action rules, neural networks, etc.

Table 2. Case Study 1 Context Requirements

3.4.3 Functional Requirements

Describe the functional requirements obtained from the use cases.

ID	Name	Description	UC	Notes
CS1-FR1	Specification of data sources	The system should provide an easy-to-use mechanism for specifying different data sources.	CS1-UC1	Interfaces should be easy to understand, visualization of different data sources could be an optional requirements to help user visualize complex environments with heterogeneous data sources
CS1-FR2	Loading data	The system must allow storing data coming from streams into the LeanBigData Store.	CS1-UC1	Data will be stored without information loss.
CS1-FR3	Definition of data use policies	The system must allow to establish policies to rule the use of data	CS1-UC1	

CS1-FR4	Initial model input	The system must support the input of structural information from the data center	CS1-UC2	
CS1-FR5	Model creation	The system must support the creation of models using multiple monitoring datasets with workload and energy consumption data	CS1-UC2	
CS1-FR6	Model recalibration	The system must support the re-calibration of the models	CS1-UC2	
CS1-FR7	Compute metrics on a new collected data	The system must allow computation of metrics on new collected datasets.	CS1-UC3	The Energy Predictor actor will be generating a stream of predictions, but the throughput of this stream needs not necessarily be the same as the throughput of data center stream
CS1-FR8	Compare new metrics with predictions computed in the last period.	The behavior of the system should be constantly compared with the last predictions to detect changes.	CS1-UC3	In the future, during the project, we may require different options such as studying dynamically changing windows or time periods that may require accessing to previous data in the past.
CS1-FR9	Identification of discrepancies	The system must support the automatic detection of discrepancies by comparing the predictions with the reals measurements of consumed power	CS1-UC3	
CS1-FR10	Discrepancy diagnosis	The system must provide an automatic method to (partially) diagnose discrepancies.	CS1-UC3	Discrepancies that could not be completely diagnosed or that require human intervention will be handed over to <Discrepancy Evaluator> actor

CS1-FR11	Extract dataset for model recalibration	The system must allow the specification of training datasets as sets of time windows.	CS1-UC3 CS1-UC4	Create training datasets
CS1-FR12	Headless visualization API	"Server side" of the API should provide fully prepared data set for visualization so that almost no data processing should happen on the "client side"	CS1-UC4	
CS1-FR13	Browse and compare historic information	The <Discrepancy Detector> will be able to browse the historical data of stream values and models	CS1-UC4	

Table 3: Case Study 1 Functional Requirements

3.4.4 Non-Functional Requirements

Describe the non-functional requirements (security, availability, performance, quality, etc.)

ID	Requirement name	Description	UC	Notes
Performance requirements				
CS1.NF-P1	Data import/export size	The system should be able to import/export tens of terabyte data volumes.	CS1-UC1	
CS1.NF-P2	No data loss in streams	The system should be able to store and process the streams received or generated without dropping any data.	CS1-UC3	
CS1.NF-P3	Latency of real-time query	Latency of database queries should not negatively affect user experience and usability.	CS1-UC4	
CS1.NF-P4	Interactive latency	Initial drawing of visualization should not take more than 2 seconds. Update of visualization should not take more than 250 milliseconds Click delay during interaction with visualization no more than 2 seconds.	CS1-UC4	
Integration requirements				

ID	Requirement name	Description	UC	Notes
CS1.NF-I1	Capacity to deal with heterogeneous data sources	The system should be ready to cope with heterogeneous data sources, extracted from different proprietary sources, and integrate them so that they can be used together in an integrated way during the analysis process.	CS1-UC1	Depending on the type of data collected and the type of operations to be performed on top of these data, they will be stored in different types of storage system, i.e. relation or NoSQL.
Stability requirements				
CS1.NF-S1	Past data should be available	Data should be available in the system for long periods to perform further analysis in the future that are not defined at this time of the project.	CS1-UC1 CS1-UC2 CS1-UC3	The system is not a real-time system per se. We require the data to be available in general, although we can cope with short downtimes.
Maintainability requirements				
CS1.NF-M1	Update of models	We should be able to delete, modify or update the models created at any time	CS1-UC1	
CS1.NF-M2	Updates of data sources	We should be able to update the data sources used in our system.	CS1-UC1	
CS1.NF-M3	Updates of data use policies	We should be able to update data use policies at any time.	CS1-UC1	
CS1.NF-M4	Update notifications	We should be able to update notification rules used in CS1-UC3	CS1-UC4	
CS1.NF-M6	Update visualization layout	We should be able to update the visualization layout at any time	CS1-UC4	
Scalability requirements				
CS1.NF-SC1	System should be able to scale in case we need to monitor many different metrics and models.	We may need to monitor several models related to different data sources. The system should be prepared to scale up in case the number of monitored sources grows significantly.	CS1-UC3	
CS1.NF-SC2	Visualizations should be able to scale for the display of large data sets on	We may need to visualize datasets of multiple sizes and different metrics from heterogeneous data	CS1-UC4	

ID	Requirement name	Description	UC	Notes
	multiple devices	sources. The visualization system must be capable of presenting usable visualization on diverse devices such a mobile devices and large displays.		

Table 4: Case Study 1 Non-Functional Requirements

4. Financial/Banking: Electronic Alignment of Direct Debit transactions Case Study

4.1. Case Study Overview

The recent adoption of the Single Euro Payments Area (SEPA), which follows the European Union (EU) payments integration initiative, has moved more attention to the mechanisms to avoid frauds in banking/financial transactions. Considering an SDD (SEPA Direct Debit) transaction, we have that the SEPA standard has simplified a lot the payment process, while moving the consequences of a fraud from the user to the bank. In particular in an SDD transaction one person/company withdraws funds from another person's bank account. Formally, the person who directly draws the funds ("the payee or creditor") instructs his or her bank to collect (i.e., debit) an amount directly from another's ("the payer or debtor") bank account designated by the payer and pay those funds into a bank account designated by the payee. Typically examples of SDD transactions are services that requiring recurrent payments such as pay per view TV, energy distribution, credit card etc. To set up the process, the creditor has to acquire an SDD mandate from the debtor and advise his/her bank about that mandate. Each time it will be needed, the Creditor sends a direct debit request (with amount specification) to his/her bank that will start the process to request the specified amount on the Debtor's bank account. The debtor only has to provide the signature of the mandate and the debtor could not receive communications about the SDD request. He/she can identify an unauthorized SDD amount only when it receives its bank statement. Of course this exposes the debtor to a number of possible frauds (many in the following). For this reason, with SEPA, in case of error/fraud with an SDD, a debtor can request for a refund until: 8 weeks after the SDD deadline or 13 months for unauthorized SDD (no or revoked mandate). The SDD process is shown in Figure 1.

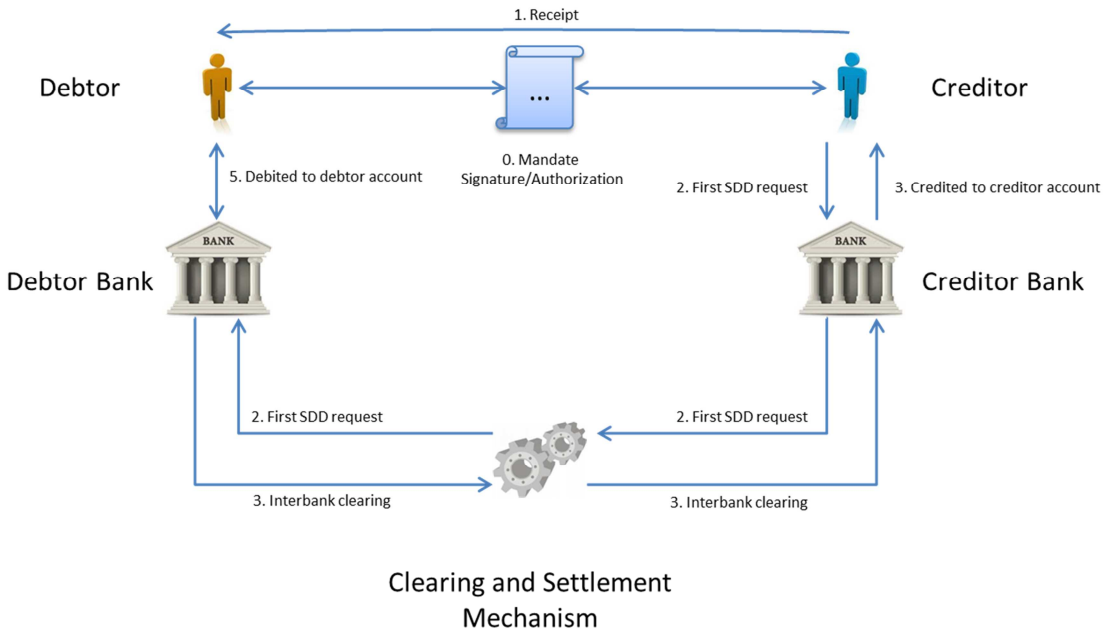


Figure 1. SEPA Direct Debit process

The Financial Banking use case wants monitor the debit request to try to recognize unauthorized debit, thus providing an anti-fraud system helping the financial institution to reduce costs due to frauds. Unauthorized debit are typically due to the Identity Theft as shown in Figure 2.

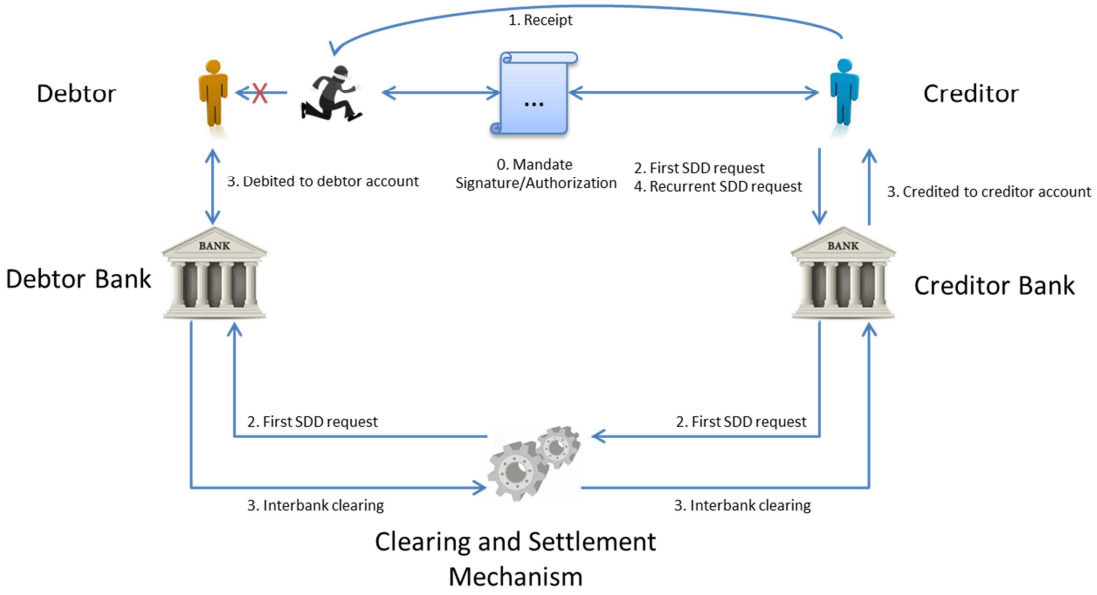


Figure 2. SDD unauthorized

To allow the detection of possible unauthorized SDD, the Financial Banking use case will correlate different information coming from different sources (social, questionnaires, personal info etc) to create a debtor profile that summarizes the habits and interests of the debtor. When an SDD transaction for a specific service is detected the anti-fraud system will measure the "distance" of the SDD service from the debtor profile to detect if the service is in the "interest" of the debtor.

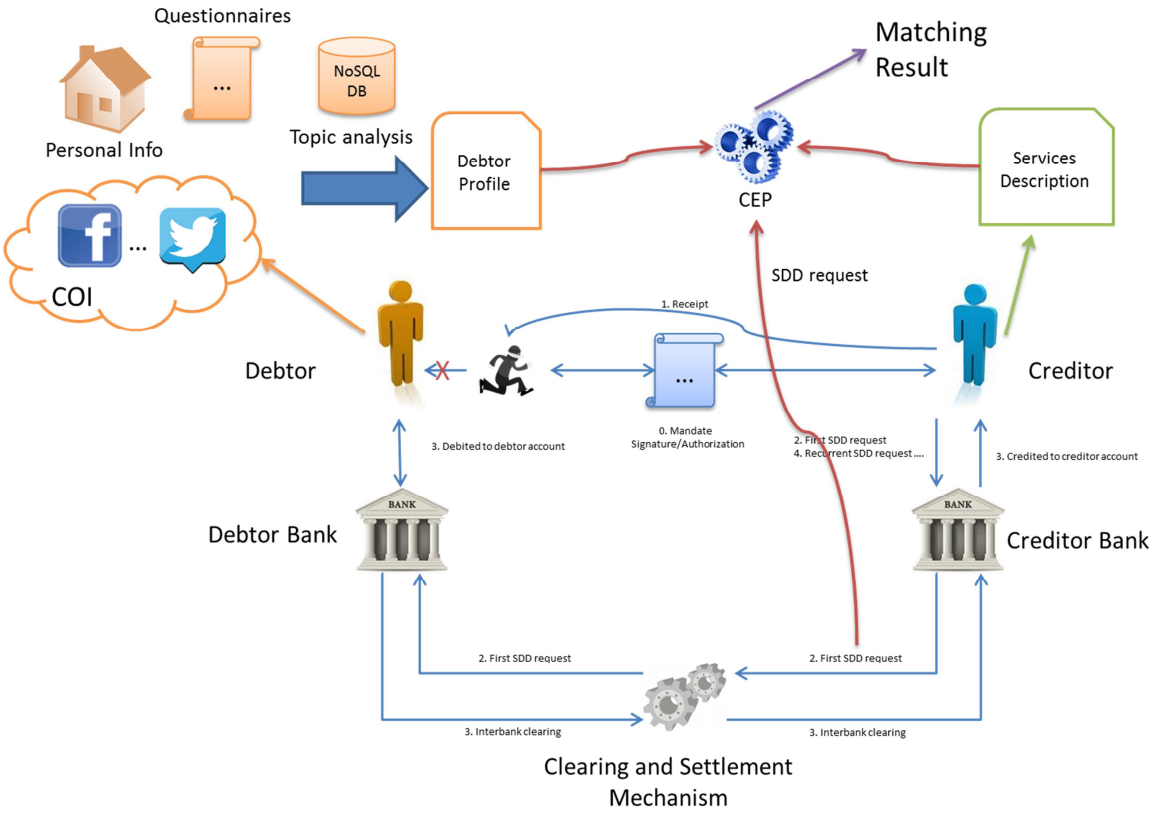


Figure 3. SDD anti-fraud system

Inputs of the anti-fraud system are SDD events, social data, personal info, service description etc. Output will be details on the monitored SDD, details on the unauthorized SDD, level of confidence of an alert etc.

4.2. Use Cases

Describe several use cases related to the case study following a given template IE: use case name, involved actors, short description, preconditions, effects, inputs, and outputs.

4.2.1 Use Case 1: Authorized SDD

Use case name: authorized SDD

Involved actor: Fraud Analyst; Debtor; Creditor.

Precondition: Debtor subscribed to the anti-fraud system with one or more social accounts; Creditor provided a description of its services.

Description: the Debtor subscribes for a service of a Creditor which is of his/her interest. An SDD request for that service is detected from the anti-fraud system. The anti-fraud system analyses the service typology and checks if it is compatible with the user profile. The anti-fraud system evaluates that the SDD request is legit.

Effects: none

Inputs: Social debtor data, SDD data, Service description.

Outputs: none

4.2.2 Use Case 2: Topic of Interest – Identity Theft

Use case name: Topic of Interest – Identity Theft

Involved actor: Fraud Analyst; Debtor; Creditor; Fraudster.

Precondition: Debtor's identity stolen by a fraudster; the fraudster signed a mandate for a service with the stolen identity; Mandate registered on the system; Debtor subscribed to the anti-fraud system with one or more social accounts; Creditor provided description of its services.

Description: a fraudster steals the identity of a creditor and requires a service using such an identity. An SDD request for that service is detected from the anti-fraud system. The anti-fraud system analyses the service typology and checks if it is compatible with the user profile. The anti-fraud system recognizes the SDD as not compatible with the debtor profile (it is beside his/her area of interest) and warns the fraud analyst for further possible investigations.

Effects: Alert on the system

Inputs: Social debtor data, SDD data, Service description.

Outputs: Alert on the system, Alert Report with all info related to it.

4.2.3 Use Case 3: Topic of Interest – Direct Debit Theft

Use case name: Topic of Interest – Direct Debit Theft

Involved actor: Fraud Analyst; Debtor; Creditor; Fraudster.

Precondition: Debtor identity theft performed by a fraudster; the fraudster signed a mandate for a service with the theft identity; Mandate registered on the system; Debtor subscribed to the anti-fraud system with one or more social accounts; Creditor provided description of its services.

Description: a fake company, registered as a biller for SDD, activates SDDs in name of the debtor unaware of the happening for a service not in the interest of the victims. In this case the company will be credited with amount stolen from the bank accounts of the debtor. An SDD request for that service is detected from the anti-fraud system. The anti-fraud system analyses the service typology and checks if it is compatible with the user profile. The anti-fraud system detects that the SDD request is not compatible with the debtor profile and warns the fraud operator for further investigations.

Effects: Alert on the system

Inputs: Social debtor data, SDD data, Service description.

Outputs: Alert on the system, Alert Report with all info related to it.

4.2.4 Use Case 4: Geographic coherence of the service

Use case name: Geographic coherence of the service

Involved actor: Fraud Analyst; Debtor; Creditor; Fraudster.

Precondition: Debtor identity theft by a fraudster; the fraudster signed a mandate for a service with the stolen identity; Mandate registered on the system; Debtor subscribed to the anti-fraud system with one or more social account; Creditor provided description of its services.

Description: a fraudster steals the identity of a creditor and, by using such an identity, pays for a service that can be provided in a specific location. The SDD request for the service is analysed by the anti-fraud system. The anti-fraud system checks the compatibility with user profile. The anti-fraud system detects that an anomalous location for the provisioning of the service (e.g. one location that was never visited by the user) was used and launches a warning to the fraud analyst.

Effects: Warning on the system

Inputs: Social debtor data, SDD data, Service description.

Outputs: Warning on the system, Warning Report with all info related to it.

4.2.5 Use Case 5: Geographic coherence - Identity Theft

Use case name: Geographic coherence - Identity Theft

Involved actor: Fraud Analyst; Debtor; Creditor; Fraudster.

Precondition: Debtor identity theft performed by a fraudster; the fraudster signed a mandate for a service with the stolen identity but uses a different home address; Mandate registered on the system; Debtor subscribed to the anti-fraud system with one or more social account; Creditor provided description of its services.

Description: a fraudster thefts the identity of a creditor and requires a service using that identity with a different home address to allow the receiving of service equipment. An SDD request for that service is detected by the anti-fraud system. The anti-fraud system analyses the SDD request and checks if it is compatible with the user profile. The anti-fraud system detects that an incorrect home location was used and launches a warning to the anti-fraud people.

Effects: Warning on the system

Inputs: Social debtor data, SDD data, Service description.

Outputs: Warning on the system, Warning Report with all info related to it.

4.2.6 Use Case 6: un-classifiable SDD

Use case name: un-classifiable SDD

Involved actor: Fraud Analyst; Debtor; Creditor.

Precondition: Mandate signed and registered on the system; Debtor subscribed to the anti-fraud system with one or more social account; Creditor provided description of its services.

Description: the Debtor subscribes for a service of a Creditor which is of his/her interest. An SDD request for that service is detected by the anti-fraud system. The anti-fraud system analyses the service typology and checks if it is compatible with the user profile. The anti-fraud system detects that the SDD request could be legit, and warns the fraud analyst for further investigations.

Effects: Warning on the system

Inputs: Social debtor data, SDD data, Service description.

Outputs: Warning on the system, Warning Report with all info related to it.

4.3. Case study requirements

4.3.1 Actors

In Table 5 specific actors and their primary user roles are listed. General actors of LeanBigData are also listed here, although later might be summarized in a common section if needed (i.e. LeanBigData admin). The section Notes describes the required data for each specific actor.

ID	Actor name	Involvement	Notes
CS2-A1	Fraud Analyst	Detects frauds and trigger reaction activities	
CS2-A2	Fraudster	Thefts debtor's identity and try to activate an SDD transaction	
CS2-A3	Debtor	In the SEPA Direct Debit (SDD) schema it is the person who has a debit that satisfies by providing funds form her bank account to the bank account of the biller by mean of an SDD transaction.	
CS2-A4	Creditor	In the SEPA Direct Debit (SDD) schema it is the person who has a credit that will be satisfied by collecting funds from the debtor's account by means of an SDD transaction.	

Table 5. Case Study 2 Actors

4.3.2 Context Requirements

The specific “environment” requirements generated by Case study 2 are listed in Table 6.

ID	Name	Description	UC	Notes
CS2-CR1	Data capture maximization	The system will be using free versions of wide known data capture APIs (Twitter search, tweeter streaming, etc.).	ALL	<side notes>
CS2-CR2	Easy data analysis integration	The system will be designed to ease the addition of new data analysis algorithms		
CS2-CR3	Social Data availability	The SDD debtor protected by the antifraud system should have live social media accounts		
CS2-CR4	Social Data privacy	The SDD debtor must grant access to his/her social media data.		

Table 6. Case Study 2 Context Requirements

4.3.3 Functional Requirements

Describe the functional requirements obtained from the use cases.

ID	Name	Description	UC	Notes
CS2-FR1	Alerts	The system will provide different alerting mechanisms to communicate a fraud tentative to the fraud analyst.	All	
CS2-FR2	Warnings	The system will provide different warning mechanisms to communicate a probable fraud tentative to the fraud analyst.		
CS2-FR3	Alerts and Warnings Reports	The system will provide reports on alerts and warning happened in a specific timeline.		
CS2-FR4	Personal data capture	The system will provide the means to capture personal data.		(Questionnaire, etc)

CS2-FR5	Social data configuration	The system will provide a GUI allowing the fraud analyst to create, update and remove social user accounts.		
CS2-FR6	User profile creation	The system using personal and social data will support the creation of user profile.		
CS2-FR7	Service description	The system will support the acquisition of the services description.		
CS2-FR8	Alert configuration	The system will provide a GUI allowing the user to configure Alert aspects of the system behavior.		
CS2-FR9	Historical report	The system will provide a GUI allowing the user to show historical report of the system behavior.		
CS2-FR10	Data sources integration	The system will provide mechanism allowing a simple integration of new data sources.		
CS2-FR11	Detection accuracy	The system will provide a good balance between false positive and false negative.		
CS2-FR12	Timeline preferences	The system will be able to report results of data analysis with respect to different timeline.		
CS2-FR13	Topic analysis	The system will provide mechanisms for topic analysis useful for the creation of the debtor profile and analysis.		

Table 7: Case Study 2 Functional Requirements

4.3.4 Non-Functional Requirements

Describe the non-functional requirements (security, availability, performance, quality, etc.)

ID	Requirement name	Description	UC	Notes

ID	Requirement name	Description	UC	Notes
Performance requirements				
CS2.NF-P1	4 Million transactions/day	The system will be able to monitor at least the number of transaction actually managed by the SyncLab bank system.		
CS2.NF-P2	Near real-time threats identification	In order to guarantee a promptly detection of fraud tentative, the environment have to guarantee a near-real time analysis.		
Integration requirements				
CS2.NF-I1	Unified data format for input data	The system will use an internal unified data format to facilitate the adding of new sources.		
CS2.NF-I2	Unified communication channel	The system will use an unified communication channel to facilitate the gathering of data from sources.		
CS2.NF-I3	Unified integration mechanism	The system will use a modular approach to facilitate the integration of new components/functionalities.		
Stability requirements				
CS2.NF-S1	Availability	The system will provide an high level of availability.		
Maintainability requirements				
CS2.NF-M1	Configuration update	The system will provide a GUI to support its configuration update.		
Scalability requirements				
CS2.NF-SC1	High scalability	The system will provide efficient scalability mechanism.		
Storing				
CS2.NF-ST1	SQL/NoSQL support for	The system will provide both SQL and NoSQL		

ID	Requirement name	Description	UC	Notes
	querying	query support		
CS2.NF-ST2	NoSQL DB	The system will provide a NoSQL Database		
Privacy				
CS2.NF-PY1	Data privacy	The system will provide mechanism to guarantee the data privacy of sensible data.		
Usability				
CS2.NF.US1	Usability	The system will provide a user friendly GUI.		

Table 8: Case Study 2 Non-Functional Requirements

5. Social Network Analytics: Social Network-based Area Surveillance Case Study

5.1. Case Study Overview

The Social Network analytics case study is dealing with the challenging issue of using social media as complementary input for city officers and other actors in the prevention and detection of troubles in potentially problematic city areas. To that extent, the use case gathers data from social networks (mainly Twitter) related to the area (geo-located, mentioning the area or POIs in the area, etc.), monitors the activity and if necessary creates alerts to the city officers in charge of the security of the city in order to mobilise the appropriate resources.

The case study will be focused in a specific city area (i.e. a neighbourhood in a city with lots of bars, restaurants and suffering potential or periodic security issues) as a proof-of-concept of the validity of the approach in a Smart City environment. To that extent, the case study will start collecting data from Twitter in a very short time in order to have an extensive dataset for historical usage in the third year of the project in order to text the results.

The case study will implement a solution for social networks gathering and analysis based on the content of dedicated data channels. A data channel will be “listening” different set of conversations in the social web, meaning that the data channels will implement configurable user queries (based on keywords, hashtags, locations, etc.) to gather data (i.e. tweets) associated to the channel. The users will be able to set up several data channels for different purposes or events, providing that the search limits provided by the APIs of the social networks (i.e. the limits of the public Twitter search and/or streaming APIs) are respected. Therefore the analysis of the data would be able to discriminate the data in order to analyse it in very flexible ways.

It is important to point out that the case study intends to deliver a couple of implementations of the software: one using big data “traditional” technologies and a second one offering the same results using LeanBigData components. Therefore, besides the functional aspects and the clear business oriented approach of the case studies, one of the goals is also to serve as a benchmark on how LeanBigData could hopefully help to develop, ease the development and improve the performance of this type of applications.

Figure 4 shows a big picture of the case study. The components are divided into three tiers according to their function.

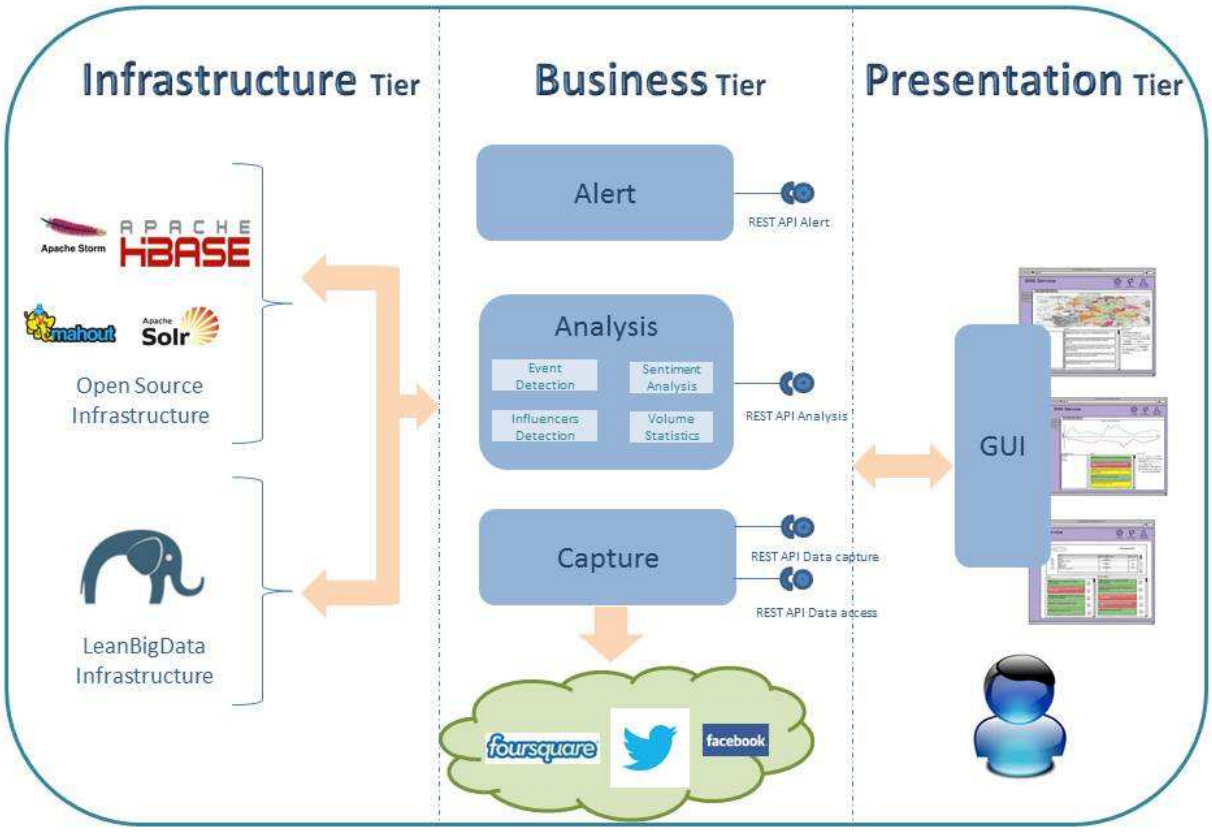


Figure 4 High level architecture Social Network Analytics case study

The *infrastructure tier* put together components related to data processing and storage capabilities. Apart from the computational and data storage capabilities provided by Lean Big data, an open source platform is provided to perform high performance analytics on an extremely large amount of data in the social media case study. This platform will be use also to measure the performance capabilities with the Lean Big data infrastructure.

The *business tier* is focused in all the components that encode the functional requirements of the case study; the data capture components, the data analytic process, and the alert system are the main building blocks in this tier.

- The "Capture" system is in charge to amass and storage large information from several social networks. It is based in the concept of data channels that allow to the user canalizing topic-related data coming from distinct data sources into individual and configurable channels. Therefore all the data that converge into a single data channel share the user subject matter as well as a set of common properties such us expiration date.For a specific social networks as Twitter[1] historical access (until the past 7 days) and streaming (low latency tweets published in is in the process of being transmitted) access will be provided by the Capture system.
- The "Analysis" system is in charge of bring together all the analytical processes implemented in the case of study: "Volume analytics" to measure volume of tweets, "Sentiment analytics" to classify tweets as positive, negative or neutral, "Event detection" to detect not planned events and "Influencers detection" to detect user with high

[1] https://twitter.com/

influence in the data channels. These are the considered analysis but system will be designed as far as possible to allow easy integration of new and forthcoming analysis.

- The "Alert" system in charge of provided needed functionality to warning the user about possible risky situations.

Finally the presentation tier is in charge of provided the graphical user interface to allow the users interact with the system. The GUI will provide optimal and user friendly views to configure and visualized the functionality exposed by the system.

5.2. Use Cases

Following sections provide general use cases related with the social networks surveillance case study. Each use case is completed with basic information as well as the sequence of user actions in the scenario.

5.2.1 Use Case 1: Data Channel management

5.2.1.1 CHARACTERISTIC INFORMATION

Goal in Context: Creation, update and deletion of data channels in the system. Test the data channel created by checking the description as well as the analysis and monitoring of tweets volume in the data channel.
Scope: Data capture.
Level: Primary task
Preconditions: None.
Success End Condition: Optimal data channel created.
Failed End Condition: The data channel is not created or there is some misconfiguration on the data channel.
Primary Actor: Emergency supervisor.
Trigger: The creation/update/deletion of a data channel is performed on demand by the user.
Related requirements: CS3-FR12.1, CS3-FR12.3, CS3-FR13.5, CS3-FR13.6, CS3-FR1, CS3-FR2.1, CS3-FR2.2, CS3-FR2.3, CS3-FR3, CS3-FR4.1, CS3-FR4.2

5.2.1.2 MAIN SUCCESS SCENARIO

- John Doe, checks the data cannel by default (geo-localized + geo-tagged tweets) in the monitoring panel.
- Taking into account last news, John decides create a new data channel to monitoring the upcoming word cup quarter-finals match between Netherland and Argentina:
    - To create a new data channel, John goes to the configuration panel, to the "data channel creation" section and after setting the required parameters (keywords, data sources…) a new data channel containing keywords related with the event, i.e.: "quarter-finals", "match", "Netherland", "Argentina", "Brazil"…
    - To check the correctness of the created data channel, John goes to the monitoring panel, concretely to the new data channel created, and after checking the introduced keywords in the "channel description section" and the tweets in the "data channel samples" section, decides to update the data channel due to it is not recovering the adequate tweets.

- To update a data channel, John goes to the configuration panel, to the “data channel update” section and changes the keywords in the queries.

- Once the match is over the data channel is not relevant anymore so John decides to remove the data channel:
  - To delete an existing data channel, John goes to the configuration panel, to the “data channel delete” section and removes the data channel, leaving from the monitoring panel.

5.2.2 Use Case 2: Real-time monitoring

5.2.2.1 CHARACTERISTIC INFORMATION

Goal in Context: Monitoring the default / streaming data channel. Test the near real-time analytics and tracking: volume, sentiment and bursty words.
Scope: Analysis and Monitoring.
Level: Primary task
Preconditions: None.
Success End Condition: Adequate measure and analysis of tweets volume, sentiment and brusty words in the default data channel. Adequate visualization mechanisms in the default data channel.
Failed End Condition: Non-detection or bad-detection of tweets volume, sentiment and bursty words in the default data channel.
Primary Actor: Emergency supervisor.
Trigger: The monitoring of a data channel is performed on demand by the user.
Related requirements: CS3-FR13.1, CS3-FR13.2, CS3-FR13.3, CS3-FR13.4, CS3-FR10, CS3-FR9.1, CS3-FR9.2, CS3-FR7.1, CS3-FR7.3, CS3-FR7.4.

5.2.2.2 MAIN SUCCESS SCENARIO

- John is monitoring the default data channel and detects a high and non-expected number of tweets in the data channel.
- Taking into account the tweet volume, John decides to alert the health public service to let them know about a possible serious amount of people in the area.
- After checking the most relevant terms visualized in the “bursty word cloud” section, he detects a non-expected concert taking place in the area.
- Before setting as critical the situation, John decides to check the “stress” level in the area by checking the “sentiment” section of the data channel. Once he realized the level is negative, John decides to alert the police to let them about a possible risky situation in the area.
- Finally the policeman asks for a concrete location, and after checking the “heat map tweet volume” section, John provides a more precise and limited troubled area.

5.2.3 Use Case 3: Event detection

5.2.3.1 CHARACTERISTIC INFORMATION

Goal in Context: Non-expected event detection. Notification of alerts. Events visualization.
Scope: Events and alarms.
Level: Primary task
Preconditions: None.
Success End Condition: Adequate detection of non-expected events in a data channel. Adequate notification of alerts. Adequate visualization of events.
Failed End Condition: Non-detection or bad-detection of new events in the default data channel. Bad or fail in alerting about a new event. Bad or fail in visualize events in the calendar section.
Primary Actor: Emergency supervisor.
Trigger: The event detection is performed systematically on each data channel.
Related requirements: CS3-FR15, CS3-FR11.4, CS3-FR7.2

5.2.3.2 MAIN SUCCESS SCENARIO

- John is defining a new data channel for his weekly surveillance. For the definition of the data channel he decides to take into account the expected and planned events in the area.
- To visualize the planned and already detected events, John goes to the calendar panel, and after checking the events, he creates the data channel with the appropriated keywords.
- When a new detected event is detected, John receives a warning in the “notification” area alerting about a new event detected. John goes to the calendar section again, check the information of the new event and after considering it relevant, he decide to update the data channel by adding new keywords related with the new event.

5.2.4 Use Case 4: Alarm definition and notification

5.2.4.1 CHARACTERISTIC INFORMATION

Goal in Context: Configuration and notification of alerts.
Scope: Alarm configuration.
Level: Primary task
Preconditions: None.
Success End Condition: Adequate configuration of alert. Adequate notification of alert.
Failed End Condition: Bad-detection of new alert for a data channel. Bad or fail in notify about alert.
Primary Actor: Emergency supervisor.
Trigger: The alert configuration is performed on the demand by the user. The notifications are performed systematically by the system when an alert is lunched.
Related requirements: CS3-FR12.2, CS3-FR14, CS3-FR11.1, CS3-FR11.2, CS3-FR11.3

5.2.4.2 MAIN SUCCESS SCENARIO

- Several months ago, John decides to set up a set of alarms to be notified automatically about a critical situation with no need to be checking the pertinent monitoring panel. After a meeting with the security experts, a set of measures and thresholds have been established so that the John will be alerted when the alarm system detects some of them.
- To configure the alerts, John goes to the configuration panel, to the “alert configuration” section, and set following alarms for the default data channel:
  - He will be notified when the number of tweets overcomes a certain threshold.
  - He will be notified when the number of negative tweets overcomes a certain percentage of total amounts of tweets.
  - He will be notified when certain words appears as bursty words
- In order to notify the user, the system will generate a warning in the notification area when some of the stetted threshold will be overcome.

5.2.5 Use Case 5: Influencers detection and visualization

5.2.5.1 CHARACTERISTIC INFORMATION

Goal in Context: Detection of influence users. Influencers statistics visualization.

Scope: Users and influencers.

Level: Primary task.

Preconditions: None.

Success End Condition: Adequate detection of influence users. Adequate visualization of statistical information about the influence users.

Failed End Condition: Non-detection or Bad-detection of influencers in a data channel. Bad visualization information related with the influencers.

Primary Actor: Emergency supervisor.

Trigger: The influencer’s visualization is performed on the demand by the user. The detection of influencers is performed systematically by the system in a data channel.

5.2.5.2 MAIN SUCCESS SCENARIO

- A demonstration will take place in the surveillance area. The demonstration is planned by a violent organization. John, who is aware of his leaders, decides to check the influence of the users in the current data channels.
- To check the influence users, John goes to the “influencers” panel. As a glance he detects that one of the leader appears in the top N influencers list and decides to check the influence information about the user.
- To check the information about an influence user, in the “influencers” panel, John clicks on the desired user and the system retrieves last tweets of the user as well as the most retweeted tweets of the user. Based on the content and the sentiment of the visualized tweets, John decides take (or not) actions.

5.2.6 Use Case 6: Historical analysis of surveillance data

5.2.6.1 CHARACTERISTIC INFORMATION

Goal in Context: Provide a flexible mechanism for data analysts to generate and visualize information aggregations based on stored data and analysis results.

Scope: Information aggregation and visualization.

Level: Primary tasks.

Preconditions: None

Success End Conditions: The data analyst is able to generate and visualize, in an easy way, and arbitrary data aggregation and visualize it in meaningful and configurable way.

Failed End Conditions: The data is not available in a reasonable time. The visualization fails to present the data in an appropriate way.

Primary actor: Data analyst.

Trigger: The data aggregation is built / retrieved by the system. The data aggregation is shown in a proper widget / graphical representation.

5.2.6.2 MAIN SUCCESS SCENARIO

- A data analysis is hired by the city council in order to identify some data trends that may be used as basis to analyse and improve the city security policies.
- The data analyst accesses the system query console in order to start the interaction with the system.
- The data analyst introduces a query in the system asking for an aggregation containing the dimensions: term, time, and number of occurrences. The system retrieves / builds the data aggregation.
- The data analyst is able to select between several kinds of visualization. Each visualization should be configurable taking into account different dimensions.
- The data analyst is able to perform this procedure for every surveillance analysis generated by the system.

5.2.6.3 MAIN ANALYTICAL QUERIES

5.2.6.3.1. CS3-AQ1- Number of tweets per time unit and data-channel

This query obtains the number of tweets per time unit and per data-channel. Thus, the OLAP cube needed has two dimensions: time and datachannel; the measure is summation of tweets. The time unit dimension time has four levels: day, month, period and year; and the dimension datachannel has two levels: datasource and datachannel.

Following table shows an example of results:

	Date									
	(+) 2013	(-) 2014								
			(-) Q1				(-) Q2			
Datachannel				(+) Jan	(+) Feb	(+) Mar		(+) Apr	(+) May	(+) Jun
(-)Total data	1628	1541	675	231	214	230	671	216	233	222
(+) datachan	579	820	336	114	104	118	334	107	116	111
(-) datachan	1054	721	339	117	110	112	337	109	117	111
datasou	543	930	127	48	45	34	456	111	234	111
datasou	554	1369	692	69	567	56	441	190	128	123

Table 9. Number of tweets per time unit and data-channel results

Each cell on the table contains the amount of tweets for a concrete period, for a concrete data channel with the possibility to make a drill down exploration through year, period, month and day, on the “time” dimension, and through datachannel, data source in the “datachannel” dimension.

For this aggregation, tables containing linkages between tweet, date and data channels information will be needed. A dimension mapping should be performed by applying text-search techniques to the database containing raw tweets in order to extract and normalize the date information contained in the json field "created_at” in the "column-family:column" “tweet:json".

So, in OLAP terms, the facts table will be the table “Tweets” and a table “Time” and "Datachanel" will be the two dimension needed for the query.

5.2.6.3.2. CS3-AQ2 Aggregated sentiment (average) per time unit and data-channel

Give the aggregated sentiment per month and per data-channel. The measure or aggregate function is average sentiment score. Two dimension: time and datachannel. The dimension time has four levels: day, month, period and year; and the dimension datachannel has two levels: datasource and datachannel.

Following table shows an example of results

	Date						
	(+) 2013	(-) 2014					
			(-) Q1				(-) Q2
	nel			(+) Jan	(+) Feb	(+) Mar	
(-)Total da	0.7	"0.7"	0.5	0.3	"0.3"	0.9	0.9
(+) datach	0	0.2	"-0.1"	0.8	"-0.9"	0	0.3
(-) datach	0.3	0.9	0.9	0.7	"-0.7"	0.9	0
datas	0.4	"-0.1"	"-0.4"	"-0.7"	"-0.2"	0	0.3
datas	0.8	"-0.1"	"-0.4"	"-0.5"	0.7	"-0.6"	0.3

Table 10. Aggregated sentiment (average) per time unit and data-channel results

The aggregation will be based on existing information provided by the use case. Therefore information between the tweet and creation date, data-channel and sentiment score will be pre-computed (when apply) and make it available for the aggregation.

A data transformation will be take place in order to normalize the date information contained in the table of raw tweets. Therefore a “Time” dimension mapping should be performed by extracting and applying text processing techniques over the json field "created_at” contained in the column-family:column “tweet:json" of the raw tweet table.

From the OLAP point of view the facts table will be the table “Tweets” (with the score associated) and the tables “Time” and "Datachannel" will be the two dimensions needed for the query.

5.2.6.3.3. CS3-AQ3: Number of tweets per time unit, data-channel and sentiment type

This query obtains the number of tweets per month, per data-channel and per sentiment. Thus, the OLAP cube needed has three dimensions: time, datachannel and sentiment; the measure is summation of tweets. Time dimension has four levels: day, month, period and year; datachannel dimension has two levels: datasource and datachannel.

Following table shows an example of results*:

	Date									
	(+) 2013	(-) 2014								
			(-) Q1				(-) Q2			
Sentiment				(+) Jan	(+) Feb	(+) Mar		(+) Apr	(+) May	(+) Jun
(+) Positive	1628	2361	1011	345	318	348	1005	323	349	333
(+) Neutral	1628	1541	675	231	214	230	671	216	233	222
(-) Negative	579	820	336	114	104	118	334	107	116	111
(-) datachani	1054	721	339	117	110	112	337	109	117	111
datasou	543	930	127	48	45	34	456	111	234	111
datasou	554	1369	692	69	567	56	441	190	128	123

Table 11. Number of tweets per time unit, data-channel and sentiment type results

In this table, each cell contains the amount of tweets for a concrete period, for a concrete data channel and for a concrete value of possible sentiment (positive, negative, and neutral), with the possibility to make a drill down exploration through year, period, month and day on the “time” dimension, and through data channel, data source in the “data channel” dimension.

The aggregation will be based on information generated by the use case such us sentiment type. Therefore the linkage between the tweet and creation date, data-channel and sentiment score will be pre-computed (when apply) and make it available for the aggregation. A data transformation will take place in order to normalize the date information contained in the table of raw tweets. Therefore a “Time” dimension mapping should be performed by extracting and applying text processing techniques over the json field "created_at” contained in the column-family:column “tweet:json" of the raw tweet table. On the other hand and based on existing sentiment score information for a tweet, a “Type Sentiment” dimension mapping should be calculated in order to categorize the score into the three types: “positive”, “neutral” or “negative”.

From the OLAP point of view the facts table will be the table “Tweets” and tables “SentimentType”, “Time” and "Datachanel" will be the three dimensions needed for the query.

*Other option should be show an interactive pie chart by datachannel that change along the time.

5.2.6.3.4. CS3-AQ4: Aggregated sentiment (average) per region and time unit

This query obtains the aggregated sentiment per region and per time. Thus, the OLAP cube needed has two dimensions: location and time unit; the measure or aggregate function is average sentiment score. The time unit dimension has four levels: day, month, period and year; the location dimension has two levels: country and city.

Following table shows an example of results*:

	Date							
	(+) 2013	(-) 2014						
			(-) Q1				(-) Q2	
Data Channel				(+) Jan	(+) Feb	(+) Mar		(+) Apr
(-)Total	0.7	"0.7"	0.5	0.3	"0.3"	0.9	0.9	0
(+) Germa	0	0.2	"-0.1"	0.8	"-0.9"	0	0.3	0
(-) Spain	0.3	0.9	0.9	0.7	"-0.7"	0.9	0	0.9
(+) CC	0.4	"-0.1"	"-0.4"	"-0.7"	"-0.2"	0	0.3	0.4
(+) As	0.8	"-0.1"	"-0.4"	"-0.5"	0.7	"-0.6"	0.3	"-0.3"

Table 12. Aggregated sentiment (average) per region and time unit results

The aggregation will be based information generated by the use case such us sentiment type. Therefore linkage between the tweet and creation date, location of the tweet and sentiment score will be pre-computed (when apply) and make it available for the aggregation.

A data transformation will take place in order to normalize the date information contained in the table of raw tweets. Therefore a “Time” dimension mapping should be performed by extracting and applying text processing techniques over the json field "created_at” contained in the column-family:column “tweet:json" of the raw tweet table. Concerning to the location, a data transformation will be apply in order to normalize the location information, if exists, stored in the column “tweet:json" of the raw tweet table.

From the OLAP point of view the facts table will be “Tweets” table (with the score associated) and the tables “Time” and "Location" will be the two dimensions needed for the query.

*Other option should be an interactive map changing the colour along the time[1].

5.2.6.3.5. CS3-AQ5: Frequency (occurrences) of topics per time unit and data-channel.

This query obtains the occurrences of topics per data-channel and time unit. Thus, the OLAP cube needed has three dimensions: term, time unit and data channel; the measure or aggregate function is the number of occurrences of a term. The time unit dimension has four levels: day, month, period and year; the datachannel dimension has two levels: datachannel and data source.

Following table shows an example of results:

		Date						
		Date						
		(+) 2013	(-) 2014					
				(-) Q1				(-) Q2
Data channel					(+) Jan	(+) Feb	(+) Mar	
(-)Total data channels		14695851	789	336	114	104	118	334
(+) datachannel1	Nelson Mandel	3998	721	157	48	157	34	234
	Referendum	2989	789	157	48	157	34	789
	Toni nadal	1289	1369	692	69	657	56	55
	Real Madrid	678	789	245	64	245	77	789
	Ucrania	467	33	34	34	233	456	345
	(+) Other	6789887	2342342	2342334	234234	234234	456456	345345
(-) datachannel2		7896543	234234	234234234	234234	23423422	456456	345345

Table 13. Frequency (occurrences) of topics per time unit and data-channel results

In this table, each cell contains the number of occurrences for a term for a concrete period, for a concrete data channel, with the possibility to make a drill down exploration through year, period,

[1] http://www.yankeefoliage.com/peak-foliage-forecast-map/

month and day on the “time” dimension, and through data channel, data source in the “data channel” dimension.

The aggregation will be based on information generated by the use case, which processes the tweets in order to analyse their content. Therefore the linkage between the terms, the tweet and their creation date and data-channel should be computed (when apply) and make it available for the aggregation in advance. A data transformation will take place in order to normalize the date information contained in the table of raw tweets. Therefore a “Time” dimension mapping should be performed by extracting and applying text processing techniques over the json field "created_at” contained in the column-family:column “tweet:json" of the raw tweet table.

From the OLAP point of view the facts table will be the table “Tweets” and the tables “Time” "Datachannel" and “Terms” will be the three dimensions needed for the query.

5.2.6.3.6. CS3-AQ6: Frequency evolution of a concrete term per time unit and data-channel.

This query obtains the occurrences of a concrete term per data-channel and time. The measure or aggregate function is the summation of occurrences. The dimension time has four levels: day, month, period and year; and the dimension datachannel has two levels: datasource and datachannel.

	Date							
	Date							
	(+) 2013	(-) 2014						
			(-) Q1				(-) Q2	
Data channel (botin)				(+) Jan	(+) Feb	(+) Mar		(+) Apr
(-)Total data ch	3998	789	336	114	104	118	334	107
(+) datachanne	98	721	157	48	157	34	234	12
(-) datachanne	3998	789	157	48	157	34	789	122
(+) datasou	12	1369	692	69	657	56	55	190
(-) datasou	3998	789	245	64	245	77	789	55

Table 14. Frequency evolution of a concrete term per time unit and data-channel results

The aggregation will be based on information generated by the use case. Therefore information about the terms and the time and datachannel associated will be pre-computed (when apply) and make it available for the aggregation. The data transformation will depend on the schema of the “Terms” table pre-computed.

From the OLAP point of view the facts table will be the table “Terms” (with the ocurrences) and the tables “Time” and "Datachannel" will be the two dimensions needed for the query.

5.2.6.3.7. CS3-AQ7: Burstiest term per time unit and data-channel.

This query obtains the burstiest term per data-channel and time unit. The measure or aggregate function is the maximum difference of the brusty word in that period. Thus, the OLAP cube needed has two dimensions: time unit and data channel; the measure or aggregate function is the burstiest term. The dimension time has four levels: day, month, period and year; the dimension datachannel has two levels: datasource and datachannel.

Following table shows an example of results*:

	Date							
	Date							
	(+) 2013	(-) 2014						
			(-) Q1				(-) Q2	
Data channel				(+) Jan	(+) Feb	(+) Mar		(+) Apr
(-)Total data channe	Mandela	Botin	Spring	2015	loves	CIMC	Botin	iphone
(+) datachannel1	Scotland	Botin	Spring	2015	lovers	spring	Botin	iphone
(-) datachannel2	Mandela	Botin	Wall Stree	Christmas	wall street	CIMC	Botin	iphone
(+) datasource1	Times	Santander	Merkel	Christmas	wall street	iphone	iphone	CNA
(-) datasource2	Mandela	Passed	Draghi	2015	merkel	CIMC	Botin	apple

Table 15. Burstiest term per time unit and data-channel results

The aggregation will be based on information generated by the use case, which processes the tweets in order to analyse their content. Therefore information about the busrtiest word and the time and datachannel asociated will be pre-computed (when apply) and make it available for the aggregation.

The data transformation will depend on the schema of the pre-computed burstiest word.

From the OLAP point of view the facts table will be the table “Burstiest words” (with the increase with respect the previous period associated) and the tables “Time” and "Datachannel" will be the two dimensions needed for the query.

5.3. Case study requirements

5.3.1 Actors

In Table 16 specific actors and their primary user roles are listed. General actors of LeanBigData are also listed here, although later might be summarized in a common section if needed (i.e. LeanBigData admin). The section Notes describes the required data for each specific actor.

ID	Actor name	Involvement	Notes
CS3-A1	Emergency supervisor	Emergency supervisors interact with the system in order to identify, prevent, follow and react to potential situations that needs public authorities intervention	
CS3-A2	System administrator	Systems administrators interact with the system in order to ease other actors work by configuring the system to adapt to changing environment conditions	
CS3-A3	Data analyst	Data Analysts interacts with the system in order to identify data trends that may improve security policies by identifying new system functionalities, developing new action protocols, etc.	

Table 16. Case Study 3 Actors

5.3.2 Context Requirements

The specific "environment" requirements generated by Case Study 3 are listed in Table 17.

ID	Name	Description	UC	Notes
CS3-CR1	Data consistency	In order to maintain data consistency, data channel modifications should be kept to its minimum		
CS3-CR2	Data capture maximization	The system will be using free versions of wide known data capture APIs (Twitter search, tweeter streaming, etc.). The system will provide the means to maximize data capture taking into account the limitations of this APIs		
CS3-CR3	Easy data analysis integration	The system will be designed to ease the addition of new data analysis algorithms		

Table 17. Case Study 3 Context Requirements

5.3.3 Functional Requirements

Describe the functional requirements obtained from the use cases.

5.3.3.1 Capturing of non-structure content from social networks

Following requirements describe the functionality of the Data Capture module.

ID	Name	Description	UC	Notes
CS3-FR1	Capturing massive non-structured data	The system will provide the means to capture massive non-structured data from several social networks	UC_N1	Initially, consider mainly Twitter, but desirable to be extended to others

CS3-FR2	Data channel	The data acquisition will be based on a data channel approach, allowing the user to gather data from different social sources grouped in a topic-based channels		Respecting the limitation of the APIs of the social networks
CS3-FR2.1	Data channel creation	The system will support the creation of data channels by introducing a set of search elements as keywords, users, buzzwords, etc. indicating the topic of interest as well as a set of data sources indicating the desired social networks	UC_N1	
CS3-FR2.2	Data channel update	The system will support the update of data channels by updating the data sources and search elements associate to the data channel	UC_N1	
CS3-FR2.3	Deletion of data channels	The system will support the deletion of data channels	UC_N1	
CS3-FR3	Twitter specific capture	The data capture from Twitter will be done attending to data temporal nature: Historical data, Real-time data	UC_N1	
CS3-FR4	Geo-localized default data channel	The system will provide the means to create a default data channel to capture data related with the surveillance geographical area		“Default” means the main channel from a functional perspective, but it is in fact just a data channel
CS3-FR4.1	Geo-localized tweets	Tweets located (location activated) in the defined area, tweets mentioning location on the defined area, calendar events taking place in the area, tweets from user registered in the area	UC_N1	

CS3-FR4.2	Geo-tagged tweet	Tweets containing keywords related with the geographical area like main streets, adjacent streets, neighbourhood name or locations in the area such as bars, shops, business, etc.	UC_N1	
CS3-FR4.3	Event-tagged tweet	Calendar events taking place in the area		Optional
CS3-FR4.4	User Profile location	Tweets from user registered in the area		Optional
CS3-FR5	Restrictions on the default data channel	The system will be able to create new data channels by adding restrictions to the default data channel.		**DEPRECATED**
CS3-FR6	User profile capture	The system will be able to capture the user profile of each tweet's author including followers and following users lists	UC_N5	As long as this is available via the SN API

Table 18: Case Study 3 Functional Requirements (I)

5.3.3.2 Social Media Analytics

Following requirements describe the functionality of the Data Analysis module.

ID	Name	Description	UC	Notes
CS3-FR7	Near real-time analytics	The system will be able to perform social analytics by means of analysing the data stream provided by the data capture module		By text
CS3-FR7.1	Bursty/Buzz word detection	The system will be able to identify and measure the most emergent topics currently discussed by the social network community	UC_N2	

CS3-FR7.2	Event Detection	The system will be able to discover ongoing or future not planed events. The identification of an event will be mainly defined by the location and time detection for bursty words	UC_N3	Optional
CS3-FR7.3	Sentiment Analysis	The system will be able to extract the general sentiment expressed in a tweet.	UC_N2	The method will be used a 3-class method (positive, negative or neutral) for the classification of tweets
CS3-FR7.4	Tweet volume analysis	The system will be able to extract the volume of tweets gathered by the system in a given time.	UC_N1 UC_N2	
CS3-FR8	Historical analytics	The system will be able to perform social analytics based on historical data captured by the system.		
CS3-FR8.1	Influencers detection	The system will be able to detect “influencer” users. The detection might be based on: users with a high number of followers, users with a high number of mentions, users with a high number of retweets, etc	UC_N5	Optional
CS3-FR8.2	Historical sentiment Analysis	Extract positive, negative or neutral opinions about future/planed events.		DEPRECATED

Table 19: Case Study 3 Functional Requirements (II)

5.3.3.3 Social media monitoring and alerts

Following requirements describe the functionality of the Monitoring and Alert module

ID	Name	Description	UC	Notes

CS3-FR9	Tweet volume tracking	The system will be able to provide tracking of tweet volume for the created data channels.		
CS3-FR9.1	Time-based tweet volume	The system will be able to capture the amount of tweets for a time interval in a data channel	UC_N1 UC_N2	The aim is to create a timeline.
CS3-FR9.2	Time-based location-based tweet volume	The system will be able to capture the amount of tweets localized in a certain area for a time interval in a data channel	UC_N2	The aim is to create a heat map.
CS3-FR10	Tweet sentiment tracking	The system will be able to measure the sentiments (positive, negative, neutral) in a particular data channel.	UC_N2	
CS3-FR11	Critical situation alert	The system will provided means to alert about critical situation		
CS3-FR11.1	Overcrowded area alert	The system will be designed to alert when the volume of tweets in a data channel exceeds a threshold defined by the user	UC_N4	
CS3-FR11.2	Overstressed area alert	The system will be designed to alert when the positive / negative sentiments in tweets in a data channel exceeds of a threshold defined by the user	UC_N4	Review
CS3-FR11.3	Bursty / Buzz word alert	The system will be able to alert the user in case a word from a predefined list becomes a bursty word.	UC_N4	
CS3-FR11.4	New event alert	The system will be designed to alert about new detected event	UC_N3 UC_N4?	Optional

Table 20: Case Study 3 Functional Requirements (III)

5.3.3.4 Visualization Requirements

Describe the functional requirements related to the graphical user interface.

ID	Name	Description	UC	Notes
CS3-FR12	System configuration	The system will provide a GUI allowing the user to configure key aspects of the system behavior.		See section 8.1
CS3-FR12.1	Data channel management	The system will provide a GUI allowing the user to create, update and remove data channels.	UC_N1	See the mockup in Figure 5 and Figure 6
CS3-FR12.2	Alert configuration	The system will provide a GUI allowing the user to create, delete and update alerts or system notifications	UC_N4	An alert will be defined by expressing a parameter to measure, a data channel, a threshold value. See the mockup in Figure 7
CS3-FR12.3	Timeline preferences	On timeline visualizations, the user will be able to view different statistical functions (mode, mean, variance) depending on their needs	UC_N1	Optional
CS3-FR13	Data channel monitoring and analytics visualization	The system will be able to offer an integrated dashboard showing different visualizations related to a data channel		See section 8.2
CS3-FR13.1	Timeline visualization of tweet volume	Tweet volume is shown on a timeline (line chart). On mouse hover, exact amount of tweets will be shown	UC_N1 UC_N2	See mockups in Figure 8
CS3-FR13.2	Heat map visualization of tweet volume	Tweet volume is shown on a heat map. On mouse hover, exact amount of tweets will be shown	UC_N2	See mockups in Figure 9
CS3-FR13.3	Timeline visualization of tweet volume per sentiment	Tweet volume per sentiment is shown in a timeline representation. On mouse hover, exact amount of tweets classified in a given sentiment is shown	UC_N2	See mockups in Figure 10

ID	Name	Description	UC	Notes
CS3-FR13.4	Tag cloud visualization of bursty words	For a given data channel, the GUI will show a tag cloud including recent bursty words	UC_N2	See mockups in Figure 11
CS3-FR13.5	Data channel sample	For a given data channel, a sample of the last N tweets will be shown to the user	UC_N1	See mockups in Figure 11
CS3-FR13.6	Data channel description / most frequent terms	For a given data channel, a list of most frequent terms will be shown to the user	UC_N1	See mockups in Figure 11 section
CS3-FR14	Alerts and notifications	The system will provide a mechanism to inform the users about notifications or alerts produced by the system.	UC_N4	At first stage, notifications will be shown only in the regular surveillance user interface
CS3-FR15	Calendar visualization	The system will provide a calendar visualization displaying events identified by the system	UC_N3	Still under discussion Optional
CS3-FR16	Influencers visualization	The system will provide a list of most influential users taking into account followers, mentions and retweets	UC_N5	See mockups in Figure 12 Optional

Table 21: Case Study 3 Functional Requirements (IV)

5.3.4 Non-Functional Requirements

Describe the non-functional requirements (security, availability, performance, quality, etc.)

ID	Requirement name	Description	UC	Notes
Performance requirements				
CS3.NF-P1	Storage capability	The system will provide scalable and efficient storage capabilities able to deal with the huge amount and variety of data captured from social networks.	All	
CS3.NF-P2	Response Time	The system will provide the fast search capabilities.	All	
CS3.NF-P3	Near-real time processing	The system will be able to process streaming data coming from different	All	

ID	Requirement name	Description	UC	Notes
		social networks.		
Integration requirements				
CS3.NF-I1	LeanBigData integration	The use case should provide loosely coupled components to ensure an easy integration with other big data storage/computation solutions.	All	
CS3.NF-I2	Data channels	The system will use unified communication channels to facilitate the gathering and integration of data from different social networks.	All	
CS3.NF-I3	Unified integration mechanism	The system will use a modular approach to facilitate the integration of new components/functionalities.	All	
CS3.NF-I4	Software API	The system will provide the required API's to allow easy integration with external components.	All	
Stability requirements				
CS3.NF-S1	Availability	The system will provide an high level of availability.	All	
Maintainability requirements				
CS3.NF-M1	Updates of data sources	The system will be able to aggregate and combine different social networks used in our system.	All	
CS3.NF-M2	Configuration update	The system will provide mechanisms to configuration update: update of thresholds, update the data channels, update of list of keywords…	All	
Scalability requirements				
CS3.NF-SC1	High scalability	The system must be ready and able to handle the increased usage.	All	
Usability requirements				
CS3.NF.U1	Usability	The system will provide a user friendly GUI.		

Table 22: Case Study 3 Non-Functional Requirements

6. Targeted Advertisement Case Study

6.1. Case Study Overview

SAPO, the internet technology department at Portugal Telecom, responsible for the most visited online portal in Portugal with associated services competing directly with Google and Yahoo services in the country, runs a vast big data and analytics infrastructure which tops up the data coming from the portal with data from the quadruple-play telecommunications business where PT is market leader in all sectors: mobile, internet, landline and TV.

Figure 1. SAPO Homepage

As part of PT's internet business, SAPO sells multiplatform ads online covering the whole spectrum of web, mobile and TV. Similarly to other industry standards, such as Google AdSense and AdWords, SAPO allows advertisers to define their own campaigns, set their campaign goals and budget, their choice of paid words, as well as many other constraints including geographic and demographic of the targeted customers.

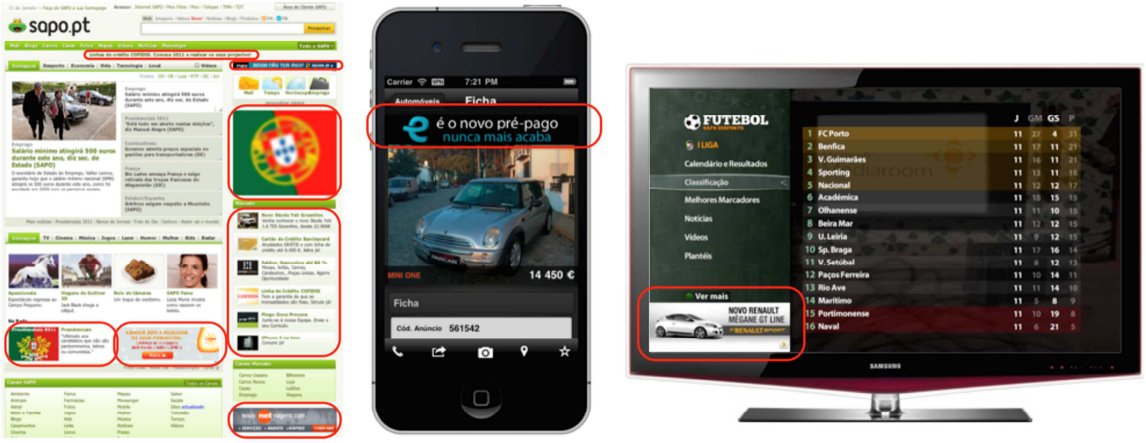


Figure 2. Ads being served on three platforms: web, mobile, TV.

Recent trends point to a maximization of convergence synergies between all the distribution channels by allowing profiling to happen across the whole spectrum of data so that ads are served in a much more targeted fashion. An example of an intricate target ad strategy, would be to use EPG (Electronic Programming Guide) information and the customer's own use of the set top box to infer the program you are watching and to deliver ads on your laptop at home which are related to the programs you are watching (e.g. a Mercedes Benz ad if you are watching Top Gear). A plethora of such examples exists.

Decisions on which ads to show in which client need to be made in a fraction of a second and should be informed by all the batch processing associated with profiling. To cope with these large streams of data SAPO currently uses a hodgepodge of big data technologies currently leading to high communication overheads and undesired operational complexity. The goal of this case study in the project is to make use of the Lean Big Data platform to improve efficiency around the management of the existing data to allow faster and better queries at the database level and also to improve cross-domain analytics made possible through the convergence of the various data streams.

6.2. Use Cases

In this section we describe several use cases related to the case study following the same template as previous case studies, namely: use case name, involved actors, short description, preconditions, effects, inputs, and outputs.

6.2.1 Use Case 1: Ad Serving

Use case name: Ad Serving

Involved actors: Client; Ad Server

Precondition: Existence of campaign published by some advertiser.

Description: A client uses any of PT's multiplatform services, whether mobile, web or TV, and receives a targeted contextualised ad which takes into consideration the user profile and specific campaign requirements by the advertiser together with general inferred clusters/models from previous usage of the platform.

Effects: The platform is updated with logging information on the ad served and the client request.

Inputs: HTTP request.

Outputs: Served ad via the appropriate channel.

6.2.2 Use Case 2: Forecasting

Use case name: Forecasting

Involved actors: Advertiser; Sales Person; Ad Analyst

Precondition: Ad serving history of a particular campaign

Description: Given the history of a campaign including all the associated specific metadata (e.g. data about the users and kind of user profiles who have consumed the ads), compute the expected behaviour of a campaign in time in terms of prints and user clicks. The forecast will be useful for Advertisers to review their campaign specifics, Sales people to predict revenues and Ad Analysts to possibly revise their ad strategy on specific cases where the forecasting of the campaign is not good.

Effects: None

Inputs: Ad consumption history.

Outputs: Forecast of ad consumption.

6.2.3 Use Case 3: Custom Reports and Dashboards

Use case name: Custom Reports and Dashboards

Involved actors: Sales Person; Ad Analyst; Data Analyst

Precondition: Relevant usage data available on the database.

Description: Any set of features/values should be prone to visualisation via insightful dashboards. Sales Persons, Ad Analysts and Data Analysts can all gain from the system allowing both ad-hoc custom sophisticated queries and real time serving of custom dashboards based on existing data. In this use case users can create and save their own dashboards based on ad-hoc queries which are executed on the fly from available data streams.

Effects: None

Inputs: Relevant sections of the existing database, depending on the custom queries defined.

Outputs: Custom real time dashboard

6.2.4 Use Case 4: User profiling

Use case name: User Profiling

Involved actor: Data Analyst; Ad Analyst

Precondition: Considerable past usage data available on the database.

Description: Through state-of-the-art real time analytics knowledge discovery algorithms the Data Analyst and/or Ad Analyst should be able to both discover clusters of users effectively profiling existing users and to find patterns in the data

Effects: None

Inputs: Full history of the platform.

Outputs: User profiles/clusters and detected patterns.

6.2.5 Use Case 5: Campaign Setup

Use case name: Campaign Setup

Involved actor: Advertiser; Ad Server

Precondition: None

Description: The Advertiser wishes to setup a campaign on the Ad Server. Through a back office the Advertiser defines the specific properties of the campaign and submits it getting feedback on expected KPIs for the campaign.

Effects: The Campaign is added to the list of campaigns to be served by the Ad Server

Inputs: Campaign properties.

Outputs: Bootstrap forecast.

6.3. Case study requirements

6.3.1 Actors

In Table 23. Case Study 4 Actors Table 5 specific actors and their primary user roles are listed.

ID	Actor name	Involvement	Notes
CS4-A1	Client	The human end client who visits a webpage or uses a service and received contextualised ads.	
CS4-A2	Advertiser	The company which sells ads on the platform. This may be an agency or a network of advertisers.	
CS4-A3	Sales Person	The person at PT/SAPO who interacts with Advertisers and analyses success of campaigns as well as high-level ad strategies.	
CS4-A4	Ad Analyst	The technically-oriented person at PT who implements specific ad strategies.	
CS4-A5	Data Analyst	The person making sense of data, responsible for inferring knowledgeable insights from available data.	
CS4-A6	Big Data Admin	The person managing the Big Data infrastructure and ensuring QoS.	
CS4-A7	Ad Server	An automatic actor that selects an ad to show based on	

ID	Actor name	Involvement	Notes
		existing campaign inventory, inferred clusters/models and information about the users.	

Table 23. Case Study 4 Actors

6.3.2 Context Requirements

The specific "environment" requirements generated by Case Study 4 are listed in Table 24. Case Study 4 Context Requirements

ID	Name	Description	UC	Notes
CS4-CR1	Data interoperability	The system should allow direct usage of existing databases and streams, including SQL, Hadoop, Esper, Storm, MonetDB.	All	
CS4-CR2	Privacy	Data fields that contain identifiable information should be anonymised and/or encrypted with appropriate access control restrictions.	All	

Table 24. Case Study 4 Context Requirements

6.3.3 Functional Requirements

The functional requirements obtained from the use cases are described in the table below.

ID	Name	Description	UC	Notes
CS4-FR1	Data loading from streams	The system must allow storing data coming from real time streams of events in a lossless fashion.	All	
CS4-FR2	Data policy enforcement	The system must allow policy enforcement for the use of data		

CS4-FR3	Knowledge discovery integration	The system must allow seamless integration with business intelligence, knowledge discovery and clustering algorithms which lift data into insightful knowledge.		
CS4-FR4	Profiling	The system should allow inference of profile clusters based on usage.		
CS4-FR5	Forecasting	The system should allow forecasting of behavior of individual users and campaigns		
CS4-FR6	Ad-hoc custom queries	The system should allow for ad-hoc custom sophisticated queries and their quick execution.		
CS4-FR7	Dashboards	The system should allow for serving real time custom dashboards based on the current data.		

Table 25: Case Study 4 Functional Requirements

6.3.4 Non-Functional Requirements

Various non-functional requirements obtained from the use cases are described in the table below.

ID	Requirement name	Description	UC	Notes
Performance requirements				
CS4.NF-P1	Load	90M requests per day, over 150M / day in peak days over 3600 req/s in peak hour	All	
CS4.NF-P2	Response time	90% of the requests served in less than 30 ms		
CS4.NF-P3	Latency	An increased latency is expected but not more than 10% over baseline		
Integration requirements				

ID	Requirement name	Description	UC	Notes
CS4.NF-I1	Legacy Compatibility	The system integrate with our existing tools as much as possible		
CS4.NF-I2	Java ready	Some form of Java API for data access/query (JDBC strongly preferred) or easily wrappable in Java.		
CS4.NF-I3	Modular and Extensible	The system will use a modular approach and cater for the integration of new components.		
Stability requirements				
CS4.NF-S1	Availability	The system will provide robust industrial-strength levels of uptime (>99.9% of the time)		
Maintainability requirements				
CS4.NF-M1	GUI Configuration	The system will provide a GUI to support its configuration update.		
CS4.NF-M2	API Configuration	The system will provide an API for configuration update		
CS4.NF-M3	Backup and restore	The system will feature robust mechanisms for backup and restore		
Scalability requirements				
CS4.NF-SC1	Elasticity	The system will provide reliable and efficient scalability mechanisms which allow for high scalability measures.		
CS4.NF-SC2	Redundancy and resilience	The system should be redundant to cater for node failure and resilient.		
Storing				
CS4.NF-ST1	SQL/NoSQL support for querying	The system will provide both SQL and NoSQL query support		
CS2.NF-ST2	Merge/Upsert	The storage should allow merge/upsert functionality when performing bulkloads.		
CS2.NF-ST3	Multivalues	The "storage system" / "sql interface" should allow multi-value data types (e.g. arrays and/or dictionaries)		

ID	Requirement name	Description	UC	Notes
Privacy				
CS4.NF-PY1	Data privacy	The system will provide mechanism to guarantee the data privacy of sensible data through anonymisation and/or encryption.		
Usability				
CS4.NF.US1	Usability	The system will provide a user friendly GUI and dashboards.		

Table 26: Case Study 2 Non-Functional Requirements

7. Conclusion and future work

This document provides the software requirement analysis that defines the main functionalities to be implemented by each LeanBigData use case. The result from these analysis will be used as starting point:

- For the development and implementation process of the use case applications addressed in work package 8,
- For defining the evaluation and testing process defined in task 7.2 and used to measure the performance of the LeanBigData platform. In order to ensure a most effective evaluation, the defined requirements will allow a cross-sectorial evaluation due to the use cases came from many different sectors: financial, social media, marketing and software.

In the document has been defined the methodology followed in the gathering requirements process. A set of uses cases specific for each case of study has been included from where the requirements has been obtained and classify according following categories: context requirements, end-user requirements, functional requirements, non-functional requirements.

To this end, work package 7 has collaborations with other work package especially in following fields:

- Definition of requirements for the software visualization tools implemented by work package 6.
- Definition of mockups specially created to help in the design of the visualization tool implemented by work package 6.

At the date of publication of this deliverable, the WP7 responsible of each use case provided a set of almost-final requirements but due to actively nature of the software requirements, we foresee that this activity will continue throughout the development lifecycle.

8. References

Nuseibeh, B. a. (2000). Requirements Engineering: A Roadmap. Proceedings of International Conference on Software Engineering (ICSE-2000). Limerick, Ireland.

Wikipedia. (n.d.). RequirAnalysis. Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Requirements_analysis

Zinc K. (January, 2013), Efficiency, Optimization and Predictive Reliability. White paper. CA Technologies. Link: www.ca.com/us/~/media/Files/whitepapers/Capacity-Management-Resource-Scoring-WP.pdf

Annex 1. Social Network-based Area Surveillance Case Study mock-ups

8.1. System configuration

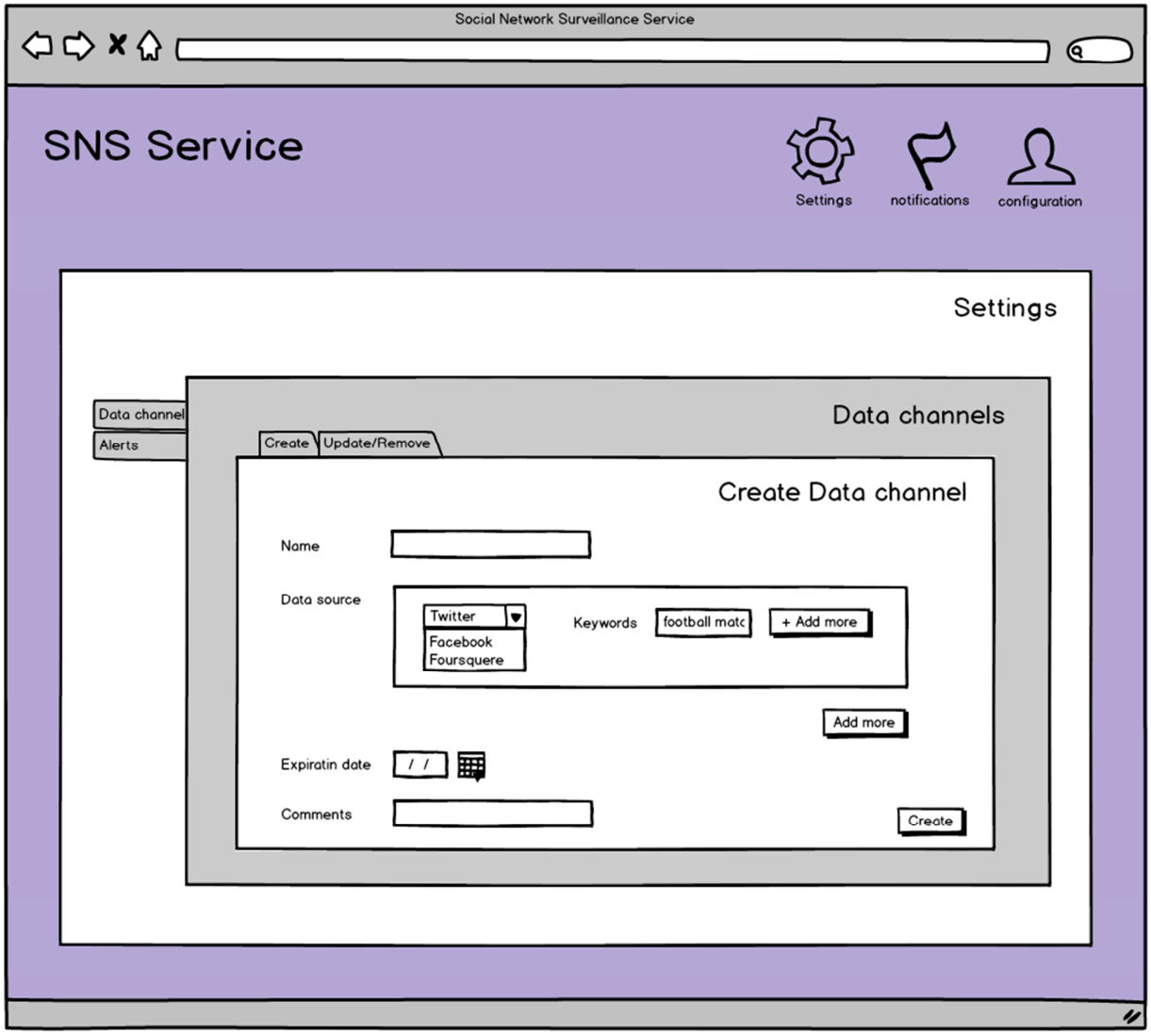


Figure 5. Data channel creation

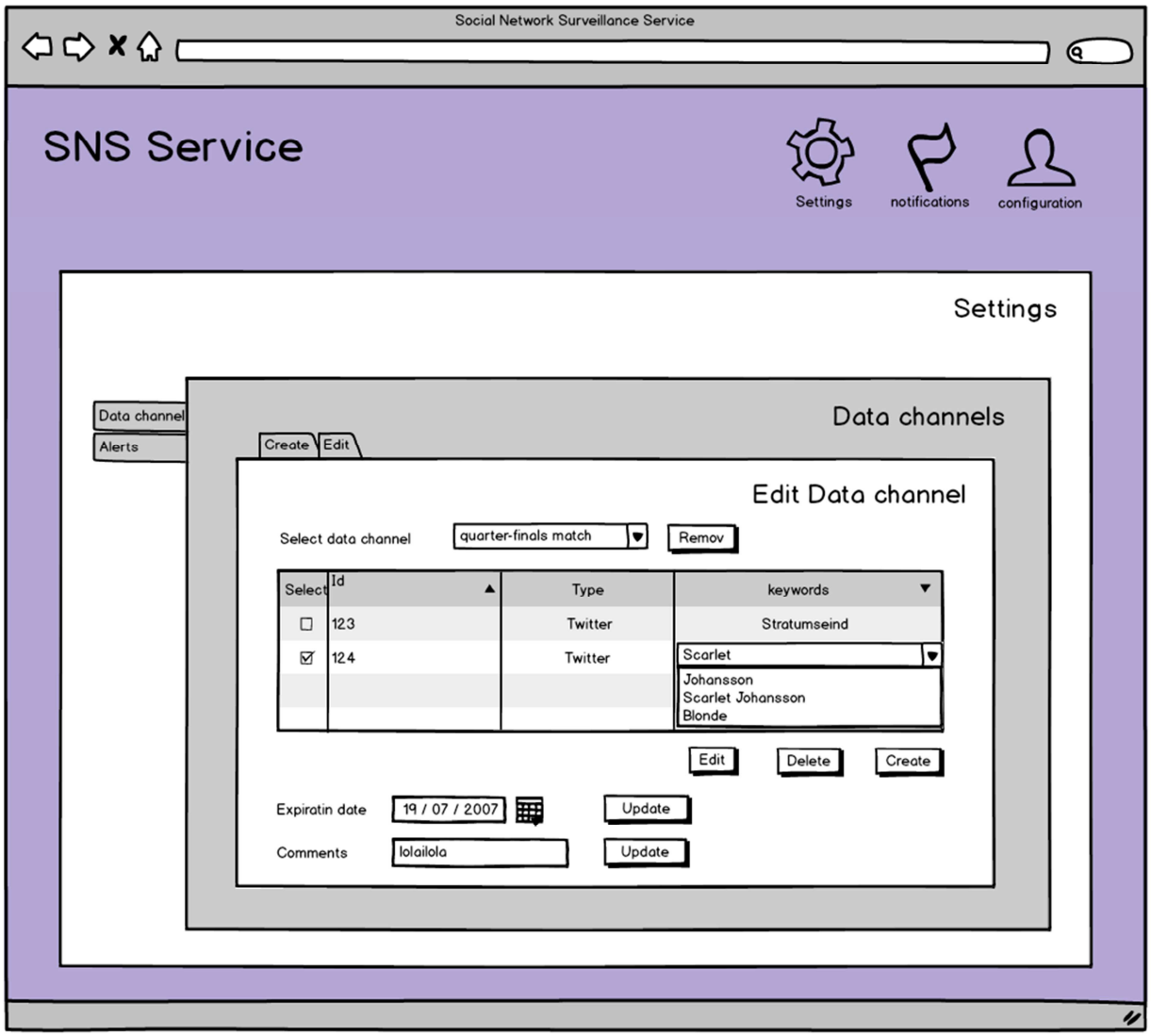


Figure 6. Data channel edition

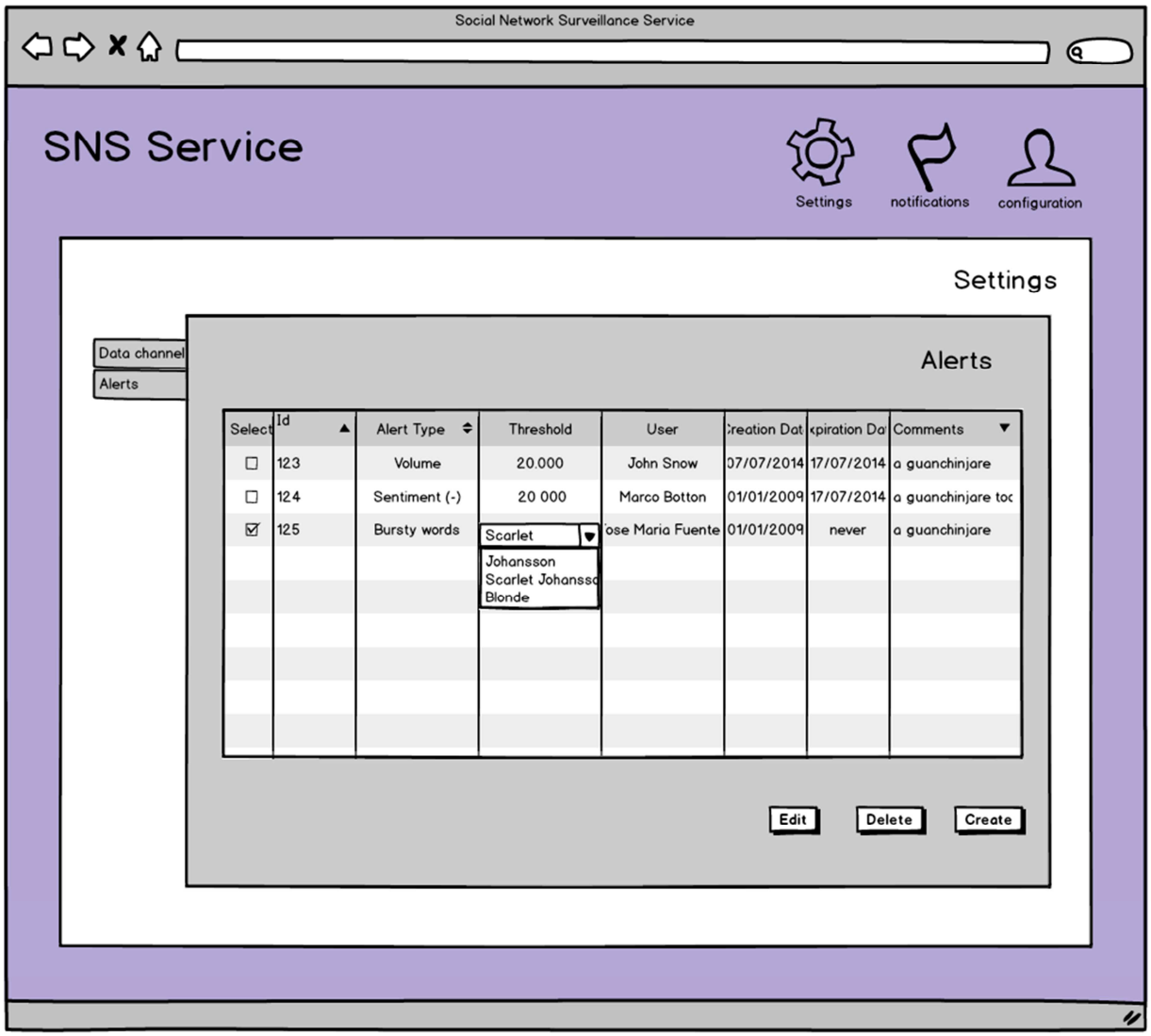


Figure 7. Alert configuration

8.2. Data channel monitoring and analytics visualization

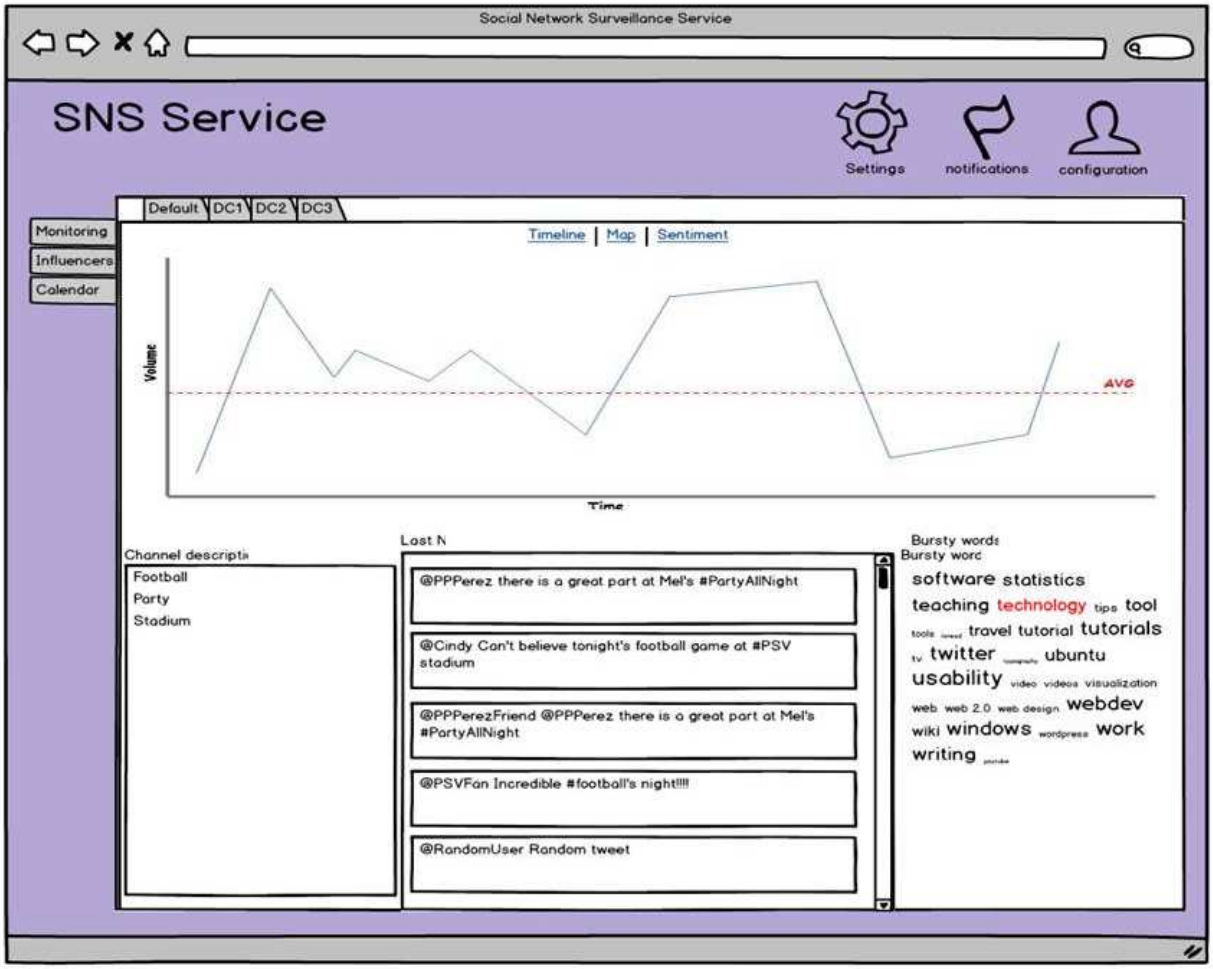


Figure 8. Monitoring Visualization: tweet volume

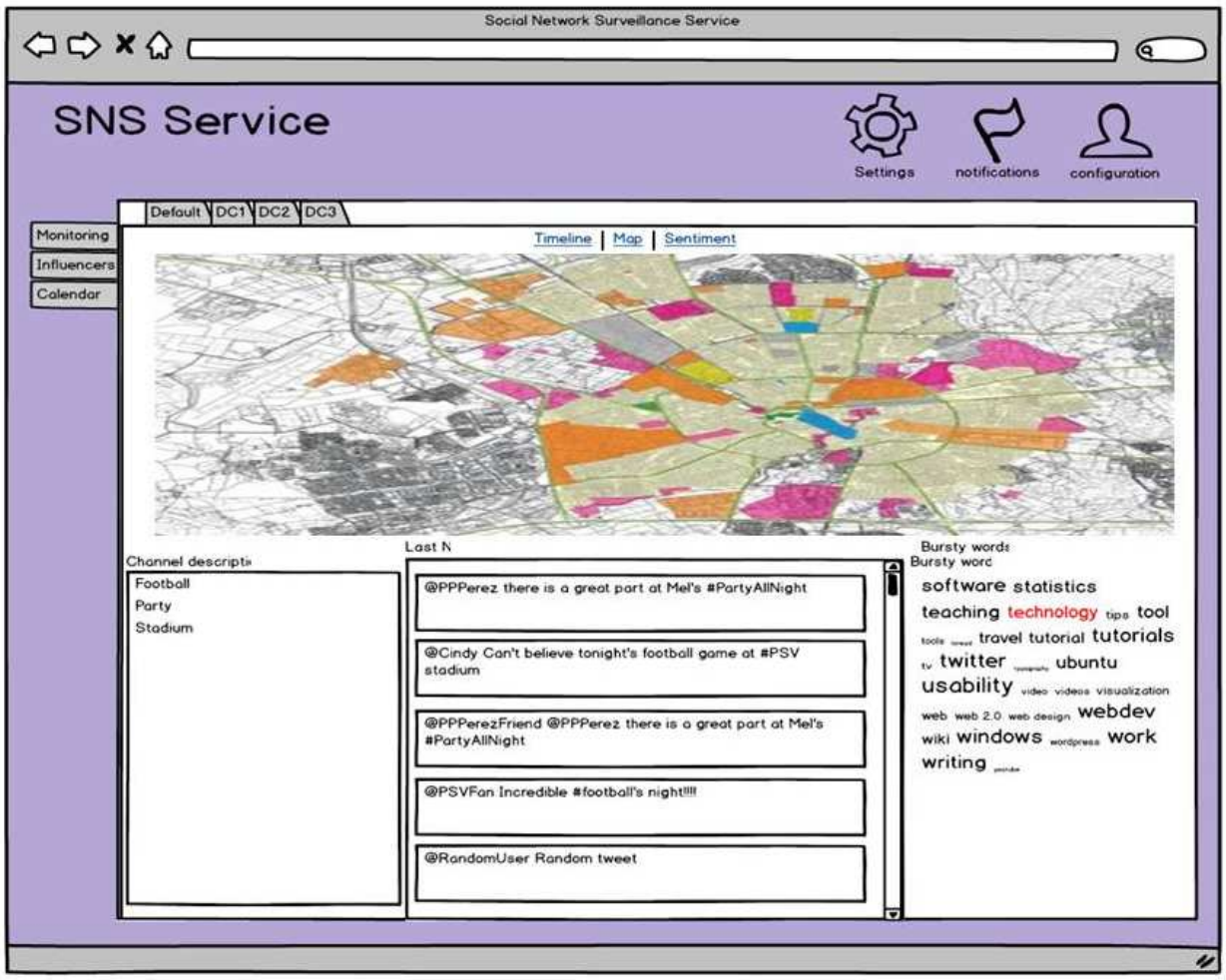


Figure 9. Monitoring Visualization: Heat map of tweet volume

Figure 10. Monitoring Visualization: tweet volume per sentiment

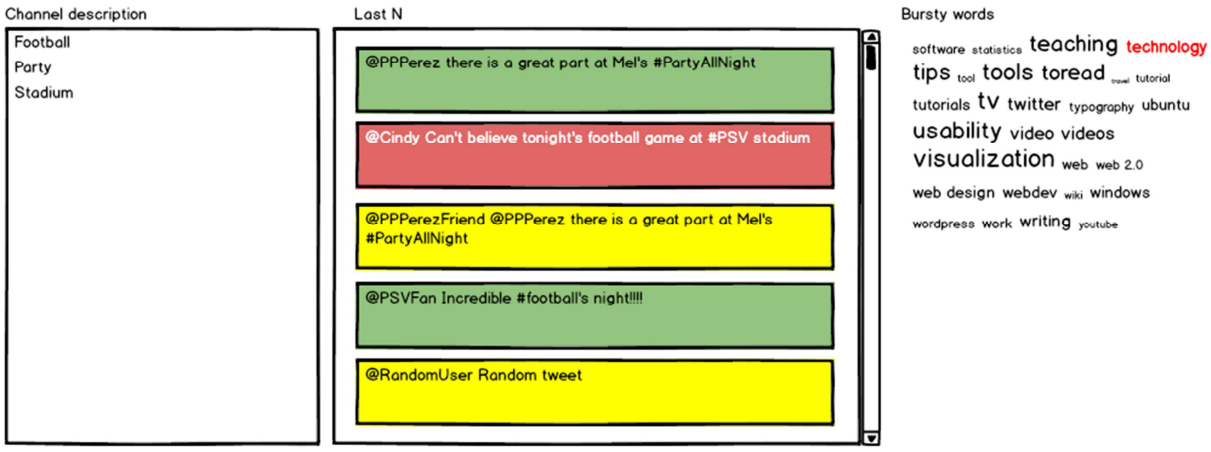


Figure 11. Visualization of data channel description, data sample and bursty cloud

8.3. Influencers visualization

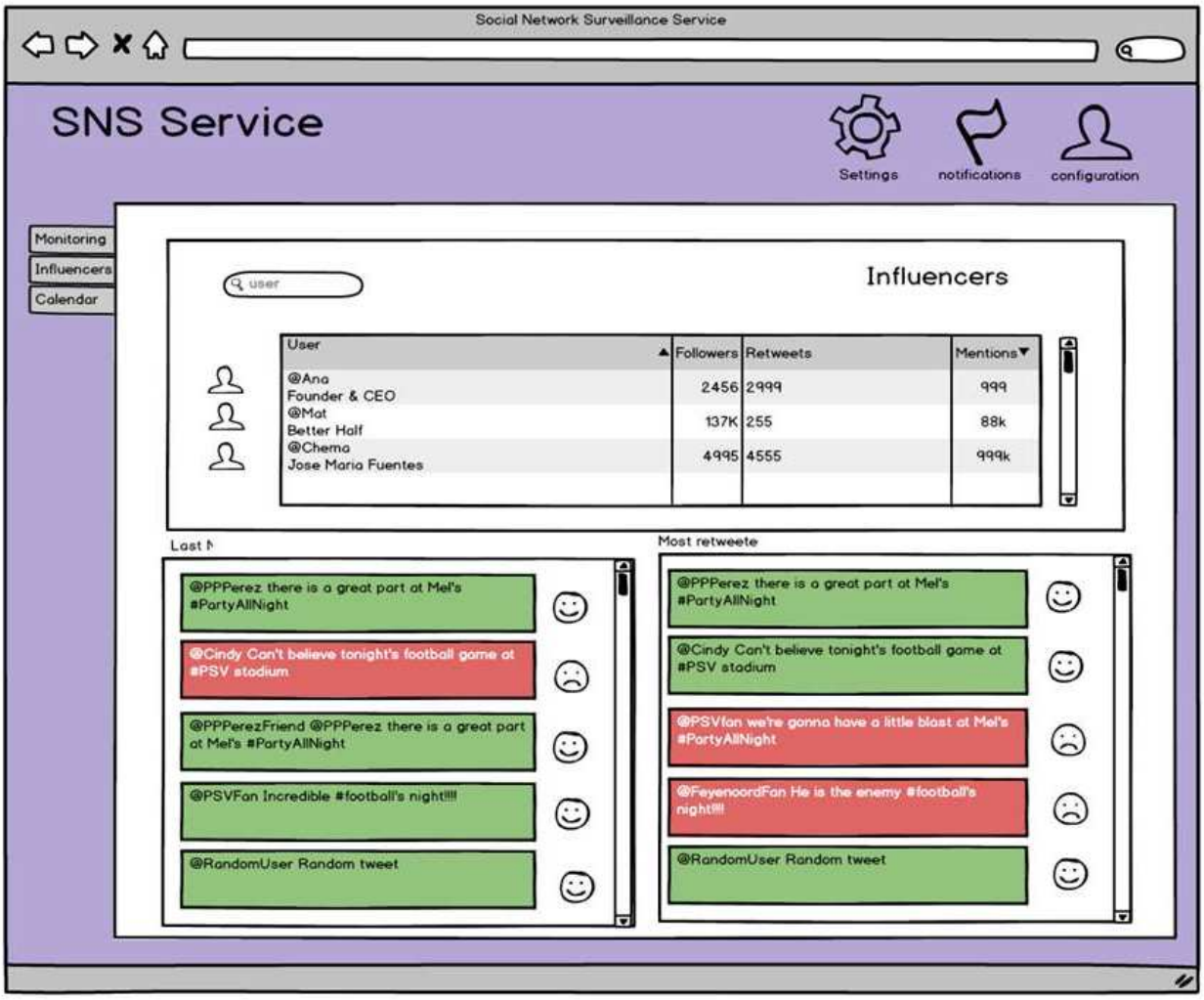


Figure 12. Influencers visualization