Horizon 2020 Program (2014-2020)

Big data PPP

Research addressing main technology challenges of the data economy

Industrial-Driven Big Data as a Self-Service Solution

D5.4: Big-Data-as-a-Self-Service Test and Integration Report v2[†]

Abstract: The current deliverable presents the second version of Big-Data-as-a-Self-Service Test and Integration Report. The work gives important perception towards the release of the envisioned I-BiDaaS solution, ensuring a smooth and effective integration of the separate *I-BiDaaS* components. It also presents the revised integration plan, which details how and when the individual technical elements of the I-BiDaaS solution are adapted and integrated in a common solution.

Contractual Date of Delivery	31/12/2019
Actual Date of Delivery	31/12/2019
Deliverable Security Class	Public
Editor	*George Bravos (ITML)*
Contributors	ITML, BSC, UNSPMF, CAIXA, CRF
Quality Assurance	*Raül Sirvent (BSC)* *Enric Pages (ATOS)* *Kostas Lampropoulos (FORTH)*

[†] *The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780787.*

The *I-BiDaaS* Consortium

Foundation for Research and Technology – Hellas (FORTH)	Coordinator	Greece
Barcelona Supercomputing Center (BSC)	Principal Contractor	Spain
IBM Israel – Science and Technology LTD (IBM)	Principal Contractor	Israel
Centro Ricerche FIAT (FCA/CRF)	Principal Contractor	Italy
Software AG (SAG)	Principal Contractor	Germany
Caixabank S.A. (CAIXA)	Principal Contractor	Spain
University of Manchester (UNIMAN)	Principal Contractor	United Kingdom
Ecole Nationale des Ponts et Chaussees (ENPC)	Principal Contractor	France
ATOS Spain S.A. (ATOS)	Principal Contractor	Spain
Aegis IT Research LTD (AEGIS)	Principal Contractor	United Kingdom
Information Technology for Market Leadership (ITML)	Principal Contractor	Greece
University of Novi Sad Faculty of Sciences (UNSPMF)	Principal Contractor	Serbia
Telefonica Investigation y Desarrollo S.A. (TID)	Principal Contractor	Spain

Document Revisions & Quality Assurance

Internal Reviewers

1. *Raül Sirvent, (BSC)*
2. *Enric Pages, (ATOS)*
3. *Kostas Lampropoulos, (FORTH)*

Revisions

Version	Date	By	Overview
0.0.1	10/10/2019	Editor	TOC
0.1	5/11/2019	Editor	1st draft
0.5	30/11/2019	Editor, V. Chatzigiannakis	2nd draft
1.0	12/12/2019	Editor	Final

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ASR: Automatic Speech Recognition
AVT: Advanced Visualisation Toolkit
BDVA: Big Data Value Association
CI: Continuous Integration
CC: Call Center
CNN: Convolutional Neural Networks
CPU: Central Processing Unit
CVS: Concurrent Versions System
DFP: Data Fabrication Platform
E2E: End-to-End
GPU: Graphics Processing Unit
GUI: Graphical User Interface
ISTQB: International Software Testing Qualifications Board
IOPs: Input/output operations per second
IoT: Internet of Things
JSON: JavaScript Object Notation
MIT: Massachusetts Institute of Technology
MQTT: Message Queuing Telemetry Transport
MVP: Minimum Viable Product
PoCs: Proof-of-concepts
QA: Quality Assurance
RMO: Resource Management Orchestrator
RTC: Real-time Computing
SCM: Source Control Management
SQL: Structured Query Language
SVN: Subversion
SW: Software
TDF: Test Data Fabrication
UAT: User Acceptance Testing
UM: Universal Messaging

Executive Summary

The current deliverable presents the second version of Big-Data-as-a-Self-Service Test and Integration Report. The work, that corresponds to tasks T5.2, T5.3, and T5.4, gives important perception towards the release of the final I-BiDaaS solution, ensuring a smooth and effective integration of the separate I-BiDaaS components, taking into consideration their availability, their interoperability potential, their scalability and performance requirements.

Following a brief introduction, the second part of this document (Section 2) is dedicated to the delivery of the 1st integrated I-BiDaaS platform. The 1st integrated solution comprises an updated architecture with respect to the one presented within WP1, and a number of complementary use cases. To that end, Section 2 describes in detail the architecture and the deployment of a generic use case; it also summarizes the I-BiDaaS components and technologies that have been integrated in the unified solution for this 1st version. All these technologies and the integrated version were delivered by M18.

The integrated version comprises 3 modes of operation with different functionalities, depending on the end user:

1. The Self-service mode allows industry in-house personnel that has domain knowledge and some insights about data analysis (non-experts) to easily construct Big Data pipelines in a user-friendly way, selecting a pre-defined data analytics algorithm from an available list.
2. The Expert mode allows experts (developers) to upload their own data analytics code based on the available I-BiDaaS highly reusable templates.
3. The Co-Develop mode corresponds to an end-to-end solution for a given industry project developed by the I-BiDaaS team (the I-BiDaaS use cases).

The use cases implemented up to now, on top of the generic one, comprise: two use cases provided by CAIXA (analysis of relationships through IP addresses and advanced analysis of bank transfer payment in financial terminal); one use case provided by CRF (production process of aluminum die-casting); and one use case designed by TID (quality of service in call centers). All these use cases and the way they were implemented in the integrated platform are described in detail in Section 3.

Section 4 presents the updated integration plan (with respect to the plan described in D5.2 [1]), detailing how the individual technical elements of the I-BiDaaS solution were actually adapted and integrated in a common solution. Within the framework of solid and efficient development management throughout the whole software development lifecycle, a number of software tools have been employed to empower the integration activities. Finally, Section 4 presents a detailed list of integration tests of the I-BiDaaS platform that are used in order to validate and evaluate the performance of the integrated system with respect to industry-validated benchmarks.

1 Introduction

The current deliverable presents the second version of Big-Data-as-a-Self-Service Test and Integration Report. The work gives important perception with respect to the release of the integrated I-BiDaaS solution and towards the deployment of the final version, ensuring a smooth and effective integration of the separate I-BiDaaS components, taking into consideration their availability, their interoperability potential, their scalability and performance requirements. It also provides insights about the updated integration plan and describes in detail the use cases implemented in this first integrated version and the exploitation of the different I-BiDaaS technologies and components.

1.1 Relation to other Tasks and Work Packages

In general, this work will provide important insights regarding the implementation and deployment of the I-BiDaaS' integrated framework, a large-scale computational infrastructure that allows real-time decision-making supporting analytics. The outcome of the work is strongly connected to the preliminary work reported in D5.2 and will be directly exploited in the next and final release of "Big-Data-as-a-Self-Service Test and Integration Report" (D5.6).

Besides, there is a close interrelation between this deliverable and Task 1.3 (Conceptual architecture specification) and Task 1.4 (Experimental protocol specification) in WP1 (Setting the scene: Baseline framework). It has been initially stated that the outcomes of Task 1.3 will be the starting point for Task 5.3 (Integration towards Big-Data-as-a-Self-Service). In particular, in Task 1.3 a thorough understanding of the challenges, technologies and the state-of-the-art in terms of data economy, big data processing and the requirements of the end users have been successfully addressed. Thus, D5.4 has been structured based on some important insights taken from T1.3 (Conceptual architecture specification).

Furthermore, D5.4 is closely connected to Task 1.4, as the final release of the envisioned I-BiDaaS solution should fulfill all the key quality tests with respect to the industrial-validated benchmarks defined in Task 1.4. Finally, the I-BiDaaS evaluation and impact analysis (Task 6.3) is closely aligned with Task 5.3; D6.1, D6.2 and D6.3 are in line and have been following the I-BiDaaS prototype releases (D5.2 and D5.4 up to now).

1.2 Contribution to the Scientific and Business Objectives

Having a central and core position within the whole concept of I-BiDaaS solution, D5.4 and the related versions contribute towards the following I-BiDaaS objectives:

Objective 1: Develop, **validate, demonstrate**, and support, a complete and solid big data solution that can be easily configured and adopted by practitioners.

Objective 2: Construct a safe environment for methodological big data experimentation, for the development of new products, services and tools.

Objective 3: Develop data processing tools and techniques applicable in real-world settings and **demonstrate** significant increase of speed of data throughput and access.

1.3 Structure of the Document

This deliverable is divided into the following sections: **Section 2** is dedicated to the description of the I-BiDaaS generic use case, the presentation of the updated architecture used in the integrated solution, and a brief description of the tools, technologies and components that were

integrated in the I-BiDaaS platform. **Section 3** presents a detailed description of the use cases provided by the I-BiDaaS end users that were implemented in the integrated platform. **Section 4** provides the updated testing and integration plan, ensuring the development of various tools and components and their further integration within the final I-BiDaaS platform. Finally, **Section 5**, gives the overall conclusions and the next steps.

2 The 1st I-BiDaaS Integrated version

2.1 1st integrated version generic use case description

In order to properly demonstrate the services and functionalities of the I-BiDaaS integrated platform, the consortium decided to proceed - on top of the developments of the tailor-made I-BiDaaS use cases - to a complementary implementation of a generic use case; the main purpose was to demonstrate, in collaboration with the use cases provided by CAIXA, CRF and TID, the functionalities of the different envisioned modes of the I-BiDaaS platform, namely (i) The Self-service mode, which allows industry in-house personnel that has domain knowledge and some insights about data analysis (non-experts) to easily construct Big Data pipelines in a user-friendly way, selecting a pre-defined data analytics algorithm from an available list; (ii) The Expert mode, which allows experts (developers) to upload their own data analytics code based on the available I-BiDaaS highly reusable templates; and (iii) The Co-Develop mode, which corresponds to an end-to-end solution for a given industry project developed by the I-BiDaaS team (the I-BiDaaS use cases).

In the generic use case, users can either initiate existing projects (with already implemented algorithms, *e.g.*, K-means) or create custom ones (by uploading and running their own code - expert mode). Then they create and execute their experiment(s) based on their data and preferences. More technical details about this process are presented in Section 2.3.5.

In the following, the basic services and functionalities of the 1st version of the I-BiDaaS platform, as delivered in M18 via the Generic Use case implementation, are summarized:

- Users with registered accounts can log in in order to use the rest of the web application.
- After logging in, users can access the different projects.
- Projects are separated in 3 different categories (categories' names are under revision):
    1. The Self-service mode allows industry in-house personnel that has domain knowledge and some insights about data analysis (non-experts) to easily construct Big Data pipelines in a user-friendly way, selecting a pre-defined data analytics algorithm from an available list.
    2. The Expert mode allows experts (developers) to upload their own data analytics code based on the available I-BiDaaS highly reusable templates.
    3. The Co-Develop mode corresponds to an end-to-end solution for a given industry project developed by the I-BiDaaS team (the I-BiDaaS use cases).
- The meaning of the entity "Project" within the I-BiDaaS platform, is using specific algorithms and, therefore, specific code in order to achieve the desired result. The users can run "Experiments" using any projects available to them. The experiment is the result of actually running the code of a specific project, using specific parameters and input.
- When a new user logs in, he/she only has access to the predefined experiments, both the default and the self-service ones.
- Predefined projects can be accessed by any user. User clicks on one of them and chooses run experiment. He/she is then called to provide the necessary parameters required, which vary depending on the project. Once these parameters are provided, the experiment starts. Although these projects are accessible by every user, the experiments and their results belong to the user who initiated them, and they can only be seen by him/her.
- The user can choose the input for each experiment either by selecting a file, which has been pre-uploaded to the server, or a Cassandra keyspace. The available files and keyspaces are provided by a dropdown list.

- While the experiment is active, its status remains on "running". Once it has finished, the user can download the logs of the experiment, which will include the output file(s), if there is one.
- The results of the experiments of predefined projects are also represented visually. Visualization differs depending on the project.
- Custom projects are created by the users. A name, a description and an input method (file, Cassandra keyspace or no input) are required in order to create a custom project. Furthermore, the user must choose whether he/she will use the default I-BiDaaS docker image or another one from Dockerhub. Finally, the user must upload a zip file with the code that will be used for this project.
- Custom projects belong to the user who created them and only he/she can run experiments on them.
- Running custom project experiments requires providing a name for that experiment, the command that should be used to run the custom code, and the input, if it is necessary for the specific project.
- Some rules/functionality apply throughout both predefined and custom experiments:
    - When running a new experiment, the user may select the resources to be utilized in the experiment, in terms of cpu number and available memory per cpu.
    - Experiments that have already taken place are available after selecting a specific project.
    - They can be deleted.
    - Their logs can be downloaded.

2.2 1st integrated version Architecture

The architecture of the 1st integrated version includes most of the modules identified in the complete I-BiDaaS architecture (Figure 1); a simplified version of the 1st version architecture is presented in Figure 2 and more details for each one of the implemented components are presented in the remaining subsections of Section 2.3 of this document. The integration of all modules of I-BiDaaS architecture is expected to be demonstrated in the 2nd and final integrated version.

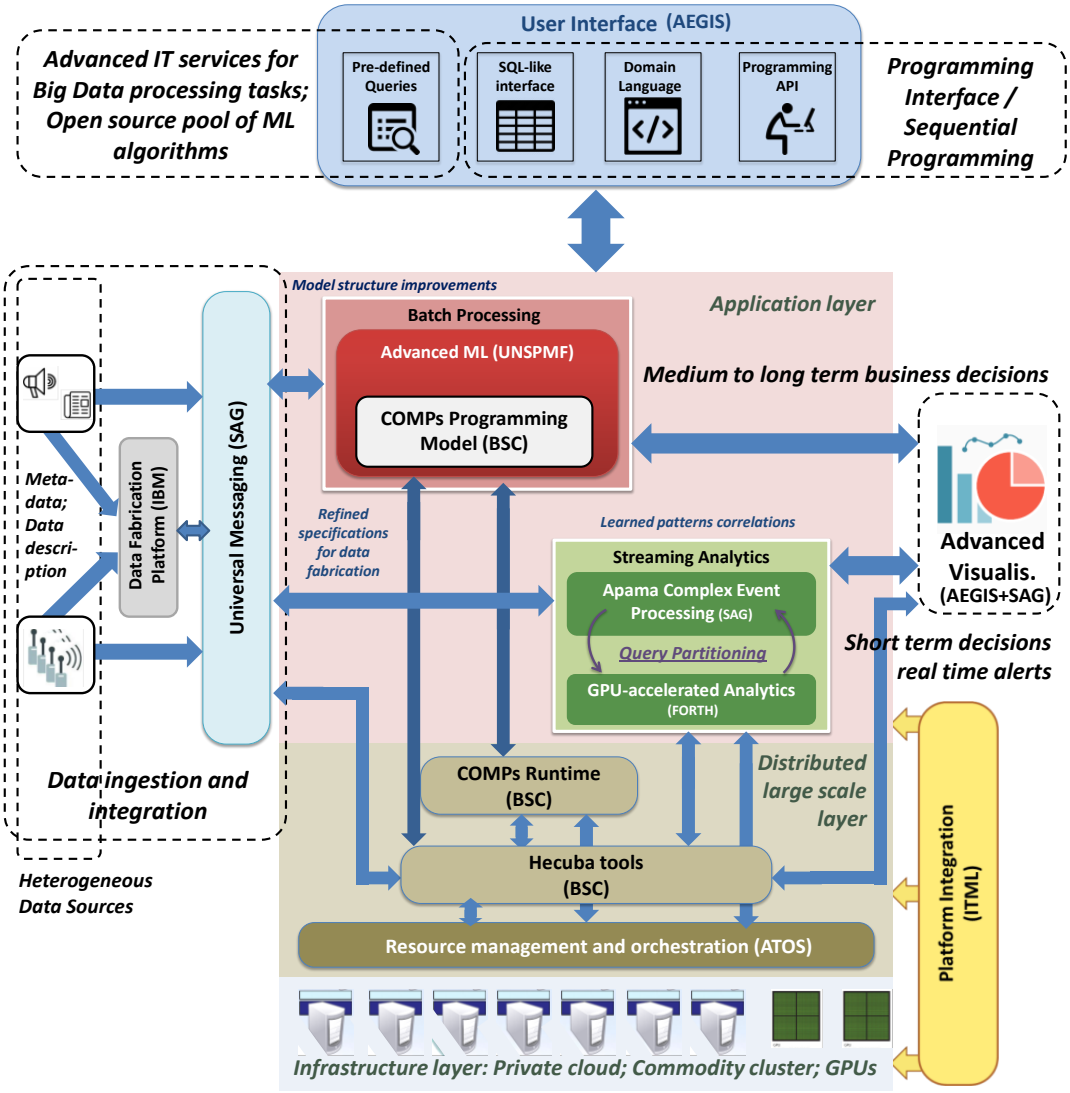


Figure 1. The I-BiDaaS architecture

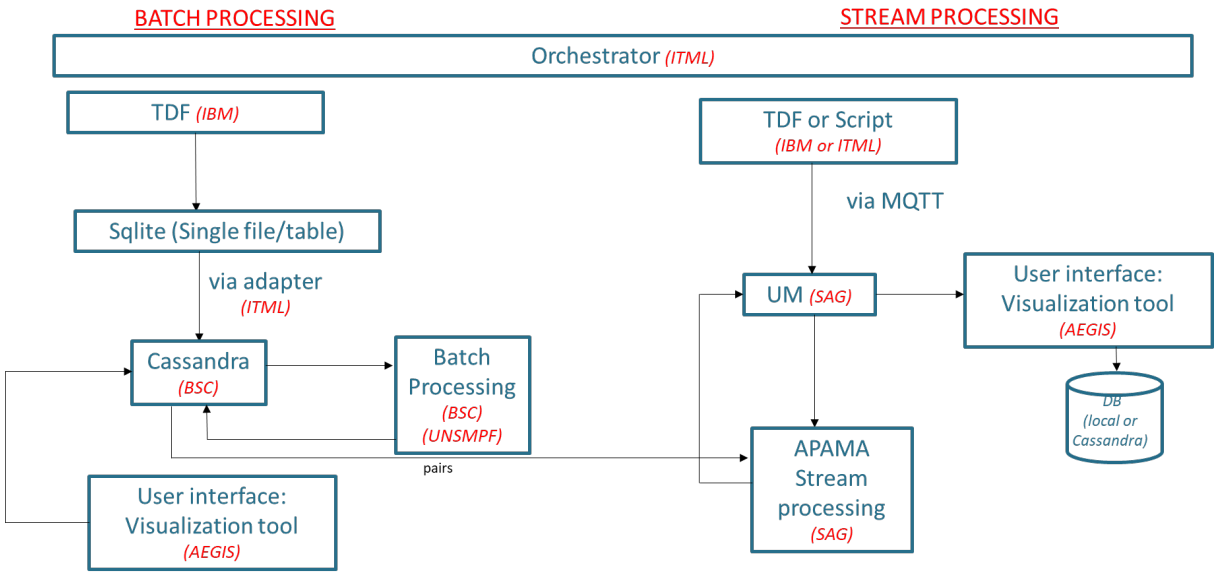


Figure 2. The 1st integrated version architecture

Focusing explicitly on batch processing, the system architecture is shown in Figure 3. The cluster that runs the batch processing jobs is based on docker swarm[1]. The swarm consists of a manager node and a set of worker nodes. The orchestrator assigns dynamically a set of workers from the set of available nodes in docker swarm, based on the user's preferences and set-up inserted through the UI. These workers exploit the Cassandra DB and the shared FS in order to complete the requested job.

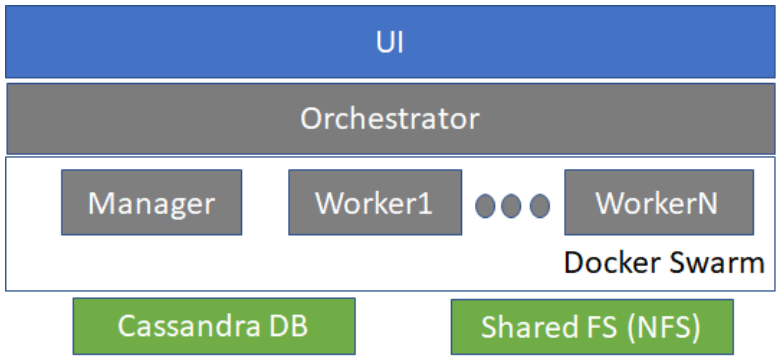


Figure 3. Batch processing architecture: The orchestrator and the docker swarm

On the other hand, Figure 4 demonstrates the case of stream processing; here, the analysis is carried out through the APAMA analytics module in collaboration with the Universal Messaging (UM) tool, rather than via docker swarm.

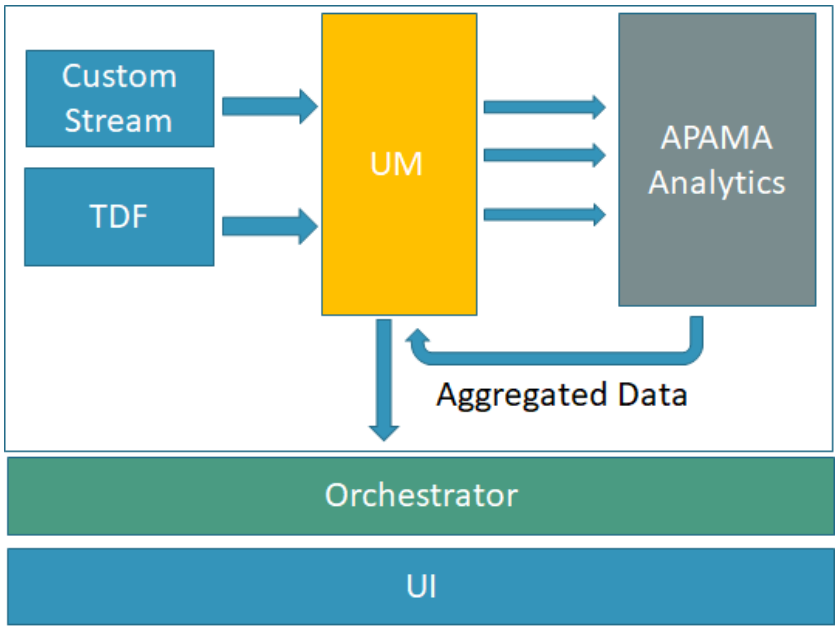


Figure 4. Streaming processing architecture: The orchestrator and APAMA analytics

[1] https://docs.docker.com/engine/swarm/

2.3 1st integrated version components and technologies

2.3.1 Universal Messaging

Universal Messaging[2] is a mature and stable commercial product. It is part of SAG's Enterprise Service Bus and provides a message broker which supports multiple message passing standards like JMS, AMQP and especially MQTT, an ISO standard. For the I-BiDaaS project, we have chosen MQTT as our message passing standard.

It can be operated in a cluster to provide high availability and reliability. The UM servers contain common messaging resources, such as channels/topics or queues.

More conceptual and technical details are given in D2.4 [2] (Universal Messaging Bus (Interim Version)) and will be also given in D2.6 (Universal Messaging Bus (Final Version)).

The UM servers can be configured and monitored via a Java tool called Enterprise Manager. With this tool, one can visualize and inspect the different message topics. This is shown in Figure 5, where the input topic for the CRF Aluminum die casting use case is shown in one message. The tree of topics is shown to the left and after an individual topic is selected, the messages published to this topic can be monitored and inspected. In addition, one can also republish a message in its original form or modify it, before republishing it, or even publish new messages.

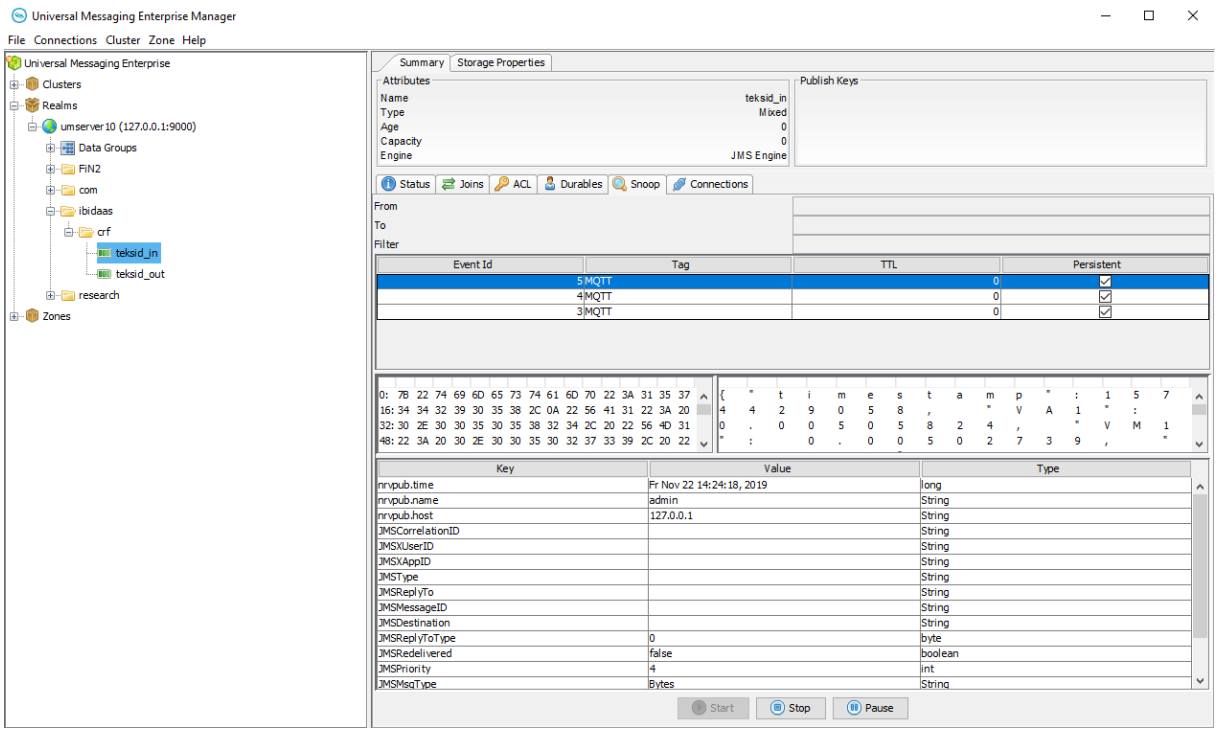


Figure 5. Example of the inspection of a message topic

If the amount of message passing data increases, *e.g.*, when more use cases are running at the same time, one can let several UMs work together.

[2] http://www.universalmessaging.org

2.3.2 Batch Processing in 1st integrated version

The batch processing module is composed of three main high-level components. The ready-to-use ML algorithms, the data management, and the integration between the batch module and the Test Data Fabrication (TDF) component[3].

The pool of ready-to-use algorithms currently consists of:

- Distributed ElNet ADMM
- Distributed Lasso ADMM
- Distributed logistic regression ADMM
- Distributed Ridge ADMM
- Parallel Differential Privacy K-means
- Ridge regression ADMM

All these algorithms are publicly available in the I-BiDaaS knowledge repository[4] and they are implemented using COMPSs/PyCOMPSs[5] to distribute the workload. At the same time, thanks to the integration between Hecuba[6] and the open-source Dislib[7] library, the following algorithms are also available:

- Random Forests Classification
- Cascade Support Vector Machines Classification
- K-Means Clustering
- DBSCAN
- Gaussian Mixture Model
- K-Nearest Neighbours
- Alternating Least Squares Recommendation

Furthermore, UNSPMF has contributed a general-purpose version of the ADMM to the Dislib project.

The integration of Hecuba and PyCOMPSs with Docker uses Docker swarm to coordinate the different parts of each deployment. At first, a Cassandra instance is created to provide backend storage; later, the Swarm stack executes the PyCOMPSs application. The whole procedure counts on two main scripts. The first script allows users to create a Docker image with PyCOMPSs, all required dependencies, and the user application code installed. Alternatively, it is possible to develop a new Dockerfile and create customized images with the desired software. The second script takes as input the number of resources (*e.g.*, number of CPUs, nodes, memory allocations, *etc.*) we need for deployment. The scripts use such information to create a *docker-compose.yml* file that describes the logical component deployment and later uses this file to start the application using the swarm manager as the PyCOMPSs master. The script also retrieves the results once the execution is finished. Figure 6 shows an example of the runcompss-docker command.

[3] https://www.ibm.com/us-en/marketplace/infosphere-optim-test-data-fabrication

[4] https://github.com/ibidaas/knowledge_repository

[5] https://pypi.org/project/pycompss/

[6] For more information check deliverable D3.2, http://www.ibidaas.eu/sites/default/files/docs/ibidaas-d3.2.pdf

[7] https://github.com/bsc-wdc/dislib

```
runcompss-docker --worker-containers=N
                 --swarm-manager='<ip>:<port>'
                 --image-name=\"DOCKERHUB_USER/IMG-NAME\"
                 --stack=stack-name
```

Figure 6. Example of the runcompss-docker command

Hecuba provides the data management functionalities that allow programmers to access data via a portable and easy-to-use interface. During the project, Hecuba's capabilities have been extended in order to meet the requirements of the project data provider. In particular, support has been added for distributed Numpy, support for new types (set, date, datetime and time) and a dynamic task granularity scheduler. The current functionalities of Hecuba are cross-class referencing, support automatic task parallelization with PyCOMPSs, a Hecuba cache for fast access to recently retrieved data, and a split method to iterate in chunks of data. For more information check deliverable D3.2, M18.

Hecuba is also the preferable interface to Qbeast [4], a novel Multidimensional Storage with Efficient Sampling that allows to index and organizes data according to its multiple dimension at high-speed. Qbeast allows approximate analytics, interactive visualization and faster machine learning. Currently, a prototype version of Qbeast's interactive visualization is used in the CAIXA's advanced bank transfer use case, and it is in the process of being fully integrated with the I-BiDaas platform. Also, currently under deployment, it is the integration of Dislib and Qbeast that will allow performance improvements of machine learning algorithms such as K-means and DBSCAN[8].

One of the goals of the I-BiDaaS project is to simplify the generation of synthetic data that can be used for testing and developing purposes. The best way to do so is to use a batch application that analyses the real data and produces descriptive statistics. This information can then be used to create rules that allow generating synthetic data automatically. For instance, the batch layer can calculate the max, min and standard deviation of a field, and provide such information to the TDF, so that it can generate internal rules that follow the detected statistical distribution.

Therefore, in the current workflow, a specific module of Hecuba that analyses the real data is generating various descriptive statistics. Once the batch application terminates, the same code uses the computed information to produce a JSON file, which follows a pre-determined schema.

The JSON file contains a set of statistics for each column of the target dataset. The file can be uploaded to TDF using its REST API first to set up the project, and then to start the automatic generation of synthetic data. An example of the JSON descriptive file is shown in Figure 7.

[8] For more information check deliverable D3.2, http://www.ibidaas.eu/sites/default/files/docs/ibidaas-d3.2.pdf

```
{
    "columnName": "PK_TSINSERCION",
    "columnAnalysisResults": {
        "nbRows": 799999,
        "minDateValue": "2019/04/01 00:00:00",
        "maxDateValue": "2019/04/01 23:59:59",
        "nbOfNullValues": 0,
        "nbOfEmptyValues": 0,
        "nbOfUniqueValues": 4839,
        "nbDistinctValues": 81961,
        "maxDistinctFrequency": 41,
        "minStringValue": "2019/01/04 00:00:00",
        "maxStringValue": "2019/01/04 23:59:59",
        "dateStats": {
            "bins": [
            ],
            "stats": {
                "count": 799999,
                "mean": 1554116169.0,
                "variance": 376923467.303314
            }
        },
        "logicalTypeDistribution": [
            {
                "count": 799999,
                "distinctCount": 81961,
                "value": "NUMERIC"
            }
        ]
    }
},
```


Figure 7. Example of the JSON descriptive file

2.3.2.1 Next steps

The major steps required in the implementation of the batch module are completed. Future work will focus on increasing the number of implemented algorithms and on improving the integration between Hecuba, Dislib and Qbeast to exploit new types of data parallelism that allows better performance and scalability. Furthermore, in the future, new complex data generation rules will be implemented and integrated in IBM's TDF.

2.3.3 Streaming analytics in 1st integrated version

SAG's Apama[9] is an event processing platform that allows to develop streaming analytic applications. It monitors rapidly moving event streams, detects and analyzes important events and patterns of events, and immediately acts on events of interest according to user's specifications.

Event-based applications differ from traditional applications because rather than continuously executing a sequence of instructions, they listen for and respond to relevant events. Events describe changes to particular real-world or computer-based objects, for example the energy consumption of a device or the temperature in a room.

In the I-BiDaaS platform, they are used to analyze the data provided by the use cases and prototypes in real-time or nearly real-time in order to gain faster insights or reduce costs or production times. More details are given in D4.1 (Real time complex event processing engine – design and approach) and D4.2 (Distributed event processing engine).

In order to detect interesting patterns in the event stream, one can write queries or the so-called monitor script files. In I-BiDaaS, we use the latter for detection. In Figure 8, we show the console output from Apama for the CRF Aluminum die casting use case, where engine parts were classified using a Random Forest Model.

[9] https://www.softwareag.com/be/products/az/apama_predictive_analytics/default.html

```
INFO  [3096] - ibidaas.CRFUseCase1 [1] - got suggested classification of 0 from Scikit Random Forest Model
INFO  [3096] - <connectivity.chain.MQTTManager-ToMQTT[ibidaas/crf/teksid_out]> Connectivity chain created, subscribed to [mqtt:ibidaas/crf/teksid_out]
INFO  [3096] - ibidaas.CRFUseCase1 [1] - got teksid event : ibidaas.Teksid(1563286442,.316248834,.446662873,2.481905222,255,4.296374321,4.578121662,7.12973
INFO  [3096] - ibidaas.CRFUseCase1 [1] - got suggested classification of 1 from Scikit Random Forest Model
INFO  [3096] - ibidaas.CRFUseCase1 [1] - got teksid event : ibidaas.Teksid(1563286443,.315619171,.453923047,2.420047998,261,4.21508646,4.551946163,12.99811
INFO  [3096] - ibidaas.CRFUseCase1 [1] - got suggested classification of 2 from Scikit Random Forest Model
INFO  [3096] - ibidaas.CRFUseCase1 [1] - got teksid event : ibidaas.Teksid(1574429058,.00505824,.00502739,.0046951,.00501589,.00451785,.00456863,.00246592,
INFO  [3096] - ibidaas.CRFUseCase1 [1] - got suggested classification of 0 from Scikit Random Forest Model
```

Figure 8. Detection and classification of CRF Aluminum die casting events

The output is also passed on from Apama to UM, so that it can be visualized in the AVT from AEGIS. The message is shown in Figure 9 by inspecting the corresponding topic.

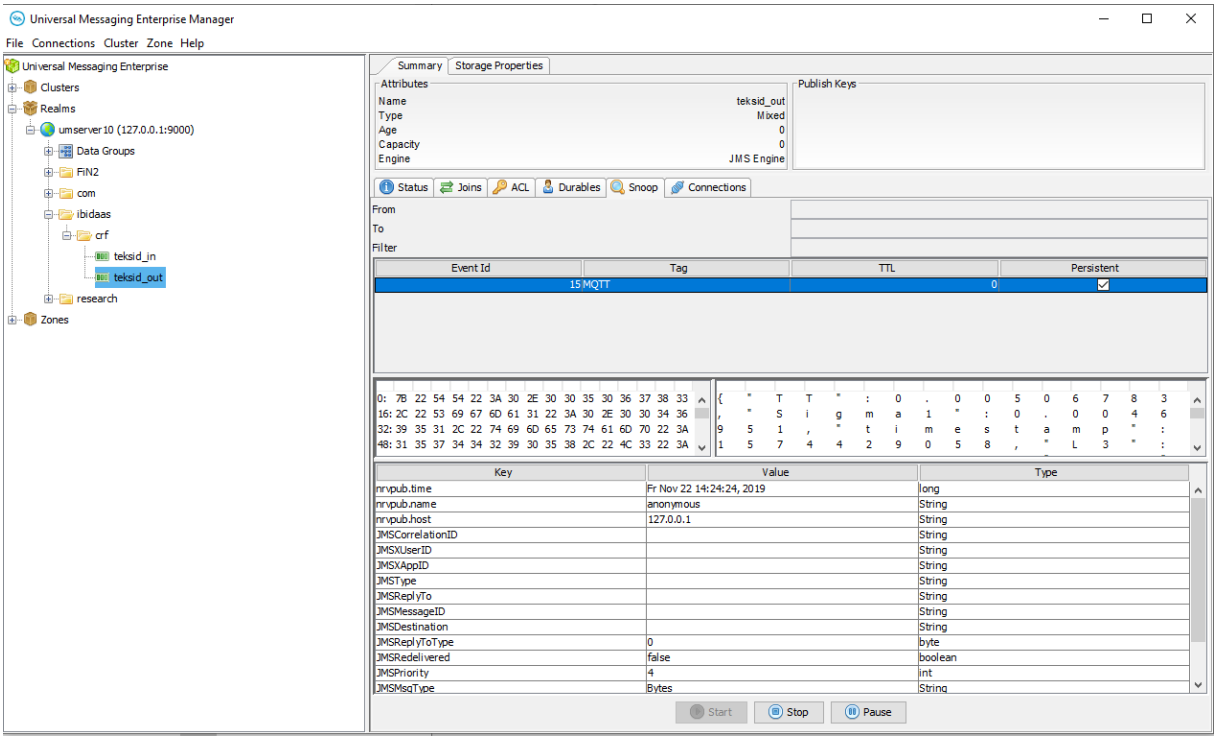


Figure 9. Classification send to UM for further processing

2.3.4 Visualization in 1st integrated version

The 1st integrated version of the I-BiDaaS prototype exposes all the offered functionalities via the Multipurpose Interface and interactive visualization framework, which is initially described in D2.3 (I-BiDaaS visualization and monitoring framework, and a multipurpose interface), but has undergone numerous adaptations in order to support the emerged visualization needs of the use cases. Following a dashboard-styled layout, the interface allows different types of users to create the so-called 'projects' and perform their analysis on their datasets, following a 3-step wizard. The first step is the project creation, where the project's dataset, data source and data processing mode are defined. A project is the basic entity of the system and depending on the target user group it has various levels of configurability. In the expert mode (see Figure 10), users can create new projects and configure the dataset, executed algorithm and data processing mode (batch or streaming).

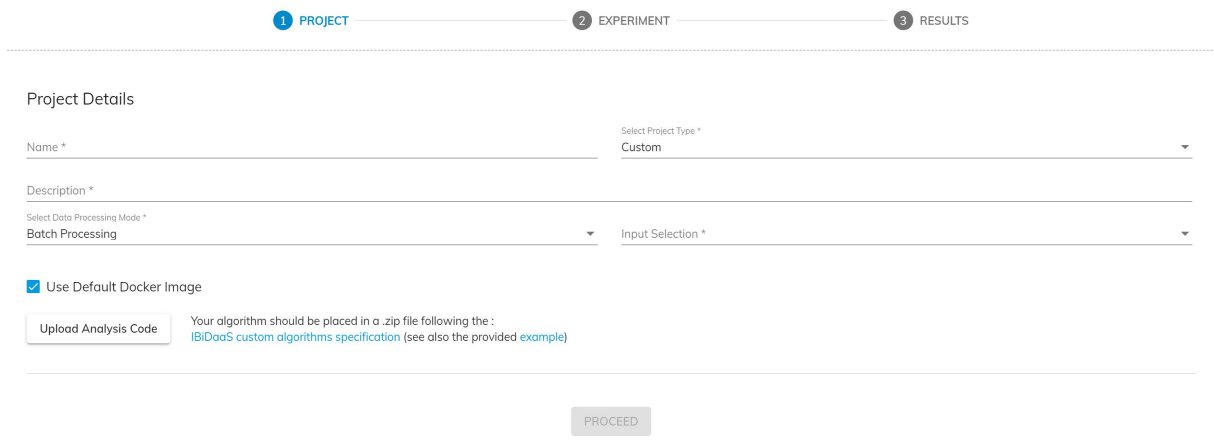


Figure 10. Project Creation - expert mode

For the self-service mode (see Figure 11), users can select a project from the predefined project options based on the algorithm that they wish to run on their dataset. Therefore, the analysis code is predefined and only the input dataset can be configured by users.

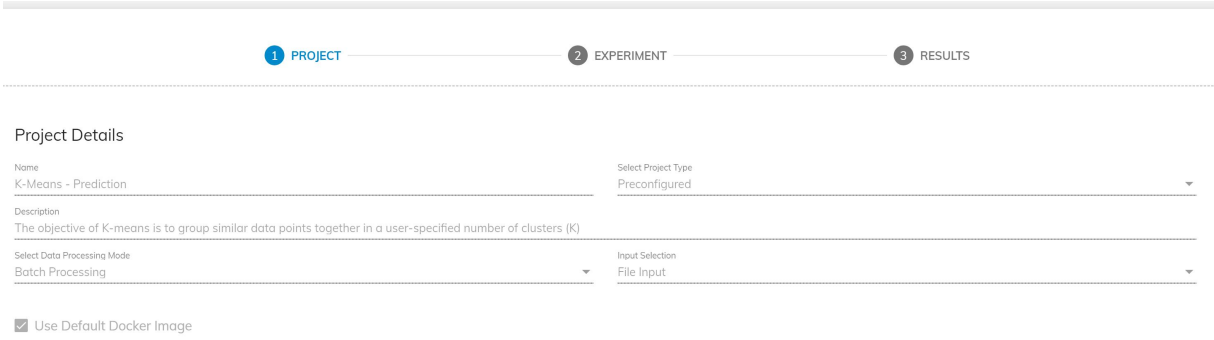


Figure 11. Project creation - self-service mode

Moreover, projects in co-develop mode have both the data source and the analysis algorithm predefined, so that users can only add experiments when selecting these projects.

An experiment is an instantiation of a project, where users experiment with different parameters and resources so as to see what is the optimal configuration for their data processing needs. The interface supports setup and execution of experiments for every project type and adapts to the configurable parameters of every project.

In expert mode (see Figure 12), users have to define the actual command that will execute the analysis of the given dataset, based on the custom analysis code that has been provided to the project.

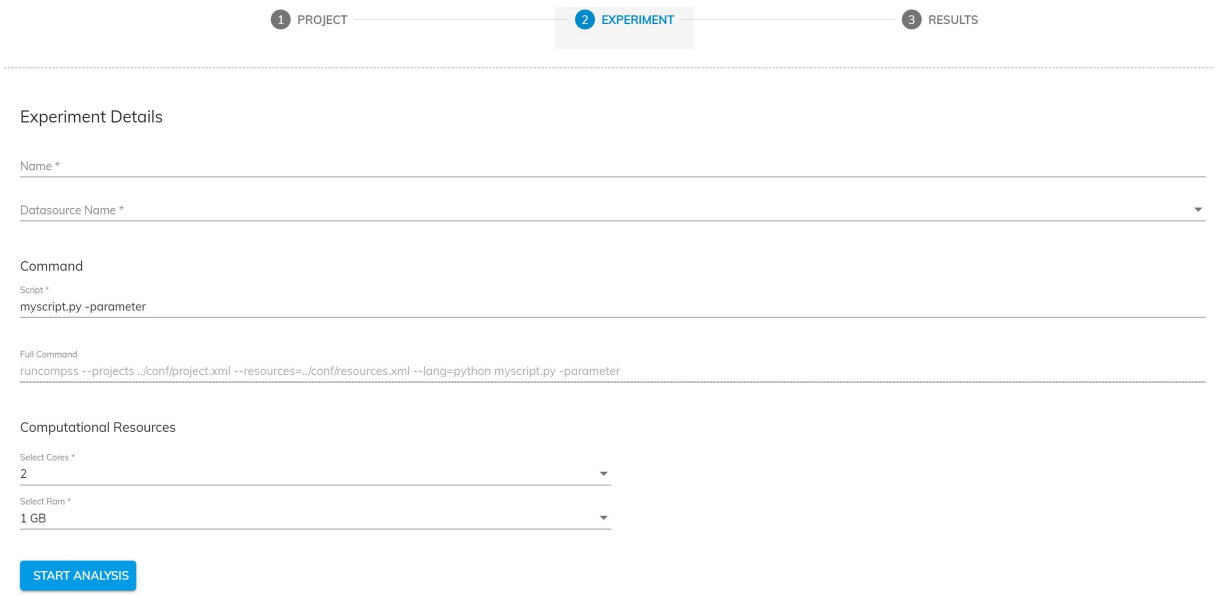


Figure 12. Experiment setup - expert mode

When selecting a project in self-service mode (see Figure 13), users can provide the values to the selected algorithm's parameters. A set of default options is provided, thus allowing users to start experimenting with some known nominal values for the specific algorithm and afterwards create more experiments with different parametrisation.

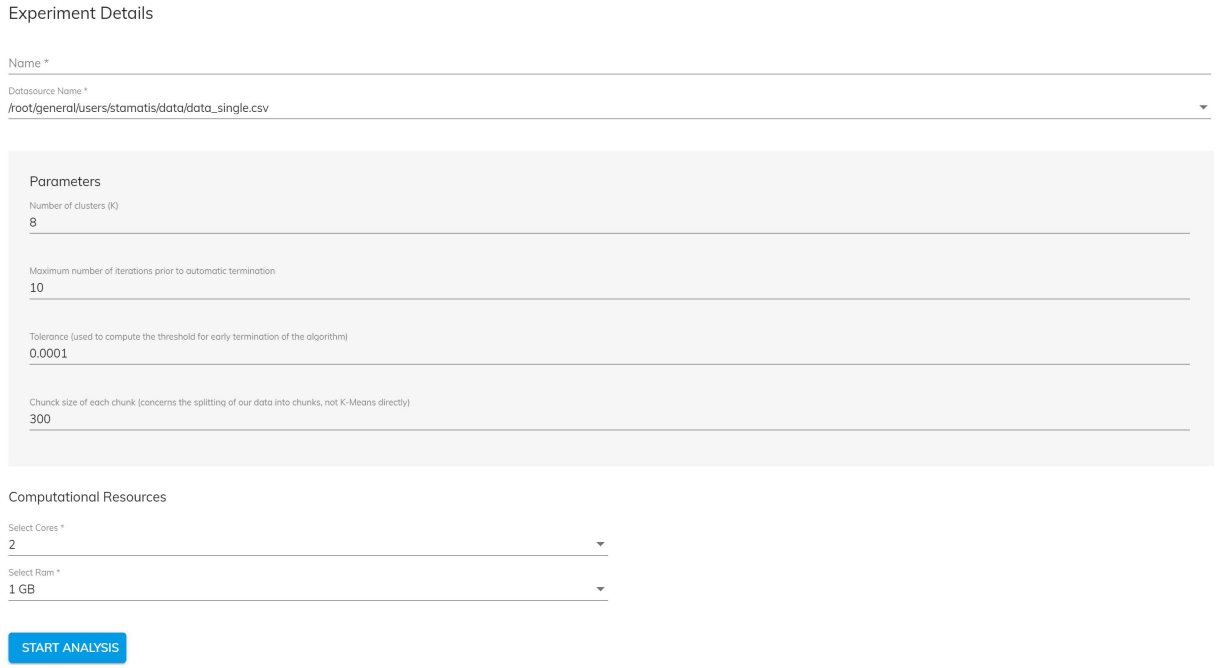


Figure 13. Experiment setup - self-service mode

Experiments in projects available in co-develop mode currently only support selection of the provided computational resources, since the rest of the algorithmic parameterisation is defined specifically for each one of the projects of this mode.

More specifically, building on the work conducted for the delivery of MVP (M12), custom visualisation options have been developed for the streaming use cases (namely CAIXA - Analysis of relationships through IP addresses, CRF - Production process of aluminum die-casting and TID - Quality of service in Call Centers), and the batch processing use case of

CAIXA - Analysis of relationships through IP addresses. The visualisations of these use cases include standard chart representations (bar charts, line charts, area plots, *etc*.), as well as other dynamic elements like information maps or data tables. The latter ones have been used to deal with the visualisation needs of the streaming use cases, where constant output of resulting data is fed to the visualisation, which in turn gets continuously updated to depict the incoming information.

The following images depict the visualisations for each of these use cases:

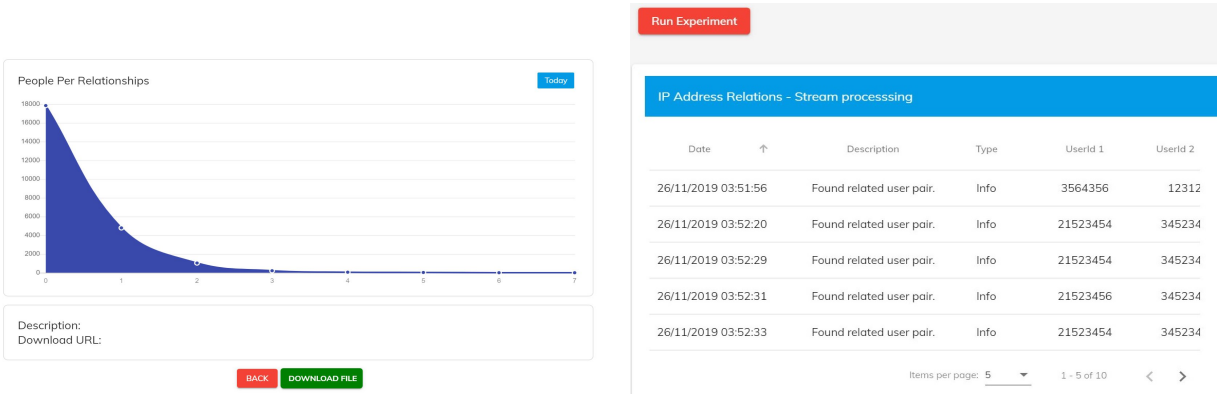


Figure 14. Visualisation: CAIXA "Analysis of relationships through IP addresses" use case (batch & stream)

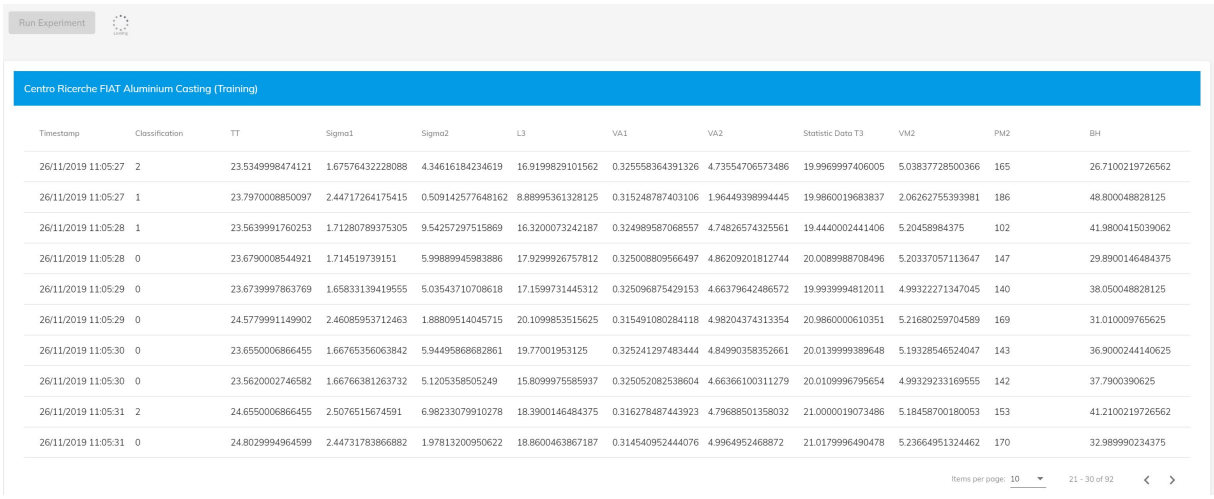


Figure 15. Visualisation: CRF "Production process of aluminum die-casting" use case (stream)

Figure 16. Visualisation: TID QoS in CC (stream)

Taking up on the user feedback on the 1st prototype and according to the feedback received by the reviewers during the 1st review of the project (M18), the future steps of the Multipurpose Interface and interactive visualization framework include a number of enhancements to better support users during their work with the platform tools. So, towards better distinguishing the functionalities addressed to expert or less expert users and in order to improve current information presented to the users, we will explore the implementation of the following updates:

- Separation of entry pages for different groups of users.
- Helping hints for the various offered elements, *e.g.*, page guides, help popups, *etc*.
- Data preview facilities to let users see a chunk of their data and verify that the system properly reads them.
- Provision of experiment execution statistics like *e.g.*, elapsed time.
- Integration of new visualization elements to support new co-develop mode projects and enhance existing ones.

Especially referring to the last item of the list with the envisioned updates, preliminary work has already been done with respect to improvements on CAIXA's "Analysis of relationships through IP addresses" batch use case and CRF's "Production process of aluminum die-casting" streaming case. These include a graph representation of the dataset used in the first case, where nodes represent the users and edges represent the relationships between them. The following figure presents an overall view of the graph and a user-based graph, where one can see the connected users with a selected one.

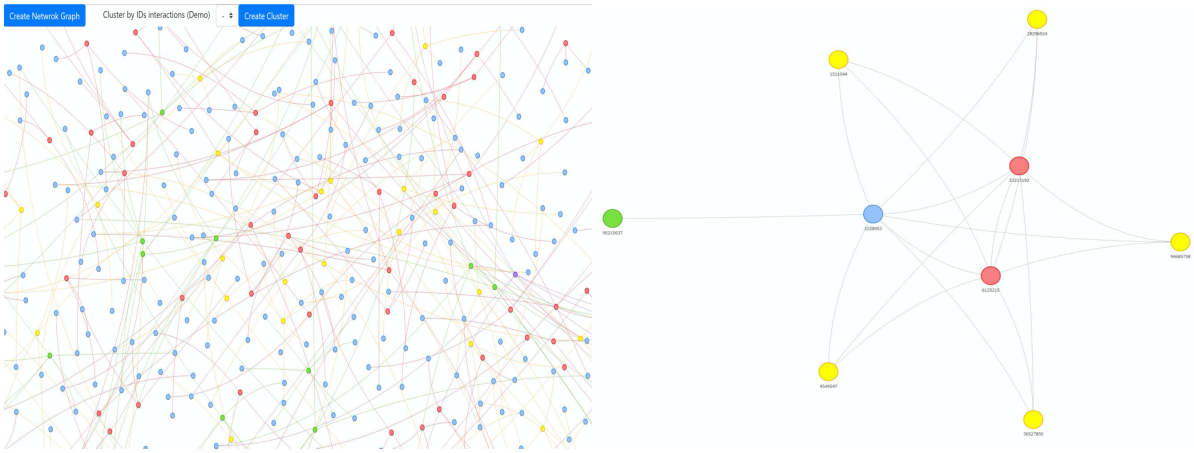


Figure 17. Visualization: graph representation of CAIXA "Analysis of relationships through IP addresses" use case

For CRF's streaming use case, the data-table being used in the 1st prototype was updated to support color-coded presentation of the results in the table according to a given dimension of data (classification in this case). Furthermore, dynamic diagrams showing the incoming streaming data in real-time, as well as aggregations of them that are constantly updated, have been added, as seen in the figures below.

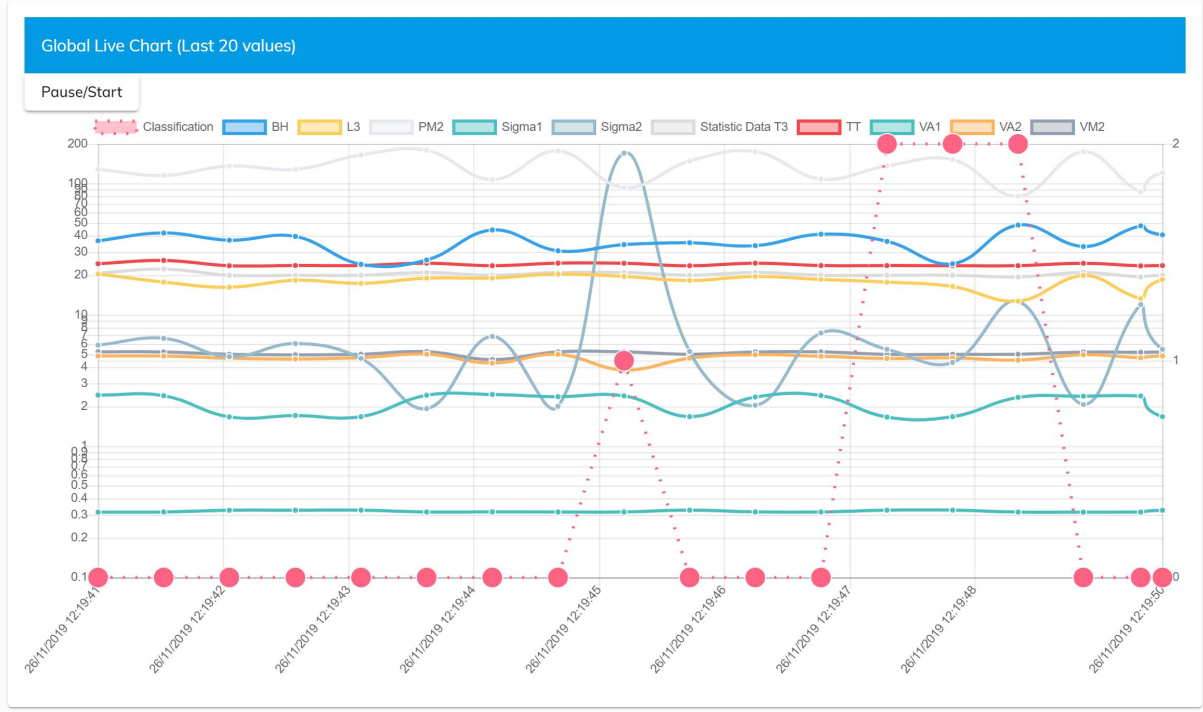


Figure 18. Visualization: Dynamic Multi-line chart

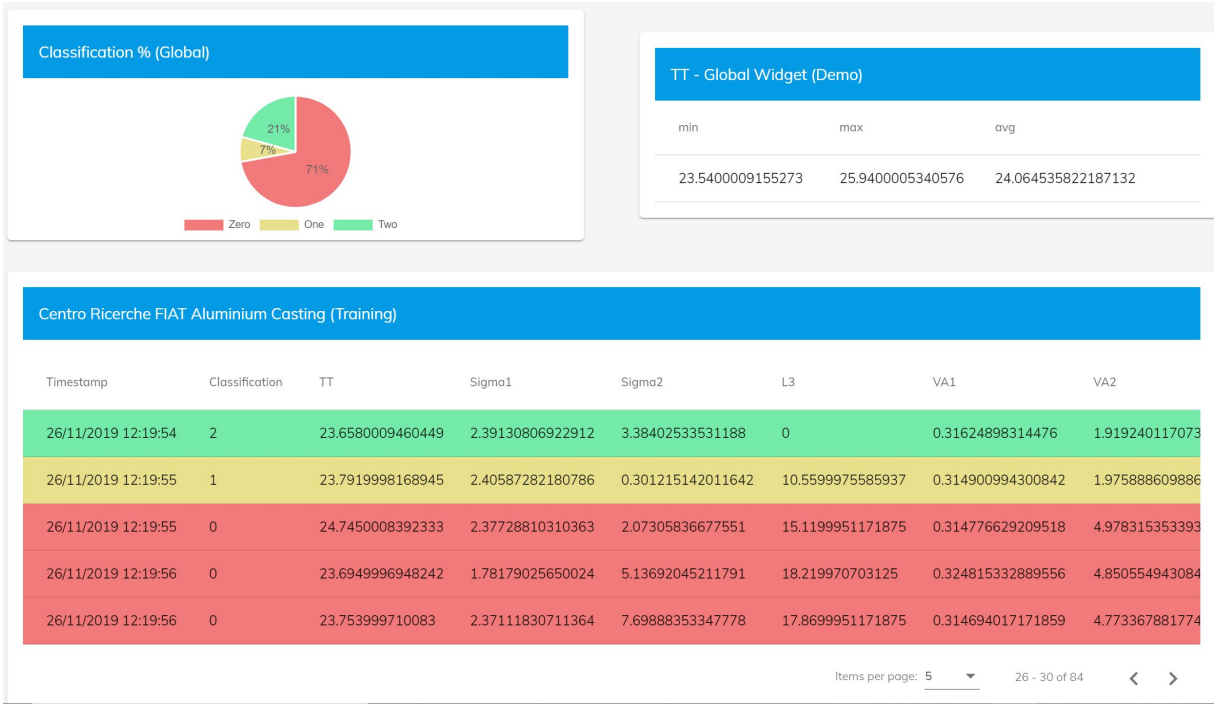


Figure 19. Visualization: Color-Coded data and cumulative statistics

2.3.5 Orchestration management in 1st integrated version

The Orchestrator in the 1st integrated version of the I-BiDaaS platform was created to fill the gap in the original architecture for orchestrating the I-BiDaaS elements; it operates as a middleware between the UI and the rest of the infrastructure (see also Section 2.2 for the 1st integrated version's architecture).

The orchestrator is a REST API with DB support that provides:

- **Scalability**: nodes/core number, RAM selection per experiment
- **Security/isolation**: Every experiment runs in docker containers that are disposed after experiment completion
- **Self-service-ness**:
  - Users do not need to maintain a COMPSs environment in the machines
  - Users can experiment on ready-to-use docker images with templates/examples
- **CRM Features**
  - User provisioning
  - Authentication, Authorization, Accounting
  - History (Custom project management, Experiment history)

The REST API is implemented in PHP Laravel[10], supported by Mysql RDBMS[11]. Its functionality in terms of the available endpoints is provided in Appendix 1.

[10] https://laravel.com/

[11] https://www.mysql.com/

The Orchestrator provides integration between the UI and the docker swarm executing the requested experiments. The sequence diagram for running a new Experiment is depicted in Figure 20.

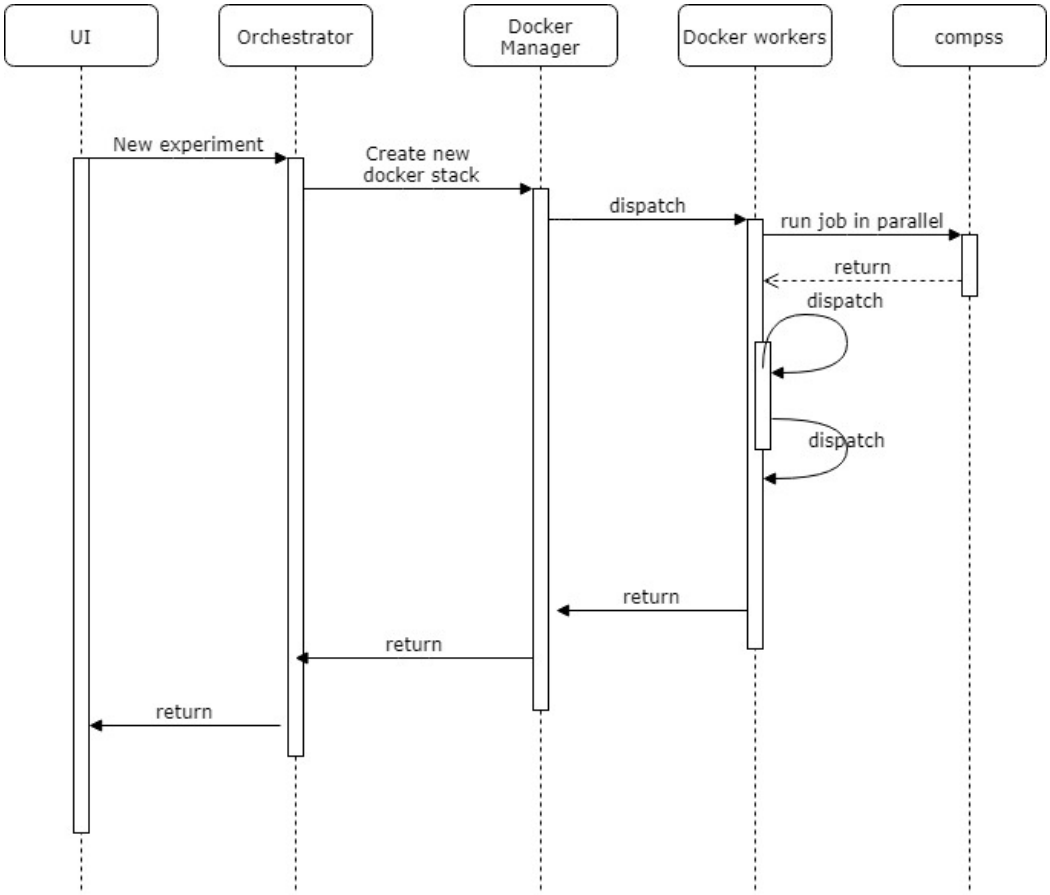


Figure 20. The Generic use case process sequence diagram

The relational model of the Mysql database supporting the REST API is depicted in Figure 21. The projects that each user can initiate have a specific code (an implementation of an I-BiDaaS use case) or a generic algorithm implementation, *e.g.*, K-means. In Custom projects, users can upload and run their own code (expert mode).

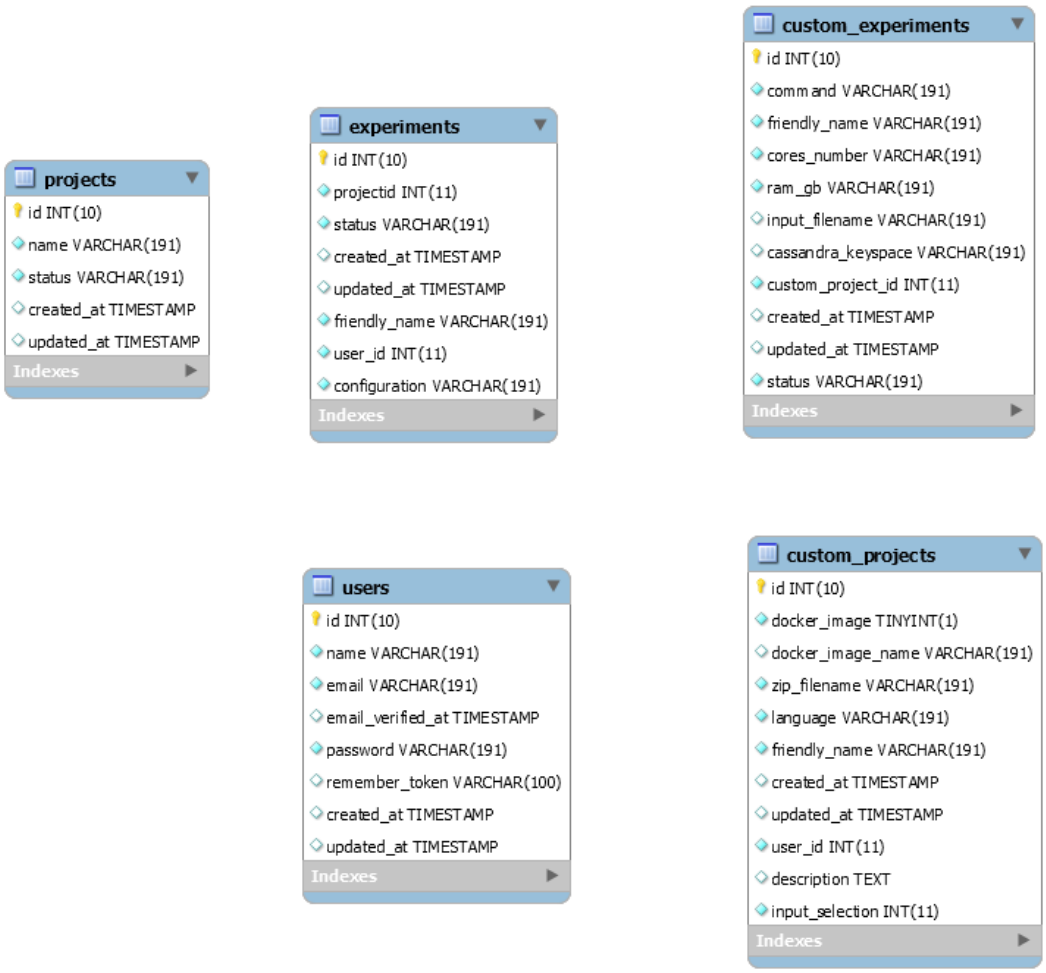


Figure 21. The I-BiDaaS generic use case database structure

The next steps with respect to this deployment can be summarized as follows:

- Integration with ATOS Resource Management and Orchestration (RMO) module.
  - Automated provisioning of nodes in the Swarm based on load
  - Monitoring of node utilization (accounting)
- Integration of the rest I-BiDaaS use cases
- Packaging of the solution, so it can be easily installed by a 3rd party

3 Report on the integrated version per Use Case

3.1 Analysis of relationships through IP addresses (CAIXA)

The "Analysis of relationships through IP addresses" use case was completed and integrated in the 1st version of the I-BiDaaS platform. This means that the user can use the I-BiDaaS platform to analyse a dataset of IP connections of clients to online banking services and extract some relationships between the users. Establishing this or any strong relationship between users (that allows to assure both users know each other) is very relevant to CAIXA in order to use the list of relationships to automatically validate future transactions between those related users, labelling them as "low fraud possibility".

Following the steps that the user should do to perform the analysis, there are three different services in the self-service section of the I-BiDaaS platform related to this use case (Figure 22). Entering in two of them, the user will be able to perform the batch analysis of synthetic or tokenized dataset of user connections in order to extract the relationships. The third allows the user to analyse a stream of bank transfers and detect in which ones the sender and receiver has an existing relationship (based on the previous batch processing of the other services).

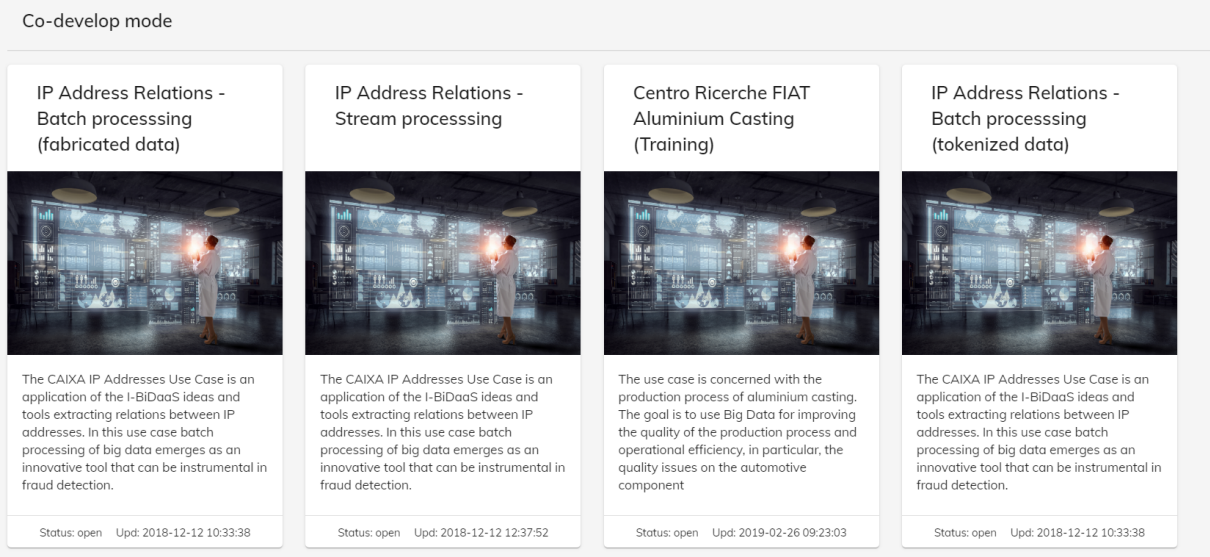


Figure 22. List of self-service available use cases (CAIXA batch and streaming use cases for "Analysis of relationships through IP addresses")

Once entering one of the batch processing services, the user is able to configure the project (Figure 23). In this case, as they are pre-prepared services of the self-service mode, most of the project details are predefined, allowing the user just to proceed with the deployment of a new experiment.

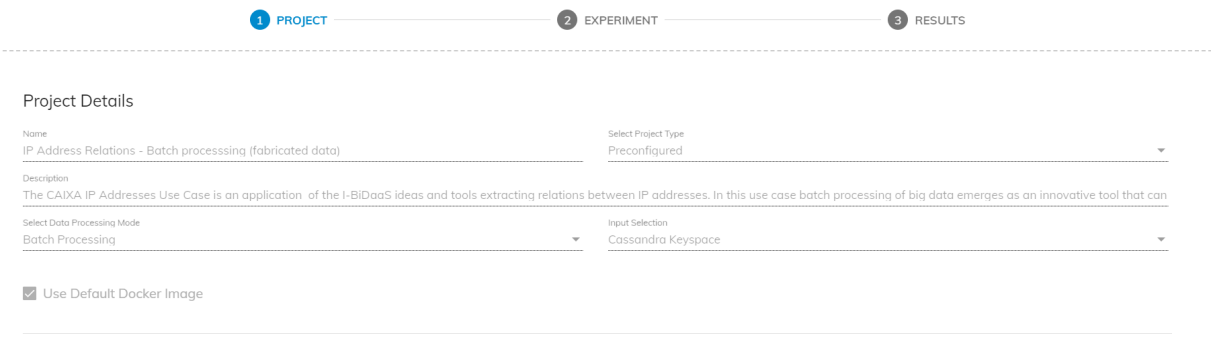


Figure 23. "Analysis of relationships through IP addresses" use case project's configuration window

In the configuration of a new experiment the user can specify a name for it, select the data source and also the computational resources to be used for its processing (Figure 24).

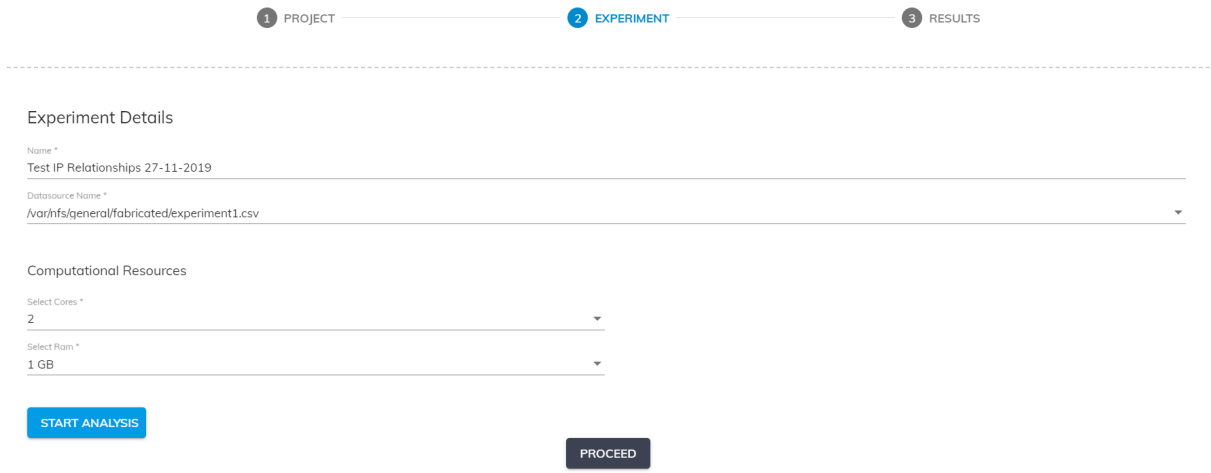


Figure 24. "Analysis of relationships through IP addresses" use case's experiment configuration window

Once the experiment is processed, the results are presented, showing a simple graphic for the fast-visual identification of the amount of people that has relationships and the number of relationships they have (Figure 25). Apart from this graphic, the user can click the "Download File" button in order to obtain the results, in the form of a .csv file with all 1-to-1 relationships that were found in the dataset (

Figure ***26***).

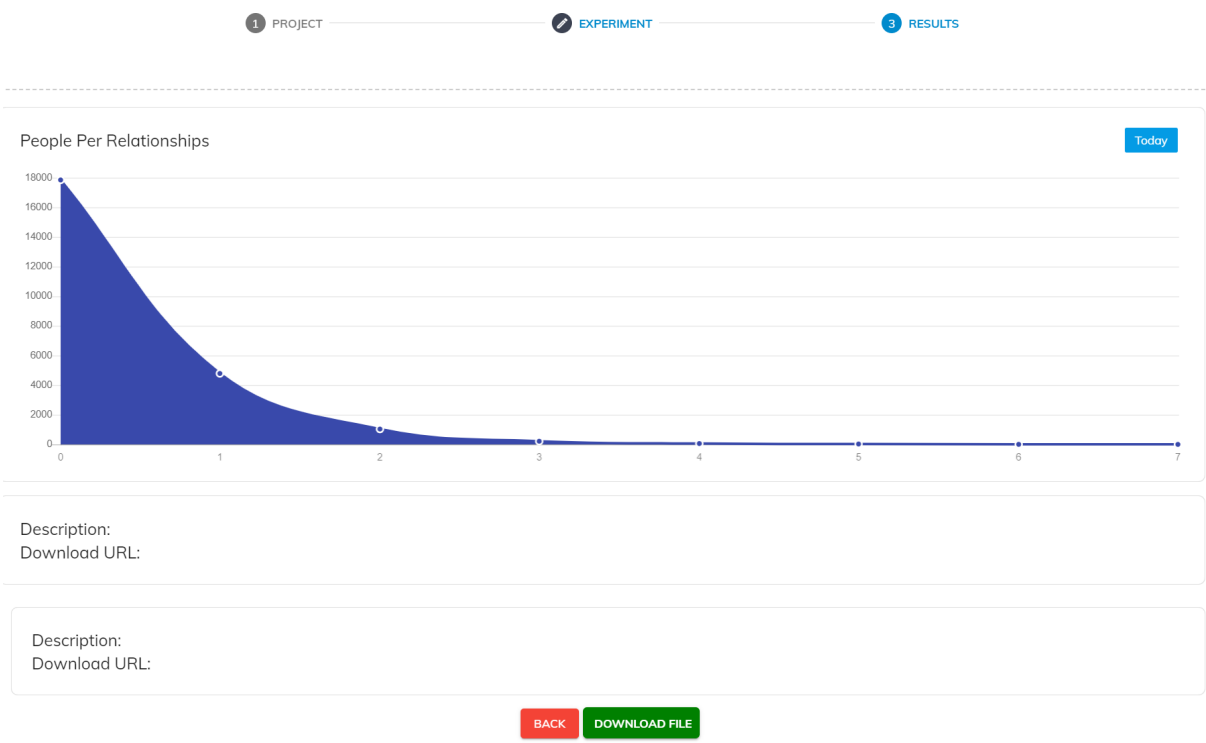
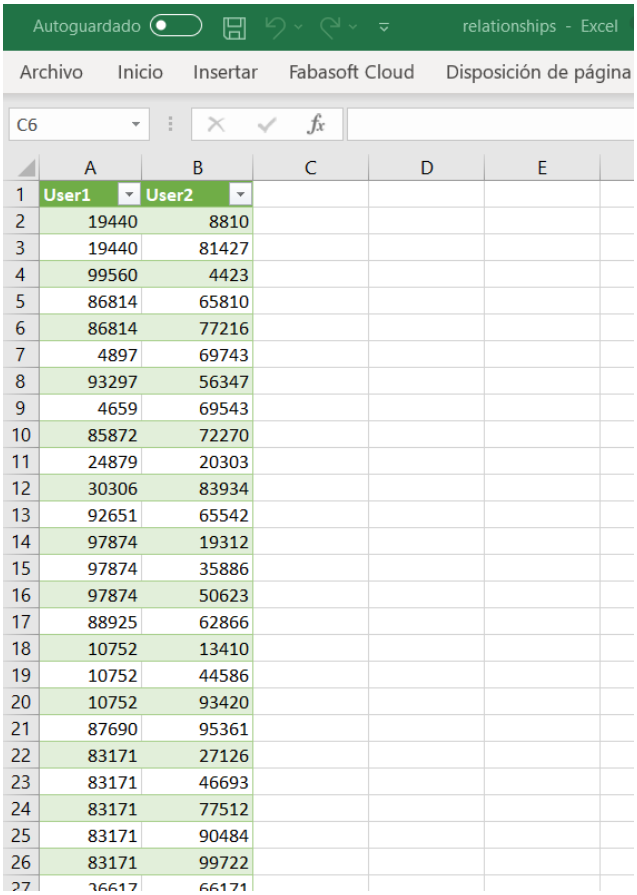


Figure 25. "Analysis of relationships through IP addresses" use case's results window



	A	B	C	D	E
1	User1	User2			
2	19440	8810			
3	19440	81427			
4	99560	4423			
5	86814	65810			
6	86814	77216			
7	4897	69743			
8	93297	56347			
9	4659	69543			
10	85872	72270			
11	24879	20303			
12	30306	83934			
13	92651	65542			
14	97874	19312			
15	97874	35886			
16	97874	50623			
17	88925	62866			
18	10752	13410			
19	10752	44586			
20	10752	93420			
21	87690	95361			
22	83171	27126			
23	83171	46693			
24	83171	77512			
25	83171	90484			
26	83171	99722			
27	26617	66171			

Figure 26. File of exported results from "Analysis of relationships through IP addresses" use case's experiment

Once the batch analysis is done, the user can go back to the main I-BiDaaS menu and select the Stream Processing service. Entering this service, it will directly start analyzing a streaming source of bank transfers and it will show to the user those transfers that are made between the already connected users (Figure 27). The stream source, in this case, is emulated by a program that randomly creates bank transfers and publishes them as messages in the UM.

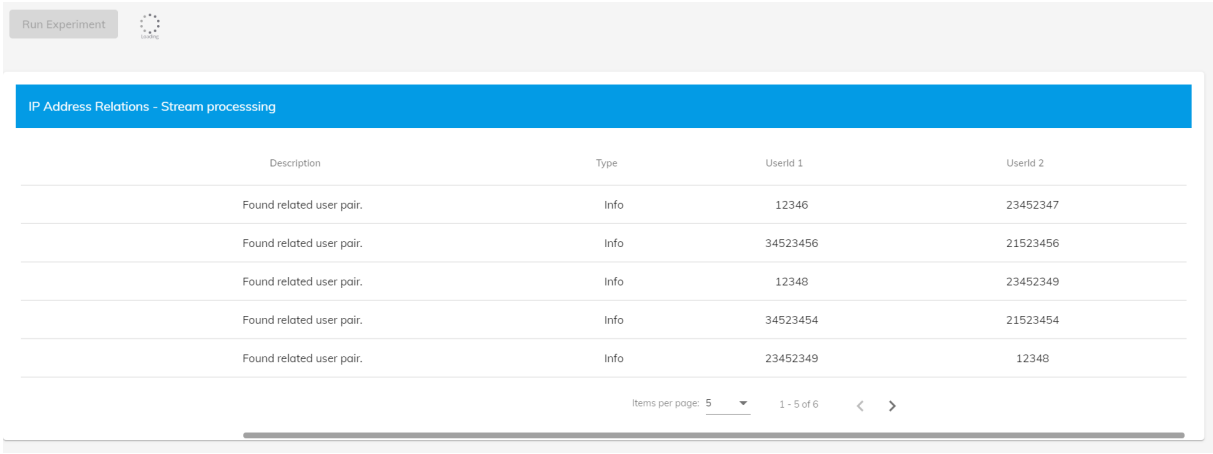


Figure 27. Results from stream processing testing phase of "Analysis of relationships through IP addresses" use case

This use case is using the PyCOMPS implementation of the relationship algorithm that was specifically defined by CAIXA, the TDF for the generation of the synthetic data, Apama for the streaming analytics and the UM for intercommunication between the different components. Every tool is well integrated in the I-BiDaaS platform and the visualization in terms of user experience was evaluated positively by CAIXA users who interacted with it.

However, potential improvements of it were identified, such as the integration of the three services in one single multi-step service, in which the user can select if the data will be synthetically generated or not, and the streaming service is directly deployed after the results page. The possibility of selecting a specific user and visualizing the relationships graph of that user was also an enhancement that was considered for the fast visual analysis undertaken by the members of CAIXA Security Operator Center.

Some additional evaluation in the framework of the use case is still being undertaken on the comparison of the implementation with synthetic and real tokenized data, and how can the results of the real data analysis be used in order to improve the similarity of the synthetic data. And finally, the quality of the data will also be analyzed, taking into account its reliability to be used strictly for testing the scalability and performance of the I-BiDaaS tools (not considering data analytics purposes).

3.2 Advanced analysis of bank transfer payment in financial terminal (CAIXA)

A first version of the "Advanced analysis of bank transfer payment in financial terminal" was also integrated in the 1st version of the I-BiDaaS prototype, allowing the main expected functionalities of the use case. The user can analyse anomalies in bank transfers that are made by employees under the name of a customer (when the customer goes to a bank office and orders it, properly identifying themselves). The I-BiDaaS platform allows in a fast and simple way the analysis of the provided dataset with pre-defined clustering algorithms that enable viewing the information in a 3D graph for a fast visual identification and selection of the bank transfer that presents anomalies.

In the main menu of the I-BiDaaS platform, a "Caixa Advanced Bank Transfer" service was created in the "Expert mode" for the analysis of this use case (Figure 28).

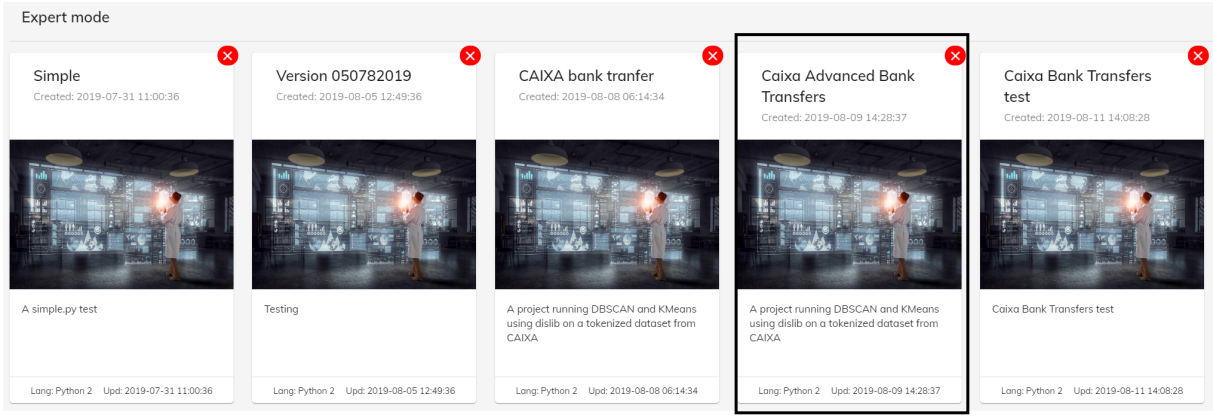


Figure 28. Selection of "Advanced analysis of Bank Transfer payment in financial terminal" use case

Following the same structure as in the other use cases, when selecting the service, the user accesses first the project configuration. In this case, as a custom project, more options can be selected (Figure 29). However, this is because it is still in an exploration and configuration phase. At the end, this service will be pre-defined and located in the self-service mode, like the other industrial use cases developed in the project. In this window, the user can select to generate a new experiment by pressing the "Add Experiment" button.

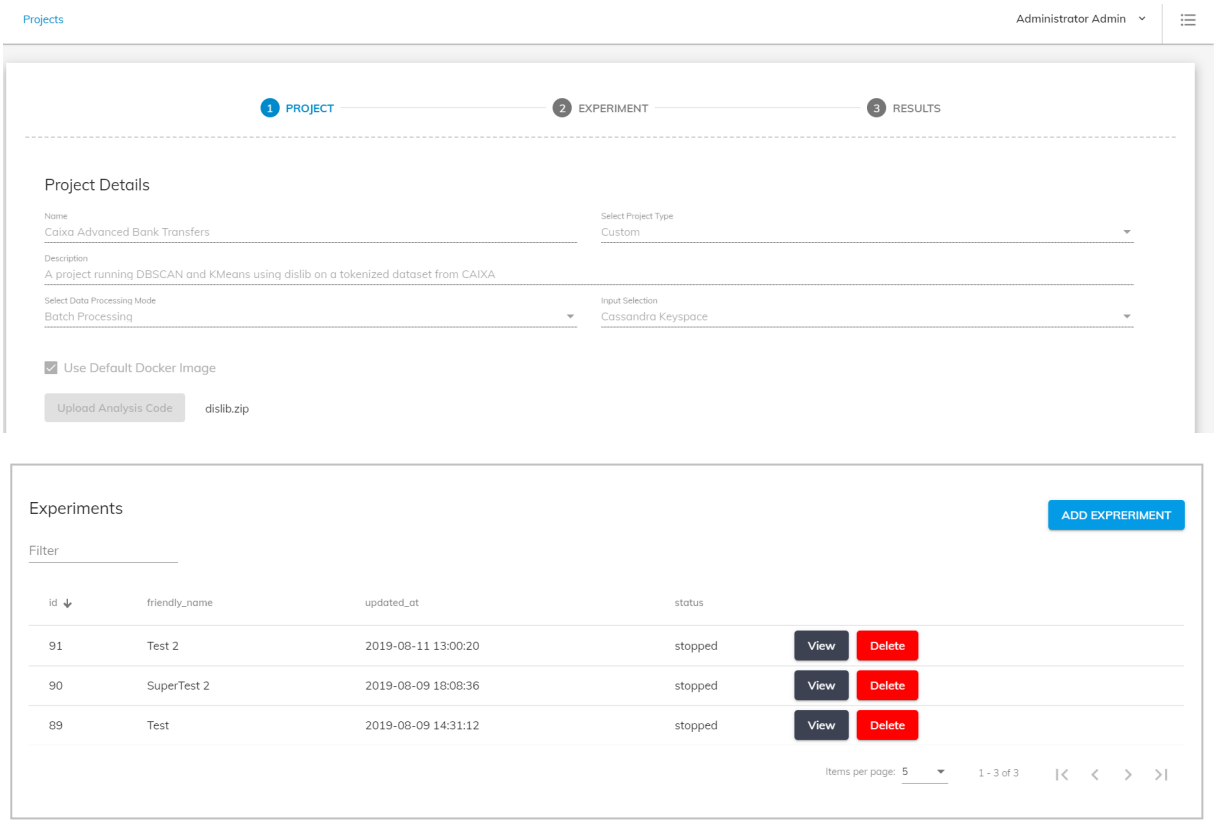


Figure 29. "Advanced analysis of Bank Transfer payment in financial terminal" project's configuration window

This brings the user to the experiment configuration window, which is very similar to the analogous window in the "Relationships through IP address" use case, but allows for some

specific commands (Figure 30). This is because this service was created as an expert mode service, allowing a higher level of customization. In any case, the default parameters for the analysis are already defined when creating the new experiment and this is also a feature that will be directly integrated in the following versions, when migrating it to self-service section.

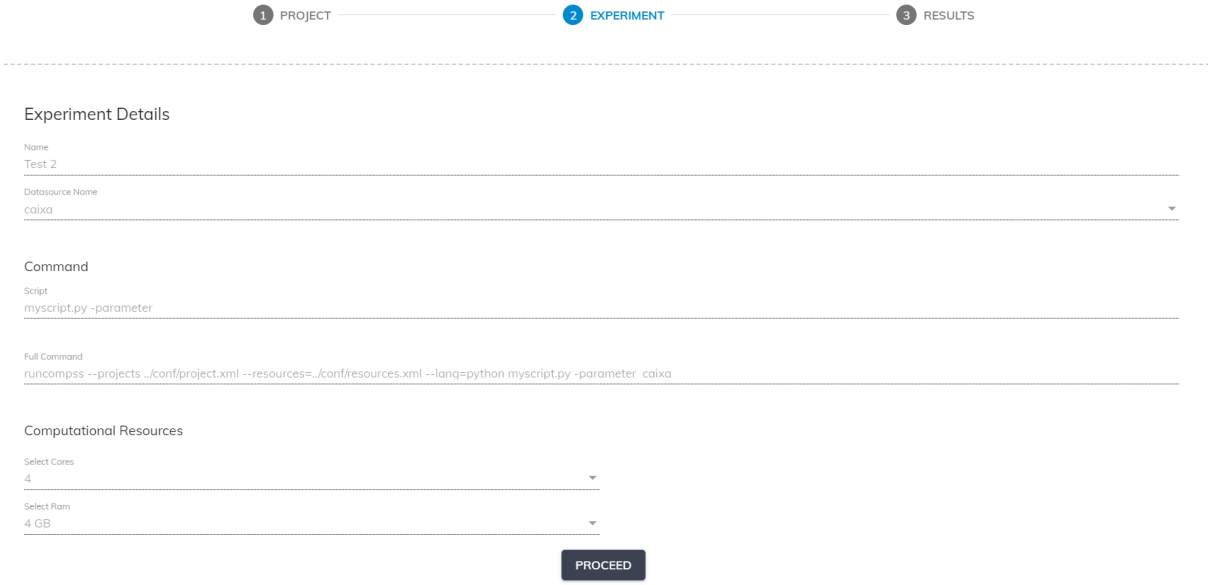


Figure 30. "Advanced analysis of Bank Transfer payment in financial terminal" experiment's configuration window

When proceeding with the experiment, the I-BiDaaS platform shows the results window, which in this case only allows the possibility to download the results file. This results file includes the set of anomalies detected (Figure 31).

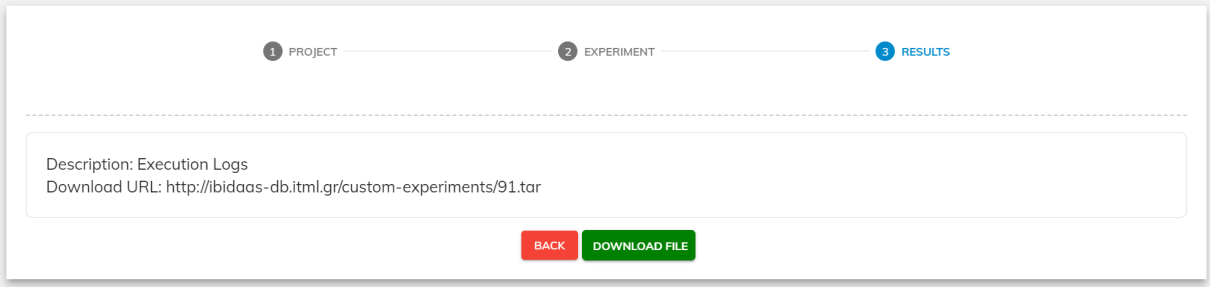


Figure 31. "Advanced analysis of Bank Transfer payment in financial terminal" results' window

However, if the user wants to go more into depth in the analysis of the dataset and the potential anomalies found, an additional custom web interface (Figure 32) was built to visualize the dataset in a 3-D graph (by means of space reduction of the dataset). It allows the user a fast visual analysis of the data and the anomalies, a custom selection of them and the extraction of the specific selected samples (Figure 33).

Figure 32. "Advanced analysis of Bank Transfer payment in financial terminal" anomalies visualisation interface

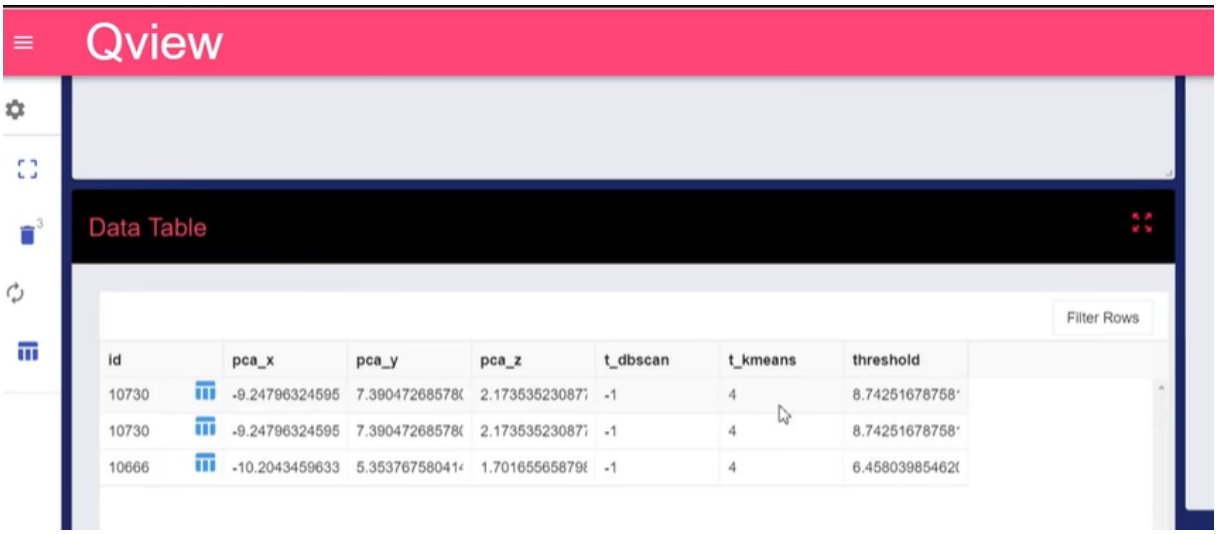


Figure 33. "Advanced analysis of Bank Transfer payment in financial terminal" anomalies extraction

In this use case, several tools and algorithms were integrated in the I-BiDaaS platform: the use case is created as a custom project in the I-BiDaas Platform, the ML algorithm (K-means) was implemented by UNSPMF and the custom visualization by BSC.

Following steps of this use case encompass the comparison of the anomalies extracted with the I-BiDaaS platform towards the ones extracted from the same dataset using the DataRobot[12] commercial product, as well as the enhancement of the user experience with this service with a better integration of the visualization tool with the whole I-BiDaaS platform.

[12] https://www.datarobot.com/

3.3 Production process of aluminum Die-casting (CRF)

The main goal for the CRF "Aluminum die casting processing" use-case is the improvement of the quality of the manufacturing process of engine crankcases. In order to reach this objective, during the first phase of the project, the focus has been on the analysis of the real anonymized production data, by investigating the possible correlations between process parameters and product quality levels.

As reported in "D6.3: Evaluation final report", CRF has provided real data related to the manufacturing of crankcases in the aluminum die casting process. A TDF project has been created on the TDF dedicated VM for the evaluation process. The generation of a synthetic dataset has been performed using the IBM TDF tool, according to the definition reported in "D2.1:Data assets and formats". A python script was used to calculate the statistical properties (max value, min value, mean value, variance value and discrete quantile distribution of the values) and verified that statistical variations are minor in real than in fabricated data.

In the framework of the hackathon held in June 2019, the I-BiDaaS partners produced a random forest model with 79% accuracy and a neural network model on the same data with 68% accuracy. Moreover, a Logistic regression – stochastic gradient method was developed in COMPSs, achieving about 60% of classification accuracy. The algorithm was tested for scalability and accuracy and scales well when the data set is large enough to show scaling properties. Classification accuracy was compared with Scikit-Learn[13] on several publicly available moderate data sets on a KNN implementation and similar accuracy is obtained (see Tables 5-7 in I-BiDaaS D3.1).

For tests, SAG's Universal Messaging (UM) and SAG's APAMA have been used, as described in previous Sections 2.3.1 and 2.3.3. As sample data, a 618 Byte long JSON string was used as the MQTT payload which corresponds to the CRF use case used for the review prototype. The analysis was displayed online using the AEGIS visualization tool, as described in Section 3.3.4.

The I-BiDaaS platform has been used in the Co-develop mode and the following images depict the visualization for the Aluminum die casting use case.

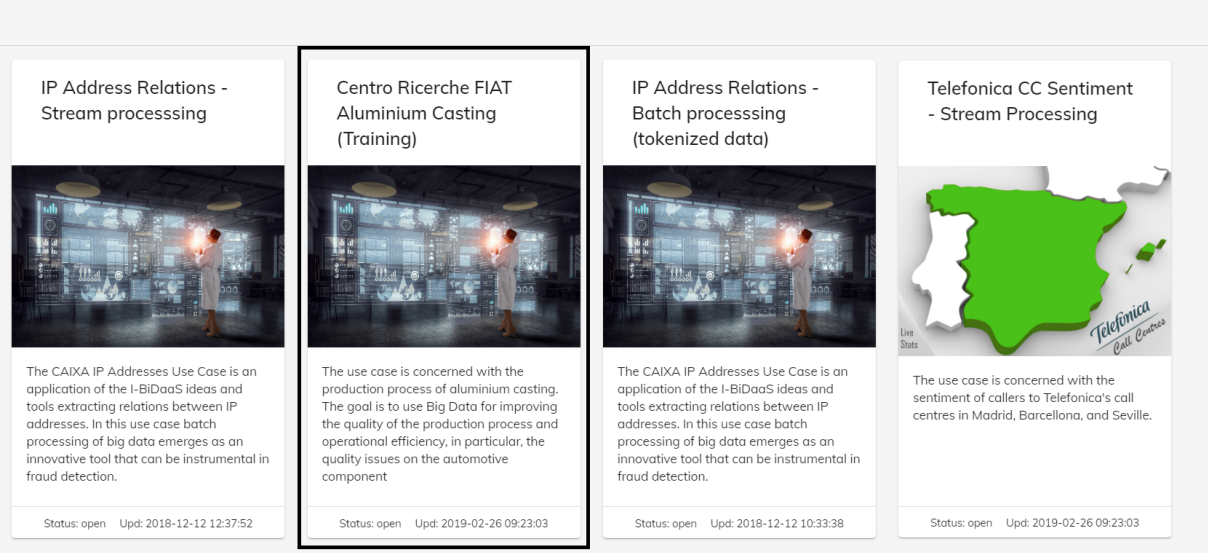


Figure 34. Co-develop mode for CRF "Production process of aluminum Die-casting" use case (Training)'

[13] https://scikit-learn.org/stable/

In Figure 35, the initial page is shown, in which it is possible to choose how many items per page will be able to be displayed and press the top-left button to select "Run Experiment".

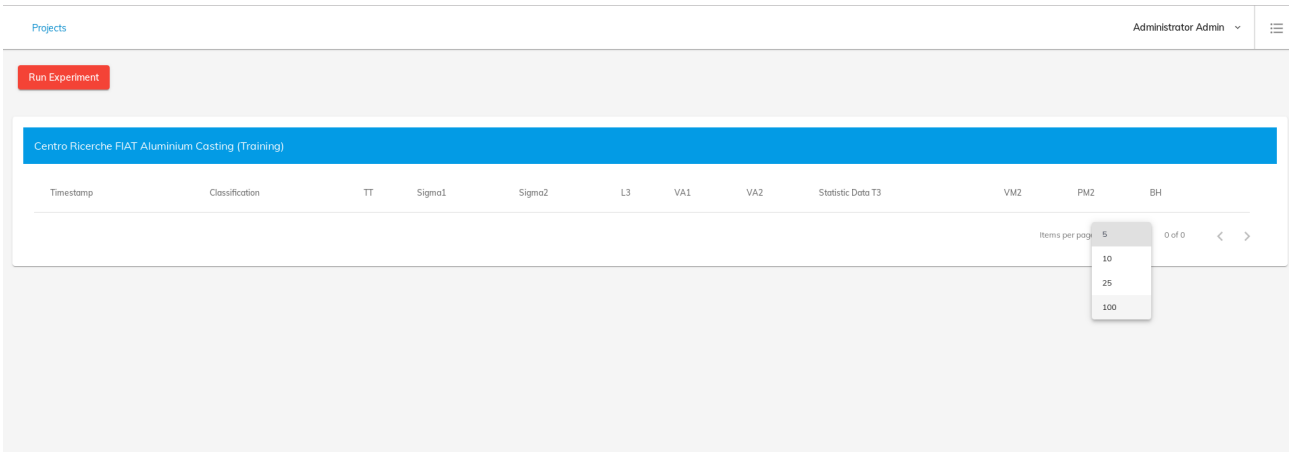
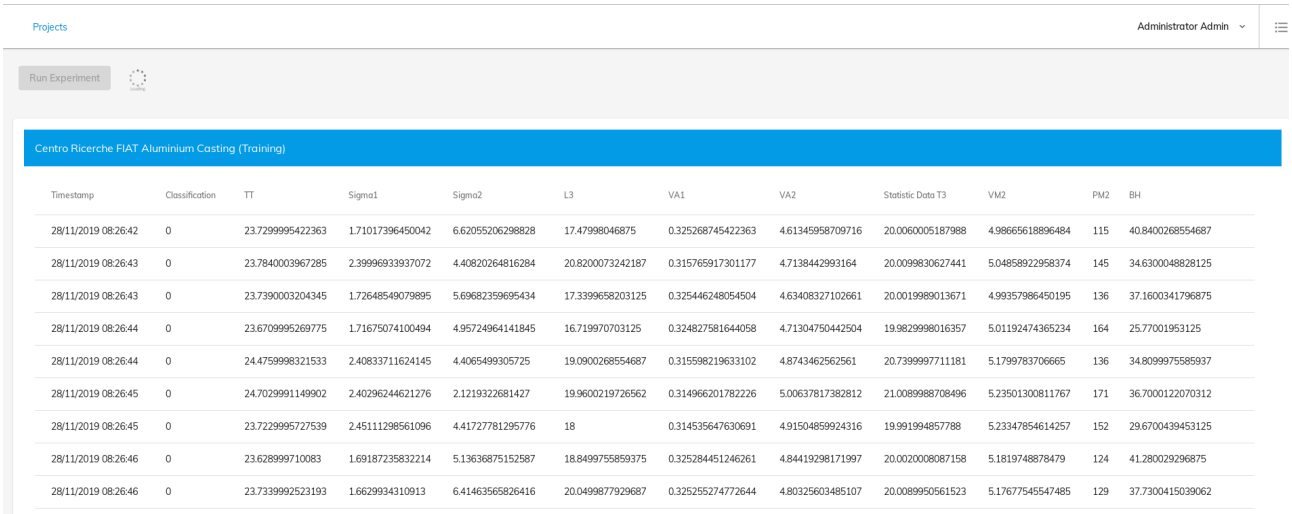


Figure 35. Initial page for the CRF "Production process of aluminum Die-casting" use case

Figure 36 shows in the second column, the classification level obtained for each row of data.



Timestamp	Classification	TT	Sigma1	Sigma2	L3	VA1	VA2	Statistic Data T3	VM2	PM2	BH
28/11/2019 08:26:42	0	23.7299995422363	1.71017396450042	6.62055206298828	17.47998046875	0.325268745422363	4.61345958709716	20.0060005187988	4.98665618896484	115	40.8400268554687
28/11/2019 08:26:43	0	23.7840003967285	2.39996933937072	4.40820264816284	20.8200073242187	0.315765917301177	4.7138442993164	20.0099830627441	5.04858922958374	145	34.6300048828125
28/11/2019 08:26:43	0	23.7390003204345	1.72648549079895	5.69682359695434	17.3399658203125	0.325446248054504	4.63408327102661	20.0019989013671	4.99357986450195	136	37.1600341796875
28/11/2019 08:26:44	0	23.6709995269775	1.71675074100494	4.95724964141845	16.719970703125	0.324827581644058	4.71304750442504	19.9829998016357	5.01192474365234	164	25.77001953125
28/11/2019 08:26:44	0	24.4759998321533	2.40833711624145	4.4065499305725	19.0900268554687	0.315598219633102	4.8743462562561	20.7399997711181	5.1799783706665	136	34.8099975585937
28/11/2019 08:26:45	0	24.7029991149902	2.40296244621276	2.1219322681427	19.9600219726562	0.314966201782226	5.00637817382812	21.0089988708496	5.23501300811767	171	36.7000122070312
28/11/2019 08:26:45	0	23.7229995727539	2.45111298561096	4.41727781295776	18	0.314535647630691	4.91504859924316	19.991994857788	5.23347854614257	152	29.6700439453125
28/11/2019 08:26:46	0	23.628999710083	1.69187235832214	5.13636875152587	18.8499755859375	0.325284451246261	4.84419298171997	20.0020008087158	5.1819748878479	124	41.280029296875
28/11/2019 08:26:46	0	23.7339992523193	1.6629934310913	6.41463565826416	20.0499877929687	0.325255274772644	4.80325603485107	20.0089950561523	5.176775455474485	129	37.7300415039062

Figure 36. Visualisation of the streaming of the CRF "Production process of aluminum Die-casting" use case

CRF has provided, during the last plenary meeting in Palma de Mallorca (M23), new real production data collected from March to September 2019. Moreover, as a next step, CRF will also provide other data about process temperature for further analysis. Images of thermal cameras, as depicted in the picture below, will be shared in order to check if there is a correlation between temperature and the other parameters in relation with the quality levels.

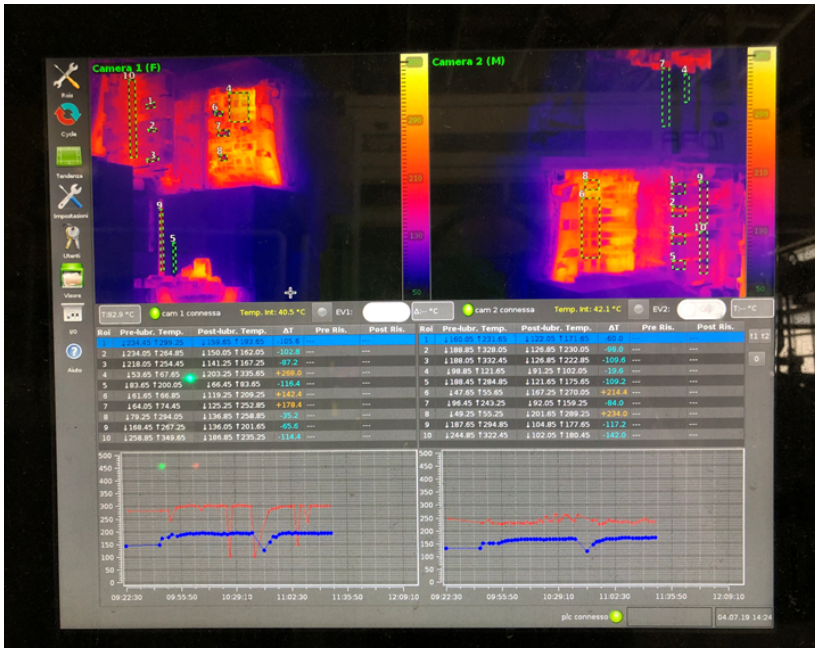


Figure 37. Frame by thermal camera

Another objective that CRF aims to reach, even in the long-term, is to have a real-time analysis of the production data, allowing to have in advance the information about the final quality of the produced crankcases, before they continue their manufacturing process. This would be possible thanks to the execution of reliable predictive algorithms that would help in reducing the risk of making additional operations for repair or for executing unnecessary operations, if the crankcase is later discarded.

The architecture for the data sharing related to this use case is represented in the image below and is described in detail in Deliverable D4.2: Universal Messaging Bus (Interim Version) [3].

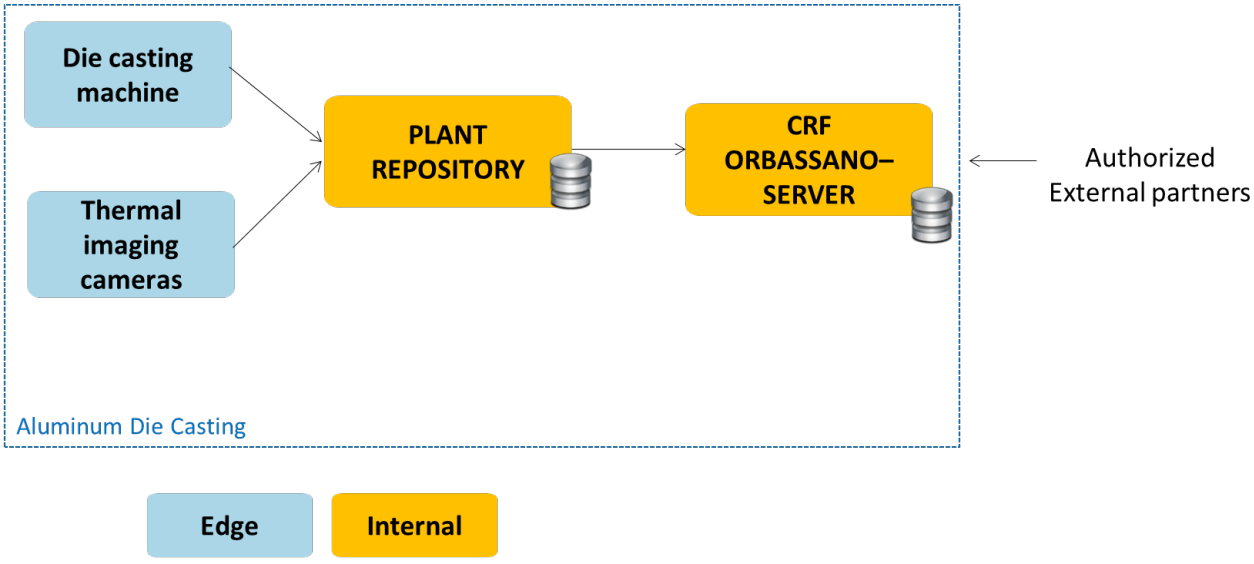


Figure 38. Picture of the architecture for the CRF "Production process of aluminum Die-casting" use case

For this use case, UNSPMF provided several models for both batch and streaming purposes. These models were initially developed at the Campus Melfi Hackathon (M17). The models were trained on the provided CRF's dataset and included a Random Forest (RF) model and a

Deep Neural Network model. Both models performed similarly in terms of accuracy, with the Random Forest model being slightly faster with respect to inference.

The integration of the Random Forest classification model for the Aluminum casting use case was three-fold. Firstly, it was integrated with the Universal Messaging component from SAG to provide real-time analytics based on our previous knowledge. The idea behind this integration was to provide real-time information to the operators about the engine being built. The Random Forest model was not the online learning type, but this modification would be trivial to carry out. We also integrated the Deep Neural Network model with Universal Messaging, but since it performed similarly as the Random Forest classifier, we decided to focus on the faster variant – the RF model. The second aspect of integration, which was considered for future releases, is with respect to IBM's TDF data fabrication platform.

Since TDF currently cannot be used for modelling of labels (classification results), we can integrate our Random Forest model to work with the data fabrication worker. This way, when all the synthetic data is generated (by defining constraints in the TDF platform), we could generate labels using our trained model. Lastly, both models were integrated in the CRF's use-case part of the main I-BiDaaS component, where they can work both in batch mode and streaming mode.

3.4 Quality of Service in Call Centers (TID)

A demo version of the TID QoS in Call Center (CC) use case has been included in the Co-develop mode of the I-BiDaaS Visualization Toolbox. The main goals of this use case are to improve CC customer experience, assess and increase centers' performance (calls/s), screen phone calls and discover problems in attention of the agent.

The current status in TID Call Centers is that there is no automated solution in place; customer audio calls are processed by human agents, which is a costly and time-consuming solution. Given an average call duration of 8.6 minutes, a human agent following a work schedule of 40 hours per week (160 hours per month), could help process up to 11,520 calls per year. This manual process allows to flag ~2,300 low customer satisfaction calls that require (further & immediate) attention and may lead to customer churn.

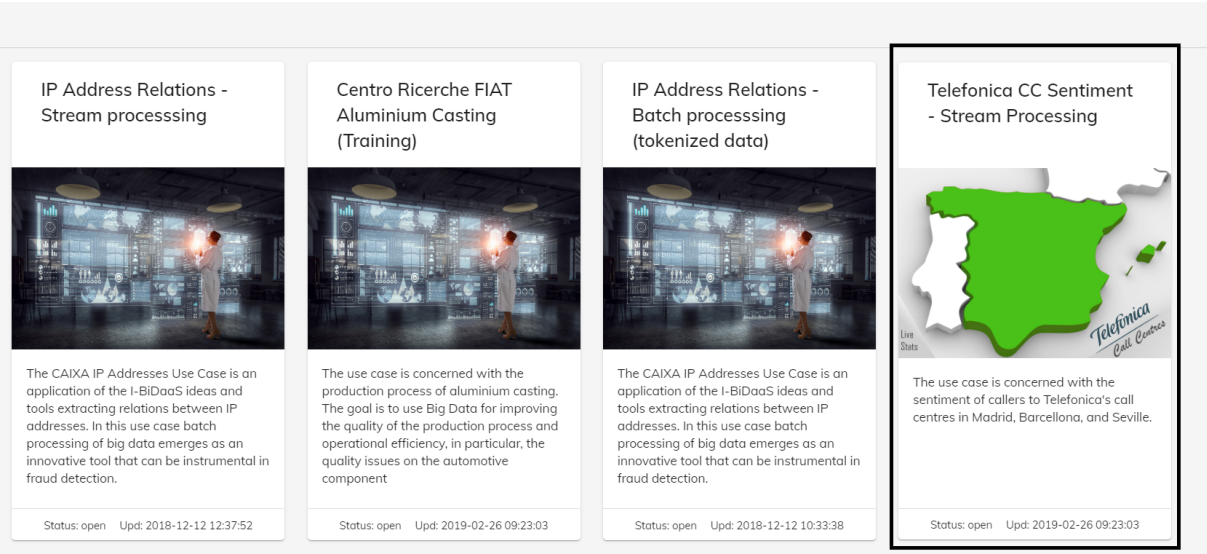


Figure 39. Co-develop mode for 'TID Quality of Service in Call Centers" use case

For the creation of the dataset to be used in the TID CC use case, TID analyzed the significance of various acoustic features extracted from customer-agents' spoken interaction in predicting self-reported satisfaction and investigated whether speech prosodic features can deliver complementary information to speech transcriptions provided by an ASR (Automatic Speech Recognition) solution developed by TID.

The demonstrator (depicted in Figure 40) simulates the aggregation of data streams originating from three different Call Centers in Spain. After processing each call in a Call Center and aggregating the results, the information produced in the demonstrator consists of the calculated satisfaction index of the customer, the number of dropped calls, the average wait time and average resolution time.

Figure 40. Screenshot of the TID "Quality of Service in Call Centers demonstrator" use case

The current version of this use case uses randomly generated data, produced within the Visualization Toolkit for demonstration purposes. The next step is to fully integrate the use case in the I-BiDaaS platform. Towards that end, TID explored the possibility of using a deep neural architecture to perform early feature fusion on both prosodic and linguistic information. Convolutional Neural Networks (CNN) were trained on a combination of word embedding and acoustic features for the binary classification task of "low" and "high" satisfaction prediction. The initial version of the satisfaction prediction software was developed in python using skicit-learn and was "dockerized" in order to allow easy integration with the existing I-BiDaaS platform. The goal is to create a streaming use case where phone call data belonging to different CCs are being fed to the streaming analytics engine of the platform (APAMA) and real-time satisfaction level is calculated.

4 Revised testing and Integration Plan

In this section, we are presenting an integrated test plan for WP2 – WP5. Integration and Testing is explicitly decoupled from the "development of the architecture components": That is, development of each SW component (Batch processing, Streaming analytics, Resource management, Visualisation, *etc*.) follows its own methodology. Some of them (especially the ones that are not open-source) can be considered as “black boxes” that will be integrated into the I-BiDaaS platform.

This section is a revised version of the Testing and Integration Plan provided in Deliverable D5.2, M12. More information about the tools and methodologies used can be found in that Deliverable.

4.1 I-BiDaaS Actors and their integration activities

The I-BiDaaS actors, integration activities, together with integrated components are listed in Table 1.

Table 1: I-BiDaaS actors and their integration activities

Actor	Integration Activities	Integrated Components
Foundation for Research and Technology – Hellas (FORTH)	Development of GPU-accelerated Analytics	Streaming Analytics
Barcelona Supercomputing Center (BSC)	Development of Hecuba Tools	Batch Processing Module
IBM Israel – Science and Technology LTD (IBM)	Development of Test Data Fabrication	Data Fabrication Tool
Centro Ricerche FIAT (FCA/CRF)	Pilot leader / Data provider	N/A
Software AG (SAG)	Development of Streaming analytics & Messaging	• APAMA Server • Universal Messaging
Caixabank S.A. (CAIXA)	Pilot leader / Data provider	N/A
University of Manchester (UNIMAN)	Resource Management and Orchestration	N/A
Ecole Nationale des Ponts et Chaussees (ENPC)	N/A	N/A

ATOS Spain S.A. (ATOS)	Resource Management and Orchestration	Resource Management Orchestrator
Aegis IT Research LTD (AEGIS)	Development of Visualization	I-BiDaaS Dashboard
Information Technology for Market Leadership (ITML)	• Development of Orchestrator • Platform Integration • QA	Orchestrator
University of Novi Sad Faculty of Sciences (UNSPMF)	Development of Batch Processing algorithms	Batch Processing Module
Telefonica Investigation y Desarrollo S.A. (TID)	Pilot leader / Use case and Data provider	N/A

4.2 Testing Plan

In this section, we describe a list of different types of tests conducted in the I-BiDaaS Platform: Unit Testing, Universal Messaging Testing, End-to-End Testing and Code Quality Testing. The actual test results for the 1st version of the integrated I-BiDaaS platform are presented in section 4.3.

4.2.1 Unit Testing

Each team is responsible for the unit testing of the component they are developing. Unit testing will verify the functions created to implement the business logic for each component, verifying the results. Although Unit Testing results per component are not in the scope of this document, specifically for the orchestrator module that is responsible for the coordination of the Platform, Unit Testing after the integration of the rest of the modules is more important, as it is equivalent for testing the whole platform (apart from the UI/Visualization Toolkit). For that reason, in Section 4.3, we will specifically present Unit Testing designed and conducted for the Orchestrator module.

4.2.2 UM Testing

The Universal Messaging (UM) handles communication between the major components of the I-BiDaaS platform for the streaming use cases. This simplifies the integration testing up to a point, since even if a component is not ready, the associated message queue receiving messages for that component will be available, thus allowing at least to test the sending of messages. The communication concerning the UM must be tested for three important aspects:

- Each queue/topic must correctly deliver the messages to the correct components.
- Each component must have a worker retrieving the messages from the message queue/topic that is assigned to that component. The use of the message after it is retrieved does not belong to the scope of this part of testing.

- Each component must correctly implement the logic that sends the messages to the correct queues, when attempting to communicate with another component.

This will verify that all messages will be delivered, that each component will be able to receive messages and that each component will be able to send messages.

Once the communication with the UM has been established and tested, the integration testing proceeds with testing the communication between pairs of components (where available) through the UM in order to verify that each request receives the appropriate response. The purpose is not to check the validity of the responses, but to ensure that they trigger the correct chain of messages between the various components, since in many cases a request to one component will be processed through another component, which will deliver the final response.

4.2.3 End to End Testing

The purpose of End-to-End (E2E) testing is to verify the desired functionality across the whole platform, offering insights on the performance and in this case of a multi-component system demonstrating the intended communication between these components.

Successful completion of this test confirms that the I-BiDaaS platform is working as intended both in software and hardware terms. Each test case will remain valid for as long as the tested functionality remains in the application and should be repeated after each update of the application in the future.

E2E tests are performed by the ITML, using the test cases described in the next section. Any case that leads to non-intended results or disruption of the application flow should be documented as detailed as possible, including information about the process that led to it, the conditions and the platform. An attempt to recreate the error should also be made.

4.2.4 Code Quality Testing

Regardless of development methodologies, languages, frameworks or tools, enforcing a high code quality is a way to achieve faster and easier development, testing and maintenance. Code quality is measured with the usage of Sonarqube[14] (Figure 41).

Specifically, the list of tests performed with Sonarqube are:

- Reliability: Discovery of coding errors that will break the code and need to be fixed immediately.
- Security: Automatic discovery of vulnerabilities and discovery of security-sensitive code that requires manual review to assess whether or not a vulnerability exists.
- Maintainability: Automatic discovery of code that is confusing and difficult to maintain (Code Smells).

[14] https://www.sonarqube.org

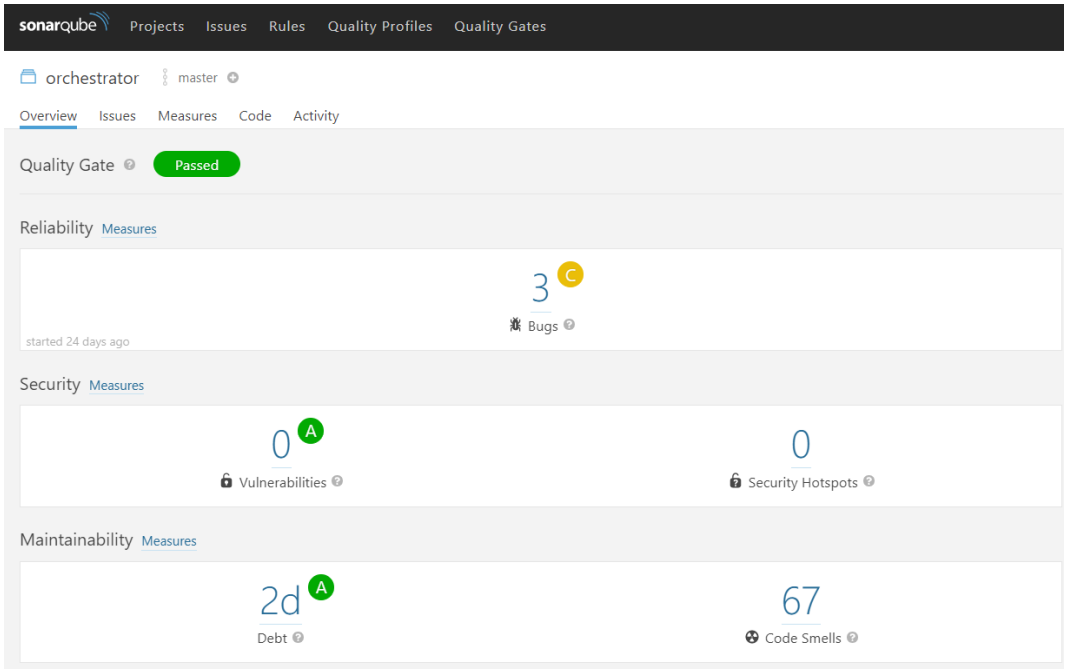


Figure 41. Sonarqube online report for Orchestrator code

4.3 Integration Test Results

The list of integration and quality assurance tests that were performed for the 1st version of the I-BiDaaS Platform is summarized in Table 2. All the tests in Table 2 have already been conducted and have passed.

The list of tests is divided to the following types:

- Orchestrator Unit Testing (OUT): Tests for verifying Orchestrator functionality. Since the orchestrator is a Rest API (described in Section 2.3.5), Unit tests are tests designed for each endpoint of the API. These tests are automated; check is done through validating the API response (in JSON format).
- Use Case Testing (UCT): E2E test for verifying a Use Case specific functionality. These tests correspond to the functionalities of the integrated use cases, presented in Sections 3.1, 3.2 and 3.3. The tester performs a specific task in the UI and verifies its validity by checking (i) the Visualization Toolkit output, (ii) the database of the orchestrator and (iii) the log files and output produced by relevant modules (*e.g.*, docker-compss).
- Generic Use Case Testing (GUCT): E2E tests for verifying the functionalities of the Generic Use Case, as presented in section 12. The tester validates the task in the same way as with the Use Case Testing.
- Universal Messaging Testing (UMT): Tests for verifying intercommunication in the streaming use cases. Testing is done by validating the contents of MQTT messages exchanged.
- Generic Advanced Visualization Toolkit Tests (AVT): Generic tests of the toolkit, such as login, logout functionalities.

Table 2: Integration and quality assurance tests

Test Name	Test Type	Description	Preconditions	Output
Update-experiment	OUT	Set status of experiment to terminated. Used internally when COMPSs terminates	• Specified experiment exists • Specified experiment has status: running	• Specified experiment status is set to "terminated"
Get-projects	OUT	Get the list of projects (predefined projects)	• User is logged in	• A valid array of the predefined projects
Get-project-experiments	OUT	Get the list of project experiments owned by the current user	• User is logged in • Specified project exists	• A valid array of experiments belonging to the specified project • Returned experiments belong to current user
Get-project-details	OUT	Get the details of a project	• User is logged in • Specified project exists	• A valid JSON representation of the specified project
New-experiment	OUT	Start a new experiment for a specified project	• User is logged in • Specified project exists	• A new experiment is created in the DB • A new docker stack is created in docker swarm • The experiment is running in the infrastructure
Get-experiment	OUT	Get the details of experiment, specified by id	• User is logged in • Specified experiment exists • Specified experiment belongs to user	• A valid JSON representation of the specified experiment
Stop-experiment	OUT	Request the termination of an experiment	• User is logged in • Specified experiment exists	• The status of the new experiment is set to terminated

			• Specified experiment belongs to user • Specified experiment has status "Running"	• Docker stack belonging to this experiment is terminated
Download-experiment	OUT	Return a package (tarball) of the results of an experiment, specified by id	• User is logged in • Specified experiment exists • Specified experiment belongs to user • Specified experiment has status "Terminated"	• A tarball containing the files created by docker/COMPSs
Delete-Experiment	OUT	Delete a specified experiment	• User is logged in • Specified experiment exists • Specified experiment belongs to user	• The experiment is deleted from the database • Docker/COMPSs Output files are deleted
Get-experiment-output	OUT	Return the results of an experiment, in JSON format	• User is logged in • Specified experiment exists • Specified experiment belongs to user • Specified experiment has status "Terminated"	• Returns a valid JSON with the output of the experiment (use case specific)
Get-input-dirs	OUT	Return a JSON representation of the available directories to be used as input, for the current user	• User is logged in • User has access to the returned directories	• Returns a valid JSON with an array of directories
Get-input-files	OUT	Return a JSON representation of the available files to be used as input, for the current user	• User is logged in	• Returns a valid JSON with an array of all files that the user has access to
Get-keyspaces	OUT	Return a JSON representation of the available Cassandra keyspaces to be used as input, for the current user	• User is logged in	• Returns a valid JSON with an array of all keyspaces that the user has access to

Create-custom-project	OUT	Create a new project, owned by the current user	• User is logged in	• A new project is created in the database • Code is uploaded in user space
Get-custom-projects	OUT	Get the list of custom projects of the current user	• User is logged in	• A valid array of the custom projects belonging to the user
Get-custom-project	OUT	Get the details of a custom project	• User is logged in • Specified project exists • Custom Project belongs to current user	• A valid JSON representation of the specified project
Update-custom-project	OUT	Update the parameters of a custom project	• User is logged in • Specified project exists • Custom Project belongs to current user	• The specified project is updated in the DB
Delete-custom-project	OUT	Delete a specified custom project	• User is logged in • Specified project exists • Custom project belongs to current user	• All experiments belonging to the specified project are deleted from the DB • The specified project is deleted from the DB
New-custom-experiment	OUT	Start a new experiment for a specific custom project	• User is logged in • Specified project exists • Custom Project belongs to current user	• A new experiment is created in the DB • A new docker stack is created in docker swarm • Custom docker image is created (if specified by user during project creation) • Docker-compss logs created (showing that COMPSs is running in worker nodes)

Delete-custom-experiment	OUT	Delete the specified experiment, belonging to a custom project	• User is logged in • Specified project exists • Experiment belongs to current user	• The experiment is deleted from the database • Docker/COMPSs Output files are deleted
Get-custom-experiments	OUT	Return a list of experiments, belonging to a custom project	• User is logged in • Specified project exists • Custom Project belongs to current user	• A valid array of experiments belonging to the specified custom project • Returned experiments that belong to current user
Stop-custom-experiment	OUT	Request the termination of a specified experiment, belonging to a custom project	• User is logged in • Specified experiment exists • Specified experiment belongs to user • Specified experiment has status “Running”	• The status of the new experiment is set to terminated • Docker stack belonging to this experiment is terminated
Download-custom-experiment	OUT	Return a package (tarball) of the results of the specified custom project experiment	• User is logged in • Specified experiment exists • Specified experiment belongs to user • Specified experiment has status “Terminated”	• A tarball containing the files created by docker/COMPSs
User login	AVT	User can successfully login to the Advanced Visualization Toolkit	• User is already registered	• User is redirected to the Advanced Visualization Toolkit Dashboard page
User logout	AVT	User can successfully logout from the Advanced Visualization Toolkit	• User is already logged in	• User is redirected to the Advanced Visualization Toolkit login page

Create-custom-project	GUCT	Create a new project, owned by the current user	• User is logged in	• UI validates user input • UI returns a new project • A new custom project is created in the DB, with the same properties as the one defined in the UI
Get-custom-projects	GUCT	Get the list of custom projects of the current user	• User is logged in	• UI returns all custom projects belonging to the user
Get-custom-project	GUCT	Get the details of a custom project	• User is logged in • Specified project exists • Custom Project belongs to current user	• A valid UI representation of the specified project, as stored in the DB
Update-custom-project	GUCT	Update the parameters of a custom project	• User is logged in • Specified project exists • Custom Project belongs to current user	• The specified project is updated in the DB • A custom project is updated in the DB, with the same properties as the one defined in the UI
Delete-custom-project	GUCT	Delete a specified custom project	• User is logged in • Specified project exists • Custom project belongs to current user	• The specified project is deleted from the DB
New-custom-experiment	GUCT	Start a new experiment for a specific custom project	• User is logged in • Specified project exists • Custom Project belongs to current user	• A new experiment is created in the DB • The UI returns the current experiment with running status
Delete-custom-experiment	GUCT	Delete the specified experiment, belonging to a custom project	• User is logged in • Specified project exists	• The experiment is deleted from the database

			• Experiment belongs to current user	• The UI returns to the Dashboard page
Get-custom-experiments	GUCT	Return a list of experiments, belonging to a custom project	• User is logged in • Specified project exists • Custom Project belongs to current user	• A valid visualization of experiments belonging to the specified custom project • Returned experiments belong to current user
Stop-custom-experiment	GUCT	Request the termination of a specified experiment, belonging to a custom project	• User is logged in • Specified experiment exists • Specified experiment belongs to user • Specified experiment has status "Running"	• The status of the new experiment is set to terminated
Download-custom-experiment	GUCT	Return a package (tarball) of the results of the specified custom project experiment	• User is logged in • Specified experiment exists • Specified experiment belongs to user • Specified experiment has status "Terminated"	• User successfully downloads a tarball containing the files created by docker/COMPSs
New-LASSO ADDM-experiment	UCT	Start a new experiment for LASSO ADDM (Self-service mode)	• User is logged in	• A new experiment is created in the DB • The UI returns the current experiment with running status
View-LASSO ADDM-experiment	UCT	View the results of experiment for LASSO ADDM (Self-service mode)	• User is logged in • Specified experiment exists • Specified experiment belongs to current user	• The UI parses the JSON output of the results and visualizes it correctly

New-K Means experiment	UCT	Start a new experiment for Kmeans (Self-service mode)	• User is logged in	• A new experiment is created in the DB • The UI returns the current experiment with running status
View- K Means - experiment	UCT	View the results of experiment for K means (Self-service mode)	• User is logged in • Specified experiment exists • Specified experiment belongs to current user	• The UI parses the JSON output of the results and visualizes it correctly
New-IP Address Relations Streaming-experiment	UCT	Start a new experiment for IP Address Relations Streaming (Co-develop mode)	• User is logged in	• A new experiment is created in the DB • The UI returns the current experiment with running status
View- IP Address Relations Streaming - experiment	UCT	View the results of experiment for IP Address Relations Streaming (Co-develop mode)	• User is logged in • Specified experiment exists • Specified experiment belongs to current user	• The UI parses the JSON output of the results and visualizes it correctly
New-CRF Casting - experiment	UCT	Start a new experiment for CRF Casting (Co-develop mode)	• User is logged in	• A new experiment is created in the DB • The UI returns the current experiment with running status
View- CRF Casting - experiment	UCT	View the results of experiment for CRF Casting (Co-develop mode)	• User is logged in • Specified experiment exists • Specified experiment belongs to current user	The UI parses the JSON output of the results and visualizes it correctly
Download - experiment	UCT	Return a package (tarball) of the results of a self-service-mode or a co-	• User is logged in • Specified experiment exists • Specified experiment belongs to user	• User successfully downloads a tarball containing the files created by docker/COMPSs

		develop mode project (only for batch processing projects)	• Specified experiment has status "Terminated"	
IP Address Relations Streaming-Messages	UMT	Check the messages exchanged in UM for IP Address Relations Streaming	• Experiment has started	• Messages describing new transactions are published in topic #caixa_in • Messages for transactions between members of the same group are published in UM topic #caixa_out
CRF Casting - Messages	UMT	Check the messages exchanged in UM for CRF Casting	• Experiment has started	• Messages describing new engine data are published in topic #tecsid_in • Messages for engines with classification results are published in UM topic #tecsid_out

5 Conclusions and next steps

Current deliverable presents the second version of the "Big-Data-as-a-Self-Service Test and Integration Report". The work gives important perception towards the release of the final I-BiDaaS solution, ensuring a smooth and effective integration of the separate I-BiDaaS components, taking into consideration their availability, and interoperability potential. Moreover, this document presents the analysis of the 1st integrated solution based on (i) a generic use case and (ii) a number of 4 complementary use cases deployed, driven by the I-BiDaaS end users. The main objective of these deployments is to further validate and test the I-BiDaaS architecture and the integration process between the I-BiDaaS components. The findings obtained from the current study are summarized below:

- Prevision of the updated analysis of the testing and integration plan for the I-BiDaaS platform (Chapter 4).
- Representation of the main tools and components developed and integrated on this stage of investigation (Chapter 2).
- Presentation of the updated architecture and the generic use case deployed for the 1st integrated version (Chapter 2)
- Description of all the use cases driven by the I-BiDaaS partners that are deployed in the framework of the 1st integrated version, including descriptions of the technologies used for each use case.

Future work, starting from M25, will expand the integration of the I-BiDaaS platform with respect to the 1st integrated version towards the final implementation including all the tools and technologies by FORTH, BSC, ATOS and UNSPMF, and it will include also the remaining CAIXA, CRF and TID's use cases experiments according to a revised experimental protocol as lead by UNIMAN.

Some important milestones for the future are the integration of the RMO module (M26), the finalization of the design of the remaining Use Cases (M26) and the packaging of the platform, to allow installation from 3rd parties in private or public clouds.

Moreover, as the various additional use cases that will be implemented in the I-BiDaaS platform become well-defined, a complete list of E2E tests will be prepared and utilized for the validation of I-BiDaaS final platform integration. The time plan is depicted in the following figure:

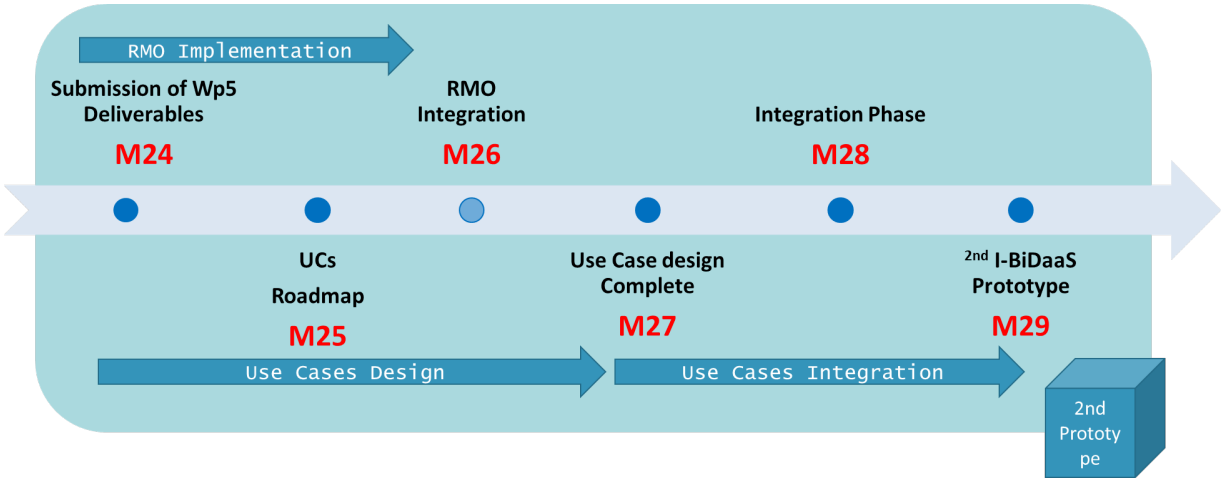


Figure 42. Platform integration time plan

6 References

[1] I-BiDaaS project 2018a. Deliverable D5.2 Big-Data-as-a-Self-Service Test and Integration Report (first version), available at https://www.ibidaas.eu/deliverables

[2] I-BiDaaS project 2019a. Deliverable D2.4 Universal Messaging Bus (interim version), Public Report, available at https://www.ibidaas.eu/deliverables

[3] I-BiDaaS project 2019b. Deliverable D4.2 Universal Messaging Bus (Interim Version)., Confidential Report

[4] Cesare Cugnasco, Hadrien Calmet, Pol Santamaria, Raül Sirvent, Ane Beatriz Eguzkitza, Guillaume Houzeaux, Yolanda Becerra, Jordi Torres, and Jesus Labarta, The OTree: Multidimensional Indexing with efficient data Sampling for HPC, *IEEE Big Data 2019*, Los Angeles.

Appendix 1 – Orchestrator REST API

Route	Type	Description
experiment/update-status/{id}	GET	Sets status of experiment to terminated. Used internally when compss terminates
custom-experiment/update-status/{id}	GET	Sets status of experiment to terminated. Used internally when compss terminates
project	GET	Gets the list of projects (predefined projects)
project/experiments/{id}	GET	Gets the list of project experiments owned by the current user, project is specified by project_id
project/{id}	GET	Gets the details of a project, specified by project_id
experiment/run/{project_id}	POST	Starts a new experiment for project, specified by project_id
experiment/{id}	GET	Gets the details of experiment, specified by id
experiment/stop/{id}	GET	Requests the termination of an experiment, specified by id
experiment/download/{id}	GET	Returns a package (tarball) of the results of an experiment, specified by id
experiment/{id}	DELETE	Deletes the specified experiment
experiment/output/{id}	GET	Returns the results of an experiment, in JSON format
input-dirs	GET	Returns a JSON representation of the available directories to be used as input, for the current user
input-files	GET	Returns a JSON representation of the available files to be used as input, for the current user
keyspaces	GET	Returns a JSON representation of the available Cassandra keyspaces to be used as input, for the current user
custom-project	POST	Creates new project, owned by the current user
custom-project	GET	Gets the list of custom projects of the current user
custom-project/{custom_project_id}	GET	Gets the details of a project, specified by project_id

custom-project-update	POST	Updates the parameters of a project, specified by project_id
custom-project/{custom_project_id}	DELETE	Deletes the specified project
custom-experiment/run/{custom_project_id}	POST	Starts a new experiment for project, specified by project_id
custom-experiment/{custom_experiment_id}	DELETE	Deletes the specified experiment
custom-project/experiments/{custom_project_id}	GET	Returns a list of experiments, belonging to a custom project, specified by custom_project_id
custom-experiment/stop/{custom_experiment_id}	GET	Requests the termination of an experiment, specified by custom_project_id
custom-experiment/download/{id}	GET	Returns a package (tarball) of the results of the specified experiment