



Ecosystem for COllaborative Manufacturing PrOceSses – Intra- and
Interfactory Integration and AutomaTION
(Grant Agreement No 723145)

D5.1 Big data mining and analytics tools I

Date: 2017-11-30

Version 1.0

Published by the COMPOSITION Consortium

Dissemination Level: Public



Co-funded by the European Union's Horizon 2020 Framework Programme for Research and Innovation under Grant Agreement No 723145

Document control page

Document file: D5.1 Big data mining and analytics tools I
Document version: 1.0
Document owner: FIT

Work package: WP5 – Integration of Internal and External Elements
Task: T5.1 – Multi-Level and Cross-Domain Big Data Analysis and Management
Deliverable type: OTHER

Document status: Approved by the document owner for internal review
 Approved for submission to the EC

Document history:

Version	Author(s)	Date	Summary of changes made
0.1	José Ángel Carvajal Soto (FIT)	2017-11-07	Initial TOC
0.2	José Ángel Carvajal Soto (FIT)	2017-11-10	Add content
0.3	Alexander Graß (FIT)	2017-11-21	Internal review
0.4	Dimosthenis Ioannidis (CERTH)	2017-11-22	Big Data Visual Analytics' chapter added
0.5	José Ángel Carvajal Soto (FIT)	2017-11-24	Merging content and taking review changes
0.6	José Ángel Carvajal Soto (FIT)	2017-11-27	First complete version with LinkSmart development synergies and conclusion
0.7	Dimosthenis Ioannidis (CERTH)	2017-11-27	Updated version of the Visualization chapter
0.8	José Ángel Carvajal Soto (FIT)	2017-11-27	Consolidated version before review
1.0	José Ángel Carvajal Soto	2017-11-30	Final version

Internal review history:

Reviewed by	Date	Summary of comments
Matteo Pardi (NXW)	2017-11-29	Minor comments.
Ifigeneia Metaxa (ATL)	2017-11-29	Minor comments

Legal Notice

The information in this document is subject to change without notice.

The Members of the COMPOSITION Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the COMPOSITION Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Possible inaccuracies of information are under the responsibility of the project. This report reflects solely the views of its authors. The European Commission is not liable for any use that may be made of the information contained therein.

Index:

1	Introduction	4
2	COMPOSITION Context	5
2.1	Scenarios	5
2.2	Scenario INTRA-2: Predictive maintenance.	5
2.2.1	COMPOSITION Data Analytics View: Intra-factory Interoperability Layer	6
3	Big Data Analytics provided by the LinkSmart® IoT Learning Agent	8
3.1	Introduction	8
3.2	What's new in the LinkSmart® IoT Learning Agent.....	8
3.3	The Complex-Event Machine Learning methodology	9
3.3.1	Data Propagation Phase.....	9
3.3.2	Data Pre-Processing (Munging) Phase	9
3.3.3	Learning Phase.....	9
3.3.4	Continuous Validation Phase.....	10
3.3.5	Deployment Phase.....	10
3.4	Design Perspective	10
3.5	Functional view: Capabilities	15
3.5.1	Stream	15
3.6	Architecture.....	17
3.7	Scalability.....	17
3.8	APIs Implementation.....	18
3.8.1	I/O API:	19
3.8.2	Process API:	20
3.8.3	Monitoring API	22
3.8.4	Management API	23
3.9	Security	23
3.9.1	TLS.....	23
3.9.2	JWS.....	24
3.10	LinkSmart development synergies.....	24
3.10.1	Continuous Integration best practices	24
4	Big Data Visual Analytics	31
4.1	Visualization Techniques Analysis	31
4.1.1	Overview of Visualization Techniques Categories	31
4.1.2	Parameters Define Visualization Technique's Selection	31
4.1.3	Brief analysis of general visualization tools	32
4.2	Overview of Visual Analytics tool architecture	33
4.3	Visual Analytics tools in COMPOSITION Use Cases	33
5	Future work and Conclusions.....	37
5.1	Future Work	37
5.1.1	Integration the real-time data of the Pilot partners	37
5.1.2	Integrating the LA with the DLT	37
5.1.3	Enabling TLS and JWS.....	37
5.1.4	Integrate the LA with the Visual Analytics	37
5.1.5	Test the Scalability of the LAs	37
5.1.6	Improve monitoring and management tooling	37
5.2	Conclusion	37
6	List of Figures and Tables.....	38
6.1	Figures	38
6.2	Tables	38
7	References	39

1 Introduction

1.1 Summary

In this deliverable, we present the current development state of the task 5.1. We start by presenting a general background in this chapter. In chapter 2, we present the COMPOSITION scenarios where Data Analytics will be used, and the place of these techniques in the general COMPOSITION architecture. In chapter 3, we present the solution for Big Data Analysis. In chapter 4, we present the analysis of the possible representation of the Big Data Analysis. Finally, we end with a short conclusion in chapter 5.

1.2 Background

The manufacturing industry is being disrupted in what is already known as the 4th industrial revolution or Industry 4.0. This revolution is driven by the need of reduction of the time-to-market [1], increment of complexity, (mass customization) [2] [3], and added value services [4] around the products -- all together in a competitive globalized world [4]. To solve these challenges, this revolution is introducing a set of new advanced networking technology, hardware, and more important, intelligent software. While in the 3rd industrial revolution, the manpower was replaced by simple "hardwired" automatization [5], which could not adapt to market trends fast enough - e.g. in mass customization where almost unlimited variations of a product can be produced [1], [6], [7], [8], [9] - the current revolution is creating so-called cyber-physical systems, where machines collect data, communicate with each other and jointly take decisions [10]. To succeed in this new industry, the technologies must be highly adaptable, manageable, and in many cases even self-managed and self-configured [11], [12].

To achieve this degree of intelligence, advanced algorithms have been incorporated into the production process to achieve embedded artificial intelligence (AI) within the process. This embedded AI has been constructed from the experiences obtained by the machines and usually designed by data scientists [13]. These techniques can be used in several manufacturing challenges such as predictive maintenance or product defect detection.

However, while many efforts has been invested in tackling these challenges, few works has been done in developing an integrated manageable platform to solve these problems [14], [15], [16], [17], [18]. Most of the solutions propose a heterogeneous set of technologies to achieve the goals, and mostly none provide tooling for a deployment or to manage the solution in a deployed running system. Most existing solutions do not provide any tooling for collecting data and management for AI technologies, such as online machine learning (ML). Additionally, the solutions provide very little integrated deployment tools for the reproduction of ML methods or models in other deployment environments.

We believe in the need of an integrated extensible solution that provides runtime management tools and is able to manage and configure itself. A platform that provides a set of mechanisms for real-time data collection, processing, and analysis. In this manner, it is possible to create common methodologies to reproduce and redeploy ML and other AI technologies reducing their cost and increasing their usage.

In this deliverable, we present our solution first shown in [19], within a manufacturing environment. More specifically, we deploy the solution in Surface-Mount Technology production in BLS plant and in other process in the KLE production plant.

2 COMPOSITION Context

2.1 Scenarios

As stated in the deliverable D2.1, the COMPOSITION IIMS will apply business intelligence to provide improved coordination mechanisms of collaborative manufacturing processes. It will be based on continuous real-time monitoring and the control of underlying complex collaborative industrial and logistics processes.

The Intra-factory scenarios will demonstrate the value added to services that address fundamental challenges in the pilot organisations by matching requirements to capabilities for internal and external processes and addressing emerging issues. The scenarios will aim at boosting collaborative manufacturing and intra-factory interoperability in marketplaces to the next level of knowledge management, agility, reliability, security, responsiveness, and cost-efficiency.

In close dialogue with the pilot owners, four Intra-factory scenarios were defined:

- Scenario INTRA-1: Production Floor Monitoring and Visualisation
- Scenario INTRA-2: Predictive Maintenance
- Scenario INTRA-3: Material Management
- Scenario INTRA-4: Automatic Data Conversion

For now, big data analytic tools and services will be needed only for INTRA-2 accordingly to the amount of data and the data generation rates.

2.2 Scenario INTRA-2: Predictive maintenance.

The COMPOSITION IIMS collects historical information as well as real-time information about the performance. E.g. speed (RPM), power (Watts) or decibel levels (dbis captured at machine level (laser power, soldering paste, fans, mechanical conveyers, etc.

The actual status is compared to static data models (performance specs, history, costs) about the optimum process performance and algorithms can predict the likely point in time where critical components in the machine or process may fail thus causing the manufacturing process to be disrupted or products to be scrapped. Based on historic performance the prediction of failures can further be improved using different machine learning technologies.

The predictions are presented to the operator to support their decision about when and what to replace before failure occurs, saving costs. The operator will thus see a selection of critical components and their estimated time of failure.

The COMPOSITION IMMS will help the pilots to efficiently and effectively manage machine downtimes and failures based on the prediction of failures of critical components. Information such as levels and temperature of solvents, vibration of machines, etc., will be provided in the IMMS. Prediction of the Blower motors within the Heller Ovens are very important to BSL from a quality and cost perspective. Potential cost of a non-recoverable oven alarm (motor/blower failure) resulting in non-conforming product being scrapped is estimated as \$60K

2.2.1 COMPOSITION Data Analytics View: Intra-factory Interoperability Layer

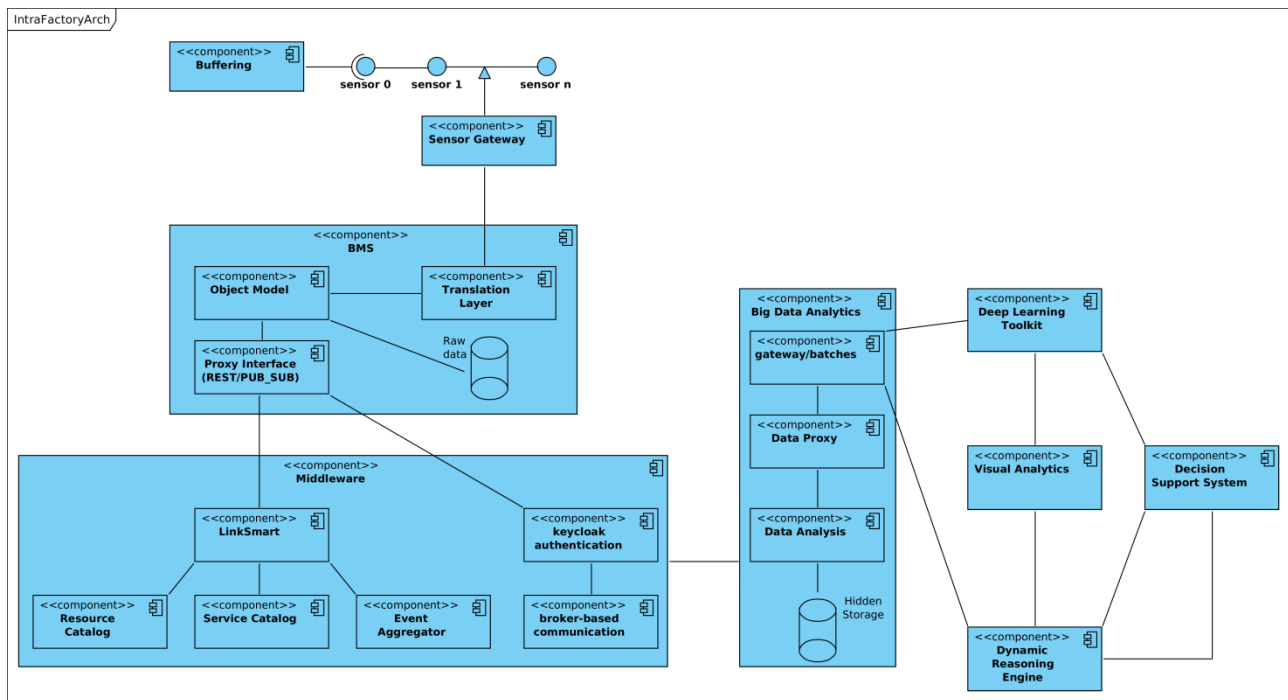


Figure 1. Intra-factory interoperability layer components and dependencies.

In this chapter, we will address the architecture of the Big Data Analytics module in the COMPOSITION context. As it was mentioned before, the INTRA-2 is currently the only scenario with big data characteristics. The INTRA-2 scenario is an Intra-Factory scenario. Therefore, we will focus in this chapter on the Intra-Factory architecture, known as intra-factory interoperability layer.

The intra-factory interoperability layer has two main goals: the first one is to provide a model for interconnecting the COMPOSITION ecosystem in the intra-factory scenario, the second one is to ensure the conformity between communications among interconnected components. The involved technology is provided by development partners of COMPOSITION and the connectors that will be defined, developed and deployed to integrate these.

Individual partners' responsibilities and work package outputs are highlighted in the followings:

- Sensors, Sensor Buffering and Sensor Gateways will be developed and adopted from existing technology. Consideration will be taken to Technical Objective 1.1 (see D2.3).
- The BMS is provided by a project development stakeholder (NXW) and is the translation layer providing shop floor connectivity from sensors to the COMPOSITION system. Raw data storage will be added for offline debug purposes.
- The middleware is the main recipient in which the interoperability single components act
 - LinkSmart is a well-known middleware solution per se and will be customized to satisfy COMPOSITION requirements. Components include
 - Service Catalog, works as service index, and provides security information for service intercommunication.
 - Event Aggregator, parse messages to ensure homogeneity in data streams
 - Keycloak is a virtual layer that ensures authorization and authentication. Like all security related measures, it will be deployed by the Security Framework.
 - The broker-based intra-factory communication system manages all internal communication in COMPOSITION.

- The Big Data Analytics component provides Complex Event Processing (CEP) capabilities for the data provided by the intra-factory integration layer
- The Hidden Storage is a storage not accessible from the outside in which aggregated data are stored for debug purposes, i.e. re-bootstrapping already trained artificial neural networks belonging to the Deep Learning Toolkit and to the Dynamic Reasoning Engine.
- The Deep Learning toolkit component for this intra-factory scenario and an example will be described in D5.3.
- The Visual Analytics component is the reporting interface of the Decision Support System and Simulation and Forecasting Toolkit.
- The Dynamic Reasoning Engine is part of the Simulation and Forecasting Toolkit.
- The Decision Support System uses process models to guide the production process.

3 Big Data Analytics provided by the LinkSmart® IoT Learning Agent

In this chapter, we present the selected solution and the development of the solution in the context of COMPOSITION. The chapter starts with an introduction of the solution. In subsection 3.2, the new developments in the COMPOSITION context

3.1 Introduction

Manufacturing in assembly lines consist in a set of hundreds, thousands or millions of small discrete steps aligned in a production process. Automatized production processes or production lines thereby produce for each of those steps small bits of data in form of events. Although the events possess valuable information, this information loses its value over time. Additionally, the data in the events usually are meaningless if they are not contextualized, either by other events, sensor data or process context. To extract most value of the data, it must be processed as it is produced, to be more precise in real-time and on demand. Therefore, in case of Big Data Analysis we propose the usage of Complex-Event Processing for the data management coming from the production facilities. In this manner, the data is processed at the moment when it is produced extracting the maximum value, reducing latency, providing reactivity, giving it context and avoiding the need of archiving unnecessary data.

The Complex-Event Processing service is provided by the LinkSmart® Learning Agent (LA). The LA is a Stream Mining service that provides the utilities to manage real-time data for several purposes. On the one hand, the LA provides a set of tools to collect, annotate, filter, aggregate, or cache the real-time data incoming from the production facilities. This set of tools facilitates the possibility to build applications on top of real-time data. On the other hand, the LA provides a set of APIs to manage the real-time data lifecycle for continuous learning. Moreover, the LA can process the live data to provide complex analysis creating real-time results for alerting or informing about important conditions in the factory, that may be not be seen at first glance. Finally, the LA allows the possibility to adapt to the productions needs during the production process.

It's worth mentioning that the LA does not learn from the data, it just facilitates the data to the models. In other words, the LA connects externally to the models for the learning process. By this, the LA enables the online real-time learning process and data deliverable for training the model. In COMPOSITION, the external learning models will be provided by a Deep Learning Toolkit. Nevertheless, the LA is capable of doing on-the-run analytics using less historical data intensive algorithms such as Random Forests, Gradient Boosting, Kalman Filters, Particle Filter, Hidden Markov Models, *boosted* Artificial Neural Networks. With them it may be possible to predict certain phenomenon without the need of historical data.

The LA has been developed and tested in different EU projects such as ALMANAC¹ and IMPReSS². However, the use cases where in the scope of Smart Cities or Smart Buildings, and it must be tailored for more Industry 4.0 oriented use cases where the events are driven by business processes and data intensive.

3.2 What's new in the LinkSmart® IoT Learning Agent

The LA start to be developed in 2014 in the ALMANAC project as a simple CEP for Smart Cities and presented in [20]. Since then, the LA has being developed and transformed in a self-managed learning orchestrator service that combined Complex-Event Processing and Machine Learning and other techniques. Specifically in COMPOSITION there had being following improvements:

- Python interoperability layer for programmers or Python SDK
- Micro-batch learning handling for non-iterative learning models
- Implementation and testing of a default detection model for SMTs using the Python SDK and Random Forest model.
- Implementation of the JWS standard for the I/O API.
- Full Dockerized distribution
- Introduction of CI for quality assurance using automatic testing. This includes

¹ http://cordis.europa.eu/project/rcn/109709_en.html

² http://cordis.europa.eu/project/rcn/185510_en.html

- Development of Docker based Integration Test for Statement API
- Development of Docker based Integration Test for CEML API
- Other smaller improvements and fixes had been done. For more detailed information, please check the LinkSmart® project documentation³ and source⁴ code release notes.

3.3 The Complex-Event Machine Learning methodology

The Complex-Event Machine Learning (CEML) [19] is a framework that combines Complex-Event Processing (CEP) [21] and Machine Learning (ML) [22] applied to the IoT. This means that the framework was developed to be deployed everywhere, from the edge of the network to the cloud. Furthermore, the framework can manage itself and works autonomously. The following section briefly describes the different aspects that CEML covers. The framework must automate the learning process and the deployment management. This process can be broken down in different phases: (1) the data must be collected from different sensors, either from the same device or in a local network. (2) The data must be pre-processed for attribute extraction. (3) The learning process takes place. (4) The learning must be evaluated. (5) When the evaluation shows that the model is ready, the deployment must take place. Finally, all these phases happen continuously and repetitively, while the environment constantly changes. Therefore, the model and the deployment must adapt as well.

3.3.1 Data Propagation Phase

Data in the IoT is produced in several places, protocols, formats, and devices. Although this deliverable does not address the problem of data heterogeneity in detail, the learning agents require a mechanism to acquire and manage the heterogeneity of the data. The mechanism must be scalable and, at the same time, the protocol should handle the asynchronous nature of IoT. Finally, the protocol must provide tools to handle the pub/sub characteristics of the CEP engines. Therefore, we have chosen MQTT5, a well-established Client Server publish/subscribe messaging transport protocol. The topic based message protocol provides a mechanism to manage the data heterogeneity by making a relation between topics and payloads. It allows deployments in several architectures, OS, and hardware platforms; basic constraints at the edge of the network. The protocol is payload agnostic and as such allows for maximum flexibility to support several types of payloads.

3.3.2 Data Pre-Processing (Munging) Phase

Usually ML is tied to stored datasets, which incurs several drawbacks. Firstly, the learning can take place only with persistent data. Secondly, usually the models generated are based on historical data, not current data. Both constraints, in the IoT, have dire consequences. It is neither feasible nor profitable to store all data. Also, embedded devices do not have much storage capacity which makes it impossible to use ML algorithms on them. Furthermore, IoT deployments are commonly exposed to ever-changing environments.

Using historical data for off-line learning could cause outdated models learning old patterns rather than current ones, producing drifted models. Although some IoT platforms like COMPOSITION support storage of historical data, it may be too time and space consuming to create large enough times series. Therefore, there is also a need for non-persistence manipulation tools. This is precisely what the CEP engine provides in the CEML framework. This means, the CEP engine decides which and how the data is manipulated using predefined CEP statements deployed in the engine. Each statement can be seen as a topic, to which each learning model is subscribed. Any update of the subscribers provides a sample to be learnt in the learning phase.

3.3.3 Learning Phase

There is no pre-selection of algorithms in the framework. They are selected by the restrictions imposed by the problem domain. For example, in extreme constrained devices, algorithms such as Algorithm Output Granularity (AOG) [23] may be the right choice. In other cases where the model changes quickly, one-shot algorithms may be the best fit. Artificial Neural Networks are good for complex problems but only with stable phenomena. This means that the algorithm selection should be made case-by-case. Our framework provides mechanisms for the management and deployment of the learning models, and the process of how the model is fed with samples. In general, the process is based on incremental learning [24] albeit with online and non-

³ <https://docs.linksmart.eu/display/LA>

⁴ <https://code.linksmart.eu/projects/LA/>

⁵ *MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. Source <http://mqtt.org/>*

persistent data. The process can be summarized as follows: the samples, without the target provided in the last phase, are used to generate a prediction. The prediction will then be sent to the next phase. Thereafter, the sample is applied to update the model. Thus, all updates are used for the learning process.

3.3.4 Continuous Validation Phase

This section describes how the validation of the learning models is done inside the CEML. This phase does not influence the learning process nor validate the CEML framework itself.

ML model validation is a challenging topic in real-time environments and the evaluation for distributed environments or embedded devices is not addressed extensively in the literature, which is why we think it needs further research. There are two addressed strategies. Either we holdout an evaluation dataset by taking a control subset for given time-frame (time window), or we use Predictive Sequential, also known as Prequential [25], in which we assess each sequential prediction against the observation. The following section describes the continuous validation we applied for a classification problem, even though it can be applied for other cases as well.

Instead of accumulating a sample for validation, we analyse the predictions made before the learning takes place. All predictions are assessed each time an update arrives. The assessment is an entry for the confusion matrix [26] which is accumulated in an accumulated confusion matrix. The matrix contains the accumulation of all assessed predictions done before. In other words, the matrix does not describe the current validation state of the model, but instead the trajectory of it. Using this matrix, the accumulated validation metrics (e.g. Accuracy, Precision, Sensitivity, etc.) are being calculated. This methodology does have some drawbacks and advantages, explained more extensively in [19].

3.3.5 Deployment Phase

The continuous validation opens the possibility for making an assessment of the status of the model each time a new update arrives, e.g. if it is accrued or not. Using this information, the CEML framework has the capability to decide if the model should or should not be deployed into the system at any time. If the model is behaving well, then it should be deployed, otherwise it should be removed from the deployment. The decision is made by user-provided thresholds w.r.t. evaluation metrics. If a threshold is reached, the CEML inserts the model into the CEP engine and starts processing the streams using the model. Otherwise, if the model do not reach the threshold then its remove form the CEP engine.

3.4 Design Perspective

In this subsection, we will discuss the design considerations taken to develop the LA as a data-processing and data analysis platform for big data and machine learning processes.

The envision platform, should allow self-managed data process mining and which can distribute the processing power over the network by creating a scalable processing overlay. As a first step, we envision the possibility of describing the complete representation of the process, namely the **Complex-Event Machine Learning Request (CEMLR)**, we need to describe in computer readable manner, the **CEML** processes according to their parts, namely: **Pre-Processing Rules** and **Feature Extraction Rules** (for Data Pre-Processing Phase): **Learning Description** (Learning Phase), **Evaluation Description** (Continuous Validation Phase), and **Actuation Rules** (Deployment Phase). The **Pre-Processing Rules** describes how the fragmented raw input data or data streams are processed and aggregated. The **Feature Extraction Rules** define how features are extracted from pre-processed data. The **Learning Description** defines the selection of an **Algorithm**, **Parameters**, and the **Feature Space** for construction of a *model*.

The **Evaluation Description** is used to construct an *Evaluator*. The *Evaluator* is attached to the *model* and is responsible for providing real-time performance metrics about the model and deciding if it reaches the expected scores. Finally, the **Actuation Rules** describe actuation of the system whenever the *model* reaches the expected performance scores. All steps are performed in an **Execution Pipeline Environment (EPE)** or distributed in a set of interconnected EPE over the network. The output of the platform is the smart actuation of the system based on the predictions of the trained model. Moreover, a real-time monitoring infrastructure enabled tracking of the distributed process. Lastly, the platform allows the export of a trained **CEMLR**, in order to redistribute and replicate the process elsewhere.

Table 1 Analysis between the phase who captures, where are executed and implemented

Phase	Pre processing	Learning	Continuous Validation	Deployment
Captured by	Pre-processing & Feature Extraction Rules	Learning Description	Evaluation Description	Deployment Rules
Produces	Streams	Models	Evaluation State	Streams
Implemented by	Statements in CEP Engines	Models	Evaluators	Statements in CEP Engines

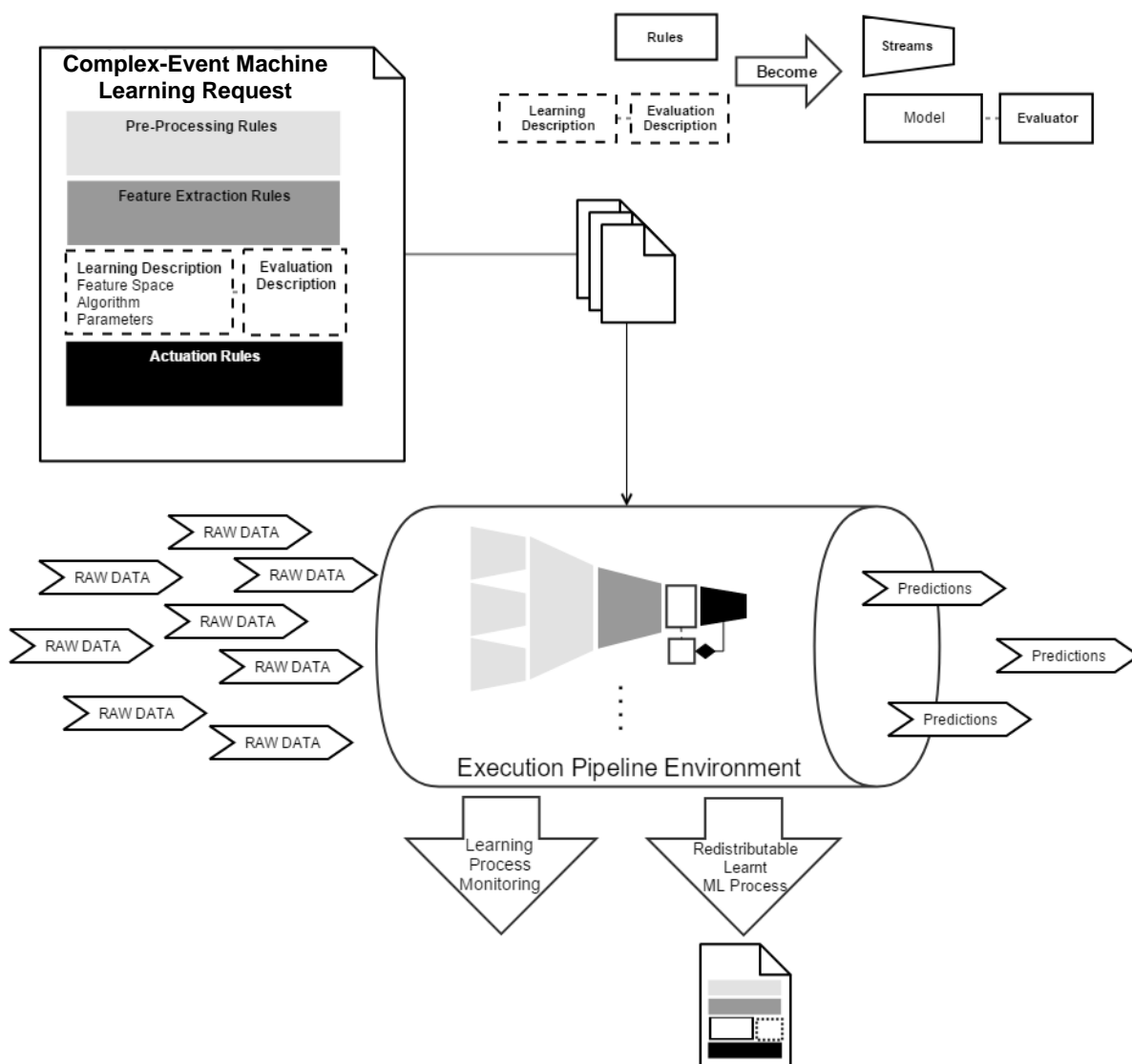


Figure 2 Design view of a single instance of CEML execution system

The possibility of describing the processes and executing them in execution pipelines allows for a reallocation and distribution of the computations according to available applications and resources. In particular, it allows to split the processes and to redistribute them among the available computational power regardless of the actual location. In parts, this enables the applications to spread them along the communication path and to

use the resources available, while reducing the costs. Nevertheless, spreading the processes inevitably adds more complexity to the applications and therefore, a new set of APIs for managing and monitoring the processes is needed.

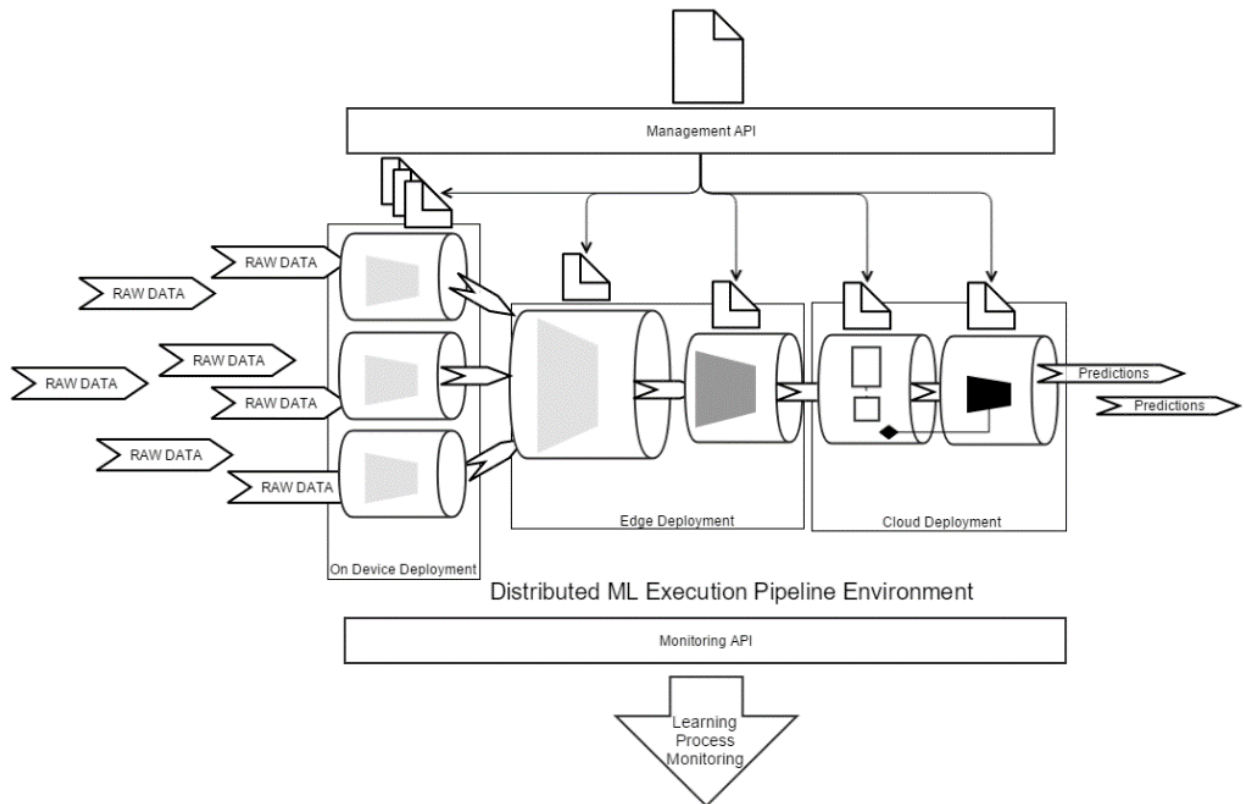


Figure 3 Design perspective of multiple CEML execution units

This set must be available for the different applications and stakeholders addressing their needs and fulfilling the different requirements to achieve the individual application goals. The APIs can be divided in the I/O, Management, Monitoring, and ML Process, each interacting with different parts of the application processes, which are Application Environment, Application Developer, System Monitor, and the External Model Backends, respectively. Moreover, each API should adapt to the users System Integrator, Application Developer, System Administrator, and Data Scientist and the goals like interconnect the data and responses, develop the application, monitor the status of the system and integrate new algorithms into the system, respectively. The I/O API allows the input of raw streams to the EPE and the output of the already processed information. The Management API is a CRUD API for the CEMLR. The Monitoring API provides tools to monitor the system performance and the machine learning processes in real-time. The ML Process API allows to connect to the execution process to add new ML *models* to the system.

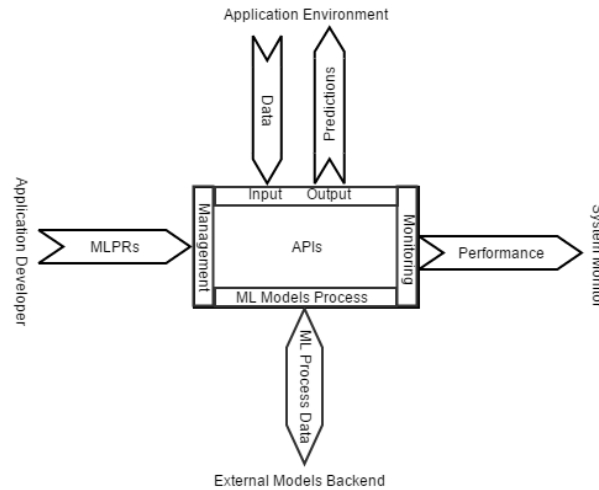


Figure 4 Structure of the API

Table 2 Mapping between users and APIs

	I/O API	Management API	Monitoring API	ML Process API
User	System Integrator	Application Developer	System Administrator	Data Scientist
			Data Scientist	

Finally, the APIs should adapt to the deployment of the system and the deployment should adapt to the applications needs. On the one hand, the degree of distribution of the system brings a different set of requirements for the APIs. On the other hand, a distribution of the system provides advantages and disadvantages. When processing data that is closer to the data sources, metrics like latency, privacy, confidentiality concerns and networking dependencies are reduced. On the other hand, if the data is processed closer to the cloud, a higher availability, computation power and reliable energy power sources are given. Thus, all this must be taken into consideration at the moment of building new systems.

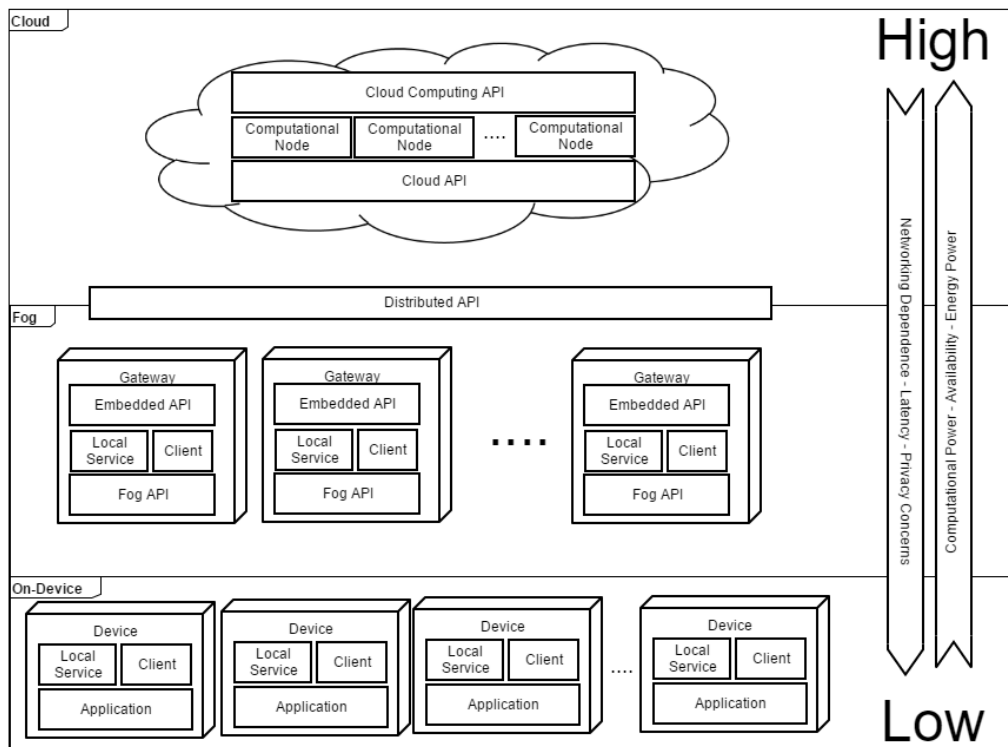


Figure 5 Deployment view of the CEML distributed system

For building the platform, we will build up using a set of state-of-the-art technologies and best practices in the Internet of Things. For the data and metadata representation and management, we may consider standards such as OGC SensorThing. For the data streams as well as for the multicast APIs in highly distributed deployments the pub/sub protocol such as MQTT is intended. For uni-cast requests, a RESTful API is considered. For the data ubiquitous data sources, lightweight JSON payloads could be used. In case of heavy data load, Google Buffers protocol could be an alternative. On the system side, Docker based technology as well as Docker Swarm or Prometheus could provide the necessary tools for deployment and monitoring. On the computational part, Complex-Event Processing (CEP) or BPMN engines could serve as Execution Pipeline Environment for the execution of rules. The Rules can be described by the CEP DSL or BPMN as a description language. Additionally, the engines should be extended to support analysis frameworks such as TensorFlow or DeepLearning4J. Moreover, we will consider incorporating well-known developer environments for data scientist such as Weka, R, python tools. However, there are no standardized way to describe a Machine Learning model, which requires fundamental research and development. Finally, security can be added using TLS for channel encryption, XAML for policy management and SAML for authentication.

3.5 Functional view: Capabilities

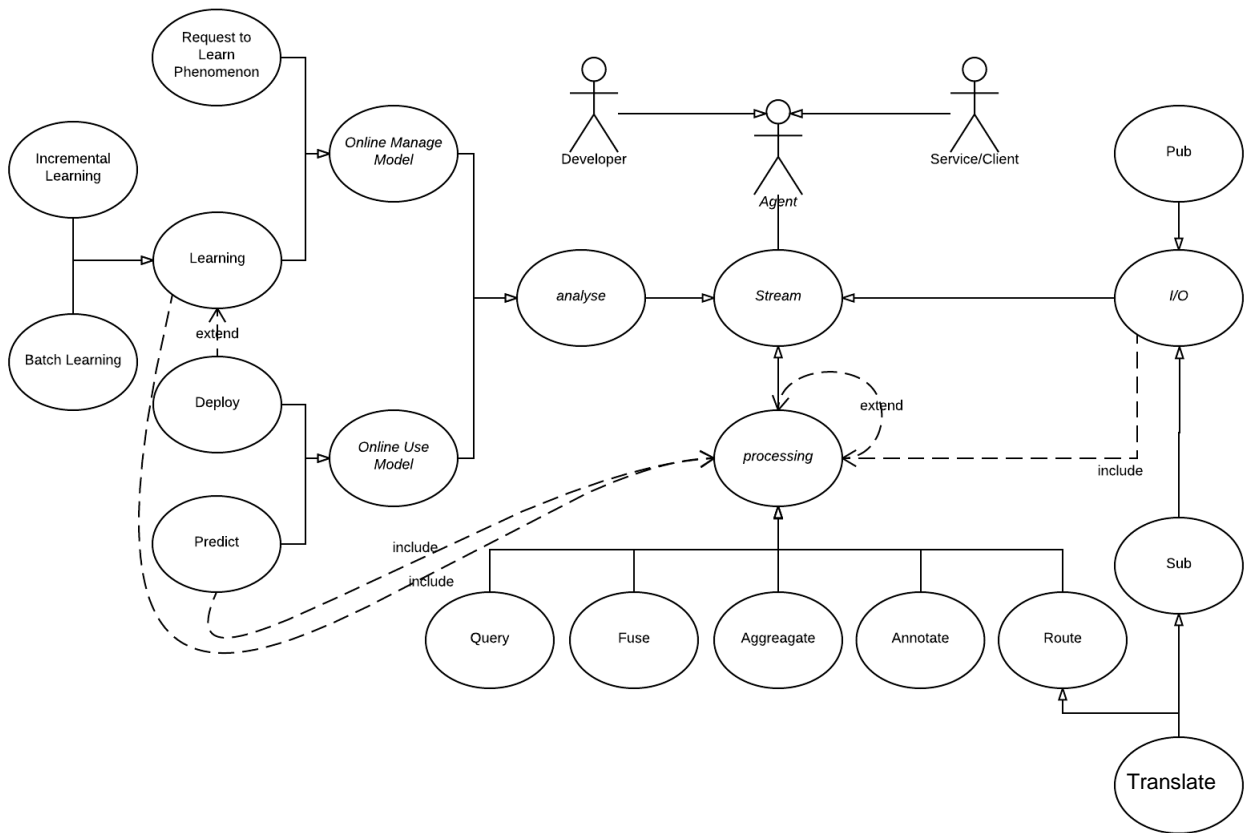


Figure 6. LinkSmart® Learning Service Enabling Use Cases.

3.5.1 Stream

The LA works executing process pre-defined processes which are triggered by incoming streams. Therefore, all operation take part inside the LA are applied on streams.

3.5.1.1 Processing

This functional use case represents requests or operations that can be Created, Read, Updated, or Delete (CRUD).

3.5.1.1.1 Aggregate

It generates a new stream with enriched information out of the original stream. E.g. calculate the average water consumption.

3.5.1.1.2 Fuse

It uses several streams and fuse them generating a new stream. As an example, the data streams of the fill level sensor of smart bins as well as the data stream of their temperature sensors can be combined to generate a smell virtual sensor or smell stream.

3.5.1.1.3 Query

This use case identifies one or several streams that fulfill a certain condition, e.g. using the fill level sensors of the bins the LA generates a new stream, which represents collection routes.

3.5.1.1.4 Route

This use case takes one stream, channeled to another stream (either by changing the stream id, or by changing the publishing output). E.g. forward an internal event in the Intra-Factory Interoperability Layer to the visualization service

3.5.1.1.5 Annotate

In this use case an event or an event stream is republished under another topic. Before being republished additional information might be added to original event. E.g. a welding gun energy consumption event is annotated with the business step.

3.5.1.1.6 Translate

Similar route, but in this UC the event is translated into another communication protocol. E.g. forward an internal event in the IIMS to a cellphone as SMS message.

3.5.1.2 I/O

This operation addresses the management of a datastream fed into the system and the output of the results of the processes in the system.

3.5.1.2.1 Pub

After any of the processes takes place, the result must be distributed. The distribution of the result depends of the target protocol where the result will be published.

3.5.1.2.2 Sub

The datastreams are unbounded data series. This series of data comes as a continuous flow of data. Therefore, the system must provide a mechanism of how the continuous data flow is inserted into the system.

3.5.1.3 Analyse

Some computational applications on the data are more complex than just applying an operation from the input data and allocate the result in the output bus. In some computation applications, the processes have states and steps and each process influences the data flow. This is the case for machine learning, and this use case is to address such process. We will call the state or “memory” of the process the model.

3.5.1.3.1 Online Manage Model

This use case addresses the actions that change the processes or algorithms that change the model.

3.5.1.3.1.1 Request to Learn Phenomena

The system receives a full detail process definition to construct and manage a model. The system “builds” the model out of the description and start to feed the model with the data accordingly.

3.5.1.3.1.2 Learning

Here the system executes the continuous learning process and feeds the data to train the model every time new data arrives according to the current description of the process. The learning process can happen in two distinctively ways: Incrementally or Batch-wise.

3.5.1.3.1.2.1 Incremental Learning

The incremental learning is the training process where the model is fed with information one-by-one. This means, as soon a new data-point can be constructed the model is trained and the state of the model is evaluate afterwards.

3.5.1.3.1.2.2 Batch Learning

The incrementally learning is the training process where the model is fed with information using blocks of data. This means, the model is trained using a dataset constructed from several data-points that had being accumulated from the input data, where the model is evaluated afterwards.

3.5.1.3.2 Online Usage Model

The sole purpose of a model is to be used in an environment. This use case describes the “use” or management of the model in the environment.

3.5.1.3.2.1 Deploy

After the model has reached the desired state, the model is introduced to the active environment to execute the specified task in the learning request. If the performance of the model drops, the model is removed from the environment.

3.5.1.3.2.2 Predict

In any moment a user can specify requests to a model to provide a prediction (providing the input corresponding) to the model regardless of the performance state of the model.

3.6 Architecture

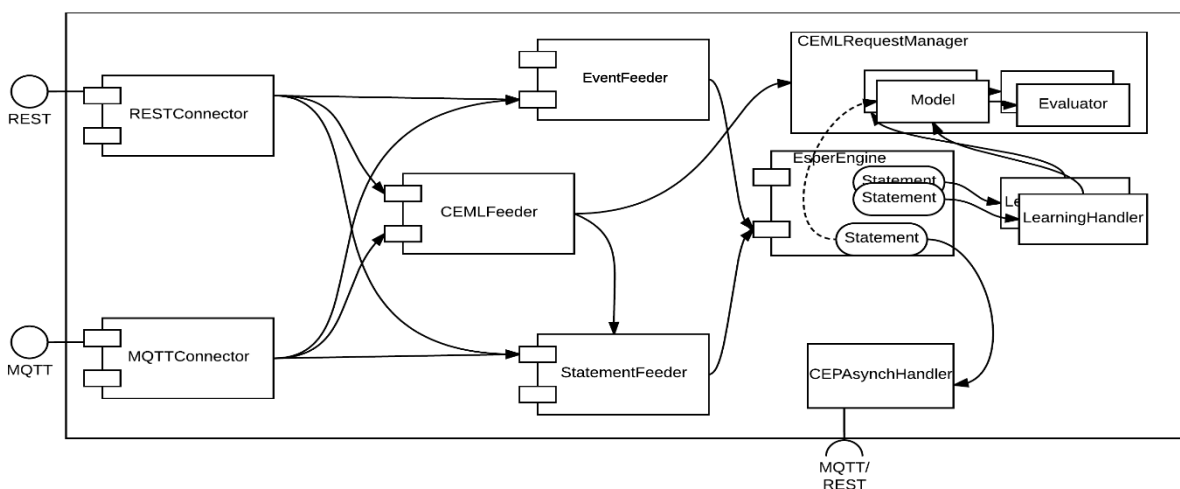


Figure 7. LinkSmart® Learning Service Architecture Sketch.

We utilize LinkSmart® LA following a modular architecture with loosely coupled modules responsible for different tasks. Figure 7 illustrates the architecture of the LA. The data and commands come via communication protocols implemented by Connectors (in Figure 7 shows two example implementations, REST and MQTT). The connectors transfer the information to the Feeders, which process the data accordingly to the API logic. This logic depends on whether it is an insertion of new Raw data, request of simple data processing (statement) or a machine learning request (CEML request). The data is inserted into the execution environment (in this case EsperEngine⁶), while the data processing requests are deployed in the same engine for the processing of the raw data. The CEML request has a more complex behaviour. Each CEML request is managed by its own CEMLManager, which contains and coordinates the model(s), evaluator for each model, and several statements. Finally, all outputs of any process (Statement) in the execution pipeline (EsperEngine) is captured or managed by a Handler. If the process should be prepared and sent through a communication protocol, then it will be handled by a Complex-Event Handler: An Asynchronous Handler, if the protocol is asynchronous (e.g. MQTT); or Synchronous Handler, if the protocol is synchronous (e.g. HTTP).

3.7 Scalability

In everyday growing amount of data, the scalability of the data processing system is paramount. Therefore, it is not only relevant but essential that the scalability of the system is accounted and tested. In this chapter, we revise the scalability of the system; while in next revision of this deliverable the scalability will be tested.

We will start by analysing the internal scalability of a single instance. The LA is a hyper-multithreaded service in which all operations are parallelized. For instance, each I/O operation is managed in individual thread, without

⁶ Esper is an open-source Java-based software product for Complex event processing (CEP) and Event stream processing (ESP) that analyzes series of events for deriving conclusions from them. See <http://www.espertech.com/>

waisting costly resources such as network NIC, sockets, and the thread itself. That a thread is reused means, each thread is not destroyed and reused for process new request after had finished to process the current one, saving the highly cost of building threads. Regarding network conectivity, the connections are shared to avoid the costly process of open sockets. On the other hand, the execution process are also parallelized per process. In summary, this allows a single instance to take all resources at hand in a machine as the data processing demands grows.

As the processing demands grows, it might be impossible or highly-conslty to address them in a single instance. Memory, CPU or NIC broadband usage can grow indefndly and scale them in a single server might not be the best approach. In such cases, the LA should be arranged in a mesh of data processing systems in a map-reduce fashion within the network. Such an arragenment allows to scale the infrastrucure depending on data processing demands, either by distribution or parallelization of the processing task (shown in figure below).

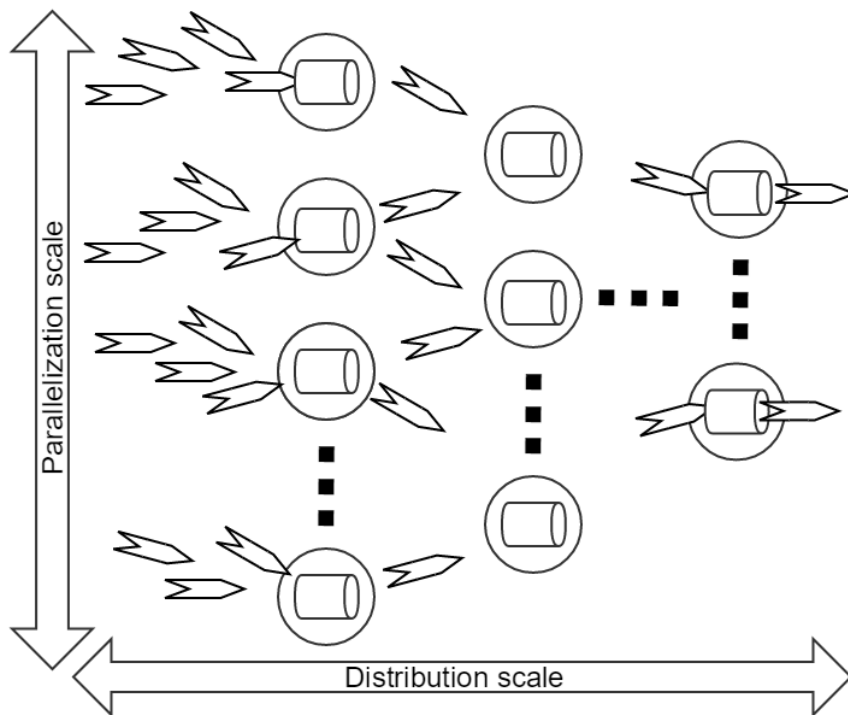


Figure 8 Scalability analys in a multi-instage processing platform

3.8 APIs Implementation

This chapter will describe the current API of the LA. To do so, we will describe the endpoints, operations and the payload send in the operation as following:

RES [PAYLOAD] DIR
 OP URI PAYLOAD

OP = operation to execute to operate with the LA.

URI = localization of the part of the API wants to be access. The URI may have values surrounded by '<' and '>', this are meta-values that describe values that the user must be field accordingly.

PAYLOAD = the document format use or returned in the operation. The format will have a suffix ending with ':' indicating the specification where the format is coming from, e.g. LS for LinkSmart or OGC for OGC SensorThings.

DIR = it is the direction of the communication. In some protocols the response of a request can be provided in the same operation, in other this is not possible. If → means that the communication is from the client to the LA. If ← means that the communication is from the LA to the client, usually a

late response of a request. \leftrightarrow Means that the response of the operations will be returned immediately after the operation overs.

[] = if the operation returns a payload, here is specified.

RES = result of the operation if successful.

3.8.1 I/O API:

The I/O APIs can be divided in two viewpoints. On one the hand, in networking communication protocols, and document event/message format. On the other hand, in the management of input or output information.

Regarding communication and format, the IoT Learning Agent builds upon standards; for both in and output. For network communications, currently, the LA supports MQTT and HTTP RESTful input/output messages, due to both are widely popular in IoT environments. In the messaging formats and endpoints definition, the LA base on OGC SensorThing, due to its wide and flexible description of the data. Moreover, the LA supports other formats, e.g. SenML, and the architecture is built to incorporate other network protocols. Although the LA supports several protocols, for simplicity this document just addresses the OGC SensorThings format (suffix OGC) and endpoints, and MQTT and HTTP.

Regarding the management of the Input and Output, the API is divided in two the Event API, and the Handling API, respectively.

3.8.1.1 Event API

This API allows events to be inserted in the “Computational Pipeline”/“Complex-Processing Engine” using their endpoints. Due to the asynchronicity nature of MQTT handles more properly this kind of operations.

MQTT Implementation

→

PUB *LS/<SW-Code>/<SW-ID>/OGC/1.0/Datastreams/<DS-ID>/* OGC:Observation

RESTful Implementation

200 [LS:Responses] \leftrightarrow

POST *LS/<SW-Code>/<SW-ID>/OGC/1.0/Datastreams/<DS-ID>/* OGC:Observation

Legend:

SW-Code = Accordingly to LinkSmart specification all component that sends data to the broker should have a software code. This can be a generic code ‘sen’ from sensor.

SW-ID = Accordingly to LinkSmart specification all component that sends data to the broker should have a unique ID.

DS-ID = Accordingly to OGC SensorThings specification the Observation should be send in a path/topic that contains the DataStream ID as is shown above.

3.8.1.2 Handling API

This API propagates the response of the result of the processing inside the “Computational Pipeline”/“Complex-Processing Engine”. In some sense, this API is the real return response of the **Event API**

MQTT Implementation

[OGC:Observation] ←

PUB *LS/<SW-Code>/<SW-ID>/OGC/1.0/Datastreams/<DS-ID>/*

RESTful Implementation

[OGC:Observation] \leftrightarrow

POST *LS/<SW-Code>/<SW-ID>/OGC/1.0/Datastreams/<DS-ID>/* OGC:Observation

3.8.2 Process API:

The process API has two aspects. Firstly, the external API that is use to define and deploy processing request of any kind. The internal API is serve by the Statement API for the simpler processing use cases, and the CEML API for the analysis use cases. Secondly, the internal API or SDK is used to add functionality to the LA that can be used in the external API.

3.8.2.1 Statement API

This API allows the user deploy instruction to process data on-demand on-the-fly. This operations can be Aggregations, Annotation, Routing, Fusion, and Translation of data streams as is explain in the use case chapter. It's important to mention that the requests are just that, requests to future process all incoming data. This means, the data will be process when and as it arrives and not when the request is deployed.

RESTful Implementation

Gets all statements created in the LA.

200 [LS:Responses] ←→

GET *statement/* LS:Statement

Gets the statement with the given ID.

200 [LS:Responses] ←→

GET *statement/<ID>* LS:Statement

Inserts a statement with the given ID.

201 [LS:Responses] ←→

PUT *statement/<ID>/* LS:Statement

Inserts new or changes an existing a statement (the id is auto-generated and returned).

201 [LS:Responses] ←→

POST *statement/* LS:Statement

Deletes a statement with the given ID.

200 [LS:Responses] ←→

DELETE *statement/<ID>/* LS:Statement

MQTT Implementation

Inserts a statement with the given ID.

→

PUB *LS/LA/<LA-ID>/SER/<LS-API-VER>/new/* LS:Statement

Inserts new or changes an existing a statement (the id is auto-generated and returned).

→

PUB *LS/LA/<LA-ID>/SER/<LS-API-VER>/add/* LS:Statement

Deletes a statement with the given ID.

→

PUB *LS/LA/<LA-ID>/SER/<LS-API-VER>/delete/* LS:Statement

To receive the responses of any of the MQTT requests described above, one of the following actions must have taken place before.

Subscribe to errors of specific Statement

[LS:Responses] ←

SUB *LS/LA/<LA-ID>/SER/<LS-API-VER>/errors/<ID>*

Subscribe to errors all errors related to the Statement API

[LS:Responses] ←

SUB *LS/LA/<LA-ID>/SER/<LS-API-VER>/errors/#*

LA-ID = the unique ID of the addresses LA.

LS-API-VER = the current API version number of LinkSmart.

ID = the unique ID of the Statement.

3.8.2.2 CEML API

RESTful Implementation

Gets all CEML requests created in the LA.

200 [LS:Responses] ←→

GET *ceml/*

Gets the CEML request with the given ID.

200 [LS:Responses] ←→

GET *ceml/<ID>*

Creates or changes a CEML request with the given ID.

201 [LS:Responses] ←→

PUT *ceml/<ID>/* LS:Statement

Inserts a new CEML request (the id is auto-generated and returned).

201 [LS:Responses] ←→

POST *ceml/* LS:Statement

Deletes a CEML request with the given ID.

200 [LS:Responses] ←→

DELETE *ceml/<ID>/* LS:Statement

MQTT Implementation

Inserts a statement with the given ID.

→

PUB *LS/LA/<LA-ID>/SER/<LS-API-VER>/ceml/add/* LS:CEMLR

Deletes a statement with the given ID.

→

PUB *LS/LA/<LA-ID>/SER/<LS-API-VER>/ceml/remove/* LS:CEMLR

To receive the responses of any of the MQTT requests described above, one of the following actions must have taken place before.

Subscribe to errors related to CEML Requests

[LS:Responses] ←

SUB *LS/LA/<LA-ID>/SER/<LS-API-VER>/ceml/errors/*

Subscribe the progress of the learning models

[LS:Prediction] ←

SUB *LS/LA/<LA-ID>/SER/<LS-API-VER>/output/<id>*

LA-ID = the unique ID of the addresses LA.

LS-API-VER = the current API version number of LinkSmart.

ID = the unique ID of the CEML request.

3.8.2.3 Extensible Model framework

The amount of models that can be applied varies widely depending on the application. I.e. in COMPOSITION the models are mostly provide by the Deep Learning Toolkit. This are external components that connect in some manner to the IoT Learning Agent framework. Therefore, the IoT Learning Agent provides an initial set of models to use; additionally, the LA provides two set of SDK to interconnect the external developed models.

3.8.2.3.1 GeneralWekaModel

Waikato Environment for Knowledge Analysis (Weka) is a suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. It is free software licensed under the GNU General Public License. This model allows to use any model or algorithm available in Weka suite that implements an `UpdateableClassifier`.

3.8.2.3.2 AutoRegressiveNeuralNetworkModel

In statistics and signal processing, an autoregressive (AR) model is a representation of a type of random process; as such, it is used to describe certain time-varying processes in nature, economics, etc. The autoregressive model specifies that the output variable depends linearly on its own previous values and on a stochastic term (an imperfectly predictable term); thus the model is in the form of a stochastic difference equation. This model is an implementation of such model using neural networks library form DeepLearning4j.

3.8.2.3.3 Java SDK

The LA is able to load in runtime any model that is located in the classpath and implements the either `RegressorModel <Input, Output, LearningObject>` or `ClassifierModel <Input, Output, LearningObject>` for regression problems or classification problems, respectively.

3.8.2.3.4 Python SDK

The Python SDK is a fully new feature of the LA developed for and in COMPOSITION. Python is one of the most popular language between Data Scientist, i.e. the Deep Learning Toolkit is fully implemented in python. Therefore, an SDK to add models implemented in python was implemented for the COMPOSITION project. The SDK allows to connect to standalone models using Pyro. Pyro establish a network connection between a Java software and a Python application handling all related to this (DNS discovery, socket connection, object translation, error handling, TLS, etc.). In this manner, any standalone python can connect to the LA in a transparent manner.

3.8.3 Monitoring API

This API is still work in progress task for the LinkSmart® IoT Learning agent. Currently, the monitoring features are powerful but the tooling might be unappropriated or difficult to use. The current functionality, allows to monitoring allows to evaluate the performance of the LA or a network of LAs using standard REST or MQTT clients leveraging on the existing APIs.

Firstly, the LA allows to monitor the CPU usage of the statements giving the possibility to monitor the computational performance of the processing Statements. To do so the, the `'connector_monitoring_mqtt_events_report_topics'` must be enable. This feature will create an MQTT event every 60 seconds (by default) where the usage of each Statement.

Secondly, the LA allows to monitor the learning performance of the model, as was mention in the CEML MQTT API implementation. This allows to get an instant performance assessment of the model. Additionally, the model can be consulted directly using the HTTP GET `ceml/<id>` endpoint. This endpoint will provide the latest state of the model. This allows to access the latest state of the model by the demand.

The most important feature of the LA regarding monitoring, it is that due to the structure of the MQTT topic allows to multicast CRUD requests. This means, it is possible to connect to unlimited amount of LA agents in a broker and be able to monitor their performance individually online at once. This allows to construct a

monitoring infrastructure of a mesh of LA either in a Cloud deployment or in distributed locations in a network or the internet.

Finally, the LA work natively together with any LinkSmart® Platform service. This means that LA can connect to the LinkSmart® Service Catalog, allowing to monitor the LA or LAs status with the catalogue. Additionally, the DataStreams status can be monitor using the HDS OGC SensorThing Catalog. In this manner, all DataStreams metadata coming from sensors, Statements, or CEML request can be queried.

3.8.4 Management API

The Management API is in similar case as the Monitoring API. The Management API leverage of other existing API. With two important remarks. Again, the structure of the MQTT topic allows to multicast CRUD requests. This features allows to address an infrastructure of LAs distributed anywhere, addressing in a single multicast request all, some or one LA in the distributed infrastructure. This means, Statements or CEML requests can be created, updated or deleted in subsets of LAs in a deployed infrastructure with a single request.

3.9 Security

The LinkSmart® IoT Learning Agent is developed in the scope of the LinkSmart® Platform and adapted to match the COMPOSITIONS needs. The agent allows natively simple security management leveraging from SSL/TLS and MQTT standard. Additionally, more advance features can be offer by other services, either by e.g. the LinkSmart® IoT Border Gateway or by other set of tools. This last is the case of COMPOSITION. In this chapter, we will just discuss the internal security features provided by the agent alone and the agent in a combined infrastructure.

3.9.1 TLS

TLS is the secure version of the TCP protocol. Currently, the LA supports three communication network protocols for different proposes. These are HTTP, MQTT and Pyro. This last one, it is actually not a protocol just a TCP socket application, here we will treated so. Although, the three of them are based on TCP, their application is extremely different. From the perspective of the agent, in HTTP the agent is a server using several TCP connections. In MQTT, the agent is a client of a server (the broker). Finally in Pyro, the connection is a single TCP socket that connect a single client. The usage of the TLS is different in each other.

3.9.1.1 HTTPS

HTTPS is the usage of TLS in the HTTP protocol context. HTTP is request/response client/server communication protocol, where clients can request to the services resources using endpoints. In most cases, the LA takes the role of the server. Loading a certificate in the LA truststore, it will allow to establish encrypted connection to the incoming clients. Additionally, it is possible to provide client certificates, with this is possible to create whitelist for the clients and blocking anonymous access. This approach is highly secure, but it might be cumbersome to maintain. Finally, there is no fine granularity regarding what a client can or cannot do after the access was given.

3.9.1.2 MQTT and TLS

In this paragraph, we discus only TLS while using MQTT, for other security mechanism see below. MQTT is a publish/subscribe client-to-client broker based communication protocol, where the clients can publish or subscribe to messages using topics. The role of the LA regarding this protocol is as client. Therefore, the security enforcement can be only applied on the server side, the broker. The LA gives the possibility to load client certificate for each MQTT Broker that wants to be established. As client, the certificate is not needed for encryption but for whitelisting.

MQTT provides additional basic mechanism authentication. MQTT allows to define user/password to authenticate clients in the broker. In it is worth mentioning that user/password authentication do not imply TLC, ergo no encryption. Moreover, no TLS means that the communication can be sniffed this including the user/password. Therefore, using TLS is a must for secure user/password communication. Finally, although MQTT standard do not define authorization mechanism; many implementations provide this feature.

Finally, the MQTT Broker implementations can only manage who publishes or subscribes to a topic, but it is impossible to verify who send a message if the topics are shared. In other to identify sender, we build JWS messages on top as payload of the messages.

3.9.1.3 Pyro

As it was mentioned before, Pyro simply opens a TCP connection such that the external Python code can connect to a Java implementation. To establish a TLS connection in Pyro, the server certificates must be loaded in the LA. This is similar as what is needed to be done in HTTPS.

3.9.2 JWS

JWS or JSON Web Sign, it is a complete new feature developed in COMPOSITION. This new standard allows to sign and validate all sent and received messages by and from the agent respectively. Combining MQTT with JWS allows the services to identify the origin of the message and if the message had been tampered. In the perspective of the LA, it is possible to enable this feature and all APIs will work accordingly to JWS.

3.10 LinkSmart development synergies

The IoT Learning Agent is a standalone service of the LinkSmart® and as such shares some common set of standards, specifications, best practices and infrastructure that all components belonging to the LinkSmart® infrastructure has. Appliances of usage of these set of technologies are not limited to the result of the developed service but also to the development process of the service itself. All components developed in the LinkSmart® ecosystem are developed using agile SCRUM process with issue tracking and continuous integration. To support this, LinkSmart® has a set of cloud infrastructure that supports the developers and ensures quality. Overall, this set of tools improves the quality of the COMPOSITION project. In this chapter, we present the set of best practices that the LA is built on as part of the LinkSmart® ecosystem.

3.10.1 Continuous Integration best practices

This section lists best practices suggested by various authors on how to achieve continuous integration, and how to automate this practice. Build automation is a best practice itself.

Continuous integration – the practice of frequently integrating one's new or changed code with the existing code repository – should occur frequently enough that no intervening window remains between commit and build, and such that no errors can arise without developers noticing them and correcting them immediately.[11] Normal practice is to trigger these builds by every commit to a repository, rather than a periodically scheduled build. The practicalities of doing this in a multi-developer environment of rapid commits are such that it is usual to trigger a short time after each commit, then to start a build when either this timer expires, or after a rather longer interval since the last build. Many automated tools offer this scheduling automatically.

Another factor is the need for a version control system that supports atomic commits, i.e. all of a developer's changes may be seen as a single commit operation. There is no point in trying to build from only half of the changed files.

To achieve these objectives, continuous integration relies on the following principles.

3.10.1.1 Maintain a code repository

This practice advocates the use of a revision control system for the project's source code. All artefacts required to build the project should be placed in the repository. In this practice and in the revision control community, the convention is that the system should be buildable from a fresh checkout and not require additional dependencies. The mainline (or master) should be the place for the working version of the software.

In LinkSmart® DevOps infrastructure, this task is done by our code Bitbucket server located in <https://code.linksmart.eu/>. In the Figure 9, we can see the dev and master branches of the LA project in code server, both with green status showing that both had been successfully tested.

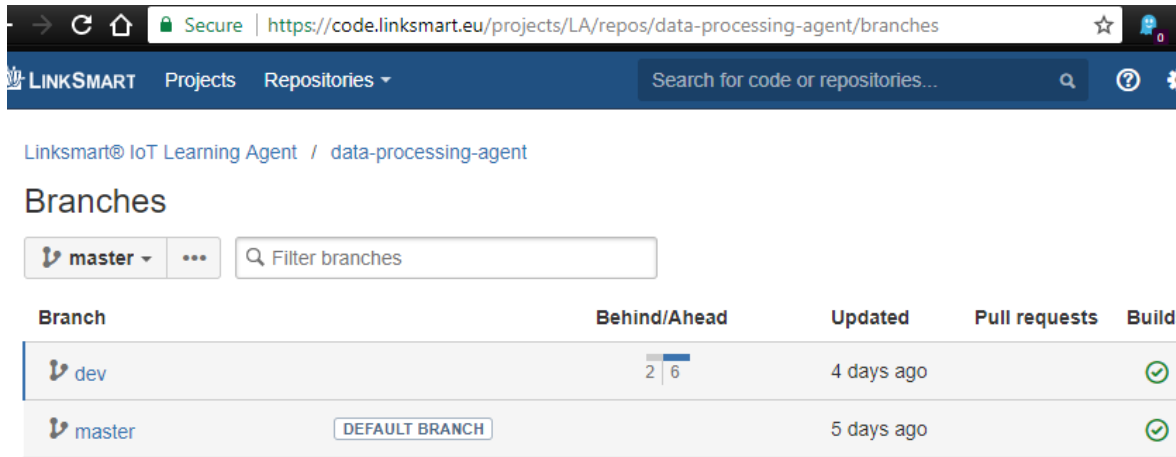


Figure 9 screenshot of the LA project in code server

3.10.1.2 Automate the build

A single command should have the capability of building the system. Many build tools, such as make, have existed for many years. Other more recent tools are frequently used in continuous integration environments. Automation of the build should include automating the integration, which often includes deployment into a production-like environment. In many cases, the build script not only compiles binaries, but also generates documentation, website pages, statistics and distribution media.

In LinkSmart® DevOps infrastructure, this task is done by our pipeline Bamboo server located in <https://pipelines.linksmart.eu/>. In the Figure 10, we can see the summary of an automatic build (#199) with all internal stages of the build.

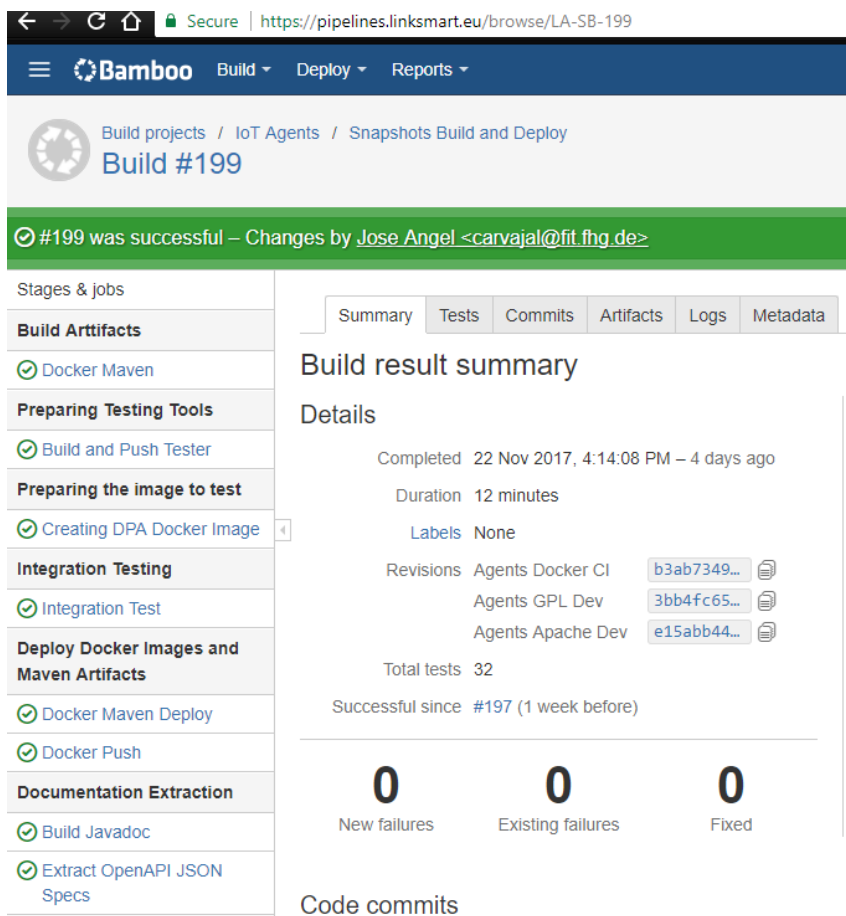


Figure 10 the summary of the automatic build no. 199

3.10.1.3 Make the build self-testing

Once the code is built, all tests should run to confirm that it behaves as the developers expect it to behave.

In LinkSmart® DevOps infrastructure, this task is done by our pipeline Bamboo server located in <https://pipelines.linksmart.eu/>. The server apply unit and component tests for each change done in the software. In the Figure 11, we can see the summary of the unit and component tests applied in build no. 199. With this test, we expect to ensure the behaviour of the different components is the expected one.

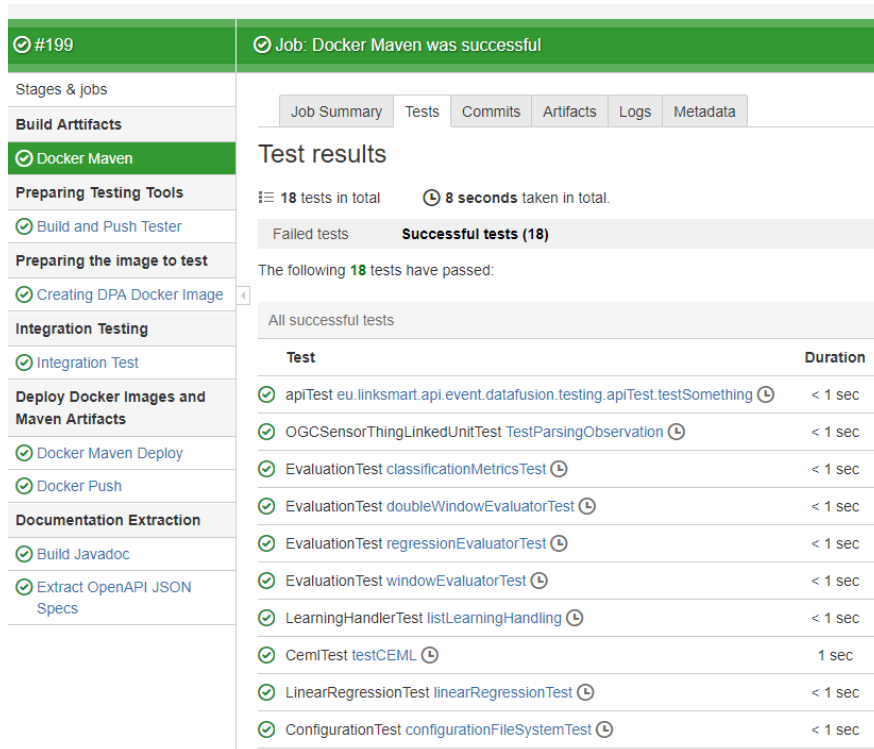


Figure 11 shows the results of the unit and component tests of the build no. 199

3.10.1.4 Everyone commits to the baseline every day

By committing regularly, every committer can reduce the number of conflicting changes. Checking in a week's worth of work runs the risk of conflicting with other features and can be very difficult to resolve. Committing all changes at least once a day (once per feature built) is generally considered part of the definition of Continuous Integration. In addition performing a nightly build is generally recommended. These are lower bounds; the typical frequency is expected to be much higher.

In LinkSmart® DevOps infrastructure, this task is done by our code Bitbucket server located in <https://code.linksmart.eu/>. In the Figure 12Figure 9, we can see the frequency of the commits and that the master branch is use only of current stable release and dev for development.

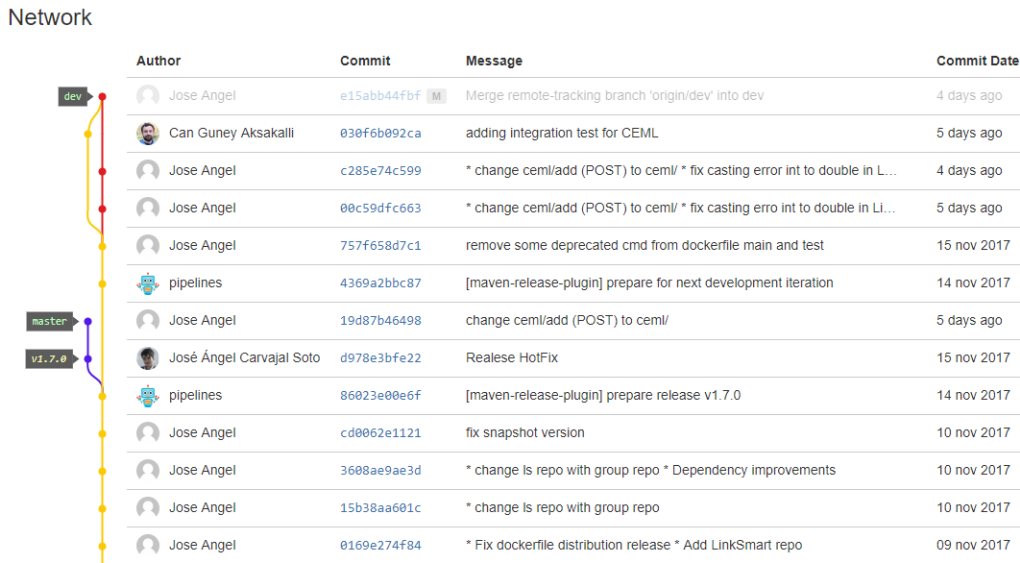


Figure 12 shows the usage of the branches in the GIT repository

3.10.1.5 Every commit (to baseline) should be built

The system should build commits to the current working version to verify that they integrate correctly. A common practice is to use Automated Continuous Integration, although this may be done manually. For many, continuous integration is synonymous with using Automated Continuous Integration where a continuous integration server or daemon monitors the revision control system for changes, then automatically runs the build process.

In LinkSmart® DevOps infrastructure, this task is done by our pipeline Bamboo server located in <https://pipelines.linksmart.eu/>. The server build every change made in the code and check if runs. In Figure 13, we can a summary of the last 3 automatic build processes.

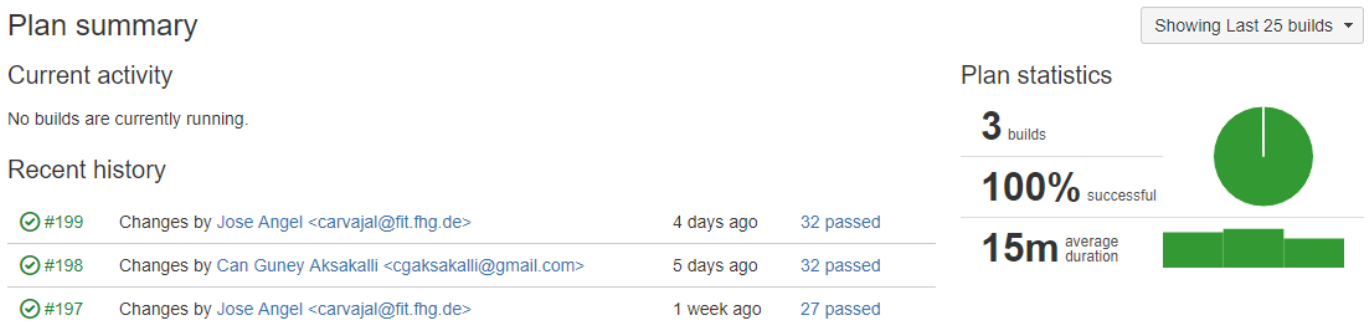


Figure 13 shows how each change triggers a building and testing process

3.10.1.6 Test in a clone of the production environment

Having a test environment can lead to failures in tested systems when they deploy in the production environment because the production environment may differ from the test environment in a significant way. However, building a replica of a production environment is cost prohibitive. Instead, the test environment, or a separate pre-production environment ("staging") should be built to be a scalable version of the actual production environment to both alleviate costs while maintaining technology stack composition and nuances. Within these test environments, service virtualization is commonly used to obtain on-demand access to dependencies (e.g., APIs, third-party applications, services, mainframes, etc.) that are beyond the team's control, still evolving, or too complex to configure in a virtual test lab.

In LinkSmart® DevOps infrastructure, this task is done by our pipeline Bamboo server located in <https://pipelines.linksmart.eu/>. The server apply integration tests for each change done in the software. In the

Figure 14, we can see the summary of the integration tests applied in build no. 199. With this test, we expect to ensure the API respect a predefined contract.

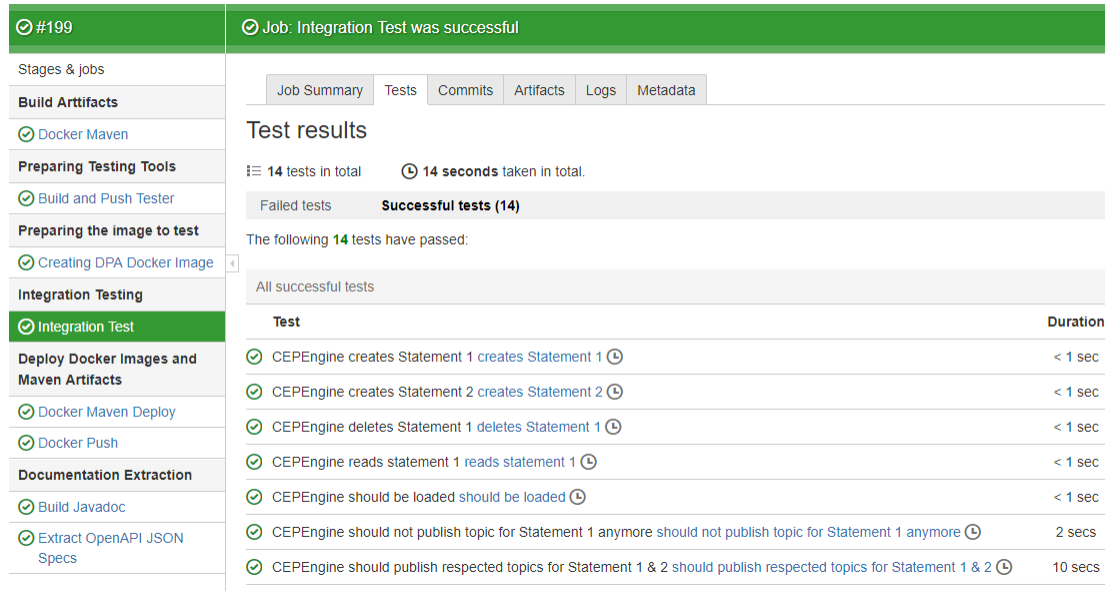


Figure 14 shows the results of the integration and system tests of the build no. 199

3.10.1.7 Make it easy to get the latest deliverables

Making builds readily available to stakeholders and testers can reduce the amount of rework necessary when rebuilding a feature that doesn't meet requirements. Additionally, early testing reduces the chances that defects survive until deployment. Finding errors earlier also, in some cases, reduces the amount of work necessary to resolve them. All programmers should start the day by updating the project from the repository. That way, they will all stay up to date.

In LinkSmart® DevOps infrastructure, this task is done by our nexus server, which provides the Docker registry service located in <https://docker.linksmart.eu/> for the Docker image distributions; and the maven repository service located in <https://nexus.linksmart.eu/repository/public/> for the java components and distributions. Both the registry and the repository contains the experimental, snapshot and release distributions of the Docker and Java. In the Figure 15, we can see in the left side the Docker distributions, and the right side we can see the release java distributions.

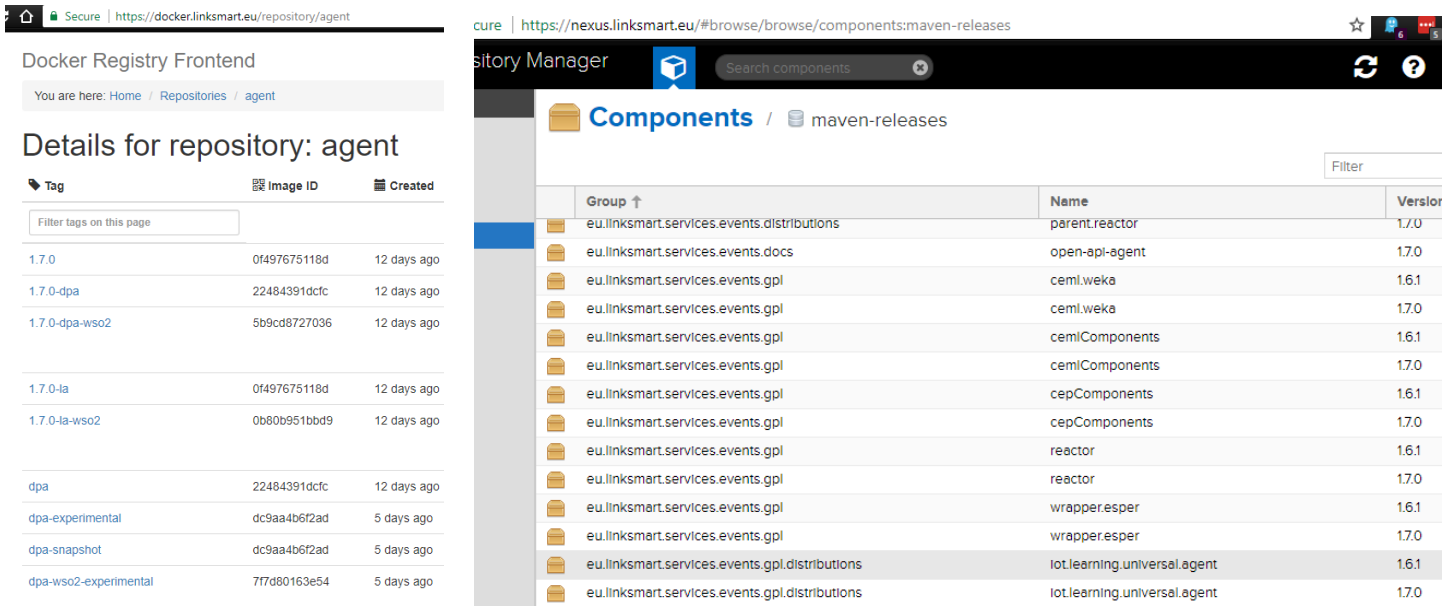


Figure 15 (left) shows the latest release, snapshot and experimental distribution Docker distributions in the LinkSmart® Docker registry server; (right) shows the latest two releases as Java applications in the LinkSmart® Nexus server.

3.10.1.8 Everyone can see the results of the latest build

It should be easy to find out whether the build breaks and, if so, who made the relevant change.

In LinkSmart® DevOps infrastructure, this task is done by our pipeline Bamboo server located in <https://pipelines.linksmart.eu/> and the boards' server located in <https://boards.linksmart.eu/>. The pipelines server builds every time a change happens and each build trace who and when the change took place (see Figure 16). Finally, in case the build fails; the pipeline server informs the developers that the fails happened and creates a report in the on the management board in the board server till the issue was solved.



Figure 16 shows the changes involved in the build no. 199

3.10.1.9 Automate deployment

Most CI systems allow the running of scripts after a build finishes. In most situations, it is possible to write a script to deploy the application to a live test server that everyone can look at. A further advance in this way of thinking is continuous deployment, which calls for the software to be deployed directly into production, often with additional automation to prevent defects or regressions.

In LinkSmart® DevOps infrastructure, this task is done (partially) by our pipeline Bamboo server located in <https://pipelines.linksmart.eu/>. While the server already deploy automatically into a building test system, it do not deploy it automatically into the production server due to administrative, security and other issues. In the Figure 17, we can see a list of automatic deployment plans.

All Deployment Projects






Deployment project	Environment	Release
Historical Datastore Release	LinkSmart Docker Registry	0.4.0 
IoT Agents Deployment Phase 1	LinkSmart Nexus Repository	1.7.0 
	LinkSmart Git Server	1.7.0 
IoT Agents Deployment Phase 2	LinkSmart Docker Registry	1.7.0 
	LinkSmart Nexus Repository - OpenAPI Specs	1.7.0 

Figure 17 list of current release automatic deployment plans

4 Big Data Visual Analytics

This chapter presents the first results of Task 5.1 related to Visual Analytics tools and techniques. In these first stages of the project, the research was focused on the analysis of the existing visualisation techniques, the specification of Visual Analytics tool architecture in overall system architecture and the creation of a high level view of the Visual Analytics tool. Besides this, some visual analytics techniques and tools have already been applied for visual analysis of the pilot partners' industrial data.

The COMPOSITION Visual Analytics tool should be able to import data from Big Data Analytics tools and Simulation tool. Based on the data of the aforementioned tools the Visual Analytics tools will apply visual analytics techniques and will present the output to the users as graphical representations. The Visual Analytics tools will provide the ability to manufacturers/end-users to evaluate the simulation results and identify possible problems.

4.1 Visualization Techniques Analysis

4.1.1 Overview of Visualization Techniques Categories

A thorough analysis of visualization techniques has been conducted at the first stage of this task. A summary of the results of this analysis is presented in this section.

Based on the aforementioned analysis we have distinguished the following categories of visualization techniques:

- **Standard 1D to 3D graphics** are widely used techniques for the visualization of data models. They are also used to view the frequency distribution of an attribute or to view the estimation of the certainty about a hypothesis. The most representative examples of Standard 1D to 3D graphics techniques are the scatter and contour plots, the line and stack graphs, the histograms and the pie and bar charts. Furthermore, combinations of the previous examples such as histogram with lines and bars with lines are also widely used.
- **Graph-based or hierarchical techniques** are used in numerous applications in the field of information visualization. They are structures which represent data relationships. To achieve this representation node-link diagrams are used. Examples of graph-based or hierarchical techniques are graphs, three maps and cone trees.
- **Pixel oriented** techniques map each data value to a coloured pixel and present the data values on the display screen in separate windows. Each window presents the data values belonging to one attribute. Query-independent technique is an example of Pixel oriented techniques. In this technique the data is sorted according to some attributes and a screen-filling pattern is used in order to arrange the data values on the display.
- **Geometric techniques** are also commonly used for the visualization of information. These techniques provide an overview of all attributes by mapping the multidimensional data into a two-dimensional plane. A scatter plot matrix is an example of Geometric technique. This matrix shows relationships and dependencies among several variables by taking a pair of them at a time. A Kiviat Diagram is another well-known example of the Geometric techniques visualization category. A Kiviat diagram is composed of axes extending from a central point and represents how multiple items compare when they are evaluated against more than two variables.

4.1.2 Parameters Define Visualization Technique's Selection

In order to select the most appropriate visualization technique to use, a set of parameters should be considered:

- **Data type** is a basic criterion for choosing a visualization technique. Moreover, the Data type parameter can be used as a criterion for visualization techniques' classification. Based on the current status of the project the following data types will be into consideration:
 - continuous or discrete quantitative data
 - data represented from time series
 - multi-dimensional data
 - and spatial data

- **Task type** is another determining factor for classifying visualization techniques. The Task type parameter is related to the activities that an end-user is able to perform. These activities refer to visualization capabilities according to his goals. Generally, the main functionalities that a user requires from a visualization tool are the following:
 - Comparison of attributes or correlation among the attributes
 - Data Overview
 - Identification of possible patterns or clusters
 - Outliers detection
- **Dimensionality** is a parameter to be observed in the data before applying a visualization technique. It refers to the number of data attributes. Dimensionality is classified based on the number of the attributes in three basic categories:
 - Small (up to 4 attributes)
 - Medium (5 to 9 attributes)
 - High (more than 10 attributes)
- **Scalability** of data is a characteristic should be considered for the selection of visualization techniques. The Scalability of data is classified in three categories:
 - Small (10^1 to 10^2)
 - Medium (10^3 to 10^5)
 - High (10^6 to 10^7)

4.1.3 Brief analysis of general visualization tools

The COMPOSITION visual analytics tool will support the analysis of large volumes of data, related with maintenance decision support, recyclables materials management and predictive maintenance processes. These functions will allow the visualization of possible correlations among predictive maintenance features, probabilities of upcoming types of machine faults, graphs of recyclables materials management, tools for decision support based on generative algorithms etc. Because of the fact that there are different needs for each pilot use case, despite general visualizations (line plots, pies charts, scatter plots, bar charts (single and with line plots), histograms (single and with line plots), bubble charts, Kiviati diagrams etc.), the visual analytics will provide targeted analysis with tools specially crafted for each use case

A brief description of some general visualization tools is given below:

- **Line plots:** A line plot is most useful in displaying data or information that changes continuously over time.
- **Pie charts:** A pie chart is a circular chart divided into sectors, illustrating proportions.
- **Scatter Plots:** A scatter plot is a type of mathematical diagram using Cartesian coordinates to display values for two variables for a set of data. The data is displayed as a collection of points, each having the value of one variable determining the position on the horizontal axis and the value of the other variable determining the position on the vertical axis.
- **Bar charts (single):** A bar chart is a chart with rectangular bars with lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally.
- **Bar charts with line plots:** Combination of bar charts and line plots
- **Histograms (single):** A histogram is a graphical representation showing a visual impression of the distribution of data.
- **Histograms with line plots:** Combination of histogram and line plots
- **Bubble charts:** A bubble chart is a type of chart that displays three dimensions of data.
- **Kiviati diagrams:** A Kiviati diagram is composed of axes extending from a central point. Each axis represents a data category. Each axis is scaled according to certain parameter values. It is used to graphically represent, on a single diagram, how multiple items compare when they are evaluated against more than two variables.

Based on the data availability and the needs of the pilots, additional visualization tools like contour plot and stack graph will be added on the visual analytics platform.

4.2 Overview of Visual Analytics tool architecture

Figure 18 depicts the architecture of COMPOSITION Visual Analytics (VA) Platform with its main functionalities. The results of data analysis made by the components of COMPOSITION IIMS Simulations and Forecasting Tools and Big Data Analytics Tools will be the main input of COMPOSITION VA Platform. The platform will have a support pre-processing component for data cleansing, transformation and normalization prior the use of data structures. Data cleaning techniques are applied to “clean” the data by filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies.

The VA Platform will support several data structures (e.g. linear structures, graphs, tables, trees etc.) so that they can be accessed and used efficiently and effectively by the general visualization modules (line plots, pies charts, scatter plots, bar charts etc.) and VA modules (clustering, probabilistic techniques, adaptive heatmaps, graphs etc.). The COMPOSITION VA Platform will provide input to the Decision Support System so as to improve decision making for business scenarios.

An API will be offered in order to provide the COMPOSITION VA Platform output to DSS. The DSS will be able to call the provided web services via HTTP requests and get the graphs. Then the DSS will be able to visualize the graphs to end users in order to support the decision making services.

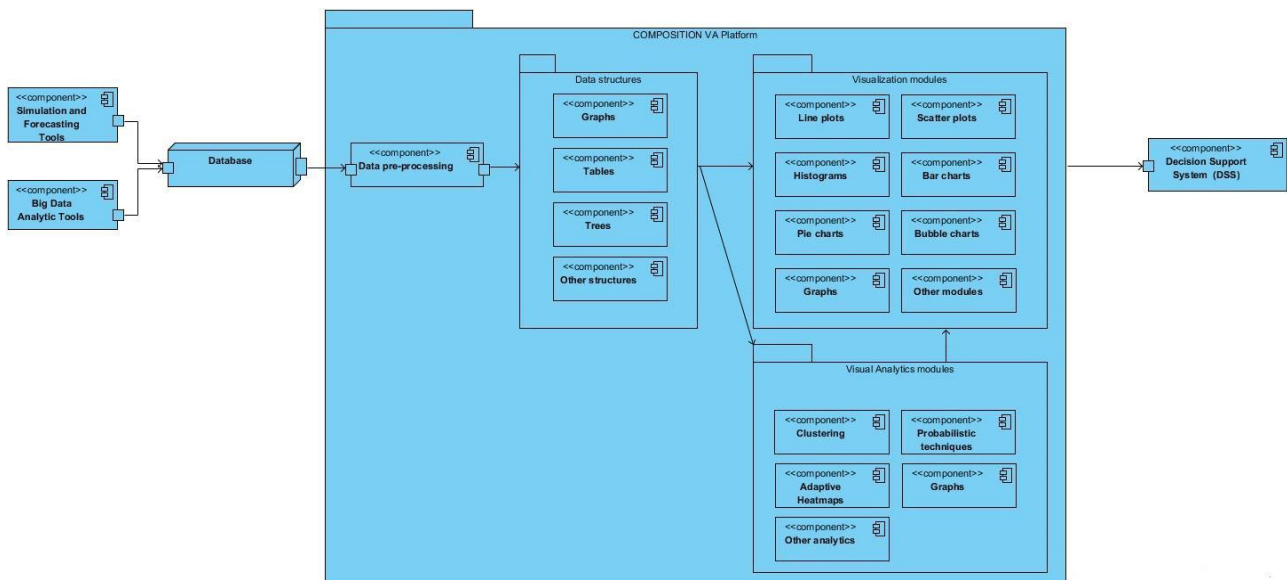


Figure 18 COMPOSITION Visual Analytics Platform architecture

4.3 Visual Analytics tools in COMPOSITION Use Cases

As we mention before during the first period of the Task 5.1 the research related to visual analytics was focused on the state of the art analysis and the architecture’s definition. Based on the sub-section 4.1.2 of this chapter, in order to select the appropriate visual analytics techniques the scalability and dimensionality of data along with data and task types, must be defined in advanced. To be able to achieve this, the Big Data Analytics and the Simulation tool should be able to export an efficient analysis. While this criterion is satisfied, the data can be evaluated and visual analytics tools and techniques can be selected and applied. Of course all the methods and techniques developed so far will be modified in the future based on new data entries and availabilities. The visual analytics techniques applied so far are presented below:

Correlation heatmaps for UC-BSL-2 Predictive Maintenance

For BSL Predictive Maintenance use case *Correlation Heatmaps* where developed so as to provide all possible correlations among the fans of the machine, during the three different stages of its operation: *wake up*, *work under load (rest)* and *cooldown* (see Figure 19).

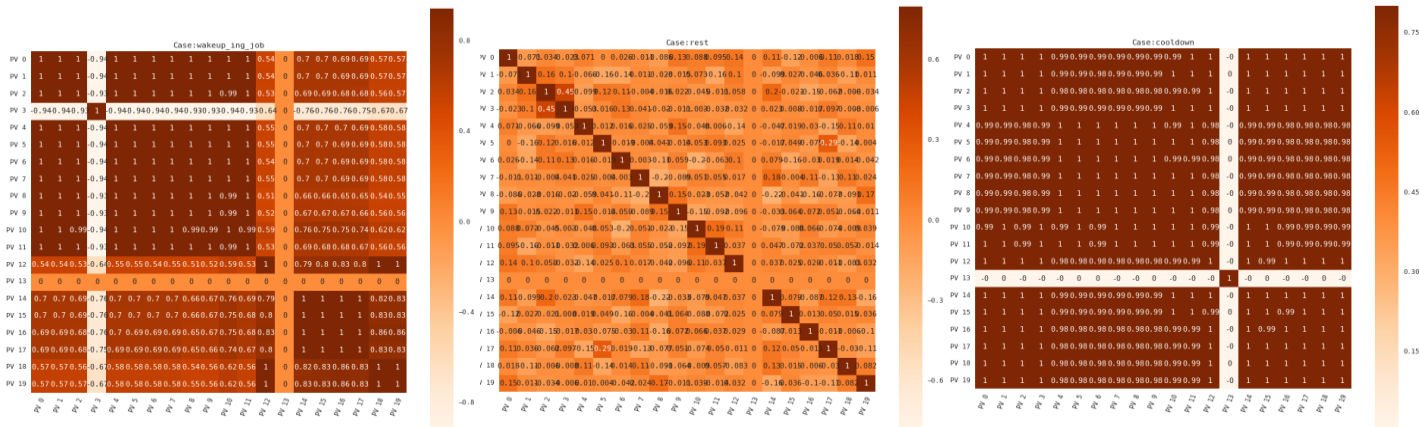


Figure 19 Correlation heatmaps for the three different stages of machine's operation.

Some of the results extracted from this correlation heatmaps plots is that: dark red areas show a direct (positive) correlation trend, whiter areas show an indirect (negative) correlation trend and orange areas indicate no apparent correlation between the fans of the machine. During wake up and cooldown variables are correlated in an obvious manner as they begin/stop functioning concurrently. Under load, fans work independently.

K-partite graph for UC-ELDIA-1 Contractual Wood and Recyclable materials (paper, plastic) Management

In graph theory, a part of mathematics, a k-partite graph is a graph whose vertices are or can be partitioned into k different independent sets. Equivalently, it is a graph that can be colored with k colors, so that no two endpoints of an edge have the same color. For UC-ELDIA-1 *k-partite graph* is used so as to provide a hint about the internal connections between several features of the waste transportation dataset (i.e. year, month, description of material, customers and transported tonnage).

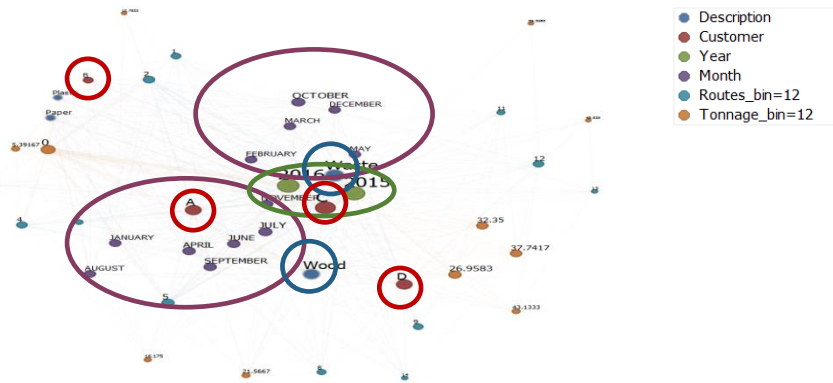


Figure 20 : K-partite graph visualization tool

Some of the results extracted from this k-partite graph is that: year (green) doesn't provide any additional information, wood and waste (materials – blue) seems to circulate more than plastic and scrap, all the customers (red) are way different from each other. As for the working months (purple), year seems to “break” in two periods (see Figure 20). Other visualizations of the data, based on k-partite graph are depicted below. For these visualizations the features that not provide any additional (or useful) information are extracted from the analysis (see Figure 21).

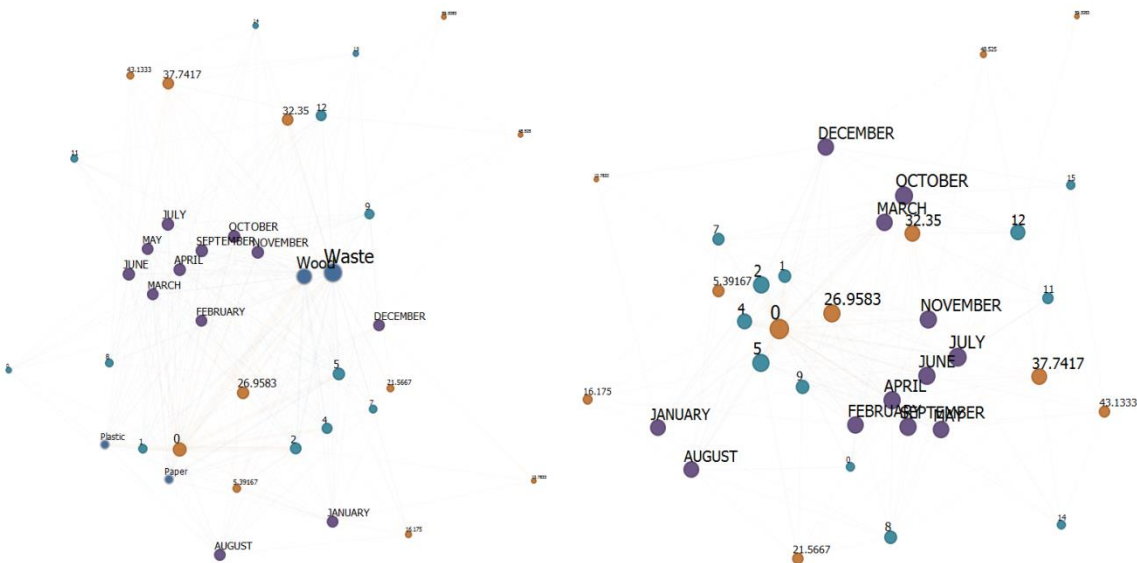


Figure 21 K-partite graph using additional features. (Left) Transported tonnage with Materials and Months. (Right) Transported tonnage with Months

Probabilities of Fault to Follow Fault (FtFF) for UC-KLE-1 Maintenance Decision Support

For KLE maintenance decision support use case *Probabilities of Fault to follow Faults (FtFF)* where developed so as to provide all possible transitions from one type of fault to another. In this use case there are three types of faults: electrical (=1), hydraulic (=2) and mechanical (=3). Thus, utilizing the time series of faults and markov models, the probability of the type of fault that will follow an existing is calculated in real time. There are nine (9) different types of FtFF and are given in Table 3 below:

Electrical to Electrical	Hydraulic to Electrical	Mechanical to Electrical
Electrical to Hydraulic	Hydraulic to Hydraulic	Mechanical to Hydraulic
Electrical to Mechanical	Hydraulic to Mechanical	Mechanical to Mechanical

Table 3: Faults transitions

The visualizations of FtFF is given in Figure 22 below, where in x-axis is the real time and in y-axis is the calculated probability for each transition.

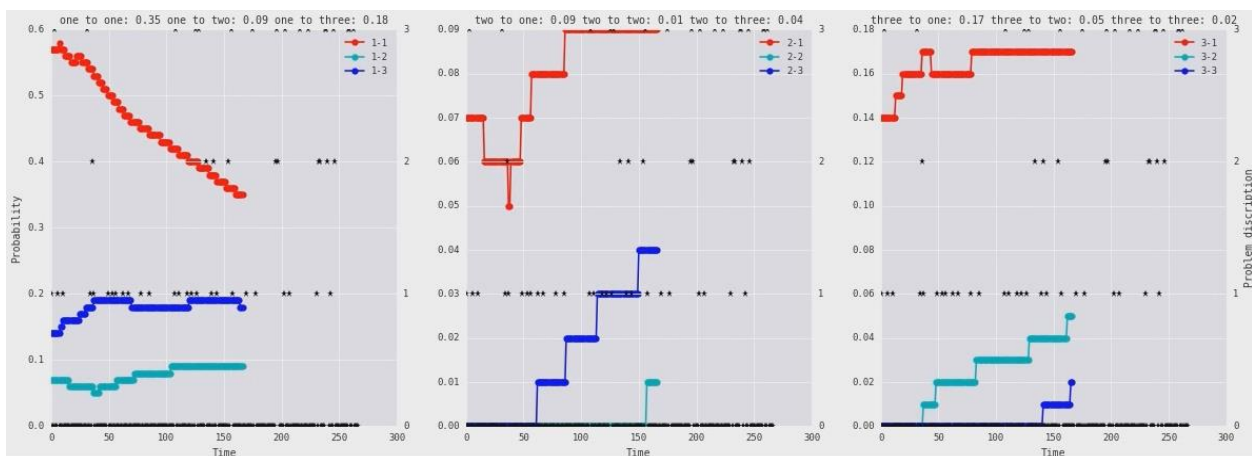


Figure 22 Real time probability calculation of Fault to Follow Fault

Along with FtFF visualization tools, a scenario based probability calculation is also developed based on the original idea of FtFF. This scenario based probability calculation is called *Fault to Follow Scenario (FtFS)* and

calculates the probability of the existence of a fault based on specific scenarios, where a number of day(s) is followed by a day without any fault ($X = 0$) or by a day with a fault X ($X = 1,2,3$), where 1,2 and 3 are denote the type of a fault. There are four (4) different types of FtFS created so far and are given in Table 4 below:

A/A	Scenario
1	One (1) day without situation X
2	Two (2) days without situation X
3	Five (5) days without situation X
4	Ten (10) days without situation X

Table 4: Scenarios of a number of day(s) followed by a situation.

The visualizations of FtFS is given in Figure 23 below, where in x-axis is the real time and in y-axis is the calculated probability for each situation X , for the 4 scenarios described above.

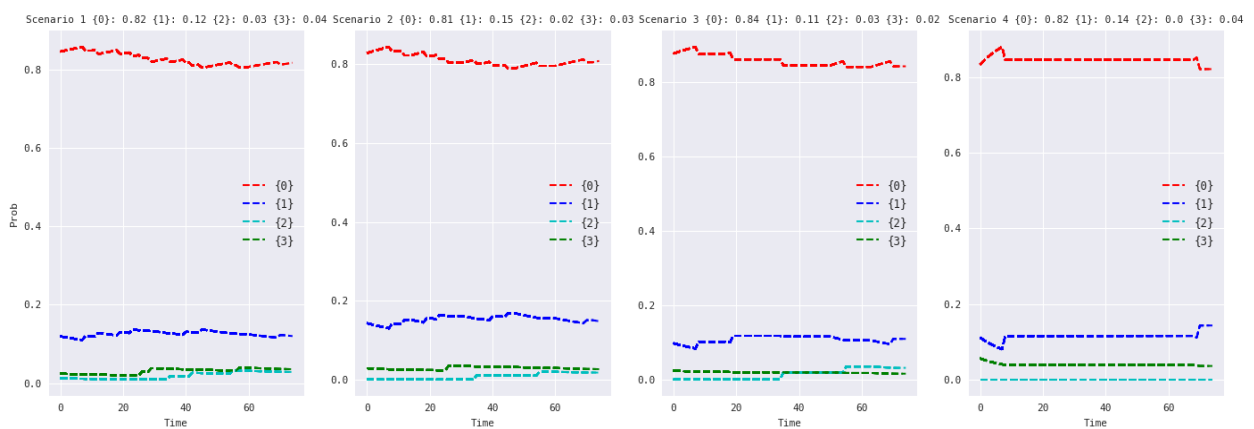


Figure 23 Real time probability calculation of Fault to Follow Scenario

5 Future work and Conclusions

5.1 Future Work

5.1.1 Integration the real-time data of the Pilot partners

Although the COMPOSITION components are deployed in the cloud, and we have received data form our partners. The real-time data streams coming from the production-line is not in place yet (Task 5.5). In order to test the proper integration of the measurements coming from the production plant, we need to get the real-time data integrations.

5.1.2 Integrating the LA with the DLT

Although the LA (T5.1) and the DLT (T5.2) are coexisting in the same cloud deployment and are connect via a common broker, there is no interconnection of messages yet. The communication cannot be in place without the feeding of real-rime data coming from the pilot partners as it was mention in 5.1.1. Additionally, although the data of the LA is OGC compliant; the data generated is strictly speaking no sensor data. Therefore, the communication between LA and DLT must be agreed between the work tasks T5.1 and T5.2. For the first stage of the project, the communication between the components will be using OGC SensorThings MQTT messages without JWS. Moreover, the communication between this two components will be revised in the near future to identify if there is more adequate protocol stack for the communication between this components.

5.1.3 Enabling TLS and JWS

At the time this deliverable was written, the TLS feature of the LA was not enable yet, due to lacking of configuration parameter and certificates to do so. This will be configured in near future. Additionally, the JWS was recently incorporate in the project, therefore, not all components has this feature yet. This is the reason why the feature is not enabled in the LA yet although its already implemented. As soon the real-time data is available, the TLS and JWS protocols will be enabled to secure the connection and data transference.

5.1.4 Integrate the LA with the Visual Analytics

Similarly as with the DLT and LA, the Visualisation tools developed in this task had not been connected with each other. As the data flows and the use case evolve the visualization tools will be incorporated to fulfil the needs of the pilot partners' scenarios.

5.1.5 Test the Scalability of the LAs

It might happened that the scenarios implemented for the pilot partners do not evaluate enough scalability of the developed system. Therefore, it is important to evaluate how the system scales while the pilot grows to become a stable big data analytics infrastructure. In the next deliverable update, a scalability test of the system will be provided for the evaluation of it.

5.1.6 Improve monitoring and management tooling

The current, the Monitoring and Management APIs are powerful but cumbersome. This can be improve by providing tools that uses and facilitates the mentioned APIs.

5.2 Conclusion

The LinkSmart® IoT Learning Agent has walk a log road since the first conceptual developed as Data-Fusion Manager in the ALMANAC project. The software has evolve and increasing its features and still been in use showing it relevance and usefulness in several domains and use cases. E.g. water leakage detection, waste container fill-level detection, waste accumulation prediction in Smart Cities scenarios; presence improvement detection and presence forecasting in Smart Buildings scenarios; real-time energy aggregation by business logic, defect detection and now in predictive maintenance in Industry 4.0 scenarios.

6 List of Figures and Tables

6.1 Figures

Figure 1. Intra-factory interoperability layer components and dependencies.....	6
Figure 2 Design view of a single instance of CEML execution system	11
Figure 3 Design perspective of multiple CEML execution units	12
Figure 4 Structure of the API	13
Figure 5 Deployment view of the CEML distributed system	14
Figure 6. LinkSmart® Learning Service Enabling Use Cases.....	15
Figure 7. LinkSmart® Learning Service Architecture Sketch.	17
Figure 8 Scalability analys in a multi-instace processing platform	18
Figure 9 screenshot of the LA project in code server	25
Figure 10 the summary of the automatic build no. 199	25
Figure 11 shows the results of the unit and component tests of the build no. 199	26
Figure 12 shows the usage of the branches in the GIT repository.....	27
Figure 13 shows how each change triggers a building and testing process	27
Figure 14 shows the results of the integration and system tests of the build no. 199.....	28
Figure 15 (left) shows the latest release, snapshot and experimental distribution Docker distributions in the LinkSmart® Docker registry server; (right) shows the lates two releases as Java applications in the LinkSmart® Nexus server.....	29
Figure 16 shows the changes involved in the build no. 199	29
Figure 17 list of current release automatic deployment plans	30
Figure 18 COMPOSITION Visual Analytics Platform architecture	33
Figure 19 Correlation heatmaps for the three different stages of machine's operation.	34
Figure 20 : K-partite graph visualization tool	34
Figure 21 K-partite graph using additional features. (Left) Transported tonnage with Materials and Months. (Right) Transported tonnage with Months	35
Figure 22 Real time probability calculation of Fault to Follow Fault	35
Figure 23 Real time probability calculation of Fault to Follow Scenario	36

6.2 Tables

Table 1 Analysis between the phase who captures, where are executed and implemented.....	11
Table 2 Mapping between users and APIs	13
Table 3: Faults transitions.....	35
Table 4: Scenarios of a number of day(s) followed by a situation.	36

7 References

- [1] R. J. Calantone and C. A. Di Benedetto, "Performance and time to market: Accelerating cycle time with overlapping stages," *IEEE Transactions on Engineering Management*, pp. 232-244, 2000.
- [2] S. M. Davis, "From "future perfect": Mass customizing," *Planning review*, pp. 16-21, 1989.
- [3] F. T. Piller, K. Moeslein and C. M. Stotko, "Does mass customization pay? An economic approach to evaluate customer integration," *Production planning & control*, pp. 435-444, 2004.
- [4] O. K. Mont, "Clarifying the concept of product- service system," *Journal of cleaner production*, vol. X, no. 3, pp. 237-245, 2002.
- [5] McKinsey Digital, "Industry 4.0 How to navigate digitization of the manufacturing sector," McKinsey Digital, 2015.
- [6] P. D. H. Kagermann, P. D. W. Wahlster and D. J. Helbig, "Recommendations for implementing the strategic initiative INDUSTRIE 4.0," National Academy of Science and Engineering, Frankfurt (Main), Germany, 2013.
- [7] M. Cheeseman, P. Swann, G. Hesketh and S. Barnes, "Adaptive manufacturing scheduling: a flexible and configurable agent-based prototype," *Production Planning & Control*, pp. 479-487, 2005.
- [8] P. Leitão and F. Restivo, "ADACOR: A holonic architecture for agile and adaptive manufacturing control," *Computers in industry*, pp. 121-130, 2006.
- [9] A. Nassehi, S. Newman and R. Allen, "TEP-NC compliant process planning as an enabler for adaptive global manufacturing," *Robotics and Computer-Integrated Manufacturing*, pp. 456-467, 2006.
- [10] M. Pellicciari, A. O. Andrisano, F. Leali and A. Vergnano, "Engineering method for adaptive manufacturing systems design," *International Journal on Interactive Design and Manufacturing (JIDeM)*, pp. 81-91, 2009.
- [11] J. Lee, B. Bagheri and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing Letters*, pp. 395-412, 2015.
- [12] V. Vyatkin, Z. Salcic, P. S. Roop and J. Fitzgerald, "Now That's Smart!," *IEEE Industrial Electronics Magazine*, pp. 17-29, 2007.
- [13] F. S. Fogliatto, G. J. Da Silveira and D. Borenstein, "The mass customization decade: An updated review of the literature," *International Journal of Production Economics*, pp. 14-25, 2012.
- [14] D. Pham and A. Afify, "Machine-learning techniques and their applications in manufacturing," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, pp. 395-412, 2005.
- [15] D.-M. Tsai and R.-H. Yang, "An eigenvalue-based similarity measure and its application in defect detection," *Image and Vision Computing*, pp. 1094-1101, 2005.
- [16] M.-H. C. Li, A. Al-Refaie and C.-Y. Yang, "DMAIC approach to improve the capability of SMT solder printing process," *IEEE Transactions on Electronics Packaging Manufacturing*, pp. 126-133, 2008.
- [17] T.-W. Hui and G. K.-H. Pang, "Solder paste inspection using region-based defect detection," *The International journal of advanced manufacturing technology*, pp. 725-734, 2009.
- [18] G. Acciani, G. Brunetti and G. Fornarelli, "Application of neural networks in optical inspection and classification of solder joints in surface mount technology," *IEEE Transactions on industrial informatics*, pp. 200-209, 2006.
- [19] G. J. Vachtsevanos, I. M. Dar, K. E. Newman and E. Sahinci, "Inspection system and method for bond detection and validation of surface mount devices". USA Patent 5,963,662, 1999.
- [20] J. Á. Carvajal Soto, M. Jentsch, D. Preuveneers and E. Ilie-Zudor, "CEML: Mixing and Moving Complex Event Processing and Machine Learning to the Edge of the Network for IoT Applications," *Proceedings of the 6th International Conference on the Internet of Things*, pp. 103-110, 2016.
- [21] D. Bonino, J. A. Carvajal Soto, M. T. Delgado Alizo, A. Alapetite, T. Gilbert, M. Axling, H. Udsen and M. Spirito, "Almanac: Internet of things for smart cities," *2015 3rd IEEE International Conference on Future Internet of Things and Cloud (FiCloud)*, pp. 309-316, 2015.
- [22] G. Cugola and A. Margara, "Processing flows of information: From data stream to complex event processing," *ACM Computing Surveys (CSUR)*, p. 15, 2012.

- [23] C. Andrieu, N. De Freitas, A. Doucet and M. I. Jordan, "An introduction to MCMC for machine learning," *Machine learning*, pp. 5-43, 2003.
- [24] M. M. Gaber, "Gaber, Mohamed Medhat; Krishnaswamy, Shonali; Zaslavsky, Arkady," *On-board Mining of Data Streams in Sensor Networks*, pp. 307-335, *Advanced Methods for Knowledge Discovery from Complex Data*.
- [25] N. A. Syed, S. Huan, L. Kah and K. Sung, "Incremental learning with support vector machines," *Citeseer*, 1999.
- [26] A. P. Dawid, "Present position and potential developments: Some personal views: Statistical theory: The prequential approach," *Journal of the Royal Statistical Society. Series A (General)*, pp. 278-292, 1984.
- [27] S. V. Stehman, "Selecting and interpreting measures of thematic classification accuracy," *Remote Sensing of Environment*, pp. 77 - 89, 1997.
- [28] M. Axling, D. Bonino, D. Ioannidis, N. K. Kaklanis, A. Nizamis, P. Vergori, M. Pardi, G. Insolubile, J. Benedicto, J. Romero, J. Á. Carvajal Soto, J. Liang, P. Kool, M. Ahlsen and P. Rosengren, "D2.3 The COMPOSITION Architecture Specification I," COMPOSITION Consortium, 2017.
- [29] COMPOSITION, "GRANT AGREEMENT 723145 — COMPOSITION: Annex 1 Research and innovation action," 2016.