D12.3
FITMAN Extended Smart Factory lessons learned and evaluations (first)

Document Owner: D. Stojanović
Contributors: A. Stojanović, A. Stojadinović, B. Stajić, J. Zlatanović, D. Kostić, K. Fischer, R. Stühmer, D. Riemer, B.Kämpgen
Dissemination: Public
Contributing to: WP12
Date: 07/05/2015
Revision: 1.0

VERSION HISTORY

VERSION	DATE	NOTES AND COMMENTS
0.1	02/03/2015	TOC PROPOSAL AND CONTRIBUTIONS DEADLINE
0.2	06/03/2015	INITIAL CONTENT PROVIDED FOR SOME SECTIONS
0.3	13/03/2015	FIRST VERSION
0.4	28/04/2015	DYVISUAL
0.5	04/05/2015	UPDATE OF SECTION 3
0.6	06/05/2015	READY FOR REVIEW
0.7	07/05/2015	INTERNAL REVIEW
1.0	07/05/2015	FINAL VERSION

Table of Contents

EXECUTIVE SUMMARY 6
1 INTRODUCTION 7
1.1 OBJECTIVES OF DELIVERABLE D12.3 7
1.2 STRUCTURE OF THE DELIVERABLE 9
2 DYNAMIC COMPLEX EVENT PROCESSING (CEP) – DYCEP 10
2.1 INTRODUCTION 10
2.2 OUR APPROACH 12
2.3 ANALYSIS OF DATA SETS - DATA DECOMPRESSION 14
2.4 ANOMALY DETECTION ON DECOMPRESSED DATA AND IMPLEMENTATION ON COSMOS 20
2.5 ANALYSIS OF THE RESULTS AND LESSONS LEARNED 28
3 DYNAMIC VISUALIZATION AND INTERACTION - DYVISUAL 33
3.1 TRW AUTOMOTIVE USE CASE 33
3.2 WHIRLPOOL USE CASE 35
3.3 LESSONS LEARNED 35
4 CONCLUSION 37

Figures

Figure 1: Final FITMAN Platform for SFs composition....8
Figure 2: Control chart example....10
Figure 3: Considering two parameters of a process separately....11
Figure 4: Multivariate analysis – Considering correlation between parameters....11
Figure 5: Considering two parameters of a process separately....12
Figure 6: Clustering based anomaly detection – The idea....13
Figure 7: Training/Detection difference....14
Figure 8: Entity/Relation model....15
Figure 9: Fields and measurement values in a compressed form for each of the parameters – Power, Speed, Total Water Inlet....16
Figure 10: Decompression result....16
Figure 11: Missing values in decompressed data....17
Figure 12: Incorrect data check....17
Figure 13: Repeating values....18
Figure 14: Power....18
Figure 15: Speed....19
Figure 16: Total Water Inlet....19
Figure 17: Correlation between power and rotation speed....20
Figure 18: Correlation between power and water inlet....20
Figure 19: Power parameter of multiple tests....21
Figure 20: Comparison of two signals – red signal is shifted for one point comparing to blue signal....23
Figure 21: Comparison of two signals – red signal is shifted even more....23
Figure 22: Comparison of two signals – counter clockwise rotation lasts longer....24
Figure 23: Comparison of two *aligned* signals....24
Figure 24: *Warping* of two signals....25
Figure 25: Difference between a *medoid* (yellow point) and *centroid* (orange point) in case of an outlier presence in the dataset....25
Figure 26: How many clusters question....26
Figure 27: Anomaly detection on COSMOS....26
Figure 28: Anomaly detection architecture....28
Figure 29: Generated medoids....29
Figure 30: First cluster....29

Figure 31: Second cluster 30
Figure 32: First anomaly 30
Figure 33: Second anomaly 31
Figure 34: Third anomaly 31
Figure 35: User Interface of the ErgoPal RISK PREVENTION Tool 34
Figure 36: Avatar indicating the problematic pose with the affected joint 34
Figure 37: Deviation Map of a Whirlpool Part in XML3D 35

Executive Summary

This document reports on the experiments which were conducted with two FITMAN Specific Enablers (SE) developed within WP12: DyCEP and DyVisual SEs. These SEs were applied in two use case scenarios of two FITMAN trials: Whirlpool and TRW Automotive.

Regarding DyCEP, we considered clustering based anomaly detection for Whirlpool functional test analysis use case. The goal was to find a way to define normal parameter values and to implement a solution that will be able to identify unusual patterns in functional tests. We discovered anomalies as behaviour that does not conform to the defined model. We analysed different clustering algorithms that could be used for that and implemented a system that is able to automatically detect anomalies, based on these algorithms.

The implementation has been done in two phases. The goal of the 1st phase was to implement anomaly detection on a single machine and to perform analysis on a subset of data to validate our methods. The goal of 2nd phase is to implement anomaly detection based on MapReduce that will run on COSMOS cluster for Big data. While COSMOS solution implementation is in progress, we have implemented single machine solution and performed validation on a small subset.

Based on results of our analysis, we conclude that our algorithm represents a good solution for the problem of detecting anomalies in functional tests provided by Whirlpool. We even got a confirmation that one of the tests represents a problem that really exists in one of Whirlpool facilities, which they are aware of. We developed a solution that is able to compare test series based on their shape, and to cluster tests based on it. We also used that similarity and clusters being produced to determine which tests look unusual comparing to normal examples found in the dataset.

Our next step is to finish MapReduce implementation that was started, so the analysis could be performed on a Hadoop (COSMOS) cluster. The implementation and the 2nd evaluation will be reported in the deliverables D12.4 and D12.5 respectively.

The summary of the lessons learned from applying DyVisual in the discussed applications is that the key aspect for the use of DyVisual is the underlying data and its interpretation. In the first place it is important that data is available but also regarding the interpretation of the data the processing of the data of DyCEP has to provide the data in an appropriate manner together with indications on how the results should be visualized. The latter needs of course only to be agreed on once when the process is implemented.

We are in the process of improving ease of use of DyVisual. This includes providing DyVisual as a service and to allow the user to more easily adapt the resulting visualization. Additionally, we consider removing obstacles like ports which need to be opened for remote access to the prototype implementations.

1 Introduction

1.1 Objectives of Deliverable D12.3

This document reports on the experiments which were conducted with the FITMAN Specific Enablers (SE) designed in Task T12.1 and implemented in Task 12.2.

The following text summarizes the objectives of the work planned in the MagniFI proposal for extending SEs for Smart Factories.

- *Obj1: To develop the **conceptual model of the solution for managing/increasing real-time situational awareness in smart factories** based on FITMAN infrastructure (MagniFI Platform)*
  - *It will consist of several components and should enable continuous sensing from different, distributed, heterogeneous sources, keeping the semantic meaning of each event in order to enable their efficient integration.*
  - *Since we want to be open for the inclusion of any source type (fixed sensors, mobile tags, wearable sensors, production real-time data, log data, etc.) semantic interoperability is a critical issue.*
- *Obj2: To provide a **novel approach for Dynamic CEP for detecting situation of interests**, that will ensure a huge agility of a manufacturing system, by*
  - *supporting an early discovery of warnings that might indicate some problems/opportunities, their dynamic monitoring through the reconfiguration of the detection process, in order to ensure their efficient detection;*
  - *enabling an efficient management of the dynamics of the detection process based on various analysis of the real-time and historical data, in order to ensure that the system will continuously (self) improve the quality of the performances.*
- *Obj3: To provide a **novel approach for real-time 3D visualization** of situation of interests that will support better decision making, through:*
  - *enabling innovative visualization of real-time events and historical data with Web technologies to support users in their decisions on how to react to complex events in the shopfloor;*
  - *supporting an attractive human-machine interaction through the contextualized 3D visualization of the complex situations, that will empower managers on different levels to make right and timely decisions.*
- *Obj4: To **realize all components by relaying on the FITMAN infrastructure and integrating through the FI-WARE GEs**, including the extensions of GEs for the purpose of an efficient realization of the proposals vision (MagniFI Platform).*
- *Obj5: To **test the MagniFI solution in at least three FITMAN trials** (Experimentation), incl. the specification of the lessons learned for the FI PPP community, esp. for Phase 3 projects*

This deliverable reports on activities performed to achieve Obj5. Indeed, in this deliverable we have analysed and evaluated the new experimentations developed in WP12/T12.2 in the light of identifying lessons learned (bottlenecks and opportunities) for a further expansion of

the selected smart factory (SF) trials. Our focus was on two SEs that have been developed in the scope of WP12, namely *Dynamic CEP (DyCEP)* and the *Dynamic Visualization and Interaction (DyVisual)*. These SEs are a part of the FITMAN Smart Factory platform that is shown in Figure 1.

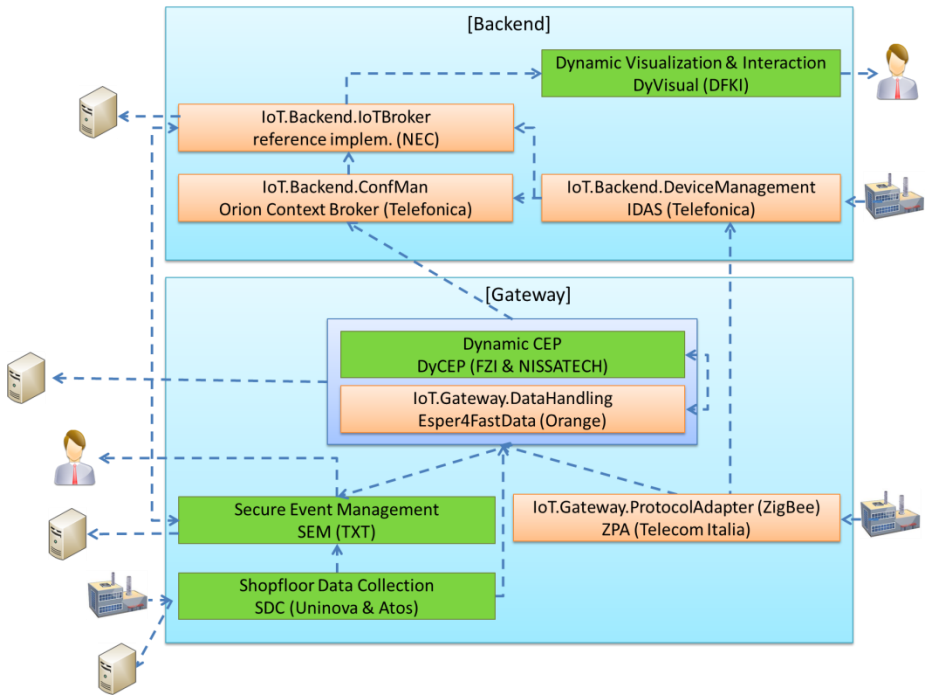

Figure 1: Final FITMAN Platform for SFs composition

The software components that compose the Smart Factory platform[1] are 5 FI-WARE GEs and 4 FITMAN SEs:

- In reference to the Generic Enablers, the *IoT.ProtocolAdapter*, a component that enables the communication with IoT Devices implementing ZigBee specification, the *IoT.DataHandling* which manages Data Handling in the Gateway, the *IoT.IoTBroker*, a middleware component enabling applications to retrieve aggregated information from IoT installations and the *IoT.ConfMan* which is an implementation of NGSI9 and NGIS10 with persistence, the *IoT.Backend.DeviceManagement*, which is a central component for the IoT backend providing the resource-level management of remote assets as well as core communication capabilities.
- In regard to SEs, the Smart Platform integrates, the *Shop floor Data collection (SDC)*, which provides mechanisms for data capture on RFID tags and for integrating wireless sensor networks, the *Secure Event Management (SEM)*, adding mechanisms for events distribution in a controlled and safety manner, the *Dynamic CEP (DyCEP)*, which enables complex real-time processing of the shop-floor events and other events relevant for manufacturing process (e.g. from MES), and the

[1] http://www.fitman-fi.eu/fitman-reference-platforms/fitman-smart-factory-platform

Dynamic Visualization and Interaction (DyVisual) in charge to provide visualization and synchronization functionalities for different clients, in particular web clients.

In the rest of this deliverable we describe two new SEs of the refined SF architecture have been evaluated.

1.2 Structure of the deliverable

Section 2 reports on the experiments done with the DyCEP SE through the analysis of the different clustering algorithms and the implementation of a system that is able to automatically detect anomalies based on these algorithms.

Sections 3 summaries the lessons learned from applying DyVisual the context of the FITMAN SF trails TRW and Whirlpool.

Section 4 summarizes the results presented in this deliverable.

2 Dynamic Complex Event Processing (CEP) – DyCEP

2.1 Introduction

The need to observe and control processes goes hand in hand with manufacturing. Monitoring a process enables us to identify early indicators of "out of control" behaviour and react momentarily. By doing so we are able to:

- Dramatically reduce variability and scrap
- Improve productivity
- Reduce costs
- Uncover hidden process personalities
- Instantly react to process changes
- Make real-time decisions on the shop floor

SPC (*Statistical Process Control*) was the number one choice for process control for many years. It relies on the following assumption - "While every process displays variation, some processes display controlled variation, while others display uncontrolled variation". The idea is to determine standard deviation of the process and use it to define boundaries of normal behaviour (variation that could be tolerated), which can be represented in a graphical form – *control charts*. An example of one such control chart is presented Figure 2.

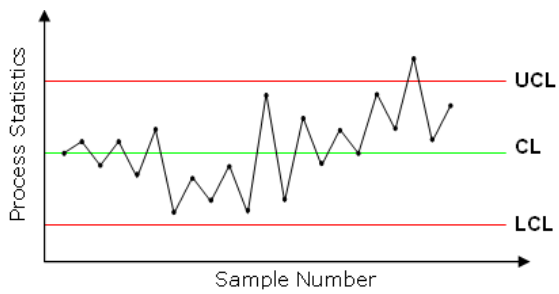

Figure 2: Control chart example

Even though SPC is well adopted, it does show some imperfection in specific cases. First of all, complexity of control charts usage greatly increases with the number of parameters increase. It becomes impractical to observe a large number of parameters side by side using such charts. Also, it is hard to correlate parameter values, and that correlation could explain process behaviour. Figure 3 and Figure 4 illustrate the value of multivariate analysis. Second, it is impossible to allow different variation during different periods of process execution. Deviation is found considering the process as a whole, but it is not strange to encounter processes that have certain *stages*, during which process behaves differently. Next, the practice has shown that many companies define the boundaries once and use them for years, not considering that there could be a shift during time in what is considered to be normal. Also, those boundaries are found by experts, which makes understanding and adoption of process control even harder. Finally, we live in the age of *Big data* that very often has a fast, streaming form, and comes from endless sources. That requires new methods for analysis, more scalable and robust than the ones used for decades.

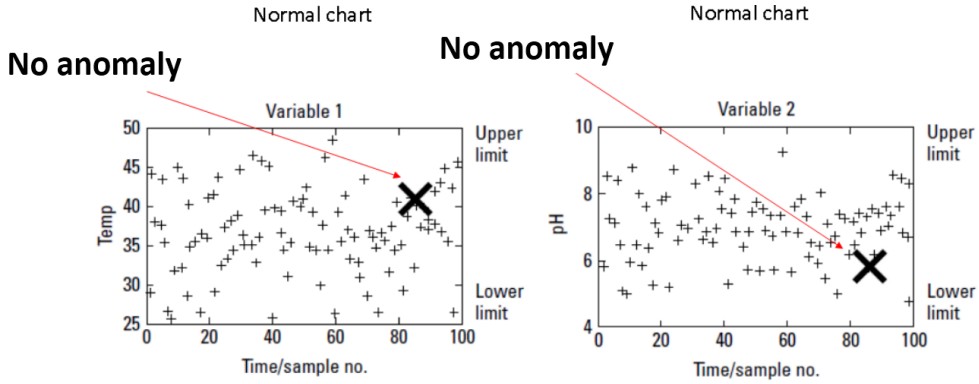

Figure 3: Considering two parameters of a process separately

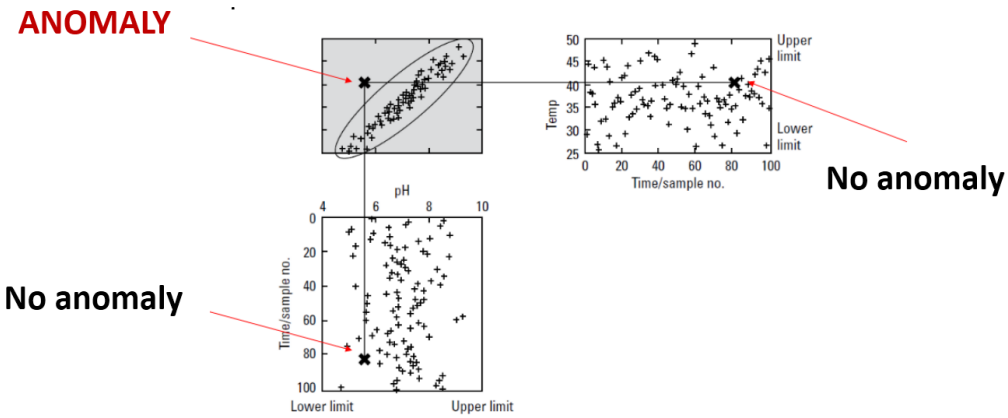

Figure 4: Multivariate analysis – Considering correlation between parameters

Large amount of data enables us to learn what is normal behaviour from the data itself, using sophisticated machine learning algorithms. Anomaly detection algorithms discover patterns in data that do not conform to a well-defined notion of normal behaviour. They represent a well-studied group of algorithms that has use in many areas such as manufacturing, fraud detection and health. There several approaches for detecting anomalies, based on:

- Clustering
- Classification
- Nearest neighbour techniques
- Statistical approaches
- Information theory

In following we consider clustering based anomaly detection for Whirlpool functional test analysis use case. Clustering is the task of grouping a set of objects in such a way that objects in the same group (*cluster*) are more similar to each other than to those in other groups. Briefly, some advantages of clustering based techniques are:

- Clustering is an unsupervised machine learning technique (which implies that examples of anomalies are not necessary, only normal examples need to be provided). It is really hard (practically impossible) to provide examples of all possible types of anomalies, since anomalies can be defined as something never seen before. Clustering based approaches avoid this problem as an unsupervised technique

- Clustering enables us to work with multivariate data
- Anomaly detection based on clustering is fast – a model of normal behaviour is produced and detection comes to comparing measurements with generated model (which can be really fast).

An illustration of clustering based anomaly detection is presented on Figure 5 giving a quick intuition about our approach.

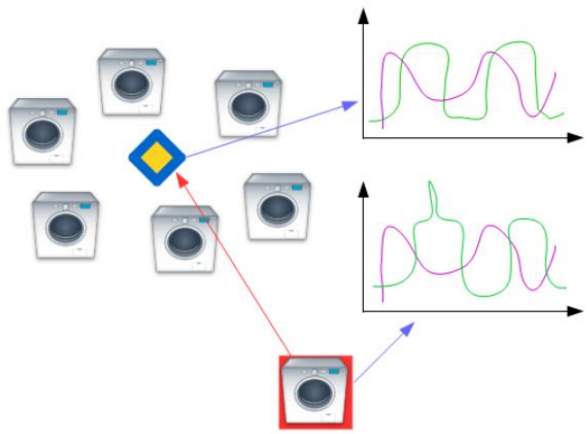

Figure 5: Considering two parameters of a process separately

Next, we describe Whirlpool functional test analysis use case and we give a detailed description of our analysis.

2.2 Our approach

In the following section Whirlpool use case is described. The problem setup is as follows:

- Washing machines functional tests are provided by Whirlpool
- Functional test is performed on every washing machine that was assembled and it is used to examine if machine functions properly
- Various parameters are measured, such as power, speed and water inlet
- The goal is to detect anomalies – washing machines that behaved *strangely* during functional test
- By detecting anomalies during functional tests quality of production can be increased
- Data is provided in a compressed form, so the first step is to decompress it
- Analysis should be performed on original (decompressed) data
- Based on analysis a solution for detecting anomalies automatically should be implemented
- The solution needs to be able to handle *Big data*
- COSMOS should be used as a platform for the solution.

As already explained, to identify anomalies we are using clustering based methods that define normal behaviour and find anomalies as behaviour that differs from what is considered to be normal. The general idea is to perform clustering over the whole dataset and find *clusters* of normal behaviour, as well as *representatives* of that clusters. In that way we are able to *learn* what the normal behaviour from the system itself is, no experts are needed. The idea is presented on Figure 6. Every blue point represents a functional test in our case. Clustering is performed on all the tests and as a result clusters and cluster representatives (yellow points)

are produced. When a new test arrives (red point), distance to representatives is found and variability in clusters is considered, and based on that values status of new functional test is determined. New test could be flagged as anomalous or normal.

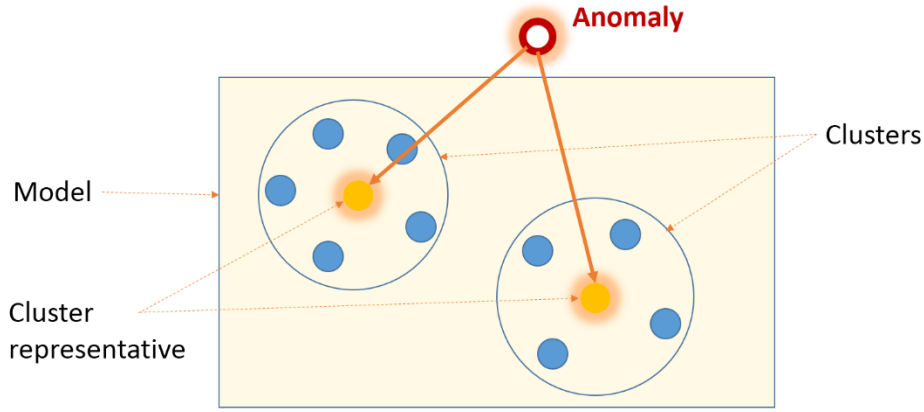

Figure 6: Clustering based anomaly detection – The idea

Different clustering algorithms exist and they are usually divided into partitioning, hierarchical and density based algorithms. Depending of clustering algorithm being used a certain number of anomaly detection techniques can be distinguished. The approach we are using is based on assumption that normal examples lie close to cluster representatives, while anomalies are placed far from the representatives. This assumption comes natural since similar behaviours tend to group together.

It should be noticed that anomaly detection comes to the scene *after* clustering has been performed. Even though clustering represents an unsupervised technique we consider this step as a *training* phase, which results in a model of normal behaviour (which consists of cluster representatives and variation that exists in each of the clusters). Only after a model is generated we approach detecting anomalies for each new example (in our case *functional test*) that gets into the system. Nevertheless, there could be some cases in which outliers already exist in the training data (mostly without knowledge of data provider), which may result in a number of small clusters that contain a very small number of elements (per example – clusters singletons). For that case, we examine the number of points in a cluster and it could happen that a small cluster is declared as an anomaly, due to its dissimilarity from the rest of the training set. That being said, we will now explain what happens to each measurement that gets into the system.

Each measurement goes through two branches – branch used for model generation and branch used for anomaly detection (Figure 7). Clustering is performed periodically and because of that each measurement is persisted, so it could be used during clustering when it is time for it. On the other hand, each measurement needs to be analysed, so an alarm could fire off in case of an anomaly. In that way, the model gets periodically updated, so it could follow the *centroid shift*, if it exists. It can be defined how much of history we want to consider during training. If new measurement is considered anomalous it is not taken into account for model generation, since we are trying to learn what is normal only based on normal instances.

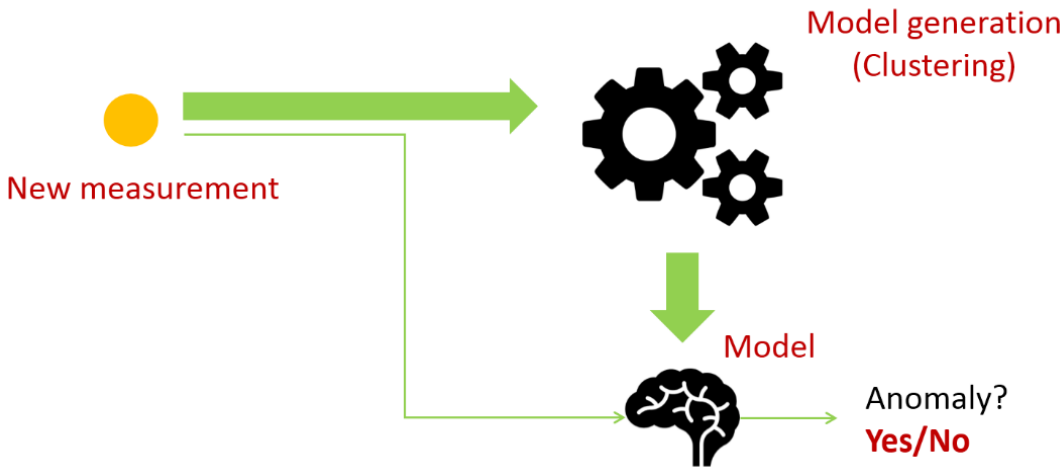

Figure 7: Training/Detection difference

We delve more deeply into concrete algorithms being used in section 2.4.
To be able to work with Big data our implementation will run on COSMOS[2]. COSMOS, as a Big data Generic Enabler, was designed for storage and processing of large amount of data, in batch mode, distributing tasks in a medium-size cluster of computers, following the MapReduce[3] paradigm. This will allow us to use Hadoop platform as a basis for our implementation that handles tasks such as distributed storage, task distribution, hardware failure and security.

Even though COSMOS makes our job easier by providing all that was mentioned, it should be noticed that versions of Java (Java 6) and Hadoop[4] (Hadoop 0.20.2) could be upgraded, to fully exploit the benefits of newer versions.

2.3 Analysis of data sets - data decompression

As already explained, washing machines functional tests, provided by Whirlpool, are being analysed. The goal is to find a way to define normal parameter values and implement a solution that will be able to identify unusual patterns in functional tests.

Dataset provided by Whirlpool contains three parameters:

- Power
- Speed
- Total water inlet

Data is originally persisted in a database in a compressed form. Figure 8 represents Entity/Relation model that explains the relations between the tables.

[2] http://catalogue.fiware.org/enablers/bigdata-analysis-cosmos

[3] http://en.wikipedia.org/wiki/MapReduce

[4] https://hadoop.apache.org/

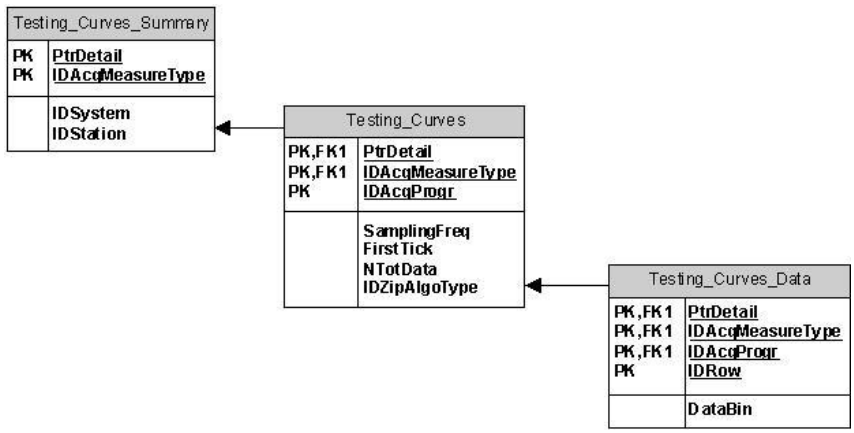

Figure 8: Entity/Relation model

Here is the description of tables and relevant attributes:

- Table *Testing_Curves_Summary* contains the summary of the acquired curve. PtrDetail has been assigned to each test and each test can contain certain number of parameters, which is specified by *IDAcqMeasureType*.
- Table *Testing_Curves* allows the possibility to split the test into more steps of acquisition. This means that it is possible to start acquisition in a particular moment with a particular frequency, then stop it and restart it after some second with a different frequency. This restart can be done as many times as necessary. All sub-acquisitions must be used to compose the complete acquisition of the test. *IDAcqProgr* identifies the sub-acquisition. If there is only one sub-acquisition there will be only one entry with *IDAcqProgr* = 1. *FirstTick* contains the first tick of the sub-acquisition and *SamplingFreq* is the frequency used for the sub-acquisition. *NTotData* represents the total number of points in sub-acquisition. *IDZipAlgoType* defines if compression is used – 1 represents no compression and 2 represents *ZLib* compression.
- Table *Testing_Curves_Data* contains the data (values) of acquisition. For each Sub-Acquisition (*IDProgrAcq*) for a test (*PtrDetail*) for a measure type (*IDAcqMeasureType*) there are from 1 to n rows to define the samples. The list of samples is splitted into more rows if the list of data exceeds 4000 bytes (the field *DataBin* that contains the data is a varbinary (4000) field).
  When all the rows that that compose the acquisition are joined, the list of samples is a string in the following format:
  <Sample 1>|<Sample 2>|….|Sample N>
  where N is equal to *NTotData* value of related table *Testing_Curves*.
  The values are all integer values. To get the real value it is necessary to use the table *Testing_Curves_CFG* where for the *IDSystem/IDStation/IDAcqMeasureType* there is a multiplier and a Measure Unit. For example, if a sample value is 346 and the multiplier is 10, the real value is 34.6.

Access to the original database was not given, but the data was provided in a form of two CSV files, *FTdata.csv* and *FTheader.csv*. The tables were already joined, so now, we have all the relevant data for our analysis in the *FTdata.csv* file that has the following structure:

PtrDetail, IDAcqMeasureType, IDSystem, IDStation, Descr, MeasUnit, Multiplier, MinTick, IDAcqProg, SamplingFreq, FirstTick, NTotData, IDZipAlgoType, idrow, DataBin

Figure 9 represents an example of measurements for each of the three parameters, *Power*, *Speed* and *Total Water Inlet*.

PtrDetail	IDAcqMeasu	IDSystem	IDStation	Descr	MeasUnit	Multiplier	MinTick	IDAcqProgr	SamplingFre	FirstTick	NTotData	IDZipAlgo	idrow	DataBin
###	1	100	3	Power	W	1	500	1	1	10	379	2	1	0x789C65940996ED20
###	2	100	3	Speed	rpm	1	500	1	1	9	380	2	1	0x789CED50DB110421
###	5	100	3	Total Water Inlet	L	10	500	1	1	9	380	2	1	0x789CED8C410E8040

Figure 9: Fields and measurement values in a compressed form for each of the parameters – Power, Speed, Total Water Inlet

As already explained, field *DataBin* contains values in a compressed form. So the first step is to decompress the data, so the analysis could be performed afterwards.

Decompression code has also been provided, but it originally worked with XML files as input, so it needed adjustment to work with CSV files. The decompression is implemented as a *MapReduce* job which executes on a Hadoop cluster. This enables to work with large amounts of data, concretely, with large number of tests, which can be decompressed in a distributed fashion over Hadoop cluster. COSMOS is used as a platform.

Part of the decompression result is presented on Figure 10. Values of measurement are separated by pipes (|).

```
PtrDetail0010000001000020002406658_T2   0010000001000020002406658,100,1,Speed,rpm,1,500,1,1,12,206,2,0|0|0|0|0|0|0|0|0|0|0|0|8|61
48|48|48|49|49|49|47|47|42|18|0|0|0|0|0|0|0|0|0|0|0|0|0|0|-28|-64|-56|-54|-52|-55|-51|-55|-53|-50|-18|0|0|0|0|0|0|0|3|65|62|44|22|0
582|605|634|651|675|684|700|704|721|699|684|679|732|802|867|905|948|829|426|138|18|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
PtrDetail0010000001000020002409287_T2   0010000001000020002409287,100,1,Speed,rpm,1,500,1,1,12,346,2,0|0|0|0|0|0|0|0|0|0|0|0|0|
49|45|45|49|51|47|46|49|46|50|52|50|49|52|52|51|48|46|45|45|47|50|49|47|47|45|46|48|52|52|55|51|48|48|51|49|46|46|46|15|0|0|0|0|0|0
33|33|39|40|43|49|55|62|63|67|73|75|86|86|93|95|102|100|154|298|393|455|518|578|621|652|694|723|744|758|779|781|750|742|784|879|943
0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
PtrDetail0010000001000020002413606_T2   0010000001000020002413606,100,1,Speed,rpm,1,500,1,1,9,245,2,0|0|0|0|0|0|0|0|0|0|0|0|0|0
52|48|47|52|47|50|51|51|49|53|48|51|50|47|52|48|50|51|47|52|52|49|53|47|49|52|49|50|54|49|49|51|53|53|49|46|52|49|50|50|49|51|51|48
0|0|0|34|56|59|55|53|56|58|52|53|41|22|10|0|0|0|43|90|133|176|225|268|327|374|414|458|562|562|611|624|628|636|409|136|6|0|0|0|0|0|0
0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
PtrDetail0010000001000020002414125_T2   0010000001000020002414125,100,1,Speed,rpm,1,500,1,1,9,255,2,0|0|0|0|0|0|0|0|0|0|0|0|0|0|3
46|47|52|48|48|52|50|54|50|44|49|49|46|46|52|49|47|52|49|52|52|46|47|49|47|47|52|49|47|52|50|51|53|46|46|48|47|48|51|51|47|49|51|51
61|52|54|58|53|50|56|48|20|7|0|0|33|89|130|179|223|270|311|356|408|460|505|556|601|621|627|635|384|129|0|0|0|0|0|0|0|0|0|0|0|0|0|0
0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
PtrDetail0010000001000020002414423_T1   0010000001000020002414423,100,1,Power,W,1,500,1,1,9,188,2,0|4|4|4|4|5|58|82|40|21|14|13|13|
94|92|92|93|93|92|93|94|92|96|100|95|98|99|99|99|101|101|101|100|103|101|103|101|100|104|104|102|103|106|106|108|108|60|24|12|9|8|7
35|35|40|66|133|119|67|46|37|30|27|25|25|25|32|83|105|99|103|102|99|100|103|105|106|105|106|105|110|114|113|111|116|118|354|428|448
23|23|23|23|14|7|5|0|0|0|0|0|0|0|0|0
```

Figure 10: Decompression result

It should be noticed that some problems were encountered during decompression procedure. These problems are represented on Figure 11, Figure 12 and Figure 13.

Figure 11 represents the case of many missing values. Decompressed data contains a number of consecutive pipes, which implies that some measurements are missing. The case of five or more missing values in a row is highlighted in yellow.

Figure 11: Missing values in decompressed data

Another problem we encountered was the problem of exception being thrown while trying to decompress the data. Concretely, *java.io.IOException* was thrown with message *incorrect data check*, which implies that there may have been some error during compression of data, or that there may have been a problem in the data itself. This problem is presented on Figure 12.

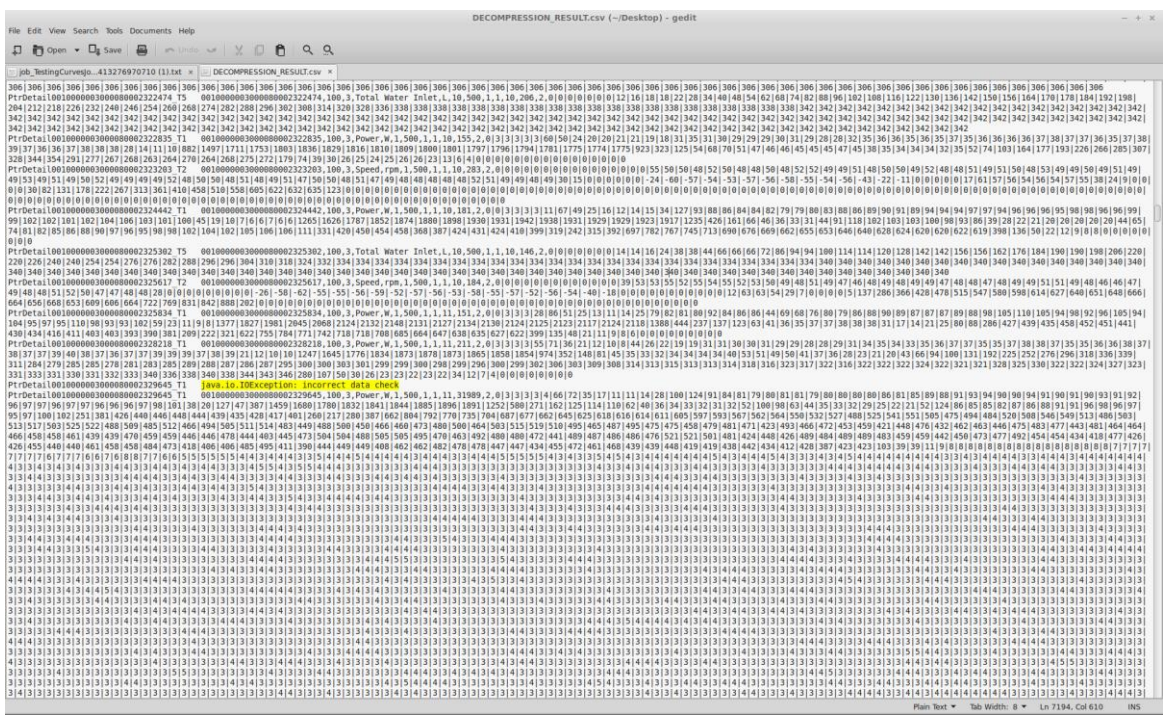

Figure 12: Incorrect data check

Finally, we consider the case of many repeating values, like the ones presented on Figure 13. Even though an explanation was provided that this may be normal, we emphasize that there could be cases of tests of much longer length.

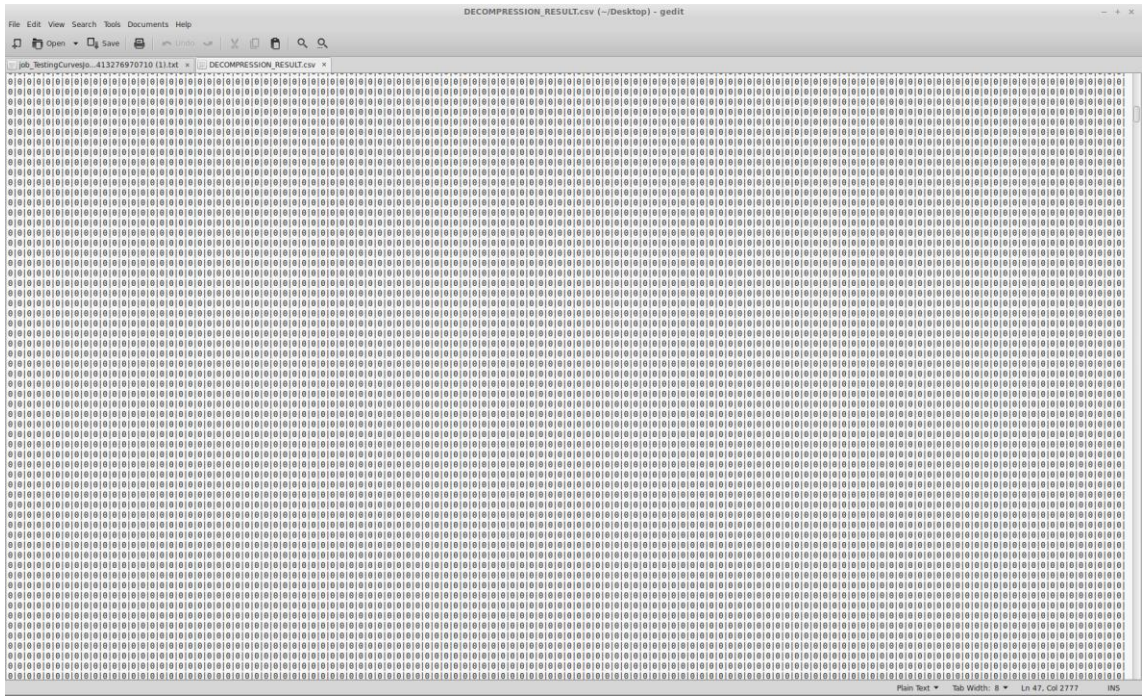

Figure 13: Repeating values

We pay special attention to these suspicious measurements, but have in mind that they represent a minority of tests found in the dataset.

Visualization of three parameters (*Power*, *Speed* and *Total water inlet)* for one of the tests, acquired after decompression, is presented on Figure 14, Figure 15 and Figure 16.

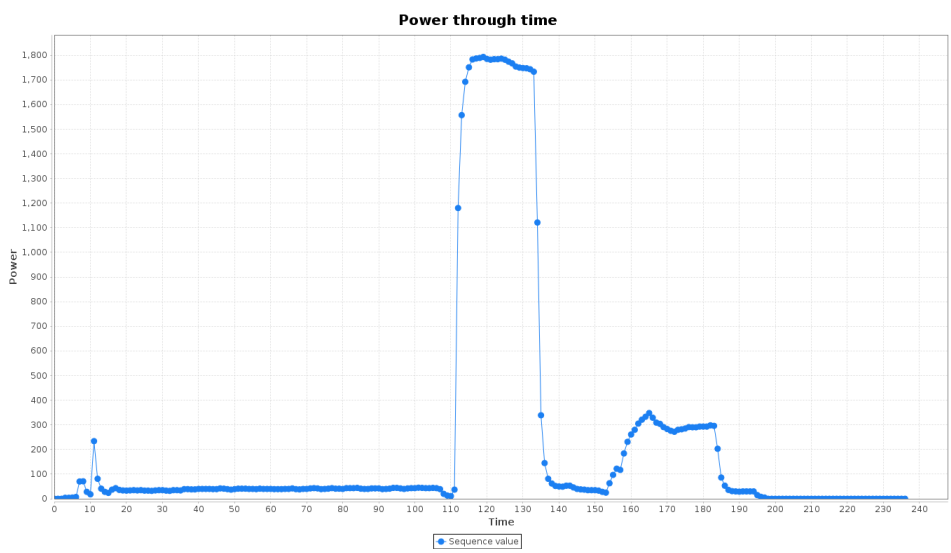

Figure 14: Power

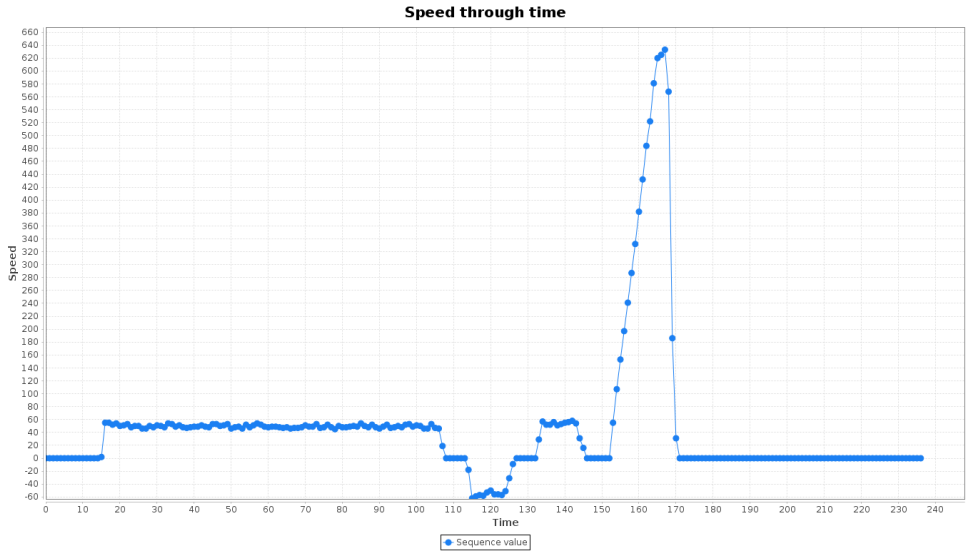

Figure 15: Speed

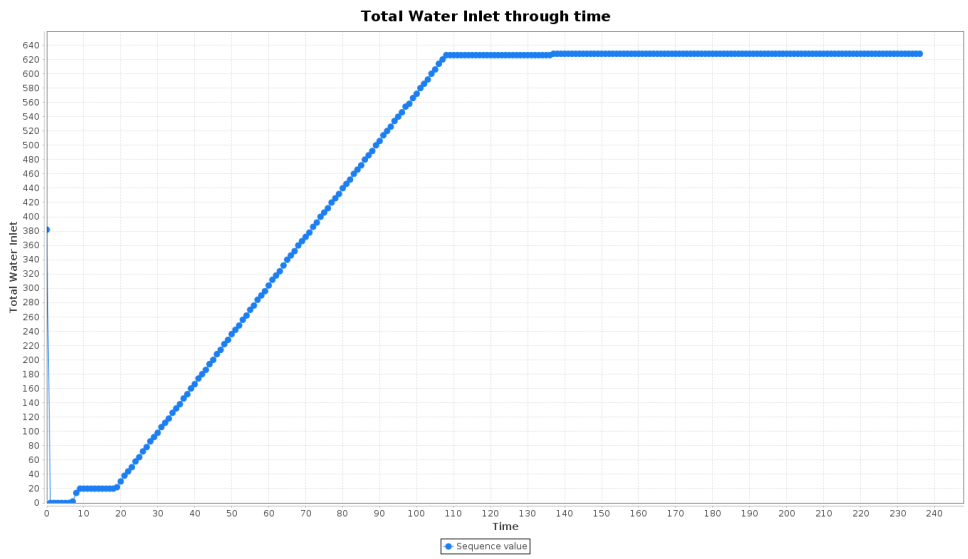

Figure 16: Total Water Inlet

Interpretation of signals and their correlation follows:

- When the machine is powered on water inlet begins;
- When a certain level of water is reached power is increased and drum rotation begins (rotation speed grows from 0 to a certain level);
- When rotation direction needs to be changed, power is decreased to zero, so that rotation will slow down, after which power is greatly increased so that direction of rotation could be changed;
- During counter clockwise rotation much greater power is needed than during clockwise rotation;
- To start the centrifuge (drum rotation at high speed) power needs to be greater than in case of regular rotation;
- During the process certain amount of power is needed to heat up the water.

Correlation between parameters is graphically represented on Figure 17 and Figure 18.

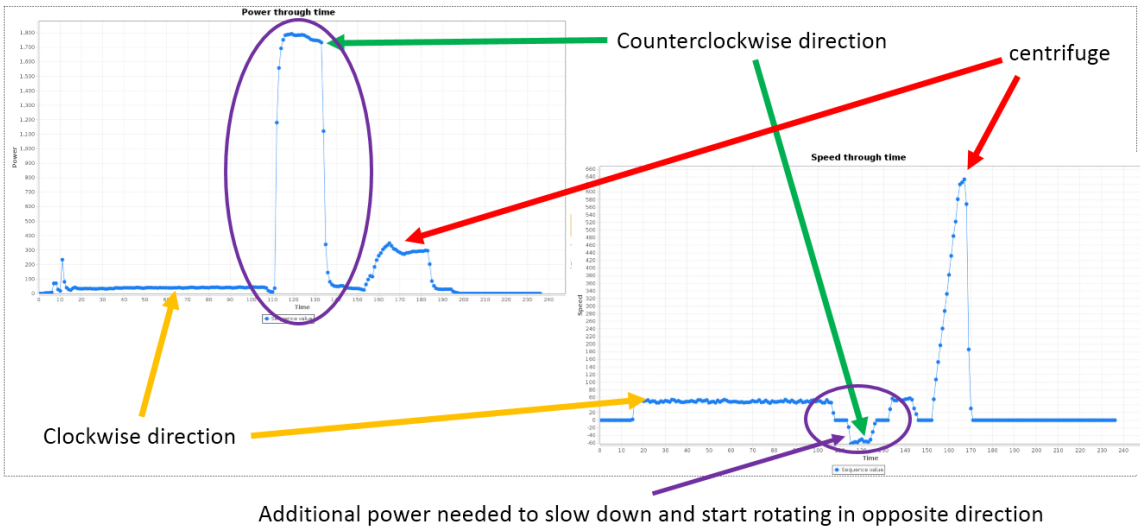

Figure 17: Correlation between power and rotation speed

Figure 18: Correlation between power and water inlet

After the dataset has been decompressed and the goal of analysis has been defined, we can approach considering concrete algorithms for our analysis.

2.4 Anomaly detection on decompressed data and implementation on COSMOS

We have previously explained that the goal of our analysis is to describe normal behaviour and discover anomalies as behaviour that does not conform to the defined model. To do that, we are using clustering. Now we will consider which concrete clustering algorithms we could use for that and how to implement a system that will automatically detect anomalies, based on these algorithms.

First of all, we have defined the process of detecting anomalies as follows:

- Cluster the data using some clustering algorithm that will not only produce clusters, but will also produce cluster representatives;
- Cluster representatives and cluster variances form the model;
- Each new measurement is compared to the existing model and the dissimilarity from the model determines its status as anomalous or normal.

Most of clustering algorithms that respect these conditions belong to the group of partitioning algorithms. Before we describe the concrete algorithm that we use from that group, it is necessary to take another look at the dataset, this time considering multiple tests at the same time.

Power parameter values of multiple tests are presented on Figure 19. We can notice that even though all the graphics present the value of the same parameter, they may have very different shapes. This has a huge impact on our analysis. There is a need for robust measure that is able to cluster time series based on the shape of the series.

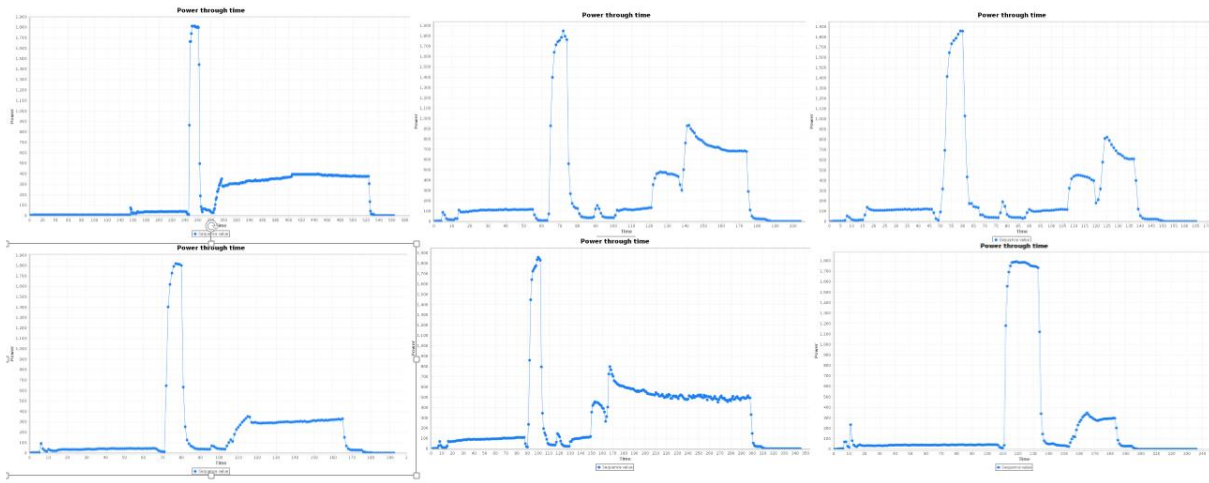

Figure 19: Power parameter of multiple tests

Another question occurs due to the latest condition, clustering time series based on shape. How should the cluster representative look like for a cluster that contains series of different shape? Some of the algorithms that produce cluster representatives give some kind of a *mean* or an *average* of all the elements in the cluster as a representative. But that would not be an acceptable solution in our case for the following reason – different tests may be performed under different conditions. By conditions we mostly mean different *timings*. In case of one test centrifuge can be started part of a second earlier than in the other, or can have slightly greater duration. That *part of a second* makes *mean* of series impossible to use as a representative. Because of that, we turn to another group of clustering algorithms, algorithms that select one of the objects from the cluster as a representative of the cluster. That object is the one that is the most similar to all the other objects in its cluster, and it's usually called a *medoid. K-medoids* is an algorithm that is based on that concept and will be described in the following.

K-medoids algorithm belongs to the group of partitioning algorithms that represent each cluster with one of the objects from the cluster, the *medoid.* Its basic steps are:

- Select initial medoids
- For certain number of iterations

- Assign objects to the most similar medoid
- Select new medoid that improves the quality of clustering

There is a number of questions that have a great impact on clustering result. For example:

- How to select initial medoids?
- How to define similarity between objects?
- What happens in case that there are outliers in the dataset?
- How many clusters exist?

We have considered each of these questions and included the answers in our implementation. But before we consider these questions, we must define our goals in more detail.

We have set two goals. First goal is to implement a solution that will execute on one machine and will be used primarily for validation of our hypotheses. Second goal is to implement a solution that will run on top of Hadoop cluster (concretely on COSMOS), after validation of our methods, that will be able to work in case of large amount of data. We form our tasks in this way so we could validate our methods quickly and then pay attention to a scalable solution that will work for Big data.

Our single machine solution follows the basic concepts of *K-medoids* algorithm, with a few adjustments that consider the relevant questions that have been defined. Now, we will pay more attention to them.

First of all, we consider initial medoids selection. This question has more relevance than it may seem. Quality of final clustering heavily depends on initial medoid selection due to phenomenon of local minimum. One option would be to select *k* objects randomly from the dataset, but that can result in poor clustering. That is why we use a different kind of initialization in our implementation. The idea is to select objects that are placed somewhere in the core of existing clusters as initial medoids. In this way, only a small number of iteration is needed to produce final clusters, and its role is just to refine initial clustering. This makes are algorithm faster and more precise than in case of random initialization.

Second, we consider the question of objects similarity. To perform clustering it is necessary to define a measure that will describe how similar objects are. Similarity measure is often compared to distance measure, since the objects are observed in N-dimensional space. Maximal similarity between two objects can have a value of 1, which means that distance between them is minimal, that is, equal to 0. There are different kinds of distance measures, such as *Euclidean*, *Manhattan* or *cosine*, but we will see that they are not very useful in our case. The reason for that is that we are comparing time series that can be shifted in time, or skewed, and distance measures like Euclidean do not tolerate this. That is why we approach to another kind of distance measures that considers shapes of two signals. The measure is called *Dynamic Time Warping* (*DTW*) and it is able to find the optimal alignment between two signals. To show this we will observe two very similar time series, represented on Figure 20 to Figure 23. Actually, the blue signal represents power for one of the tests from the dataset, and the red signal represents a modification of the blue signal. So it's basically one same signal, just shifted or skewed, as we will see.

On each figure, next to the graphic, *Euclidean*, *cosine* and *DTW* distances of two signals are displayed.

Different distances in case when red signal is shifted for one point comparing to blue signal are presented on Figure 20. We can see that *Euclidean* distance is quite large for such a *small*

difference and *cosine* distance is greater than 0. On the other hand, *DTW* distance is equal to 0, which already gives an impression of how powerful this distance measure is.

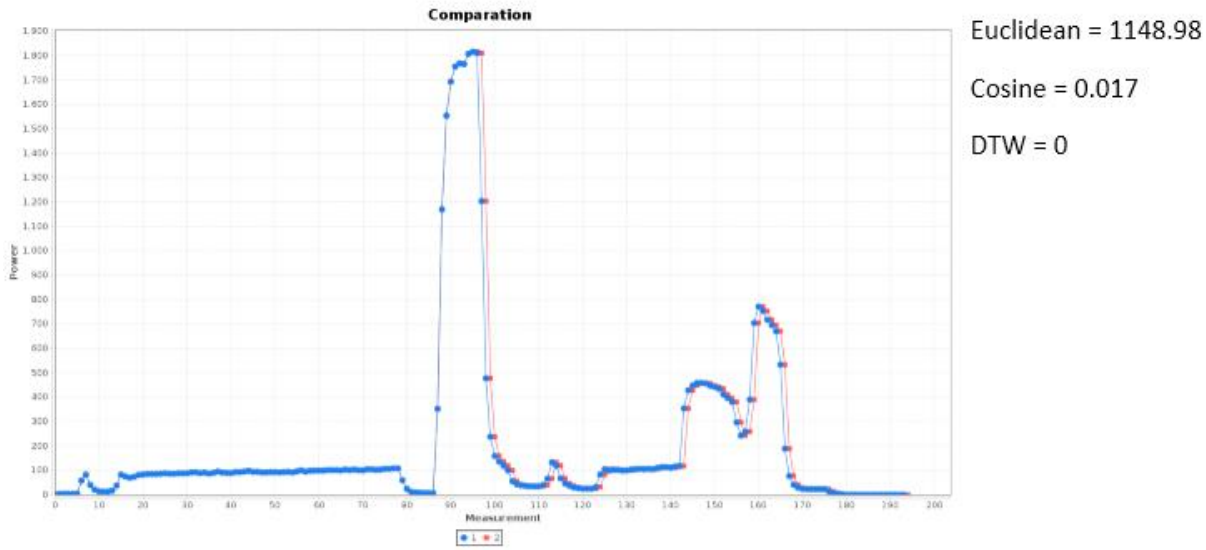

Figure 20: Comparison of two signals – red signal is shifted for one point comparing to blue signal

Distance values in case of a larger shift of red signal comparing to blue signal are presented on Figure 21. It can be noticed that *Euclidean* distance is even larger, and so is the *cosine* distance. *DTW* still has the same value – 0. This implies that *DTW* is immune to the phenomenon of *shift*, no matter how big it is. This could be very useful in our case, since different conditions are used while performing functional tests. We could interpret this shift in the following way – counter clockwise rotation was started later in the case of red signal than in the case of the blue signal, so every next step in the test (for example, centrifuge being started) also gets shifted. But this doesn't mean that something is wrong with the other test. The values are normal, they are just shifted in time.

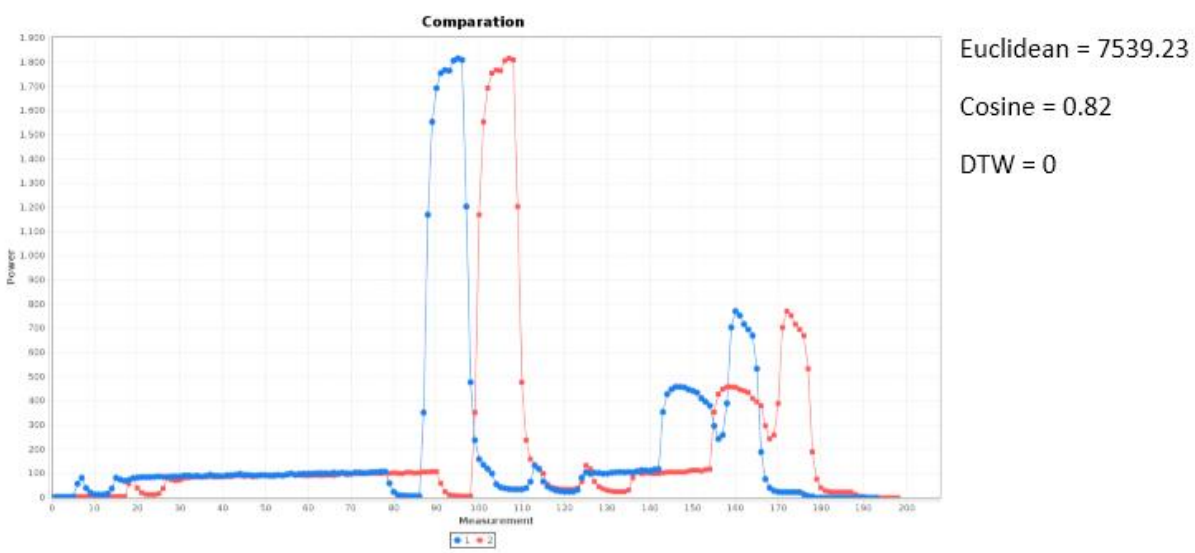

Figure 21: Comparison of two signals – red signal is shifted even more

On Figure 22 comparison of two signals is displayed in case when one part of red signal is skewed. This could also be considered as a shift, of course. *DTW* distance stays equal to zero, while *Euclidean* and *cosine* distances are again increased. This shows that *DTW* is immune to skewing of signals, which is another attribute that we were looking for. In this case, for example, rotation in clockwise direction of red signal started later comparing to blue signal, but also lasted longer. Nevertheless, we think of these signals as representing similar behaviour, and that is why we consider that the distance between them is equal to 0.

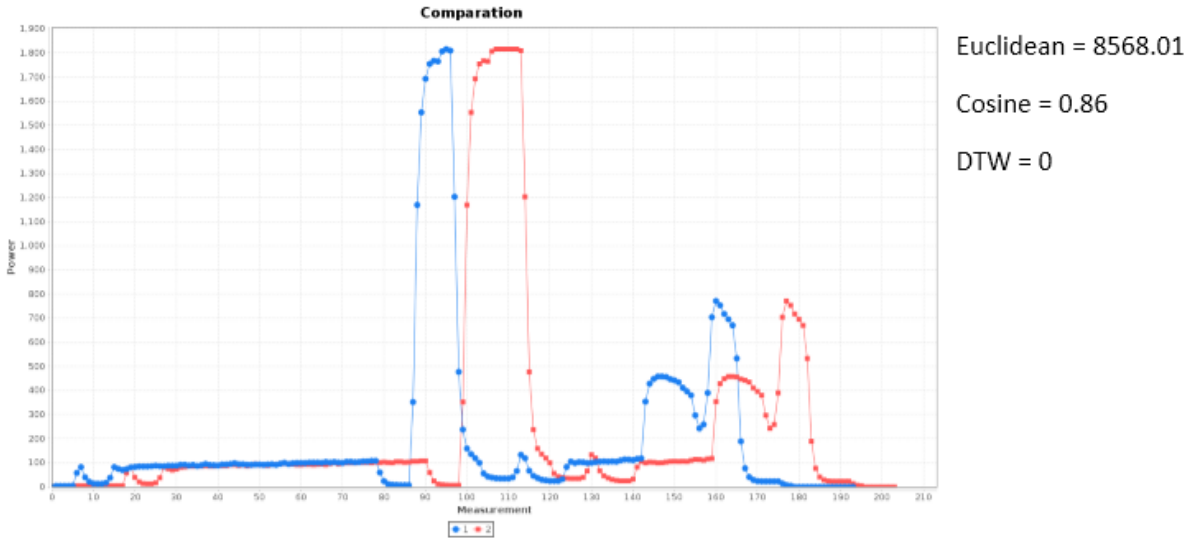

Figure 22: Comparison of two signals – counter clockwise rotation lasts longer

Just as a *sanity check* we give an example of two signals that are considered dissimilar on Figure 23. We can see that *DTW* in this case is not equal to 0, as expected. When we compare red signal to the blue one we can conclude that there is something really *strange* about it that hasn't been seen before. That is why we consider *DTW* value appropriate. It is interesting to notice that *cosine* distance was not able to notice this *abnormality* of the red signal.

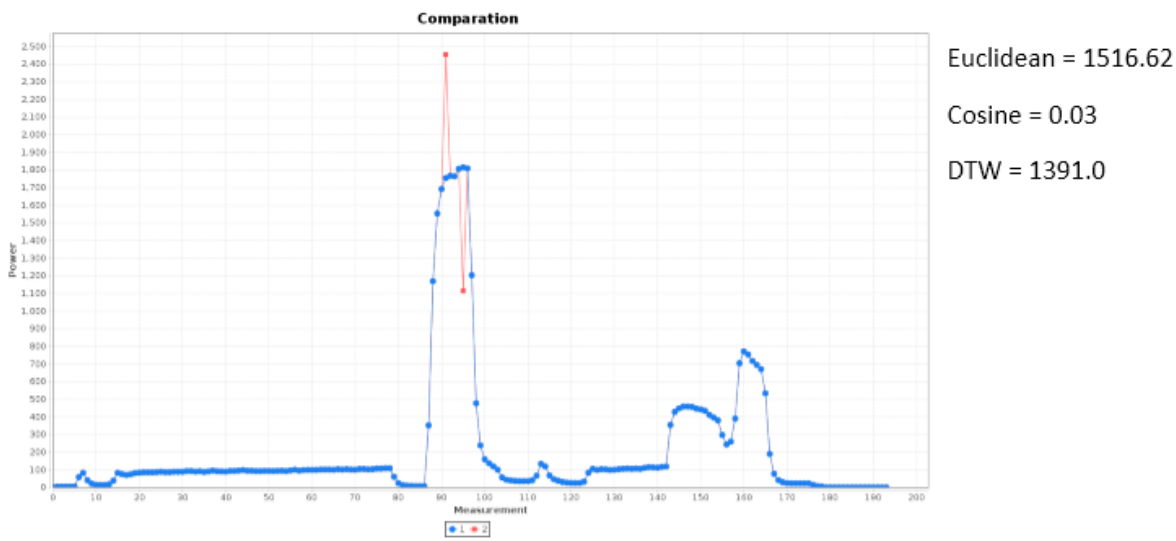

Figure 23: Comparison of two *aligned* signals

DTW originates from the field of speech recognition. For example, it can compare speech of two persons with different speaking speeds. Or it can be used to compare walking patterns of two persons, when one person walks faster than the other, or there are accelerations during the walk. This makes it an ideal measure for our case. It finds the optimal alignment between two signals by performing *warping* of sequences. An illustration of such a warping is presented on Figure 24.

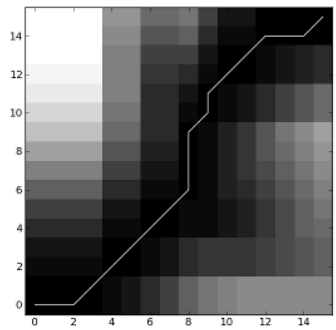

Figure 24: *Warping* of two signals

Next, we consider the case when there are outliers in the training dataset. In our case outliers are considered as anomalous test measurements. Fortunately, our method is robust, so the existence of outliers will not decrease the quality of the result. The reason for that is that we are not considering *centroids* during clustering, but *medoids*, which makes our algorithm robust. A *medoid* is one of the objects from the dataset, while *centroid* is an average of all the objects in the cluster. The worst thing it could happen is to get some small clusters that will be discarded after clustering and reported as anomalies that exist in the training set. Figure 25 shows the difference between *centroid* and the *medoid* on an example dataset that contains an outlier. It seems that the *medoid* represents the cluster better than the *centroid*.

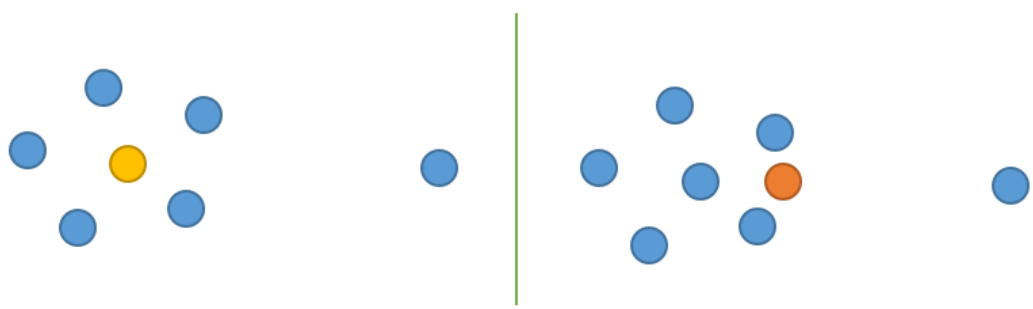

Figure 25: Difference between a *medoid* (yellow point) and *centroid* (orange point) in case of an outlier presence in the dataset

One more thing that needed to be implemented in our solution is determining the number of clusters. The question of how many clusters exist in the dataset is a very difficult question and people may have different answers to it. An illustration of this problem is presented on Figure 26. The number of clusters is a rather arguable question. Since it is necessary to specify this value for our clustering algorithm we needed to implement an extra step that will determine this figure. For that, we have implemented some modification of what is called the *Elbow method*. This way we are able to automatically determine how many clusters are there. Of

course, the implementation still allows specifying this number manually, if there is knowledge about it.

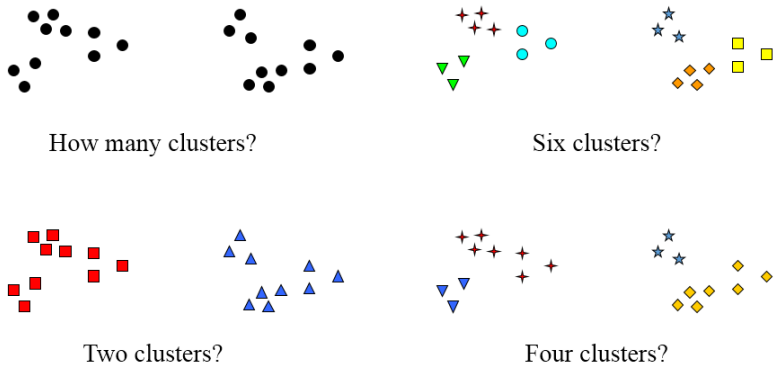

Figure 26: How many clusters question

Before we get into concrete results of our analysis, we will spend some time explaining the concept of our solution that will run on COSMOS.

We have explained that our goal is to be able to bear with large amount of data. Our implementation will run on a Hadoop cluster, using all the benefits it provides. As a Hadoop cluster we use COSMOS Generic Enabler. The position of COSMOS and anomaly detection is presented on Figure 27. Our solution needs to be implemented following the MapReduce concepts so it could run on a Hadoop cluster. This requires a scalable implementation that consists of series of MapReduce jobs. The idea is to perform training (in our case clustering) on Hadoop cluster, while detection itself can be performed in a service or even on a *Storm*[5] *topology*, depending on deployment conditions and the frequency and speed of test ingestion.

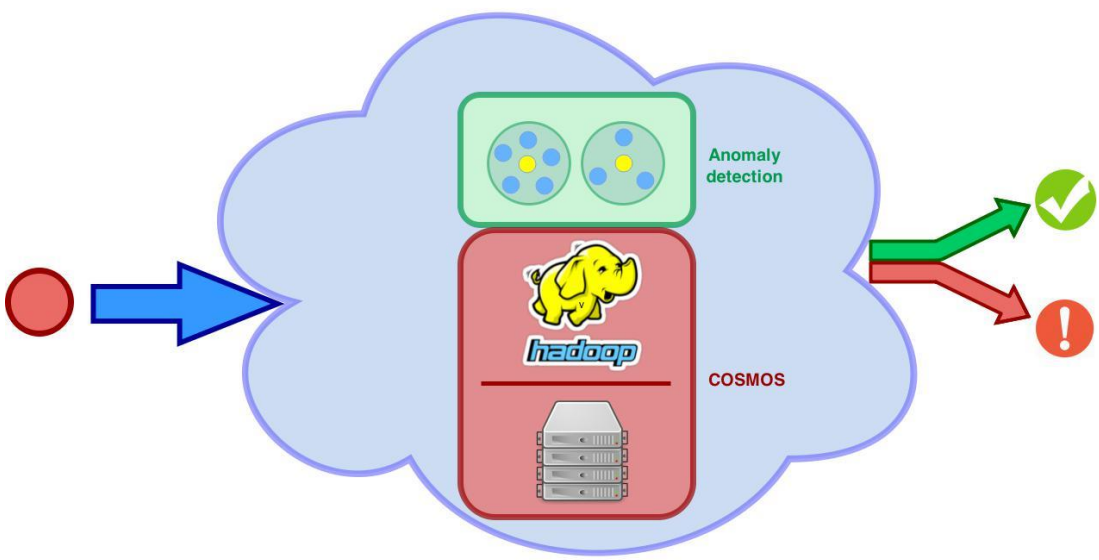

Figure 27: Anomaly detection on COSMOS

[5] https://storm.apache.org/

Figure 28 represents an architecture of such a system. Tests could be uploaded through a web service that stays in front of Hadoop cluster and its purpose is to collect the tests for the detection and for the training. We assume that tests are uploaded in a compressed form so we introduce a decompressor which has the purpose of decompressing the data, of course. Currently, the decompression is performed on COSMOS, but the reason for that is because large amount of data is compressed at the same time. We assume that tests would be uploaded one by one or in a small group, so they can be decompressed in the service itself. If that is not the case, we could easily use decompression that was already mentioned and implemented as a MapReduce job. Next in the pipeline is the *Preparator*. This component has the purpose of preparing the uploaded test (with decompressed values) in a form appropriate for clustering job that runs on Hadoop. Also, this component has the purpose of selecting only those fields that are relevant for analysis and possibly perform some conversion of values, for example, divide each value by *multiplier* attribute value to get floating point numbers. Finally, when the test values have been uploaded, decompressed and prepared, anomaly detection can be performed by comparing the test with the model generated by *Model generator.* As a result of that comparison we get the status of the test, anomalous or not anomalous. This status is returned to the caller of the service and in the case the test was not anomalous it is persisted (on HDFS, or in database like HBase[6] or Cassandra[7]) so it can be used for model generation later. Model is generated periodically using a MapReduce job that performs clustering of the dataset. Oozie[8] can be used to schedule clustering job and the period of clustering can be defined. Model is a collection of medoids and variances in their clusters. It should be noticed that we are making an assumption that tests are uploaded with a low rate, but if that is not the case the difference would be that instead of using the service itself to perform anomaly detection *Storm topology* could be used with *bolts* implementing comparison to the model.

[6] http://hbase.apache.org/

[7] http://cassandra.apache.org/

[8] http://oozie.apache.org/

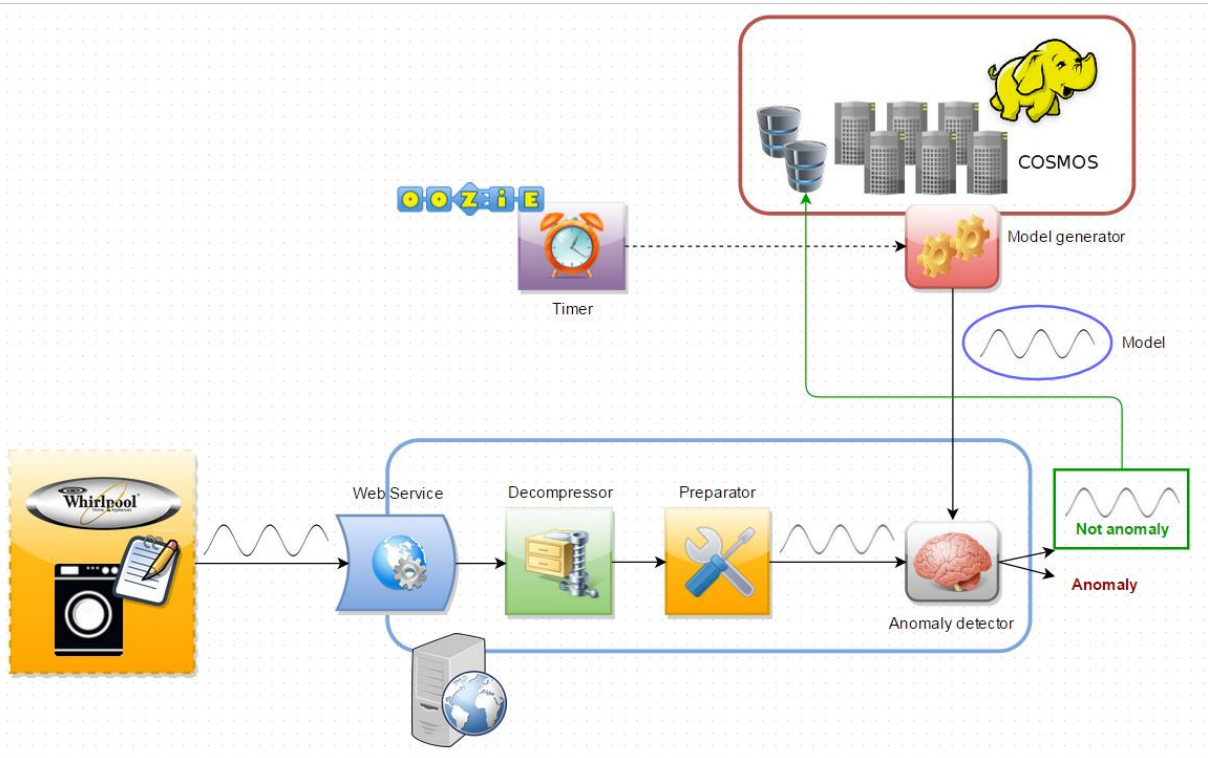

Figure 28: Anomaly detection architecture

2.5 Analysis of the results and lessons learned

We explained that we have set two goals, to implement anomaly detection on a single machine and to perform analysis on a subset of data to validate our methods, and to implement anomaly detection based on MapReduce that will run on COSMOS cluster for Big data. While COSMOS solution implementation is in progress, we have implemented single machine solution and performed validation on a small subset. Here, we present the results of our analysis.

We have initially considered *Power* parameter and took a small sample of twenty tests to validate our methods. We used a variant of *K-medoids* that we have mentioned and got five clusters, while three clusters where clusters *singletons*. Cluster singleton is a cluster that contains only one test. We consider these clusters anomalous, since the number of objects they contain is very small, so they differ from the rest of the dataset. Medoids that were produced are presented on Figure 29, while clusters are presented from Figure 30 to Figure 34. By observing the shape of signals contained in the sample we can conclude that there really are two groups of signals. At the same time we can notice that signals that belong to clusters singletons have different shape than signals that exist in non-singleton clusters. We must emphasize that our primary goal is not to find anomalies while clustering, since this can be a long-term operation, but to generate a model that can be used in *real time* to detect anomalies. Even so, we may detect *suspicious* tests in the dataset, like in our example, so the best solution is to report them, and remove them from the dataset, for safety reasons.

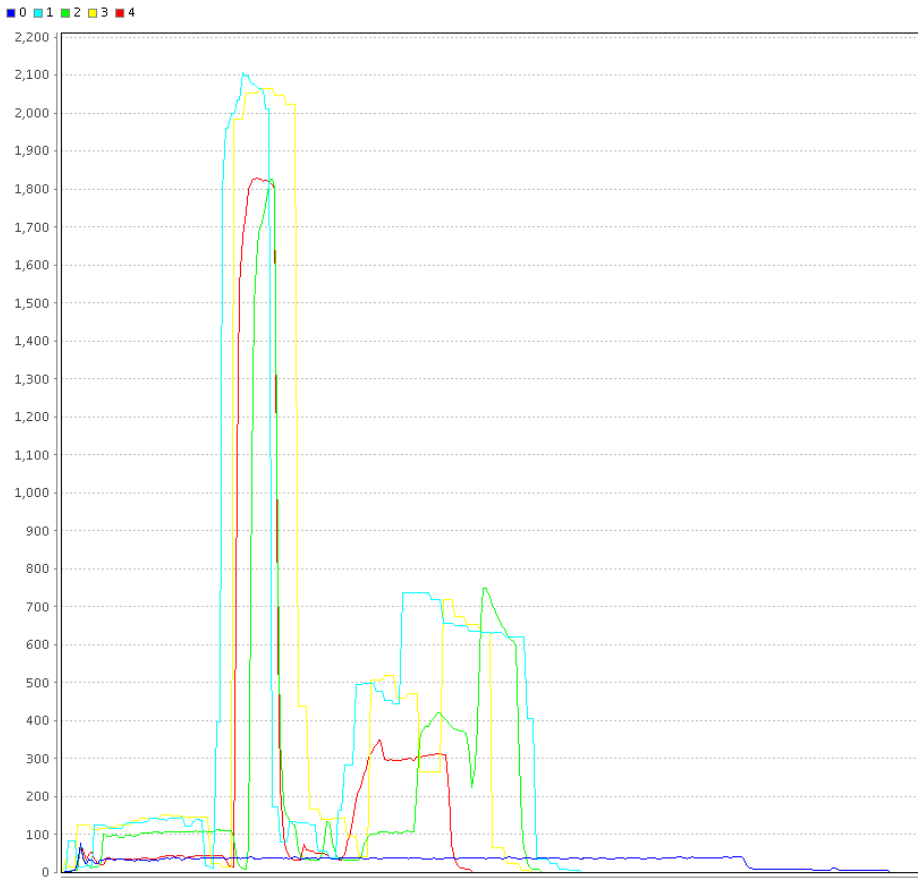

Figure 29: Generated medoids

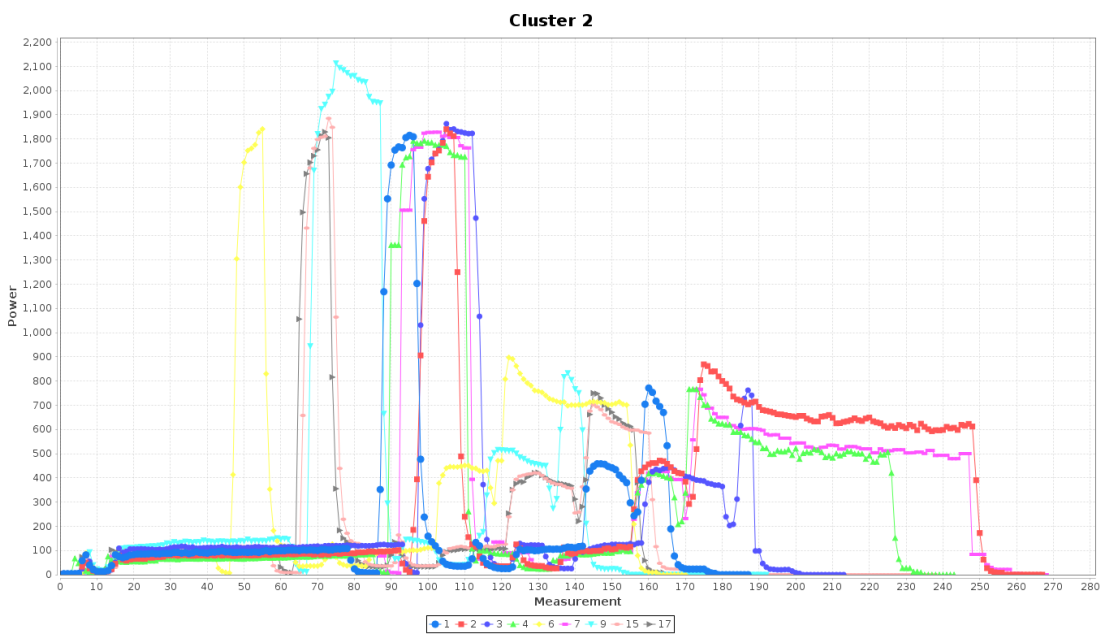

Figure 30: First cluster

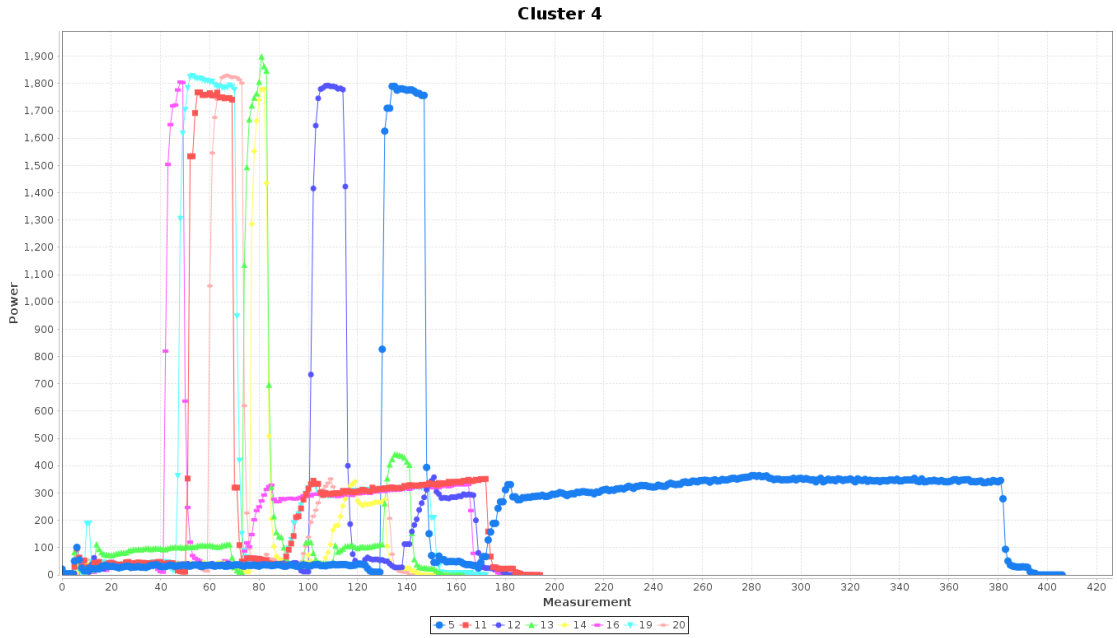

Figure 31: Second cluster

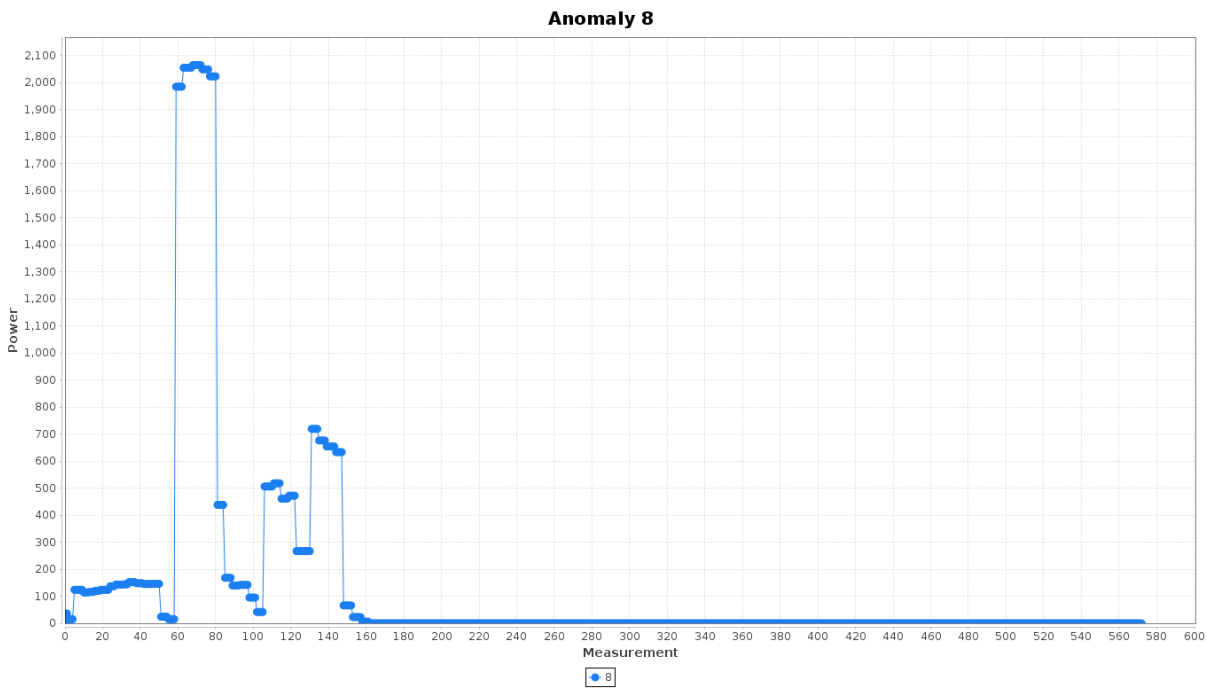

Figure 32: First anomaly

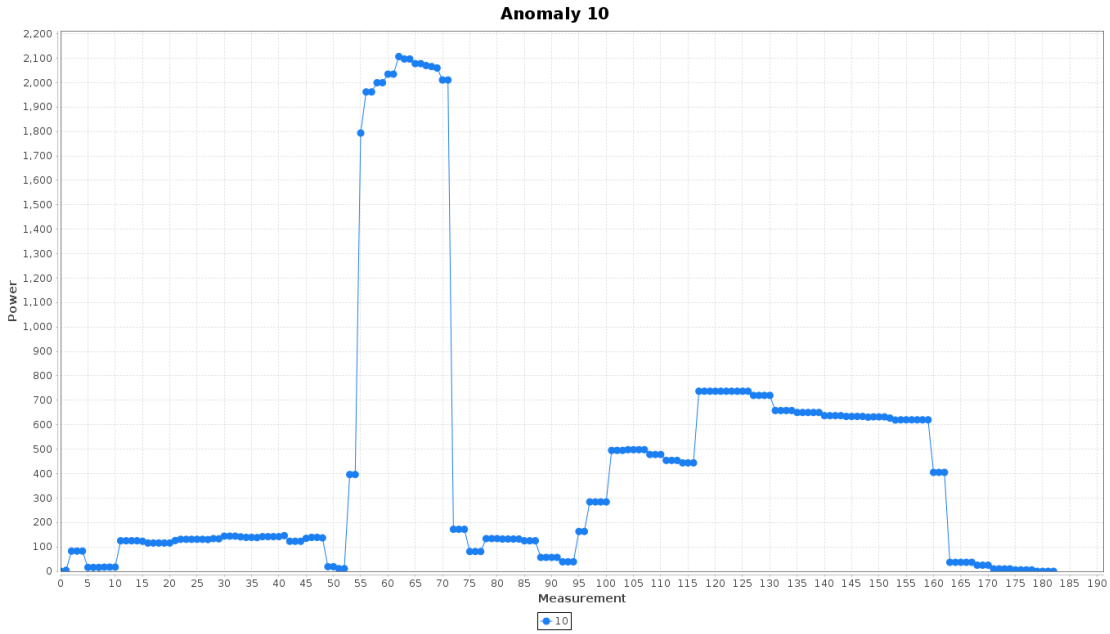

Figure 33: Second anomaly

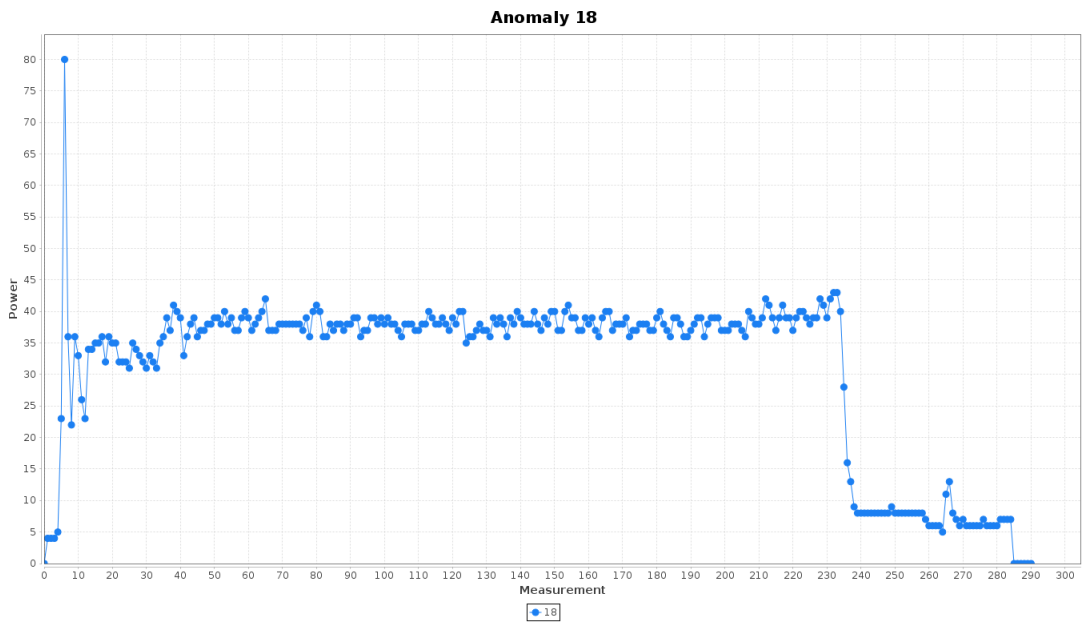

Figure 34: Third anomaly

Based on results we conclude that our algorithm represents a good solution for the problem of detecting anomalies in functional tests provided by Whirlpool. We even got a confirmation that one of the tests (the one on Figure 33) represents a problem that really exists in one of Whirlpool facilities, which they are aware of. We developed a solution that is able to compare test series based on their shape, and to cluster tests based on it. We also used that similarity and clusters being produced to determine which tests look unusual comparing to normal examples found in the dataset.

After finishing single computer implementation and validation of our technique the next step is to finish MapReduce implementation that was started, so the analysis could be performed on a Hadoop (COSMOS) cluster.

Also, the idea is to consider more parameters at the same time, so that correlation between them could be used to define what is normal and what is not.

After completing these goals, deployment could be possibly considered to enable to fully exploit the benefits that our solution offers.

3 Dynamic Visualization and Interaction - DyVisual

DyVisual is based on the FI-WARE generic enablers XML3D/XFlow[9] and the Flexible Virtual Environment Server (FiVES). XML3D allows to integrate 3D content in HTML Web pages in a seamless manner. With this it is possible to view 3D content in a standard Web browser like for example Google Chrome or Mozilla Firefox. XFlow adds to this the ability to animate 3D content and with this makes it alive or gives behavior to individual entities in the 3D scene. With FiVES multiple users can view concurrently the same content. The view port of the users can be synchronized, i.e. the users have exactly the same view to the 3D content, or each user can have his or her own perspective.

Extending the power of more traditional 2D Web based visualization (like for example http://d3js.org/) XML3D can be combined with such technologies in a straight forward manner. With this DyVisual supports many interesting new applications where realistic rendering of 3D content is important. This especially applies for commercial applications in industry, like for example in the context of product design, but especially also for applications in manufacturing where real-time data needs to be visualized in an easy to access and intuitive manner.

In FITMAN DyVisual is applied in two use case scenarios of two FITMAN trials:

- TRW Automotive: In this use case motion capture date from a Kinect system, which captures the motion of a work in a manufacturing environment, is analyzed by DyCEP. The results from this analysis are visualized with the help of an avatar for which problematic poses are visualized.
- Whirlpool: Parts of a washing machine are scanned with a laser scanner. The data of the scan is compared with the original CAD model of the part resulting in a deviation map, i.e. a model that contains the deviation between the CAD model and the scan. The deviation map is analyzed by DyCEP and the result of this analysis is visualized in DyVisual.

For DyVisual the following requirements were specified:

1. Workers should have graphical help to identify the risk level and the body areas affected
2. Prevention technicians could reproduce the worst movements, average movements, etc. of the workers (anonymously)
3. Events are communicated to interested user through mobile device

In the following we present some more details of the two use case scenarios.

3.1 TRW Automotive Use Case

In this use case motion capture data is analyzed, which is provided by a Kinect system, which is tracking a worker in a manufacturing environment. The idea is to use DyCEP to analyze the motion capture data to identify poses in the motions of the worker which are ergonomically problematic. The identified problems are then visualized by DyVisual.

[9] F. Klein, K. Sons, D. Rubinstein, and P. Slusallek. Xml3d and xflow: Combining declarative 3d for the web with generic data flows. *IEEE Computer Graphics and Applications*, 33(5):38–47, 2013

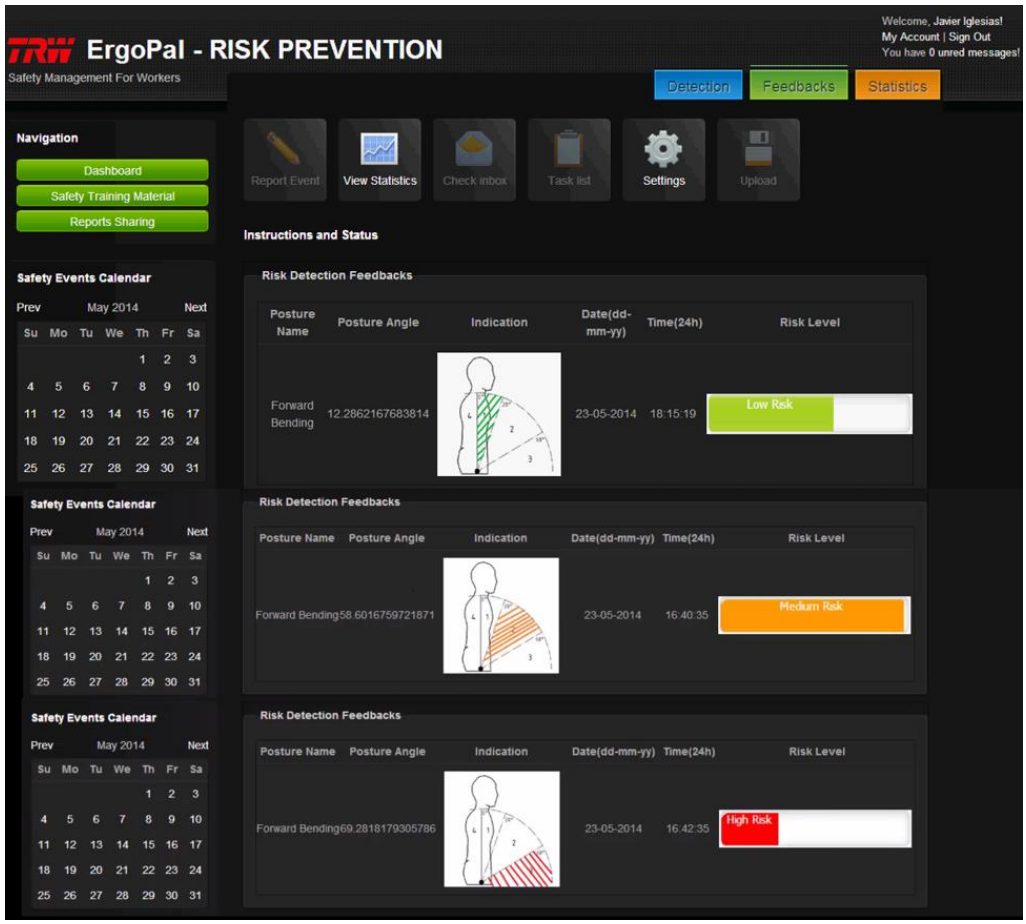

Figure 35: User Interface of the ErgoPal RISK PREVENTION Tool

The data which was provided for the work on DyVisual came from the ErgoPal system which already could display results in a 2D user interface (see Figure 35). The data was provided in an SQL database. The main problem in the interpretation of the data was that the data could not directly be mapped to the joints of the avatar.

Figure 36: Avatar indicating the problematic pose with the affected joint

In the work so far, a reasonable mapping of the given data to the joints of the avatar was calculated. Figure 36 shows the marking of a problematic joint with a colored sphere. The color scheme goes from green (not problematic) to read (very problematic). Till now only static data could be analyzed but in current work it is investigated how motion capture data from a Kinect II can be used in real time to derive the information on problematic poses. However, also in this case the visualization of the poses depends on the result of analysis of the real time data.

3.2 Whirlpool Use Case

In the Whirlpool use case parts of a washing machine are scanned with a laser scanner. The resulting point cloud from the scan of a laser scanner is then compared with the CAD model of the part. The result of the comparison is a deviation map which provides the information in how far the model resulting from a scan deviates from the CAD model.

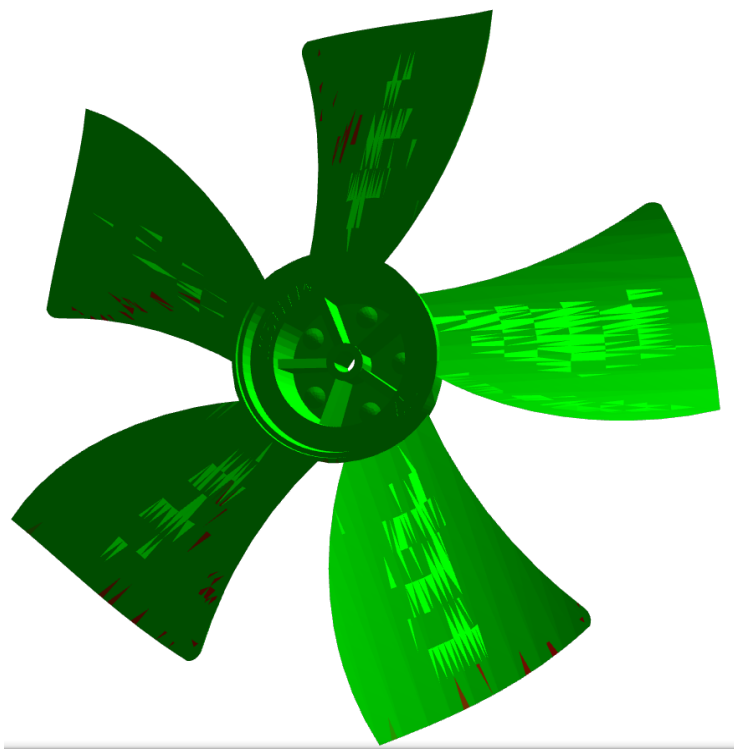

Figure 37: Deviation Map of a Whirlpool Part in XML3D

Figure 37 shows an example of a deviation map which was transformed into XML3D which means that it can be viewed over the network by standard browsers like Google Chrome and Mozilla Firefox.

The idea of this work is to run DyCEP on the data of the deviation maps and display the results of DyCEP with the help of XML3D. Currently the problem with the visualization of the deviation maps is that there is no defined mapping of deviations to colors of the visualization. Additionally, the deviations a in a significant number of cases unrealistically high which makes a consistent mapping of deviations to colors difficult.

3.3 Lessons Learned

The summary of the lessons learned from applying DyVisual in the discussed applications is that the key aspect for the use of DyVisual is the underlying data and its interpretation. In the

first place it is important that data is available but also regarding the interpretation of the data the processing of the data of DyCEP has to provide the data in an appropriate manner together with indications on how the results should be visualized. The latter needs of course only to be agreed on once when the process is implemented.

From the list of requirements specified for DyVisual 1 and 3, are successfully achieved. For TRW the prototype implementation described in section 3.1 meets requirement 1. Regarding requirement 3, DyVisual is able to be accessed from all mobile devices which can run a browser which supports WebGL. Unfortunately, because of security problems some graphics chips of mobile devices have been black listed. However, at least for more recent devices there is hope that new firmware updates for the devices will resolve the problems. For now DyVisual was successfully tested on a Samsung Galaxy SIII with the latest update of Android 4.3 installed.

Regarding requirement 2, more data for the use cases and information on the interpretation of the data is necessary. DyVisual can only visualize results and events which have been produced or detected by DyCEP or by some other means.

We are in the process of improving ease of use of DyVisual. This includes providing DyVisual as a service and to allow the user to more easily adapt the resulting visualization. Additionally, we consider to remove obstacles like ports which need to be opened for remote access to the prototype implementations.

Additionally, to the prototypes for TRW and Whirlpool DyVisual is foreseen for visualization of models in the process of 3D printing. This is a very interesting use case for DyVisual but, unfortunately, no data for the evaluation of visualization results for DyVisual was received so far.

4 Conclusion

This deliverables provides a first evaluation of two new Specific Enablers provided for SF domain: DyCEP responsible for advanced real-time processing of IoT (Internet of Things) data and DyVisual responsible for 3D visualization of the results of processing.

These two SEs will be applied in several business scenarios in two selected trials in SF domain: TRW and Whirlpool. The document contains a detailed analysis of results showing the benefits for the selected business scenarios of using proposed SEs.