AH2020/PPS2 Documentation

Release latest

Aug 05, 2021

CONTENT

In this documentation, we describe some of the outputs of the Project "Augment Humanity" (AH2020), related with the PPS2 - Big Data and Predictive Analytics for i4.0.

CHAPTER

ONE

USE CASES

1.1 AH2020.U1A - Predictive maintenance in the production line of an advanced water heater

Last info update 2021/08/03

Aims Improve value stream by adding **embedded AI maintenance devices**, **forecasting equipment failure** (focus in smart presses), **generating intelligent scheduling of maintenance times**, and identifying and predicting bottlenecks

Location BoschTT

Co-promotors BoschTT, CM, UA

State Algorithmic study

Current attained results First predictions attained with good machine learning metric but for only some types of failures

Classes of frameworks The solution will use the following classes of frameworks

- *FAMTS - Frameworks for Adaptive Maintenance Time Scheduling*
- *FEFF - Frameworks for Equipment Failure Forecasting*

1.2 AH2020.U1B - Bottleneck identification and prediction in the production line of an advanced water heater

Last info update 2021/08/03

Aims Improve value stream by adding embedded AI maintenance devices, forecasting equipment failure, generating intelligent scheduling of maintenance times, and **identifying and predicting bottlenecks**.

Location BoschTT

Co-promotors BoschTT, CM, UA

State Algorithmic study

Current attained results Several algorithms implemented for bottleneck identification; bottleneck prediction with good machine learning metrics

Classes of frameworks The solution will use the following classes of frameworks

- *FBIP - Frameworks for Bottleneck Identification and Prediction*

1.3 AH2020.U2 - Improvement of leakage test reliability of (gas) water heaters

Last info update 2021/08/03

Aims Reduce product rejections by validating (false) rejections with external sensors and predicting product rejections.

Location BoschTT

Co-promotors BoschTT, CM, FCUP, UA

State Data curation loop and algorithmic study

Current attained results First results obtained under some algorithms but results are not yet satisfatory

Classes of frameworks The solution will use the following classes of frameworks

- *FPRVF - Frameworks for Product Rejection Validation and Forecasting*

1.4 AH2020.U3 - Improvement of quality control tests on equipment testing PCBAs

Last info update 2021/08/03

Aims Improve PCBA quality test control by determining product rejection root causes and forecasting product NOKs.

Location BoschBT

Co-promotors BoschBT, CM, UA

State Data curation loop and algorithmic study

Current attained results First results obtained under some algorithms but results are not yet satisfatory

Classes of frameworks The solution will use the following classes of frameworks

- *FNPRC - Frameworks for NOK Prioritization and Root Cause*

1.5 AH2020.U4 - Improvement of maintenance plans of injection molding machines

Last info update 2021/08/03

Aims Improve maintenance plans by adding embedded AI maintenance devices; forecasting equipment failure and intelligent maintenance time scheduling.

Location OLI

Co-promotors CM, OLI, UA

State Data acquisition

Current attained results No results obtained; It is expected to apply a similar approach of U1A

Classes of frameworks The solution will use the following classes of frameworks

- *FAMTS - Frameworks for Adaptive Maintenance Time Scheduling*
- *FEFF - Frameworks for Equipment Failure Forecasting*

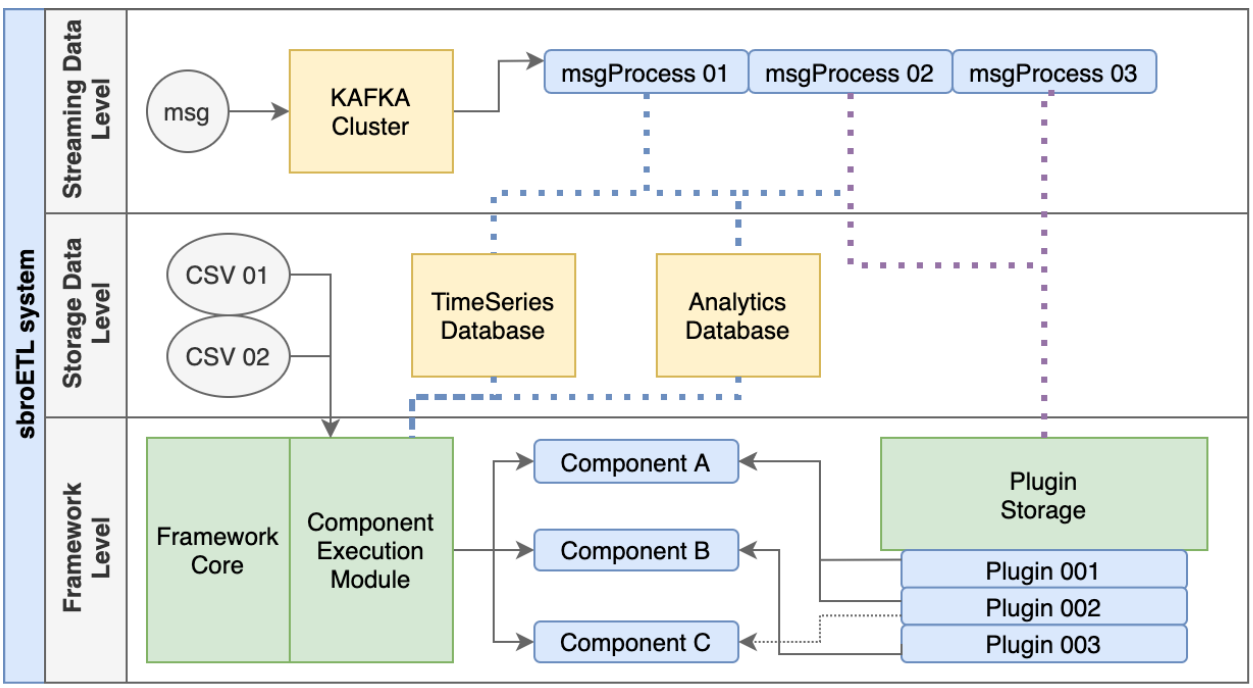
CHAPTER

TWO

CLASSES OF FRAMEWORKS

Frameworks are pieces of software (in the form of microservices) that are targeted for solving specific problems, e.g., for data injection, for ETL and analytics, for ML training, or for prediction. Each framework may have a set of components and/or plugins that implement some functionalities of the framework. All frameworks follow their own predefined input/output data format and communication channel, described in the so-called *Common Data Formats (CDFs)* of the framework.

All frameworks run under the sbroETL system, described in the following figure, and they are formed by a set of components that may use *Plugins*.



Mainly, there are two groups of frameworks: project frameworks and supplementary frameworks. Project frameworks are developed in the context of the project "Augmented Humanity" (AH2020), as described in the "Anexo Técnico", whereas supplementary frameworks are frameworks adding extra funcionalities, developped by third parties without using project resources.

2.1 FAMTS - Frameworks for Adaptive Maintenance Time Scheduling

To improve the existing preventive maintenance models, we propose an innovative framework which merges methodologies from time based maintenance, risk maintenance, condition based maintenance, and predictive maintenance; by generating maintenance times obtained from applying techniques of risk analysis, logic theory and machine learning.By far the most common approach in the Portuguese industry context is time based maintenance (TBM), which assumes that equipment service life and failure events can be determined by age, relying in the premise that TBM is (reasonably) cost effective when compared to methods with higher effort of assessment accuracy. However the majority of failure modes in a plant are not of TBM type. In FAMTS, to optimize the maintenance program we use time series forecasting techniques and, taking into account the inherent uncertainty associated with the occurrence of failures, merge it with algorithms based on the use of risk measures, such as the conditional value at risk. Those risk measures will also consider equipment failure forecasts (see *FEFF - Frameworks for Equipment Failure Forecasting*), when available. Although risk measures were initially developed to finance problems, their use rapidly spread to other management problems since they allow to measure, and consequently control, the occurrence of undesirable events provided that the distributions of the uncertain parameters is known. For the conditional based approach and when dealing with the discrete, nominal data, we intend to use the team expertise in Inductive Logic Programming (ILP) to extract interpretable rules from the data. Because ILP can handle relational data, other kinds of rules can be generated such as: "equipment with certain given characteristics that are near to other equipment/devices which have a given behavior are more likely to produce a bad behavior or a false bad behavior". These rules, that would be continuously inferred from the information gathered, will be also used to build an evolutive checkable model on a suitable Dynamic Logic, that will pave the way for a continuous and iterative maintenance analysis, in two ways. First, by supporting periodic checks of a set of obligation properties, in order to trigger valuable maintenance warnings; and secondly, by offering to the user the opportunity of checking directly complex execution scenarios and event combinations, using dynamic logics expressions. The combination of these techniques seems to be a significant push in maintenance time scheduling approaches.

2.1.1 FAMTS_csvBosch (cdf)

version 0.9

framework class *FAMTS - Frameworks for Adaptive Maintenance Time Scheduling*

input data format batch / csv file

data injection technique cron file watch

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team –

conception team

D. Almeida (BoschTT), P. Ramalho (BoschTT), E. Rocha (UA)

1. Description

Read the BoschTT file format and inject it to the internal analytics database.

2. Input Data

Source	Field	Type	Data Level	Observations
Sensor Data	dt	datetime	A	[1]
Sensor Data	locationID	varchar(150)	A	
Sensor Data	param_name	varchar(255)	A	[2]
Sensor Data	result_value	float	A	[3]
Sensor Data	unit	varchar(16)	A	[4]
Sensor Data	lower_tolerance	float	A	[5]
Sensor Data	upper_tolerance	float	A	[6]
Sensor Data	result_state	float	A	[7]
Sensor Data	workcycle_counter	float	A	[8]
Prod.Man.Data	locationID	varchar(150)	A	[9]
Prod.Man.Data	category	varchar	A	[10]
Prod.Man.Data	cause_number	int	A	[11]
Prod.Man.Data	description	nvarchar2(2000)	A	[12]
Prod.Man.Data	i_class	int	A	[13]
Prod.Man.Data	i_start	datetime	A	[14]
Prod.Man.Data	i_end	datetime	A	[15]
Prod.Man.Data	duration	datetime	A	[16]
Main. Data	order_type	varchar	A	[17]
Main. Data	order_number	varchar	A	[18]

Observations:

- [1]: Timestamp obtained when the measure is taken.
- [2]: Name of the Sensor
- [3]: Value Measured by the sensor
- [4]: Unit of the value that was measured
- [5]: Lower limit for the value measured
- [6]: Upper limit for the value measured
- [7]: Evaluation of the process of execution
- [8]: Indicates the number of reworks or if a part was processed multiple times
- [9]: Location where has ocurred the interruption
- [10] Interruption category
- [11]: Cause number of the interruption
- [12]: Interruption description
- [13]: (0 or 1) Indicates if an interruption was planed(0) or unplanned(1)
- [14]: Starting date and time of the interruption
- [15]: Ending date and time of the interruption
- [16]: Interruption duration

- [17]: Interruption type (PM01,PM02,PM03)
- [18]: Number of the order launched to resolve the interruption

3. Output Data

Output data is kept in the internal analytics database.

4. Components

Identification INJ01

Description Watch a directory for a new csv file and inject it to the internal analytics database in a normalized format.

Implementation team

E. Rocha (UA)

Plugins Do not use plugins.

2.1.2 FAMTS_csvOLI (cdf)

version 0.9

framework class *FAMTS - Frameworks for Adaptive Maintenance Time Scheduling*

input data format batch / csv file

data injection technique cron file watch

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team –

conception team

R. Antunes (OLI), E. Rocha (UA)

1. Description

Read the OLI file format and inject it to the internal analytics database.

2. Input Data

(Information will be added, when available)

3. Output Data

Output data is kept in the internal analytics database.

4. Components

Identification INJ02

Description Watch a directory for a new csv file and inject it to the internal analytics database in a normalized format.

Implementation team

E. Rocha (UA)

Plugins Do not use plugins.

2.1.3 FAMTS_GFTtrain (cdf)

version 0.9

framework class *FAMTS - Frameworks for Adaptive Maintenance Time Scheduling*

input data format internal analytics database

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team

E. Rocha (UA), P. Nunes (UA), I. Pereira (UA), A. Almeida (UA), J. Santos (UA)

conception team

E. Rocha (UA)

1. Description

From sensors data, this framework finds the best Generelized h-Fault Trees (GFTs) structures that models several classes of machine failures.

2. Input Data

Input data is obtained from the internal analytics database, already inject by *FAMTS_csvBosch (cdf)* or *FAMTS_csvOLI (cdf)*.

3. Output Data

Output data is kept in the internal analytics database to be used by the framework *FAMTS_GFTpredict (cdf)*.

4. Components

Identification GFTfit01

Description Find the best GFT structure to model the machine failure

Implementation team

E. Rocha (UA)

Plugins The following are the plugins used

- *x01_GFTfit*

2.1.4 FAMTS_GFTpredict (cdf)

version 0.9

framework class *FAMTS - Frameworks for Adaptive Maintenance Time Scheduling*

input data format internal analytics database

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team

E. Rocha (UA), P. Nunes (UA), I. Pereira (UA), A. Almeida (UA), J. Santos (UA)

conception team

E. Rocha (UA)

1. Description

Using Generelized h-Fault Trees (GFTs) structures, fitted from real data, this framework makes predictions about the failure events of a component in a machine, answering questions as

- What is the expected time of failure of a machine?
- What is the probability of failure of a component in the next T minutos?

2. Input Data

Input data is obtained from the internal analytics database, in particular, generated by the framework *FAMTS_GFTtrain (cdf)*.

3. Output Data

Results are kept in the internal analytics database, shared in PDF reports, and available for other PPS modules, e.g. for use in augmented reality.

4. Components

Identification GFTpredict01

Description Calculates the expected time of failure of a machine

Implementation team

E. Rocha (UA)

Plugins The following are the plugins used

- *x02_GFTpredict*

Identification GFTpredict02

Description Calculates the probability of failure at an instant of time

Implementation team

E. Rocha (UA)

Plugins The following are the plugins used

- *x02_GFTpredict*

2.1.5 Scientific Outputs

- Papers: [FAMTS01]

2.2 FBIP - Frameworks for Bottleneck Identification and Prediction

In industrial manufacturing environments, characterized as being stochastic, dynamic and often chaotic, disturbance handling is a crucial issue in the development of intelligent and reconfigurable manufacturing control systems. The occurrence of expected or unexpected bottlenecks, due to the cadence of some equipment or work stations, lead to deviations in the production plans. Usually it degrades the performance of the system, causing the loss of productivity and business opportunities, which are crucial roles to achieve competitiveness. Traditional manufacturing systems are not prepared to exhibit responsiveness and reconfigurability capabilities, since they are built upon centralized and hierarchical control structures. Those systems present good production optimization but a weak response to change due to the rigidity and centralization of their control structures, usually requiring the shutdown of the partial or even whole system when a bottleneck occurs. Hence the challenge faced by the manufacturing companies, to remain competitive, copes with the need to implement manufacturing control systems that exhibit innovative features, as agile response to the occurrence of the bottlenecks and dynamic re-configuration without stopping, re-programming or re-starting the process. A decision support system appears as suitable emergent paradigms to address this challenge, suggesting the definition of distributing control based on autonomous and cooperative units that account for the realization of efficient, flexible and robust overall plant control. However, a complete and integrated approach for bottlenecks analysis, that is able to detect, predict, diagnosis, re-planning and prognosis the major types of shop floor bottlenecks with impact at planning and scheduling level, is still missing.

The proposed bottleneck detection and handling decision support system, introduces the following innovative aspects: (a) automatic detection of bottlenecks; (b) differentiation between typical and atypical bottlenecks; (c) besides the bottlenecks detection and classification, also considers other kind of shop floor disturbances, such as delays and rush orders, that may have an impact at planning and scheduling level; (d) considers a distributed and reactive approach to the re-scheduling problem, executed as fast as possible to avoid the risk of degradation of the production activity; (e) integrates a prognosis component, allowing planning in advance the future occurrence of bottlenecks, transforming the traditional "fail and recovering" into "predict and prevent" practices. Furthermore, predictive analysis will be also pursued using machine learning. The expected results will contribute to help industrial companies to improve their competitiveness, namely through mechanisms and products that will enable these companies to be highly responsive to bottlenecks occurrences.

2.2.1 FBIP_csvBosch (cdf)

version 0.9

framework class *FBIP - Frameworks for Bottleneck Identification and Prediction*

input data format batch / csv file

data injection technique cron file watch

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team –

conception team

D. Almeida (BoschTT), P. Ramalho (BoschTT), E. Rocha (UA)

1. Description

Read the BoschTT file format and inject it to the internal analytics database.

2. Input Data

Field	Type	Data Level	Observations
dt	datetime	A	[1]
locationID	varchar(150)	A	
partnumberID	varchar(150)	A	
processingTime	float	B	[2]
processingTags	string	B	[3]

Observations:

- [1] Timestamp obtained when the partnumID leaves the locationID;
- [2] Time in seconds representing the processing time of the partnumberID on the locationID;
- [3] Space separated string with tags labelling the processingTime (e.g. stop_maintenance_curative)

3. Output Data

Output data is kept in the internal analytics database.

4. Components

Identification INJ01

Description Inject dataset D01 (L7 / day 2020-11-23 / shift T01)

Implementation team

E. Rocha (UA)

Plugins Do not use plugins.

Identification INJ02

Description Inject dataset D02 (L7 / day 2021-02-08 / shift T01)

Implementation team

E. Rocha (UA)

Plugins Do not use plugins.

Identification INJ03

Description Inject dataset D03 (L7 / day 2021-02-08 / shift T02)

Implementation team

E. Rocha (UA)

Plugins Do not use plugins.

Identification INJ04

Description Inject dataset D04 (L7 / day 2021-02-09 / shift T01)

Implementation team

E. Rocha (UA)

Plugins Do not use plugins.

Identification INJ05

Description Inject dataset D05 (L7 / day 2021-02-09 / shift T02)

Implementation team

E. Rocha (UA)

Plugins Do not use plugins.

Identification INJ06

Description Inject dataset D06 (L7 / day 2021-02-17 / shift T01)

Implementation team

E. Rocha (UA)

Plugins Do not use plugins.

Identification INJ07

Description Inject dataset D07 (L7 / day 2021-02-17 / shift T02)

Implementation team

E. Rocha (UA)

Plugins Do not use plugins.

Identification INJ08

Description Inject dataset D06 (L7 / day 2021-02-18 / shift T01)

Implementation team

E. Rocha (UA)

Plugins Do not use plugins.

Identification INJ09

Description Inject dataset D07 (L7 / day 2021-02-18 / shift T02)

Implementation team

E. Rocha (UA)

Plugins Do not use plugins.

2.2.2 FBIP_identify (cdf)

version 0.9

framework class *FBIP - Frameworks for Bottleneck Identification and Prediction*

input data format internal analytics database

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team

A. Moura (UA), A. de Sousa (UA), E. Rocha (UA), A. Brochado (AH2020 PhD fellow / UA), D. Almeida (BoschTT)

scientific outputs [FBIP02]

conception team

E. Rocha (UA), A. Brochado (AH2020 PhD fellow / UA)

1. Description

Calculate several bottleneck metrics. This framework has the following capabilities [level>=A]:

- CA01: Automatic determination of the locationId sequences;
- CA02: Calculation of the transition probability matrix from location i into j;
- CA03: Identification of reprocessed partnumberID at the same locationID;
- CA04: Identification of partnumberID belonging to the previous shift;
- CA05: Identification of partnumberID not concluded in the current shift;

and the capabilities [level>=B]:

- CA06: Determination of the Ideal Cycle Time (ICT) for each locationID;
- CA07: Calculation of several bottleneck metrics as sole bottleneck, shifting bottleneck, queue bottleneck, beside others.

2. Input Data

Input data is obtained from the internal analytics database.

3. Output Data

Results are kept in the internal analytics database, shared as PDF reports, and available for other PPS modules, e.g. for use in augmented reality.

4. Components

Identification DASH01

Description Generate data for dashboardings visualization

Implementation team

E. Rocha (UA)

Plugins Do not use plugins

Identification FBIP01

Description Extract unique locationID, extract unique partnumberID, and determine the predominant localtionOrder and fix internal IDs for locations and partnumbers

Implementation team

E. Rocha (UA)

Plugins The component uses the following plugins

- *fbip01_path_order*

Identification FBIP02

Description Validate paths and find part numbers not valid in this scenario and remove marked partnumberID

Implementation team

E. Rocha (UA)

Plugins The component uses the following plugins

- *fbip02_path_validation*

Identification FBIP03

Description Calculate node bottleneck metrics (as AMPM, AQPM, and ATPM) and show some statistics

Implementation team

E. Rocha (UA)

Plugins The component uses the following plugins

- *fbip03_locstats*

Identification FBIP04

Description Generate records for the start of the processingTime

Implementation team

E. Rocha (UA)

Plugins The component uses the following plugins

- *fbip04_end2start*

Identification FBIP05

Description Recalculate processingTime metrics

Implementation team

E. Rocha (UA)

Plugins The component uses the following plugins

- *fbip05_stats*

Identification FBIP06

Description Generate the state variables of the system

Implementation team

E. Rocha (UA)

Plugins The component uses the following plugins

- *fbip06_state*

Identification FBIP07

Description Realculate node bottleneck metrics (AAPM variants as AMPM, AQPM, ATPM or median variants) and show some statistics

Implementation team

E. Rocha (UA)

Plugins The component uses the following plugins

- *fbip07_nodeBottleneck*

Identification FBIP08

Description Propagate metrics from nodeID to data

Implementation team

E. Rocha (UA)

Plugins The component uses the following plugins

- *fbip08_node2data*

Identification FBIP09

Description Generate the bottleneck metric graphs

Implementation team

E. Rocha (UA)

Plugins The component uses the following plugins

- *fbip09_graphs*

Identification FBIP10

Description Build FBIP reports

Implementation team

E. Rocha (UA)

Plugins The component uses the following plugins

- *fbip10_report*

2.2.3 FBIP_train (cdf)

version 0.9

framework class *FBIP - Frameworks for Bottleneck Identification and Prediction*

input data format internal analytics database

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team

E. Rocha (UA), P. Georgieva (UA), M.J. Lopes (BoschTT)

scientific outputs [FBIP01], [FBIP04]

conception team

E. Rocha (UA), M.J. Lopes (BoschTT)

1. Description

Train several machine learning classifiers to determine the best model that forecast the bottleneck machine in the next N minutes.

2. Input Data

Input data is obtained from the internal analytics database, in particular, from the outputs of the framework *FBIP_identify (cdf)*.

3. Output Data

Input data is obtained from the internal analytics database.

4. Components

Identification TRAIN01

Description Train a Random Forest model to forecast of the next bottleneck

Implementation team M.J. Lopes (BoschTT)

Plugins This component uses the following plugins

- *x03_MJ_FBIPpredict*

Identification TRAIN02

Description Train several classifiers for the forecast of the next bottleneck

Implementation team

E. Rocha (UA)

Plugins This component uses the following plugins

- *tb01_generalClassifier_train*

2.2.4 FBIP_predict (cdf)

version 0.9

framework class *FBIP - Frameworks for Bottleneck Identification and Prediction*

input data format internal analytics database

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team

E. Rocha (UA), P. Georgieva (UA), M.J. Lopes (BoschTT)

scientific outputs [FBIP01], [FBIP04]

conception team

E. Rocha (UA), M.J. Lopes (BoschTT)

1. Description

Deploy the best model, trained with *FBIP_train (cdf)*, to forecast the bottleneck machine in the next N minutes.

2. Input Data

Input data is obtained from the internal analytics database.

3. Output Data

Results are kept in the internal analytics database, shared in PDF reports, and available for other PPS modules, e.g. for use in augmented reality.

4. Components

Identification PRED01

Description Forecast the bottleneck machine in the next N minutes

Implementation team

E. Rocha (UA)

Plugins This component uses the following plugins

- *tb02_generalClassifier_predict*

2.2.5 FBIP_routeCause (cdf)

version 0.9

framework class *FBIP - Frameworks for Bottleneck Identification and Prediction*

input data format internal analytics database

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team

E. Rocha (UA)

conception team

E. Rocha (UA)

1. Description

Determine the contributions of process features for a machine to be a bottleneck.

2. Input Data

Input data is obtained from the internal analytics database, in particular, from the outputs of the framework *FBIP_identify (cdf)*.

3. Output Data

Results are kept in the internal analytics database, shared in PDF reports, and available for other PPS modules, e.g. for use in augmented reality.

4. Components

Identification RC01

Description Determine the contributions of process features for a machine to be a bottleneck

Implementation team

E. Rocha (UA)

Plugins This component uses the followin plugins

- *rc01_processRootCause*

2.2.6 Scientific Outputs

- Papers: [FBIP01], [FBIP02], [FBIP03], [FBIP04]

2.3 FEFF - Frameworks for Equipment Failure Forecasting

Adapted to the specific features of the Portuguese industrial situation and to effectively address maintenance problems, we propose a decentralized solution for predictive maintenance, on the deploying mode part of AAFM, merged with a centralized solution on the calibration and training mode part of AAFM. The decentralized solution will have embedded predictive maintenance algorithms in edge devices (e.g. XDK/Bosch IOTiP/Fraunhofer14, or DEMA/UA), allowing an effective acquisition and preprocessing of maintenance data to be globally processed by the CTM part, e.g. using machine learning techniques. FEEF will detect and predict anomalies and failure/degradation patterns, providing early warnings and as a result allowing to adapt maintenance plans in advance. This may result in several cost savings by avoiding or minimizing the downtimes or reducing the frequency of maintenance (i.e. optimize maintenance operations). Although not new, predictive maintenance still has a growing relevance in modern industry. The existing centralized architectures for this type of problem are based on the application of predictive maintenance techniques in the cloud (central element), thus the entire process is dependent on a single element, and a failure of the communication between machine and central can inhibit an alarm which will cause possible critical equipment failure. Furthermore, existing decentralized architectures do not have the ability to predict the time window in which the failure will occur, so there is still no decentralized global solution that is effectively efficient in real time, without also merging it with a centralized global approach; which means that an orchestration of both solutions is required. Furthermore, each edge product with embedded algorithms (at the DM level of AAFM) will provide SOAP/REST services allowing easier integration into anymanufacturing environment and communication between shop-floor equipment.

2.3.1 FEFF_csvBosch (cdf)

version 0.9

framework class *FEFF - Frameworks for Equipment Failure Forecasting*

input data format batch / csv file

data injection technique cron file watch

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team –

conception team

D. Almeida (BoschTT), P. Ramalho (BoschTT), E. Rocha (UA)

1. Description

Read the BoschTT file format and inject it to the internal analytics database.

2. Input Data

Source	Field	Type	Data Level	Observations
Sensor Data	dt	datetime	A	[1]
Sensor Data	locationID	varchar(150)	A	
Sensor Data	param_name	varchar(255)	A	[2]
Sensor Data	result_value	float	A	[3]
Sensor Data	unit	varchar(16)	A	[4]
Sensor Data	lower_tolerance	float	A	[5]
Sensor Data	upper_tolerance	float	A	[6]
Sensor Data	result_state	float	A	[7]
Sensor Data	workcycle_counter	float	A	[8]
Prod.Man.Data	locationID	varchar(150)	A	[9]
Prod.Man.Data	category	varchar	A	[10]
Prod.Man.Data	cause_number	int	A	[11]
Prod.Man.Data	description	nvarchar2(2000)	A	[12]
Prod.Man.Data	i_class	int	A	[13]
Prod.Man.Data	i_start	datetime	A	[14]
Prod.Man.Data	i_end	datetime	A	[15]
Prod.Man.Data	duration	datetime	A	[16]
Main. Data	order_type	varchar	A	[17]
Main. Data	order_number	varchar	A	[18]

Observations:

- [1]: Timestamp obtained when the measure is taken.
- [2]: Name of the Sensor
- [3]: Value Measured by the sensor
- [4]: Unit of the value that was measured
- [5]: Lower limit for the value measured
- [6]: Upper limit for the value measured
- [7]: Evaluation of the process of execution
- [8]: Indicates the number of reworks or if a part was processed multiple times
- [9]: Location where has ocurred the interruption
- [10] Interruption category
- [11]: Cause number of the interruption
- [12]: Interruption description
- [13]: (0 or 1) Indicates if an interruption was planed(0) or unplanned(1)
- [14]: Starting date and time of the interruption
- [15]: Ending date and time of the interruption
- [16]: Interruption duration
- [17]: Interruption type (PM01,PM02,PM03)
- [18]: Number of the order launched to resolve the interruption

3. Output Data

Output data is kept in the internal analytics database.

4. Components

Identification INJ01

Description Watch a directory for a new csv file and inject it to the internal analytics database in a normalized format.

Implementation team

E. Rocha (UA)

Plugins Do not use plugins.

2.3.2 FEFF_csvOLI (cdf)

version 0.9

framework class *FEFF - Frameworks for Equipment Failure Forecasting*

input data format batch / csv file

data injection technique cron file watch

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team –

conception team

R. Antunes (OLI), E. Rocha (UA)

1. Description

Read the OLI file format and inject it to the internal analytics database.

2. Input Data

(Information will be added, when available)

3. Output Data

Output data is kept in the internal analytics database.

4. Components

Identification INJ02

Description Watch a directory for a new csv file and inject it to the internal analytics database in a normalized format.

Implementation team

E. Rocha (UA)

Plugins Do not use plugins.

2.3.3 FEFF_train (cdf)

version 0.9

framework class *FEFF - Frameworks for Equipment Failure Forecasting*

input data format internal analytics database

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team

E. Rocha (UA), D. Costa (AH2020 PhD fellow / UA), D. Coelho (MSc student, UA), J. Santos (UA)

scientific outputs [FEFF01], [FEFF02]

conception team

E. Rocha (UA), D. Costa (AH2020 PhD fellow / UA), D. Coelho (MSc student, UA)

1. Description

From sensors data and event data, this framework do feature enginerring, dimension reduction, anomaly detection, and trains several machine learning classifiers.

2. Input Data

Input data is obtained from the internal analytics database.

3. Output Data

Output data is kept in the internal analytics database to be used by the framework *FEFF_predict (cdf)*.

4. Components

Identification ANN01

Description Find the last (reduced) windowed anomalies by COPOD

Implementation team

D. Coelho (MSc student, UA), E. Rocha (UA)

Plugins The following are the plugins used

- *x04_DC1_FEFFann*

Identification TRAIN01

Description Train some machine learning models for failure forecasting

Implementation team

D. Costa (AH2020 PhD fellow / UA)

Plugins The following are the plugins used

- *x05_DC_FEFFtrain*

Identification TRAIN02

Description Train some machine learning models for failure forecasting

Implementation team

E. Rocha (UA)

Plugins The following are the plugins used

- *tb01_generalClassifier_train*

2.3.4 FEFF_predict (cdf)

version 0.9

framework class *FEFF - Frameworks for Equipment Failure Forecasting*

input data format internal analytics database

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team

E. Rocha (UA), D. Costa (AH2020 PhD fellow / UA), D. Coelho (MSc student, UA), J. Santos (UA)

scientific outputs [FEFF01], [FEFF02]

conception team

E. Rocha (UA), D. Costa (AH2020 PhD fellow / UA)

1. Description

For each failure event class, this framework deploys the best model, trained by the framework *FEFF_train (cdf)*, in order to forecast failures in the next T minutes.

2. Input Data

Input data is obtained from the internal analytics database, in particular, generated by the framework *FAMTS_GFTtrain (cdf)*.

3. Output Data

Results are kept in the internal analytics database, shared in PDF reports, and available for other PPS modules, e.g. for use in augmented reality.

4. Components

Identification PRED01

Description Forecast failures of machines using component TRAIN01 of *FEFF_train (cdf)*.

Implementation team

D. Costa (AH2020 PhD fellow / UA)

Plugins The following are the plugins used

- *x06_DC_FEFFpredict*

Identification PRED02

Description Forecast failures of machines using component TRAIN02 of *FEFF_train (cdf)*.

Implementation team

E. Rocha (UA)

Plugins The following are the plugins used

- plugin_tb01_generalClassifier_predict

2.3.5 Scientific Outputs

- Papers: [FEFF01]

- Master Thesis: [FEFF02]

2.4 FNPRC - Frameworks for NOK Prioritization and Root Cause

We propose a dependency network approach to the problem of minimizing errors in production and quickly identifying the root cause of such problems (e.g. NOK, failures). Complex Network Theory will be applied to approach quality control. This approach is flexible, scalable and gives good estimates with a reasonable computational overhead. Industrial products usually result from the integration of many components, in many production steps, from numerous raw materials and suppliers, involving various equipment and personal. The idea is to represent the production process as a directed network. Nodes in that network represent steps in the process, operators or components, and edges represent dependency or influence relationships. The root causes of an anomaly or failure may then be identifiedusing graph theory methods. In particular, factory maintenance teams gather information through documents detailing events, identifying involved systems and documenting solutions. This constitutes a basis of an Expert System which can be data mined in order to produce a dependency network. Such a dependency network approach has been applied to the root cause analysis problem in telecommunication systems, software development, infrastructure supply networks, among others. To the best of our knowledge, it has not been applied to factory management systems.The main discussed issue arises when a small number of original failures cascade through such a network, leading to large set of failures. The problem then becomes identifying the root cause of a failure, given the final state and the dependency network. The exact solution is an NP-complete problem. A heuristic approach to the closely related problem of finding the minimal set to damage the whole network, iteratively calculating weights for nodes in the network converges rapidly, and gives excellent results compared with other computationally intensive methods. These frameworks will significantly facilitate the work of maintenance teams, in scenarios with complex relations and big datasets of information.

2.4.1 FNPRC_csvBosch (cdf)

version 0.9

framework class *FNPRC - Frameworks for NOK Prioritization and Root Cause*

input data format batch / csv file

data injection technique cron file watch

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team –

conception team

A. Calçada (BoschBT), A. Leite (BoschBT), E. Rocha (UA)

1. Description

Read the BoschBT text files, generated by the so-called **CheckSum Analyst ems** (CAe) test machines of printed circuit board assembly (PCBA), and inject then to the internal analytics database.

2. Input Data

Text files generated by CAe.

3. Output Data

Output data is kept in the internal analytics database.

4. Components

Identification INJ01

Description Watch a directory for a new csv file and inject it to the internal analytics database in a normalized format.

Implementation team

E. Rocha (UA)

Plugins Do not use plugins.

2.4.2 FNPRC_stepStats (cdf)

version 0.9

framework class *FNPRC - Frameworks for NOK Prioritization and Root Cause*

input data format internal analytics database

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team

E. Rocha (UA), M. Pinto (MSc student, UA)

conception team

E. Rocha (UA), M. Pinto (MSc student, UA)

1. Description

Process the test files, generated by the so-called **CheckSum Analyst ems** (CAe) test machines of printed circuit board assembly (PCBA), and produce several statistics.

2. Input Data

Input data is obtained from the internal analytics database.

3. Output Data

Results are kept in the internal analytics database.

4. Components

Identification RUN01

Description Produce step statistics in quality tests

Implementation team

E. Rocha (UA), M. Pinto (MSc student, UA)

Plugins The following are the plugins used

- *x07_stepStats*

2.4.3 FNPRC_train (cdf)

version 0.9

framework class *FNPRC - Frameworks for NOK Prioritization and Root Cause*

input data format internal analytics database

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team

E. Rocha (UA), I. Pereira (UA), M. Pinto (MSc student / UA)

conception team

E. Rocha (UA), M. Pinto (MSc student / UA)

1. Description

From step information, this framework do feature enginerring, dimension reduction, anomaly detection, and trains several machine learning classifiers to find a good model for step NOK forecast.

2. Input Data

Input data is obtained from the internal analytics database.

3. Output Data

Output data is kept in the internal analytics database to be used by the framework *FNPRC_predict (cdf)*.

4. Components

Identification TRAIN01

Description Train several classifiers for the forecast step NOKs

Implementation team

E. Rocha (UA)

Plugins The following are the plugins used

- *tb01_generalClassifier_train*

2.4.4 FNPRC_predict (cdf)

version 0.9

framework class *FNPRC - Frameworks for NOK Prioritization and Root Cause*

input data format internal analytics database

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team

E. Rocha (UA), I. Pereira (UA), M. Pinto (MSc student / UA)

conception team

E. Rocha (UA), M. Pinto (MSc student / UA)

1. Description

From sensors data and event data, this framework do feature enginerring, dimension reduction, anomaly detection, and trains several machine learning classifiers.

2. Input Data

Input data is obtained from the internal analytics database.

3. Output Data

Results are kept in the internal analytics database, shared in PDF reports, and available for other PPS modules, e.g. for use in augmented reality.

4. Components

Identification PRED01

Description Forecast step NOKs

Implementation team

E. Rocha (UA)

Plugins The following are the plugins used

- *tb02_generalClassifier_predict*

2.4.5 Scientific Outputs

- Master Thesis: [FNPRC01]

2.5 FPRVF - Frameworks for Product Rejection Validation and Forecasting

Technology allowed to build devices/equipment that are quite reliable, but at the same time, very sensitive. Although this sensitiveness is controlled, equipment bad behavior can happen in situations that do not actually represent it. Factors such as changes in ambient temperature, vibration, bad localization or usage time could affect the quality of their responses. Such bad behavior on quality control equipment usually reflects on product NOKs or (false) rejections, being some rejections induced by the test process itself. An advantage today is that large amounts of data related to devices/equipment functioning, their surroundings andenvironment, and product characteristics, are collected on a daily/hourly/minute/second basis and stored in databases, clouds or as simple files in hard disks. This data becomes a very rich source of information that can allow: (1) speedy reactions to simple situations (like replacement of an old or faulty device) and to extreme situations (for example, prediction of false rejections before they happen), (2) emission of timely alerts, or (3) prediction of non-optimal or hazard situations. This data can also be used to build specialized behavior and prediction models according to the kind of product tests done. We can use the behavior models to assess differences among test machines of the same type, placed in different locations or placed in the same location, in the same pipeline or in distinct pipelines. We could also assess differences between test equipment of different types. Studies about these behaviors could be useful to plan future installations, where one wants to minimize false bad behaviors and general costs. Moreover, predicting NOKs allow to introduce corrective measures on the test plans or even correct design issues in test equipment.

2.5.1 FPRVF_csvBosch (cdf)

version 0.9

framework class *FPRVF - Frameworks for Product Rejection Validation and Forecasting*

input data format batch / csv file

data injection technique cron file watch

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team –

conception team

D. Almeida (BoschTT), P. Ramalho (BoschTT), E. Rocha (UA)

1. Description

Read the BoschTT file format and inject it to the internal analytics database.

2. Input Data

Source	Field	Type	Data Level	Observations
Sensor Data	dt	datetime	A	[1]
Sensor Data	locationID	varchar(150)	A	
Sensor Data	param_name	varchar(255)	A	[2]
Sensor Data	result_value	float	A	[3]
Sensor Data	unit	varchar(16)	A	[4]
Sensor Data	lower_tolerance	float	A	[5]
Sensor Data	upper_tolerance	float	A	[6]
Sensor Data	result_state	float	A	[7]
Sensor Data	workcycle_counter	float	A	[8]

Observations:

- [1]: Timestamp obtained when the measure is taken.
- [2]: Name of the Sensor
- [3]: Value Measured by the sensor
- [4]: Unit of the value that was measured
- [5]: Lower limit for the value measured
- [6]: Upper limit for the value measured
- [7]: Evaluation of the process of execution
- [8]: Indicates the number of reworks or if a part was processed multiple times

3. Output Data

Output data is kept in the internal analytics database.

4. Components

Identification INJ01

Description Watch a directory for a new csv file and inject it to the internal analytics database in a normalized format.

Implementation team

E. Rocha (UA)

Plugins Do not use plugins.

2.5.2 FPRVF_FCUP (cdf)

version 0.9

framework class *FPRVF - Frameworks for Product Rejection Validation and Forecasting*

input data format internal analytics database

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team Inês Dutra (FCUP), F. Rocha (AH2020 PhD fellow / FCUP)

conception team Inês Dutra (FCUP), F. Rocha (AH2020 PhD fellow / FCUP)

1. Description

(Information will be added, when available)

2. Input Data

(Information will be added, when available)

3. Output Data

(Information will be added, when available)

4. Components

Identification RUN01

Description (Information will be added, when available)

Implementation team Inês Dutra (FCUP), F. Rocha (AH2020 PhD fellow / FCUP)

Plugins The following are the plugins used

- plugin_FPRVF_FCUP

2.5.3 FPRVF_UAtrain (cdf)

version 0.9

framework class *FPRVF - Frameworks for Product Rejection Validation and Forecasting*

input data format internal analytics database

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team

E. Rocha (UA), D. Costa (AH2020 PhD fellow / UA), J. Santos (UA)

conception team

E. Rocha (UA), D. Costa (AH2020 PhD fellow / UA)

1. Description

(Information will be added, when available)

2. Input Data

Input data is obtained from the internal analytics database.

3. Output Data

Output data is kept in the internal analytics database to be used by the framework *FPRVF_UApredict (cdf)*.

4. Components

Identification TRAIN01

Description (Information will be added, when available)

Implementation team

E. Rocha (UA)

Plugins The following are the plugins used

- plugin_tb03_generalRegressor_train

2.5.4 FPRVF_UApredict (cdf)

version 0.9

framework class *FPRVF - Frameworks for Product Rejection Validation and Forecasting*

input data format internal analytics database

data processing internal analytics database

data output internal analytics database

deploy format microservice

scientific team

E. Rocha (UA), D. Costa (AH2020 PhD fellow / UA), J. Santos (UA)

conception team

E. Rocha (UA), D. Costa (AH2020 PhD fellow / UA)

1. Description

(Information will be added, when available)

2. Input Data

Input data is obtained from the internal analytics database.

3. Output Data

Output data is kept in the internal analytics database to be used by the framework *FPRVF_UApredict (cdf)*.

4. Components

Identification PRED01

Description (Information will be added, when available)

Implementation team

E. Rocha (UA)

Plugins The following are the plugins used

- plugin_tb04_generalRegressor_predict

CHAPTER

THREE

COMMON DATA FORMATS (CDFS)

Common Data Formats (CDFs) are specifications of input/output data formats and communication channels of the frameworks.

CDFs of Project Frameworks

1. *FAMTS_csvBosch (cdf)*
2. *FAMTS_csvOLI (cdf)*
3. *FAMTS_GFTtrain (cdf)*
4. *FAMTS_GFTpredict (cdf)*
5. *FBIP_csvBosch (cdf)*
6. *FBIP_identify (cdf)*
7. *FBIP_train (cdf)*
8. *FBIP_predict (cdf)*
9. *FEFF_csvBosch (cdf)*
10. *FEFF_csvOLI (cdf)*
11. *FEFF_train (cdf)*
12. *FEFF_predict (cdf)*
13. *FNPRC_csvBosch (cdf)*
14. *FNPRC_stepStats (cdf)*
15. *FNPRC_train (cdf)*
16. *FNPRC_predict (cdf)*
17. *FPRVF_csvBosch (cdf)*
18. *FPRVF_FCUP (cdf)*
19. *FPRVF_UAtrain (cdf)*
20. *FPRVF_UApredict (cdf)*

CDFs of Supplementary Frameworks

- *FBIP_routeCause (cdf)*

CHAPTER

FOUR

PLUGINS

Plugins are blocks of code that can be used in batch mode or streaming mode, which are deployed by framework components (see *Classes of Frameworks*), and implement the version of a specific algorithm generating controlled output metrics. Such allows to dynamically choose the best plugin for a particular problem solution. Plugins are i87 binary files with execution controlled by the sbroETL system and audit managed by block chain technology. Plugins may have associated royalties.

List of Project Plugins

4.1 fbip01_path_order

Version 1.0

Type Plugin

Developer

E. Rocha (UA)

Aims Extract unique locationID, extract unique partnumberID, and determine the predominant locationOrder and fix internal IDs for locations and partnumbers

4.2 fbip02_path_validation

Version 1.0

Tupe Plugin

Developer

E. Rocha (UA)

Aims Validate paths and find part numbers not valid in this scenario and remove marked partnumberID

4.3 fbip03_locstats

Version 1.0

Type Plugin

Developer

E. Rocha (UA)

Aims Calculate node bottleneck metrics (as AMPM, AQPM, and ATPM) and show some statistics

4.4 fbip04_end2start

Version 1.0

Type Plugin

Developer

E. Rocha (UA)

Aims Generate records for the start of the processingTime

4.5 fbip05_stats

Version 1.0

Type Plugin

Developer

E. Rocha (UA)

Aims Recalculate processingTime metrics

4.6 fbip06_state

Version 1.0

Type Plugin

Developer

E. Rocha (UA)

Aims Generate the state variables of the system

4.7 fbip07_nodeBottleneck

Version 1.0

Type Plugin

Developer

E. Rocha (UA)

Aims Realculate node bottleneck metrics (AAPM variants as AMPM, AQPM, ATPM or median variants) and show some statistics

4.8 fbip08_node2data

Version 1.0

Type Plugin

Developer

E. Rocha (UA)

Aims Propagate metrics from nodeID to data

4.9 fbip09_graphs

Version 1.0

Type Plugin

Developer

E. Rocha (UA)

Aims Generate the bottleneck metric graphs

4.10 fbip10_report

Version 1.0

Type Plugin

Developer

E. Rocha (UA)

Aims Build FBIP reports

4.11 x03_MJ_FBIPpredict

Version 1.0

Type Plugin

Developer M.J. Lopes (BoschTT)

Aims Train a Random Forest model to forecast of the next bottleneck

4.12 x04_DC1_FEFFann

Version 1.0

Type Plugin

Developer

D. Coelho (MSc student, UA), E. Rocha (UA)

Aims Find the last (reduced) windowed anomalies by COPOD

4.13 x05_DC_FEFFtrain

Version 1.0

Type Plugin

Developer

D. Costa (AH2020 PhD fellow / UA)

Aims Train some machine learning models for failure forecasting

4.14 x06_DC_FEFFpredict

Version 1.0

Type Plugin

Developer

D. Costa (AH2020 PhD fellow / UA)

Aims Forecast failures of machines using component TRAIN01 of *x05_DC_FEFFtrain*

4.15 x07_stepStats

Version 1.0

Type Plugin

Developer sBRO

Aims Produce step statistics in quality tests

List of Supplementary Plugins

4.16 rc01_processRootCause

Version 1.0

Type Plugin

Developer sBRO

Aims Determine the contributions of process features to occurence of a target event

4.17 tb01_generalClassifier_train

Version 1.0

Type Plugin

Developer sBRO

Aims Explore some data encodings, data normalizations, feature aggregations, dimension reductions, and train several machine learning classifiers as Bernoulli Naive Bayes, Decision Trees, Extra Trees, Gaussian Naive Bayes, Gradient Boosting, K-Neighbors, Linear Model SGDC, Linear SVC, Logistic Regression, Multinomial Naive Bayes, MLP, Random Forest, XGBoost.

4.18 tb02_generalClassifier_predict

Version 1.0

Type Plugin

Developer sBRO

Aims Predict the target label, using the best model trained with *tb01_generalClassifier_train*.

4.19 x01_GFTfit

Version 1.0

Type Plugin

Developer sBRO

Aims From data modelling a real physical phenomena (e.g., the failure of a component on a machine), this plugin finds the Generelized h-Fault Trees (GFTs) structure that best fit a target event distribution from N given basic events distributions.

4.20 x02_GFTpredict

Version 1.0

Type Plugin

Developer sBRO

Aims From a given Generelized h-Fault Trees (GFTs) structure, fitted by *x01_GFTfit*, determines the expected occurrence time of the target event or the probability of the target event to occur at a specific instant of time.

CHAPTER

FIVE

OUTPUTS

Some of the outputs are:

5.1 D11.1 - Common Data Formats (CDF) version 0.9

First specification of the common data formats of the official frameworks.

CDFs of Project Frameworks

Class *FAMTS - Frameworks for Adaptive Maintenance Time Scheduling*

1. *FAMTS_csvBosch (cdf)*
2. *FAMTS_csvOLI (cdf)*
3. *FAMTS_GFTtrain (cdf)*
4. *FAMTS_GFTpredict (cdf)*

Class *FBIP - Frameworks for Bottleneck Identification and Prediction*

1. *FBIP_csvBosch (cdf)*
2. *FBIP_identify (cdf)*
3. *FBIP_train (cdf)*
4. *FBIP_predict (cdf)*

Class *FEFF - Frameworks for Equipment Failure Forecasting*

1. *FEFF_csvBosch (cdf)*
2. *FEFF_csvOLI (cdf)*
3. *FEFF_train (cdf)*
4. *FEFF_predict (cdf)*

Class *FNPRC - Frameworks for NOK Prioritization and Root Cause*

1. *FNPRC_csvBosch (cdf)*
2. *FNPRC_stepStats (cdf)*
3. *FNPRC_train (cdf)*
4. *FNPRC_predict (cdf)*

Class *FPRVF - Frameworks for Product Rejection Validation and Forecasting*

1. *FPRVF_csvBosch (cdf)*

2. *FPRVF_FCUP (cdf)*
3. *FPRVF_UAtrain (cdf)*
4. *FPRVF_UApredict (cdf)*

CHAPTER

SIX

ABBREVIATIONS

- BoschBT: Bosch Building Technologies
- BoschTT: Bosch TermoTechnology
- CM: Critical Manufacturing
- FCUP: Faculdade de Ciências da Universidade do Porto
- OLI: Oliveira e Irmãos
- sBRO: Smart Business and Research Observatory (external)
- UA: Universidade de Aveiro

BIBLIOGRAPHY

[FAMTS01] E.M Rocha, P. Nunes, Generelized h-Fault Trees (in preparation), 2021

[FBIP01] M.J. Lopes, E.M. Rocha, P. Georgieva, N. Ferreira. General Model for Metrics Calculation and Behavior Prediction in Manufacturing Industry", chapter of the Handbook of Research on Applied Data Science and Artificial Intelligence in Business and Industry, V. Chkoniya (ed.), vol. 2, 263-290, IGI-Global, June 2021.

[FBIP02] A.F. Brochado, E.M. Rocha, D. Almeida, A. de Sousa, A. Moura. Data-Driven Bottleneck Identification with Minimal Information, pg.24, submitted in 2021

[FBIP03] E.M. Rocha, A. Brochado, A. Moura. Workers benchmarking using multi-directional efficiency analysis in a manufacturing production system, pg.11, submitted in 2021

[FBIP04] E.M. Rocha, M.J. Lopes. Bottleneck prediction and data-driven discrete-event simulation for a balanced manufacturing line, pg. 9, submitted in 2021

[FEFF01] D. Coelho, D. Costa, E.M. Rocha, D. Almeida, J.P. Santos. Predictive maintenance on sensorized stamping presses by time series segmentation, anomaly detection, and classification algorithms, 3rd International Conference on Industry 4.0 and Smart Manufacturing - ISM 2021, accepted in 2021

[FEFF02] D. Coelho, "Uma abordagem de manutenção preditiva baseada na segmentação de séries temporais", supervisors: J.Santos, E.Rocha, UA, 2021

[FNPRC01] M. Pinto, "Controlo de Qualidade de PCBA, Classificação de Erros e Recomendação de Reparação", supervisor: E. Rocha, UA, 2021