International Journal of Computer Integrated Manufacturing

ISSN: 0951-192X (Print) 1362-3052 (Online) Journal homepage: https://www.tandfonline.com/loi/tcim20

An online machine learning framework for early detection of product failures in an Industry 4.0 context

J. A. Carvajal Soto, F. Tavakolizadeh & D. Gyulai

To cite this article: J. A. Carvajal Soto, F. Tavakolizadeh & D. Gyulai (2019) An online machine learning framework for early detection of product failures in an Industry 4.0 context, International Journal of Computer Integrated Manufacturing, 32:4-5, 452-465, DOI: 10.1080/0951192X.2019.1571238

To link to this article: https://doi.org/10.1080/0951192X.2019.1571238

INTERNATIONAL JOURNAL OF COMPUTER INTEGRATED MANUFACTURING
2019, VOL. 32, NOS. 4–5, 452–465
https://doi.org/10.1080/0951192X.2019.1571238

An online machine learning framework for early detection of product failures in an Industry 4.0 context

J. A. Carvajal Soto[a], F. Tavakolizadeh [a] and D. Gyulai [b,c]

[a]User-Centered Computing Department, Fraunhofer Institute for Applied Information Technology FIT, Sankt Augustin, Germany; [b]Institute for Computer Science and Control (SZTAKI), Hungarian Academy of Sciences (MTA), Budapest, Hungary; [c]Department of Manufacturing Science and Technology, Budapest University of Technology and Economics, Budapest, Hungary

ABSTRACT

Current paradigms such as the Internet of Things (IoT) and cyber-physical systems are transforming production environments, where related processes are not only faster and with higher standards, but also more flexible and adaptable to changes in the environment. To address the ever-increasing flexibility requirements while keeping current production standards, a new set of technologies is needed. This paper presents an IoT machine learning and orchestration framework, applied to detection of failures of surface mount devices during production. The paper shows how to build a scalable and flexible system for real-time, online machine learning. Furthermore, the approach is evaluated by using a novel and realistic simulation of a production line for electronic devices as a case study. The system evaluation is done in a holistic manner by analyzing various aspects involving the software architecture, computational scalability, model accuracy, production performance, among others.

ARTICLE HISTORY
Received 31 May 2018
Accepted 4 January 2019

KEYWORDS
Manufacturing information systems; automated inspection; data mining; information technology; PCB assembly; quality; Industry 4.0; product failure detection

1. Introduction

The manufacturing industry is being disrupted in what is known as the fourth industrial revolution or Industry 4.0 (Hermann, Pentek, and Otto 2016). The main drivers and characteristics of this era are the time-to-market reduction (Calantone and Di Benedetto 2000), increase of complexity, mass customization (Piller, Lindgens, and Steiner 2012; Fogliatto, Da Silveira, and Borenstein 2012), and added value services (Mont 2002) around the products – altogether in a competitive globalized world (Bauer, Baur, and Camplone et al. 2015). To address this transformation, Industry 4.0 is introducing a set of new, advanced networking technologies, hardware, and more importantly, intelligent software. While in the third industrial revolution the manpower was replaced by simple 'hardwired' automation (Kagermann, Wahlster, and Helbig 2013), the current era creates cyber-physical systems where machines are communicating, collecting data, and making decisions in a collaborative way (Monostori 2014; Lee, Bagheri, and Kao 2015).

Expensive hardwired automation cannot adapt quickly enough to the market trends (e.g., in case of mass customization, where almost unlimited variations of a product can be produced (Calantone and Di Benedetto 2000; Cheeseman et al. 2005; Leitão and Restivo 2006; Nassehi, Newman, and Allen 2006; Pellicciari et al. 2009)). To succeed in this versatile industrial environment, the technologies must be highly scalable, adaptable, configurable, and in many cases self-managed and self-configured (Vyatkin et al. 2007; Fogliatto, Da Silveira, and Borenstein 2012).

This degree of intelligence is achieved using advanced algorithms, embedding artificial intelligence (AI) into production processes. Embedded AI is usually designed by data scientists and constructed out of the experiences obtained by the machines (Pham and Afify 2005). These techniques can be used in different domains of manufacturing, such as predictive maintenance or product defect detection in quality management. Although many efforts are invested in tackling these challenges so far, few work has been done in developing an integrated and manageable platform to solve these problems (Tsai and Yang 2005; Li, Al-Refaie, and Yang 2008; Hui and Pang 2009; Acciani, Brunetti, and Fornarelli 2006; Vachtsevanos et al. 1999). Most of the solutions propose a heterogeneous recipe of technologies to achieve the goals, and almost none of them (see Section 2) can be directly deployed or managed in a running system. Majority of the existing solutions do not offer options for data collection and management of AI technologies, such as online machine learning (ML). Additionally, the solutions provide very few integrated deployment tools for the reproduction of ML methods or models in other deployment environments.

Literature suggests the need of an integrated, extensible and extensive solution that provides runtime management tools, which is also self-managed and adaptive: a platform that provides a set of mechanisms for real-time data collection, processing, and analysis. In this manner, it is possible to create common methodologies to reproduce and redeploy ML and other AI technologies. This reduces the costs and increases the usability of ML technologies. Therefore, the authors present an extension of the solution by Soto et al. (2016a)– originally developed for smart city applications (Bonino et al. 2015, 2017; Soto et al. 2016b) – and adapt it to a manufacturing environment. The effectiveness of the solution is demonstrated by a case study in a realistic environment, focusing on

CONTACT D. Gyulai david.gyulai@sztaki.mta.hu

processes from the electronics industry, especially on *Surface-Mount Technology* (SMT).[1]

2. Literature review

The first two sections of this chapter provide a review of the state-of-the-art for product defect detection and production-related data processing techniques, respectively. After reviewing the most relevant literature, the final section (2.3) outlines the contributions of the presented work to the state-of-the-art.

2.1. Defect detection

Occurrence of defects increases costs and deteriorates manufacturing processes. Systems for defect detection exist in various industries. There is extensive research to analyze the effects (Hayek and Salameh 2001), optimize the planning (Chiu and Chang 2014; Chiu, Wang, and Chiu 2007; Kent, Kivlin, and Gasmann 2001), and reduce the costs (Taleizadeh, Wee, and Sadjadi 2010) of defect incidents. While some techniques apply solely to specific cases, others work on a wide range of domains. Physical defect detection solutions typically employ computer vision algorithms (Li, Wang, and Weikang 2002; Brosnan and Sun 2002; Gallarda et al. 2003; Vilella 2009; Viharos et al. 2016), artificial neural networks (ANN) (Kumar 2003; Chen and Liu 2000) and Gabor filters (Escofet, Navarro, and Pladellorens et al. 1998; Bodnarova, Bennamoun, and Latham 2002).

Defect detection approaches commonly use image processing techniques together with stochastic (Kim et al. 1999; Ko et al. 2000; Acciani, Brunetti, and Fornarelli 2006), analytical (Tsai and Yang 2005) or numeric (Hui and Pang 2009) algorithms, such as learning vector quantization (LVQ), multilayer perceptron (MLP), and Bayes classifier. One work achieves defect detection by applying sensor fusion (Vachtsevanos et al. 1999). In our review, we identified the work of Toth and Aach (2001) as the only visual-based system considering real-time settings. A non-vision solution with a more generic approach by Maier et al. (2011) learns automatically from production data to detect certain defects using probabilistic deterministic timed automata (PDTA). Albeit the approach is more generic and could adapt to changes, the work does not provide any information on how the model could adapt or evolve on runtime.

Previous works present systems suitable for detection of different kinds of defects on surface-mount devices. However, none of them offers a method that adapts to changes in product specifications, needed for mass-customization and dynamic manufacturing processes. These systems typically rely on a training set acquired at an offline phase, and assume that the training set applies to all new products. On the other hand, the system proposed by Maier et al. (2011) learns adaptively from production data and provides accurate defect detection. However, the system only detects failures that have an effect on the overall process of production. In the production of surface-mount devices, most defects occur when there are deviations in physical properties of devices (Kim et al. 1999; Ko et al. 2000; Acciani, Brunetti, and Fornarelli 2006; Tsai and Yang 2005; Hui and Pang 2009). Finally, the mentioned techniques do not address how the data from the production facilities are collected, aggregated, and processed in real-time, which would all impact the selection of data analytics technologies that need to be applied.

2.2. Production data processing

In order to process and manage the growing amount of real-time data, we need a set of technologies that aggregate and process the data on-demand (stream mining). Current defect detection systems rely on aggregated and pre-processed data produced by separate modules or parties. However, with the growing complexity and real-time requirements of production systems, development and management of scalable modules became an issue. Therefore, standard, well-known, or open source solutions are typically used to have a scalable, maintainable, comprehensive, and extensible system.

In the data processing for defect detection systems, current solutions can benefit from Complex Event Processing (CEP) (Cugola and Margara 2012). CEP systems can provide common configurable processing infrastructure that can be connected to other visual standard solutions such as BPMN (Buchmann and Koldehofe 2009; Baumgraß et al. 2014; Kunz et al. 2010; Backmann et al. 2013). A CEP engine is an event-based solution combining several data-stream sources, producing rich events (Luckham 2011). In recent years, CEP has received popularity in a range of disciplines for monitoring and reactive purposes (Buchmann and Koldehofe 2009), and has been used in the manufacturing for producing high enriched events (Jin et al. 2008; Hameed, Durr, and Rothermel 2011) or multi-protocol interoperability (Izaguirre et al. 2011).

Formalization and taxonomy of datastream systems can be found in the detailed study of Cugola and Margara (2012), however, without covering any ML interaction. The ML techniques applied to data streams such as active learning or stream mining have been extensively compiled in (Gama 2010) and (Aggarwal 2007); however, both works rely on the algorithmic and formal aspects of it. Moreover, they present few studies on how the models can be evaluated continuously. While Gama (2010) presents two formal models, Aggarwal (2007) leaves it as an open issue for further research.

This paper proposes a framework for management of the learning process in cyber-physical environments that answers some of the questions left open by Gaber, Krishnaswamy, and Zaslavsky (2005). Furthermore, it provides an implementation for continuous evaluation on datastreams which addresses open issues pointed by Aggarwal (2007) and by Gaber, Zaslavsky, and Krishnaswamy (2005).

2.3. Research gap and contribution

Based on the literature provided in Sections 2.1 and 2.2, the authors believe that there is a significant research gap in having a real-time, scalable, reconfigurable, and manageable framework for application fields such as the SMT. The novelty of the presented solution is twofold: on the one hand, a novel discrete-event simulation (DES) based approach is proposed, which enables evaluation of various failure identification techniques, without disturbing the physical production process. The approach is applied for simulating complex measurements and analyzing the performance of different machine

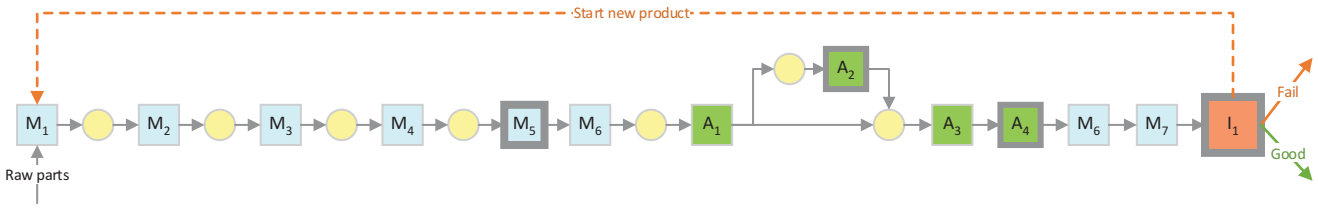

Figure 1. Scheme of the process chain, composed of machines M_n, buffers (circles), manual assembly processes A_n and the final functional test I_1. The process steps with parallel resources are highlighted by wide borders.

learning models and algorithms. On the other hand, the main contribution and scientific novelty of the proposed solution are provided by the online, self-learning, reconfigurable, failure identification method. The data feeding, incremental model training, and real-time prediction implement a closed-loop quality check that is purely based on technological and process log data, without applying the outcomes of the real functional testing. Although algorithms applied for the defect detection already exist, none of them are used for virtualising and substituting the real functional test during the production process, but they are mostly applied to evaluate the quality of individual process steps (e.g., an automated optical inspection checking the position of the components on printed circuit boards (PCB)). In contrast, the presented method provides a real-time, self-adjusting process that can be adapted seamlessly to changing production requirements. Naturally, the data-driven failure identification cannot completely replace traditional functional tests; however, it can reduce the efforts required for handling of failed products and decrease the waste by preventing the addition of components to failed products. Overall, a reconfigurable, self-managed, scalable learning system is presented that can be applied and replicated in manufacturing processes and compliant with the challenges of the Industry 4.0 era.

3. Problem statement and characteristics of the production environment

In the paper, a manufacturing system from the electronics industry is under study, in which electronic components are assembled on a generic flow assembly line. The line consists of a set of automated machines, performing the technological steps of surface-mount technology SMT. From a production management viewpoint, the resources are flexible, and the PCBs are produced in batches, applying setups to switch from one product type to another. Although any flow line that matches the above criteria could be the subject of the research, the efficiency of the proposed data-driven, functional failure identification method is demonstrated by a case study with a simulation model.

The production line under study produces mechatronic automotive products, consisting of mechanical parts, and an electronic core component, which is the controller unit of the product. The process chain consists of multiple stages, concentrated around two main processes: (i) The first, fully automated stage of the line is responsible for the production of the control unit. Afterwards, (ii) this unit is placed in a house, together with other mechanical components. The material flow follows a linear scheme, with some parallel processes to balance the line, however, all the final products leave the line at the same point. The general scheme of the process chain is illustrated by Figure 1, where processes are denoted by squares, and in-process buffers with circles. More product types are assembled on the line in batches; therefore, it is categorized as a flexible assembly line, and setups takes place when the line switches between different product variations. The line is balanced, this means that processes that take longer are performed on parallel stations.

In the first stage, the electronic components are mounted on a PCB, applying fully automated SMT machines. The PCBs enter the process at station M_1, which is a stencil printer that covers the board with solder paste. Then, optical paste inspection takes place at M_2, to check the positions of paste-covered spots. At stage M_3, an automated pick-and-place machine mounts the electronic components on the pasted areas of the board, while logging the mounting positions of the vacuum nozzle, carrying and placing the parts. Then, optical inspection is done again to check the positions of the mounted parts before re-flowing at M_4. The re-flow oven melts the paste applying specific heating zones and steps. The oven is considered as a parallel process, capable of accepting multiple products simultaneously. Leaving the oven after the re-flowing process, the solder paste becomes solid, implementing the solder joints between the board and the mounted components. As the re-flowing process might alter the components' position, an optical inspection is applied again at M_6, as the last stage of the PCB assembly.

The second stage of the process is a semi-automated, hybrid assembly (Lotter and Wiendahl 2009) segment called *backend*, including manual and automated workstations. Processes performed at these stations mostly involve the assembly of mechanical components, such as actuators, connectors, springs, etc. As the mounted components are more complex and expensive than the simple, surface mounted ones, the critical and value added activities are mostly performed at the backend segment. The first four processes A_1-A_4 are performed by human operators, and due to the longer processing times, A_2 and A_4 stations are parallelized to maintain the line balance. Depending on their type, some products skip the A_2 process, as they do not include the components assembled there. At the manually operated workstations, process parameters (e.g. screwing torques are measured and logged). After these steps, two automated tasks are performed: first, the plastic house of the product is enclosed (M_6), then the controller software is uploaded on the memory (M_7). The last step of the process chain is a complex functional test at I_1, simulating real and extreme operation conditions. This step is done in parallel on multiple parts, the products that fail the test are removed from

the line, and good parts are transferred for packing. The production is done according to pull strategy based on the customer orders, defining the production plan. Only complete batches are delivered; therefore, new products are started if a product fails the functional test, in order to complete the batch. Setup of a new product type can take place only if the batch is completed with the requested amount of functionally correct products.

Each process step is characterized by technological parameters, provided by the machine settings. The actual, sensor-measured values of the parameters can be accessed in real-time during operation. Besides the parameters of the technological resources, in-line quality inspection is also applied, and the measurements are performed by inspection devices (e.g. automatic optical inspection or AOI). These devices are measuring technical parameters of the products, that do not explicitly indicate their functionality quality; the functional test is performed only at the end of the production process. Each product can be characterized by a tuple of features F, including the numerical values of measured technological parameters and the results of the quality inspection measurements. The objective is to predict the outcome $q \in \{0, 1\}$ of the quality inspection (0 = good, 1 = fail) in the earliest possible stage of the production process, by the partial knowledge of F. In this case, the term *partial* refers to the fact that the feature values are added subsequently to F when performing the process steps one after another. The *'earliest possible'* failure identification means that the functional test needs to predict failures by using the minimal number of parameters from F.

4. Proposed solution

To build real-time, online failure identification solution that could be deployed in a real scenario, it is necessary to approach the problem in a holistic manner. Therefore, it is paramount to not only solve individually the problems of interfacing, collecting, analyzing, processing the data, and the actuation in the production environment, but proposing a scalable system that performs all that together and at the same time can adapt in real-time. The solution consists of five parts (see Figure 2) built following a Microservices architecture (Newman 2015) and the LinkSmart® Specification (LinkSmart 2017). Firstly, a test bed simulation model **Plant Simulation** is provided, implemented using commercial software (Siemens 2016) and simulating the scenario stated in Section 3. Secondly, the functionality of the production line is exposed using a **Connector**, following (LinkSmart 2017). Thirdly, the **Connector** propagates the data in a bidirectional manner using the **Mosquitto MQTT Broker**[2] (Light 2017) according to the IoT standards (Banks and Gupta 2014; Liang, Huang, and Khalafbeigi 2016). Fourthly, the live-data is processed and orchestrated using the **LinkSmart® Learning Agent** (LinkSmart 2016). Finally, a backend machine learning model is developed, called **Learning Toolkit** specifically for the use case in this work.

4.1. Simulation model

In most cases, commissioning of a software solution in a highly utilized manufacturing environment is not straightforward, due to the possibly idle times the installation and testing processes would result. Additionally, testing the long-time

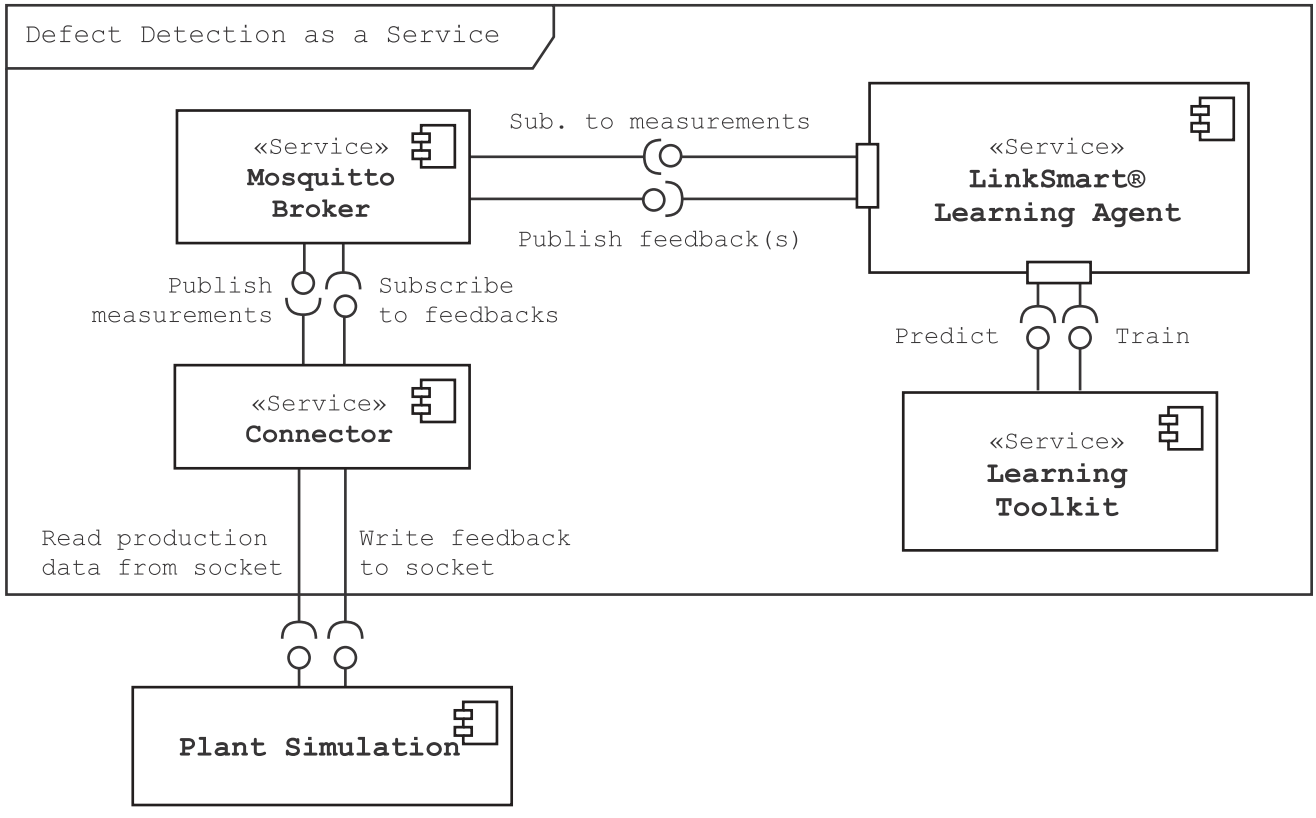

Figure 2. Components of the defect detection system, communicating with one another via provided or expected interfaces.

effect of introducing such a system might take months or years. In order to reduce this negative effect on the production and to test the long-time behaviour, a simulation-based solution which is able to imitate the behaviour of the physical line – regarding especially the measurement processes – is proposed. The DES model is part of the **Plant Simulation** component, and generates measurement data following the characteristics of the real processes. Regarding the measurements, the main strength of DES is its capability of managing efficiently stochastic parameters and random events even in case of complex models. In reality, measurement values are typically characterized by a distribution function that is specific to the technological process in question. In the DES, measurements are simulated by generating random numbers applying the distribution functions of each process. The functions are given by the type of distribution (normal distribution in most of the cases), and its parameters, which are typically the expected values and the distribution. Running the simulation, the model generates random numbers representing the measurements, and the values are streamed via the **Connector** typically using TCP-sockets.

4.2. Connector

The DES model exposes the data in simple bidirectional sockets, and then the **Connector** collects, organizes, and homogenizes the data and its structure. The **Connector** accesses the data and transforms it into data streams using a message handler or broker where the learning systems will obtain it. In this manner, the **Connector** has two roles. Firstly, it is an abstraction and integration layer between the production system and the learning system. The **Connector** is responsible for transforming the data form production site protocols such as OPC[3] (OPC Foundation 2017; Mahnke, Leitner, and Damm 2009), S7 (Siemens 2018) into live, scalable network protocols or systems such as MQTT (Banks and Gupta 2014), Kafka (Apache Foundation 2018) using IoT standards such as OGC Sensor Things (Liang, Huang, and Khalafbeigi 2016). In this implementation, we used MQTT and OGC Sensor Things as communication protocol and data structure, being both well-known standards in the IoT community. The use of standards makes our solution more interoperable between different system implementations, and therefore more open. Secondly, the use of message queuing instead of solutions such as REST[4] (Richardson and Ruby 2008) allows the system to scale horizontally (Michael et al. 2007) (see Section 4.6.1) (e.g. adding more connectors when the amount of devices cannot be handled by a single **Connector)**.

4.3. LinkSmart® Learning Agent

The **LinkSmart® Learning Agent (LA)** is the core component of the system and first introduced by Soto et al. (2016a). This service annotates, aggregates, routes, processes the data, and orchestrates all other components' behaviour. The behaviour is defined in the Learning Request, which is a reconfigurable CEML[5] process (Soto et al. 2016a). It describes: (1) how the data should be processed, (2) how the data should be delivered to internal models or external backend models (i.e. in our case, the **Learning Toolkit**), (3) how the model is to be evaluated, and (4) how the model is used (i.e. in our case, how the data goes back to the production line). The Learning Request is submitted to the **LA** via RESTful[6] and REST-like APIs exposed over HTTP and MQTT, respectively.

Concretely, the Learning Request consists of several parts: a) Data Descriptors which are the description of the feature space and the labels. b) Model, which is the description of the selected algorithm, its configuration, parameters, and evaluation methods; or the serialization of an existing trained model. c) The definition of the selected evaluation method itself consists of Metrics[7] and the associated Target or threshold, and the Evaluator. The Evaluator is selected according to the Model types (e.g. clustering, classification, regression) and is responsible for managing the evaluation process. d) Learning Statement, which is the definition of how the live-data is processed to obtain a datapoint, according to feature space described in Data Descriptors. The processing is done through CEP queries and produces the Learning Streams. e) The description of how to use the trained and validated Model is provided in the Deployment Statements, producing the Deployment Streams. The continuous validation is done when the Evaluator detects that the Model reaches the Target thresholds of Metrics using the Learning Streams, and then the **LA** deploys the Deployment Statements alongside the Model.

Finally, the selection of the backend learning algorithm depends completely on the use case and the characteristics of the data. Since **LA** is an extensible service, we extended the backend models by developing a Python module called **Learning Toolkit**. This component allows the usage of various Python-based machine learning libraries such as Numpy (Stéfan van der, Colbert, and Varoquaux 2011), Scipy (Jones, Oliphant, and Peterson et al. 2001), Scikit (Pedregosa et al. 2011), Tensorflow (Abadi et al. 2016), and Theano (Theano Development Team 2016), developed by the open source community and industry leaders.

4.4. Learning toolkit: data-driven failure detection

In this section, we briefly discuss three machine learning algorithms selected as candidates for the solution. We evaluate the performance and viability of these algorithms based on offline experiments, applying a dataset generated using the aforementioned simulation technique. The dataset consists of 30,768 datapoints with a defect ratio of 5%, previously used by Tavakolizadeh et al. (2017) to evaluate the application of random forests in offline SMT defect detection. We use these findings in the following sections to select the appropriate algorithm and evaluate the proposed system in Chapter 5.

4.4.1. Neural networks

Within the case study, experiments with Multi-Layer Perceptrons (MLP) artificial neural networks (ANN) trained with the Stochastic Gradient Decent algorithm (Bottou 1991) were conducted. Prior to training, dimensionality reduction was performed to map measurements from parallel, identical stations into single features. Furthermore, the data was standardized so that the values for each feature form a Gaussian distribution with zero mean and unit variance (Sola and Sevilla 1997). Since the production data

was imbalanced consisting only 5% defects, defect products' data was oversampled with random selection. Several network topologies were evaluated and the outperforming one was selected consisting five layers with *X*, 50, 100, 100, and 2 nodes, where *X* is the number of input features. Using state-of-the-art methodologies (i.e. cross-validation) and technologies (Pedregosa et al. 2011; Abadi et al. 2016), and with a dataset of 30,768 datapoints (i.e. from 30,768 products). The classifier achieved MCC[8] score of 0.86 on the training set. However, due to a large number of false negatives, the score was only 0.12 on the test set. The poor performance of MLP could be pinpointed to the following reasons: (i) MLP performs better when the trainset contains a large number of samples for each class (Ciresan, Meier, and Schmidhuber 2012). It is, however, infeasible to collect large amounts of data about defect products in a reasonable time in most real production systems. (ii) Since the number of defects is rather small, even with the oversampling the classifier could not gain insights into all possible defect patterns. (iii) There is low or no correlation between feature values extracted from an SMT device with numerous independent components, making it difficult for the model to learn insights.

4.4.2. Random forests

The same dataset was used, and similar data processing techniques were performed in the experiments using random forests (Breiman 2001). However, the oversampling of the minority class was not necessary since random forests algorithm forces trees to learn boundaries for both classes regardless of their size. Parameter fine-tuning was applied, and the optimal settings were found that provided high prediction accuracy with short training time. The result was an ensemble with 100 trees and max depth of 90 for each tree. In addition, the maximum number of features was set to be used for the training of each tree to the square root of total features. This configuration was used to train and test the ensemble with the implementation of Random Forests by Scikit (Pedregosa et al. 2011). The training was done in 24 s. The result was an outstanding MCC score of 0.98 and 0.96 on training and testing sets, respectively.

4.4.3. Gradient boosting

The same dataset used during the training/testing of Random Forests was used to train the Gradient Boosting ensemble. Similarly, the dimension of datapoints was reduced by assigning identical measurements into single features. These measurements were generated using the same sensors and reporting the same kind of information but at different parallel stations. Oversampling of the minority class was not necessary since the algorithm forces all weak classifiers to learn all existing classes. Several tests were performed to find the best configuration, allowing quick training of the ensemble and ideal performance. The depth of tree has a direct effect on the time needed for training, however, very shallow trees could not learn the patterns properly. An optimal setting was with 100 trees each with a maximum depth of 15. The trees were trained with a learning rate of 0.1 each taking a limited number of features. Similar to Random Forests, the maximum features was decided to be equal with a square root of total features. The ensemble was trained in 43 s reaching MCC of 0.99 on the training set and 0.98 on the testing set.

4.5. Real-time machine learning and defect prediction workflow

In this section, the technologies mentioned above are combined to create a running, reconfigurable online-learning system.

4.5.1. Data collection and feature extraction

Each measurement generated by the production line is sent to the **MQTT Broker** by the **Connector** and captured by the **Learning Agent**. The **Learning Agent** aggregates and labels the data by collecting the measurements of each instance of a product passing through the line and producing the Learning Instance. This is done by using Learning Statements and defined in the Learning Request.

4.5.2. Periodical retraining

Specifications of products change over the lifetime of a classifier (e.g. when using electronic components from different suppliers or when a new version of the same component is manufactured with different prints or dimensions). Alternatively, the product specification may be changed explicitly because of variation in production demands (e.g. slight modifications of PCB layout, adding a new connector). In these situations, the performance of the classifier will degrade or drift. Thus, it is important to retrain the classifier periodically to match the attributes of the most recent products.

To avoid this, **LA** allows for configuring online iterative learning or intervals between every model retrain for batch trained models in case of Random Forest and Gradient Boosting. This configuration is in the Learning Request. The retraining frequency value was set to 1000, which means that a retrain is performed after every 1000 products. In the simulation environment, products are synthetic objects that change less frequently than in real-world. If the products change more rapidly, a smaller interval is desired for better performance of the classifier. However, a very small interval could result in a racing condition between multiple re-training operations. At every training request, feature vectors were extracted and they were added to the training buffer. Then, sub-sampling was performed and the data was submitted to train a new classifier from scratch. Once the new classifier is ready, it replaces the previous version such that it will be used for all new predictions.

4.5.3. Prediction

During the training, only completed products' measurements were used. This is necessary because measurements and the result of the functional test are all required to train the classifier. During prediction, the system works on collected measurements as products go through the line and before they reach the functional test. Three Deployment Statements are configured in the **Learning Agent** to collect measurements and trigger events when products leave certain inspection stations. These stations perform paste and component inspection tasks followed by other stages with high operational costs. All phases of the training and prediction happen

concurrently and the system swaps learning models on runtime without affecting the prediction process.

4.6. Industry 4.0: scalability, reconfigurability, adaptability, re-deployability, interoperability and security

In this section, the proposed system's flexibility in a highly changing production environment is shortly discussed.

4.6.1. Scalability

The system follows the microservices architecture to enable scalability. There are four reasons why the system may need to scale. (1) The number of data sources increases (e.g. by adding new sensors, stations, or production lines). (2) The data rate increases (e.g. by introducing sensors operating on a higher frequency). (3) The data processings are computationally more costly (e.g. adding new algorithms or techniques into the preprocessing). (4) The learning needs more computational power (e.g. by increasing the learning set size batches).

In the case of (1), the system can scale horizontally by adding more instances of the **Connector**. Adding more connectors will increase the overall data rates, which is the case (2). In case (2), if the data rate increases due to the sensors, then the problem can be handled similarly to the problem (1). However, if the overall data rate increases over the limit of the **Broker** or the **Learning Agent**, then more instances of them can be added to scale horizontally. In the case of the broker, a cluster of brokers (balancing the data load, e.g. per topic) can be put in place to scale. If the data arriving into the **Learning Agent** reaches a critical level, additional parallel instances of the **Learning Agent** can be added to cover the incoming load. The data can be merged in further steps of the processing pipeline, see case (3). In case (3), if data processing or data processing pipeline (DPP) grows over the possibility of a single instance; then the DPP can be parallelized or/and distributed in several instances of the **Learning Agents**. In this manner, each **Learning Agent** – where parts of the DPP are processed – can be seen as processing nodes (PN). Parallelization is done by taking some independent simultaneous processes of the DPP into different parallel NPs. Distribution is done by taking parts of the DPP that happen sequentially into two or more sequential PNs. By parallelizing and distributing, the system is able to scale horizontally (see Figure 3). Case (4) is the most sensitive in terms of upscaling. While additions of more products scale horizontally by adding new backend models per product, a single backend will hardly scale horizontally. Therefore, it is paramount to take into account the computational times and power needed by the backend models (see Section 5.2). Because of this, the models are inclined that they either can be trained incrementally or with relatively small batches, for online machine learning.

4.6.2. Reconfigurability

Changes can come from many areas – for example, a new type of sensor, new ways in which the products are made, improvements in production technology and in the processing techniques. As described in Section 4.3, the **Learning Agent** orchestrates the overall system using Learning Requests. These requests can be consulted, monitored, changed, or removed online at any time. This is important as it allows the system to change while it is running, without being disturbed.

4.6.3. Adaptability

Not all changes are made, controlled, or foreseen by humans. Some systems may change in time for unknown reasons,

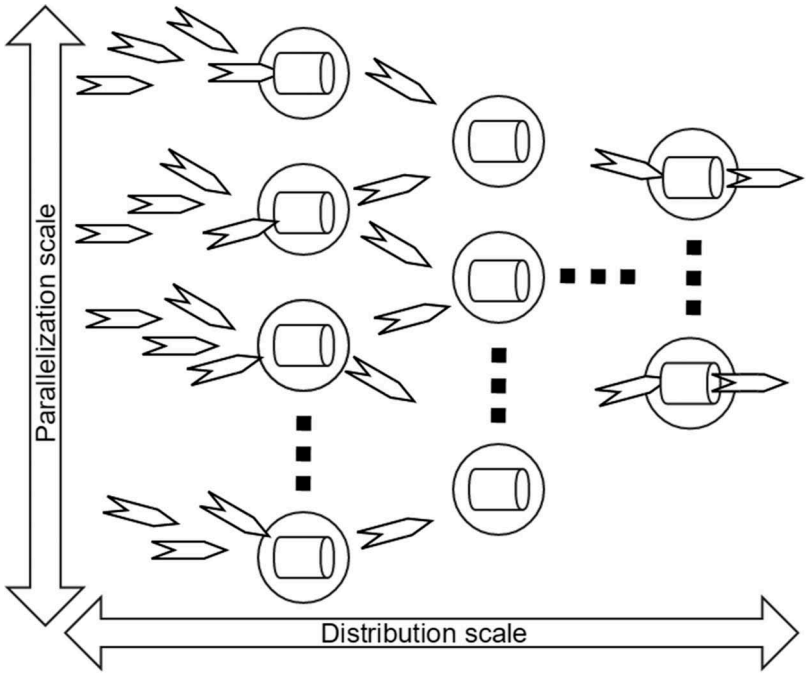

Figure 3. The figure shows how the agents or processing nodes can scale up to match the processing needs. The circles represent Processing Nodes and the cylinders are parts of the data processing pipeline.

leading to drifts in an offline learning process (Zliobaite et al. 2014). Additionally, when the changes are happening too fast; by the time the data is collected and the model has been trained, the process may already be changed or be on its way to change. Therefore, adaptability is of crucial importance. By adaptability, we mean the capability of the system to adapt by itself to changes. The adaptability of the system is addressed with the learning orchestration feature of the **Learning Agent** which allows continuous online learning of the model. With the online continuous learning, the model will adapt to the possible different types of errors that the production line could produce. In case the new type of defects cannot be detected with the current methods, several concurrent online methods can be deployed. The **Learning Agent** will select the model with the best performance for every case. In any case, the **Learning Agent** will be aware if it is adapting or not. If the **Learning Agent** is not adapting, the process can be configured.

4.6.4. Re-deployability

In production, the parallelization of resources is commonly applied to cover demand and fulfil other requirements such as mass customization (Piller, Lindgens, and Steiner 2012). In such a case, the system must be reproducible. The presented solution not only allows redeployment alongside the needed computational power but also that the learned models can be redeployed elsewhere with little efforts. This can be done by exploiting the reconfigurability of the system, allowing the possibility to redeploy a learned model at runtime, enabling running production systems to exploit the knowledge gained from each others' trained models. From an infrastructural perspective, the system relies on Docker (Docker, Inc 2018) technology; making it platform independent and easy to redeploy (e.g. in case of system updates).

4.6.5. Interoperability

Although interoperability is not the main topic of this work, any work that addresses Industry 4.0 concept in a realistic deployment must address interoperability to some extent. The data and protocol interoperability is addressed in two-folds: (1) Adding an abstraction and integration layer, and (2) using standards. There is no one-size-fits-all solution for interoperability, mostly due to concurrent coexistence of new and legacy systems.

Therefore, the presented solution applies a Microservices interoperability layer (Newman 2015), in which each third party system added into the platform that will be interconnected using **Connectors**. A **Connector** is typically an abstraction layer, translating the input and output of another component into selected IoT technologies (MQTT (Banks and Gupta 2014), RESTful (Richardson and Ruby 2008), OGC Sensor Things (Liang, Huang, and Khalafbeigi 2016), JSON (JSON 2013)); this works in a bidirectional manner.

Regarding the infrastructure interoperability, the presented solution is developed in Python and Java, and deployed and redistributed using Docker technology (Docker, Inc 2018). All these are multi-platform technologies, allowing the system to operate virtually anywhere.

4.6.6. Security

A framework that does not provide any security features cannot be applied in recent manufacturing environments. Therefore, although security is not the main aspect of the framework; it provides features to secure the system. The framework provides means to achieve identification, confidentiality, integrity, and non-repudiation by using standards, main security standards used are Transport Layer Security (TLS) (Dierks and Rescorla 2008) and JSON Web Signature (JWS) (Jones, Bradley, and Sakimura 2015). To achieve confidentiality, all services in the solution used standard communication protocols that support TLS such as HTTP, MQTT, and OPC UA. Using TLS for encrypted communication utilizing a Public Key Infrastructure (PKI) ensures a confidential communication. On the other hand, to achieve identification, a combination of TLS and JWS can be used for all direct communication and broker-based communications. Analogously, integrity and non-repudiation can be achieved using JWS. The correct use of the protocol not only allows identification of the source, but also enables verification of the integrity as well as non-repudiation. The selected standards require a PKI and private/public keys per service. The approach for PKI, private keys, and additional security features are part of future work. Finally, additional security features such as access control (e.g. authentication, authorization) are not managed by the framework, but can be achieved using open source services such as LinkSmart® Border Gateway and LinkSmart® Service Catalog.

5. Numerical results

This section presents the numerical results obtained from the case study, introduced in Section 3. First, the accuracy of the applied machine learning models is provided, comparing random forests (RF) and gradient boosting (GB) methods. Thereafter, the production-related effects of the methods' application are provided, observing the overall equipment effectiveness and its elements as selected main metrics.

5.1. Accuracy of the machine learning models

Both RF and GB classifiers showed outstanding results (see Figure 4) during offline evaluation experiments. This chapter discusses how these classifiers were optimized to suit real-time environments' needs and to offer ideal prediction accuracy.

Earlier, the results of RF and GB classifiers were presented on a dataset with more than 30,000 datapoints, which took a training time up to 43 s. With a sub-sampling technique, we reduce the training time to be less than 5 s while keeping the prediction performance. This sub-sampling technique was developed to achieve two things: first, to create a small subset of data that represents both classes and second, to have control over which part of data matters most.

For every product type, datapoints were collected in a buffer with a maximum capacity of 20,000 entries. When the buffer is full, datapoints are discarded in a first-in-first-out (FIFO) fashion. From this buffer, two independent sub-samples are selected: one for non-defects and another for defects. For each class,

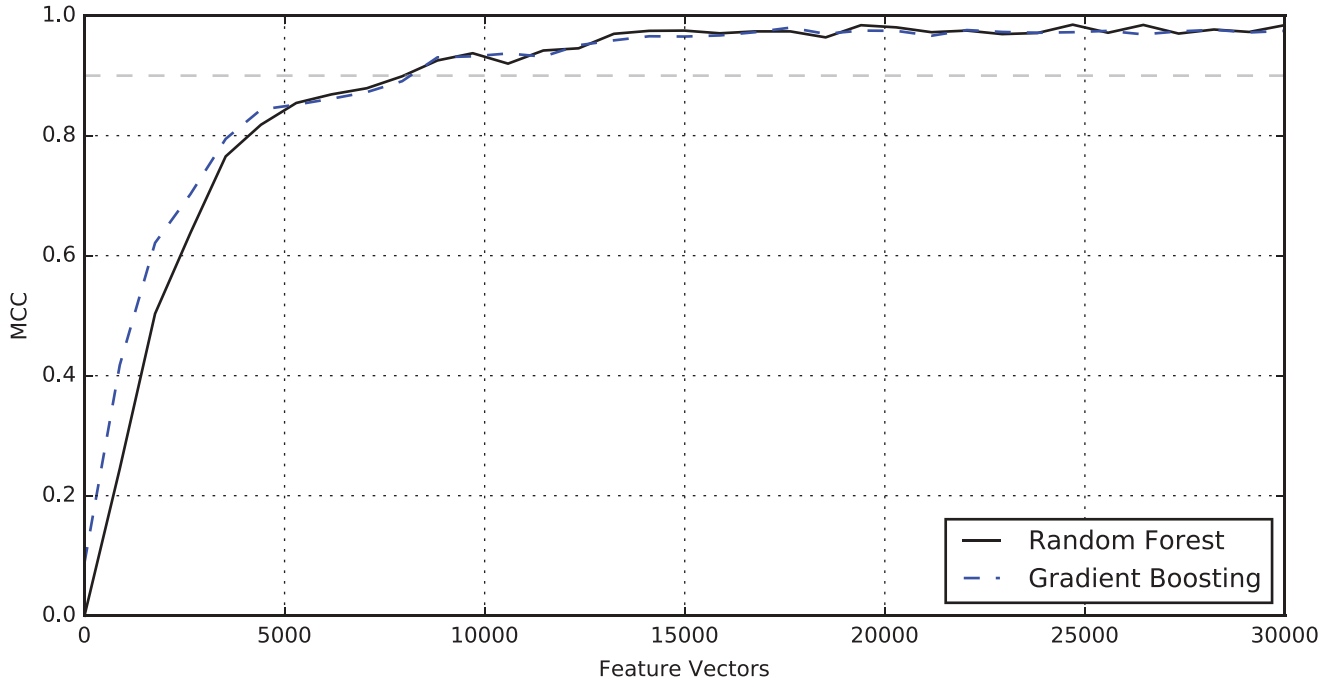

Figure 4. Random Forests (solid line) and Gradient Boosting (dashed line) classifiers. In X-axis the number of data points and Y-axis the Matthew's Correlation Coefficient.

datapoints are randomly selected while giving higher chances to the most recent entries. This is achieved by drawing samples with a logistic distribution applying the following probability density function (Scipy 2017):

$$pdf(x) = \frac{e^{-x}}{(1+e^{-x})^2} \tag{1}$$

This defines the density function in standardized form, shifted and scaled by:

$$\frac{pdf(y)}{s} \; and \; y = \frac{x-\mu}{s}, \tag{2}$$

resulting in:

$$pdf(x;\mu,s) = \frac{e^{-\frac{x-\mu}{s}}}{s\left(1+e^{-\frac{x-\mu}{s}}\right)^2} \tag{3}$$

where s and μ are the scaling and shifting factors, respectively.

The function is shifted and scaled by $s = \frac{samplesize}{5}$, and kept in the original location $\mu = 0$. The *samplesize* is set to 5000 and 1000 for non-defect and defects, respectively.

In the running system, at the beginning of a production session sub-sampling will take place because there are fewer datapoints than the expected sample size. When the size becomes larger, we will sub-sample to get 5000 and 1000 datapoints from the larger pool. Figure 5 shows the density function used for random sampling 5000 datapoints. Finally, the two sample sets are merged and permuted before being sent to the learning algorithm. Permutation (shuffling) uniformly distributes instances of both classes and is necessary to prevent overfitting of classifiers.

To evaluate the effectiveness of the sub-sampling, the dataset from Section 4.4 was used, consisting of 30,768 (28,983 non-defects, 1785 defects) entries. Subsampling resulted in 6000 (5000 + 1000) entries that were applied for training RF and GB classifiers. The classifiers were trained in 4.29 s and 5.66 s with MCC scores of 0.99 and 0.72 on the same dataset for RF and GB, respectively. The classifiers against the independent, unseen test set achieved MCC score of 0.92 and 0.51, for RF and GB, respectively. For the RF, even though the accuracy is slightly lower, the training time is about six times shorter than when trained on the whole set. While the GB performed about 8 faster than before, the MCC score dropped. The performance of GB could be improved by increasing the maximum depth of trees from 15 to 25. This resulted in a MCC score of 0.99 and 0.93 for the train and unseen test sets with an increased cost of training time by 2.4 s. In the end, RF was selected, as the classifier could be trained in a shorter time, offering a higher MCC score.

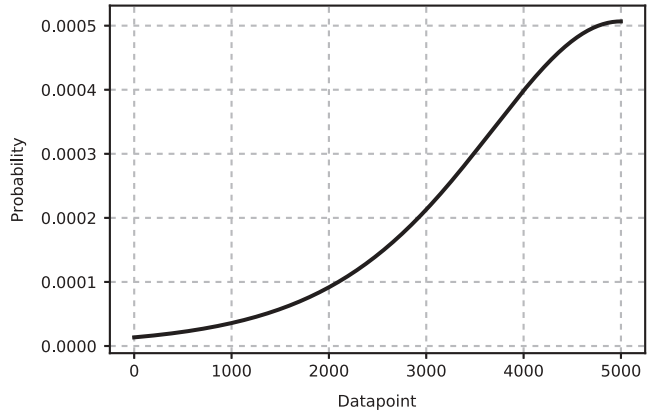

Figure 5. Logistic probability density function used for subsampling of 5000 datapoints from a large pool with $s = \frac{samplesize}{5}$ and $\mu = 0$.

5.2. Production-related effects of early fault detection

In order to evaluate the efficiency of the proposed failure identification method, not only data analytics measures are used that qualify mainly the applied learning model, but also production related metrics that represent the effect of the method on the whole production system. Most of the companies apply – among other metrics – the overall equipment effectiveness (OEE) as a main performance indicator, which is a simple numeric value qualifying more aspects of the processes, including the time, performance and quality dimensions. OEE is applied to measure the degree of the effectiveness of the system to its theoretical best performance; therefore, it is capable of representing the number of losses the operation suffers. The metric itself is a generic one calculated by the multiplication of its three main factors, availability (A), performance (P) and quality (Q), however, many of the companies derive new, customized indicators that fit their processes and needs (De Ron and Rooda 2006). In the presented case a slightly modified form of general OEE (Huang et al. 2003; Muchiri and Pintelon 2008) was applied to measure the impact of the data analytics based failure identification method on the system's overall performance.

This modification is justified by the fact the OEE is applied to characterize the operation of a system through measuring the performance of the bottleneck. However, on the one hand, the main question here is the system's overall performance, on the other hand, the quality related metric cannot be defined for the individual sub-processes, due to the fact the functional quality in question is only measured at the end of the process. Therefore, the common weighted form of the OEE is applied with weight factors to calculate A and P on a sub-process (machine) basis, and Q is calculated on a process (line) basis with (4).

$$OEE = \left(\sum_{m\in M} \frac{t_m^{NO}}{\sum_{m\in M} t_m^{NA}} \cdot \frac{t_m^{NO}}{t_m^{NA}}\right) \cdot \left(\sum_{m\in M} \frac{t_m^{NO}}{\sum_{m\in M} t_m^{NO}} \frac{o_m \cdot t_m^{C}}{t_m^{NA}}\right) \cdot \left(\frac{g-(f+n^{+})}{g}\right) \tag{4}$$

The first element of the product defines the weighted availability of machines' $m \in M$, calculated with their *net operating time* t_m^{NO} and *net available time* t_m^{NA}. The latter is the scheduled working time of the machines (including shifts), while the operating time is the net time reduced with the unplanned downtimes. The (weighted) performance element is defined as the ratio of *ideal operating time* and t_m^{NA}, by calculating the operating time as the product of the cycle times t_m^{C} and the output of the machines o_m. The last element of the OEE is the quality factor, that is calculated on a process basis as the ratio of good parts g, and the fail ones that are the sum of functional fails f and the ones that are identified by the learning engine, and classified as true negative parts n^{+} (correctly identified as fail products, already in the early stage).

As for numerical results related to the assembly line under study, simulation experiments were made by analyzing the OEE as well as its elements. The experiments were run with two settings: by switching on/off the failure identification service. Prior to the experiments, it was expected that applying the failure identification method will result in increased performance, availability and quality-related metrics, thus increased OEE as well. The horizon of the simulation analysis was 54 days, and each product type was assembled. Regarding the availability A and performance P metrics, Figure 6 visualizes the difference between applying and not applying the failure identification service, for each resource of the line. As visible, a significant increase in both metrics could be achieved by the failure identification service on $M4$, while the processes at the end of the line performed slightly better without the failure identification service. This is resulted by the fact that the failed products could be identified the earliest at $M4$, and for each fail product, a new product was started from the beginning of the process. Observing the quality element of the OEE – which is the most important from the viewpoint of the research – when applying failure identifications, experiments resulted in 2064 fail product in total (that were identified by the functional test). In contrast, when switching the service off, experiments provided 2974 functionally fail products. This is a significant improvement in regarding the quality metric, as the volume of fail products could be reduced by 31% in total, achieved only by applying the failure identification method as a service. Besides the decrease of the fail products, considering the previous metrics of performance and availability, one can conclude the following: although a slight increase was realized in the case without the failure identification service, the machines were utilized by performing processes on

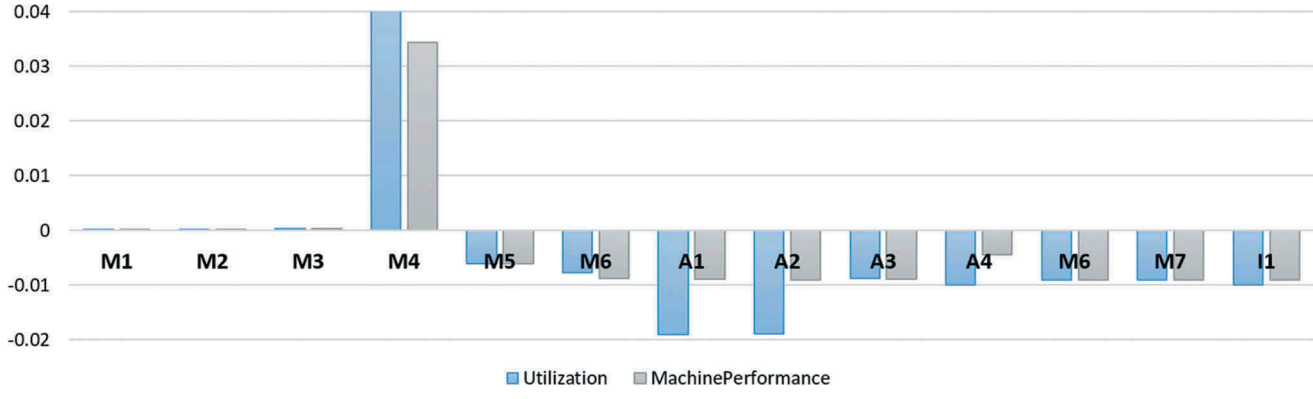

Figure 6. Experimental results: difference (applying-not applying) of machines' performance and availability metrics applying or not the failure identification service.

fail products, thus producing scrap, which is also reflected by the value of the OEE. Compared to the total volumes and weighted metrics, applying the formula (4), the overall increase in the OEE is 1.5%. This might seem a minor difference, however, on a relatively long period of 54 days, it is a significant increase, as no process improvement was needed to achieve this, but only available production data was utilized in a smart way, applying a failure identification service. In this regard, this increase is a considerably good result within the case study.

6. Conclusions and outlook

As highlighted in the paper, production environments are transforming in the era of Industry 4.0, and this change affects the opportunities and requirements of production-related data analytics solutions. As a response to the challenges and requirements, an intelligent system for failure identification was proposed that can scale and adapt to the ever-changing production conditions. This needs not only a set of advanced technologies, algorithms, and infrastructure, but also a solution that is scalable, reconfigurable, adaptable and re-deployable without hindering the production processes. A flexible learning solution was presented that addresses important needs of the production processes. The proposed system was evaluated by using the realistic simulation model of a Surface-Mount Technology-based production line. The solution's effectiveness was demonstrated through a case study, in which functionally fail products were identified during production, still before performing the actual functional test. Additionally, the authors developed a method to evaluate the result in a realistic long-term environment in a holistic manner. In this case, the term *holistic* means that the system was evaluated by applying production management and computer science related metrics in realistic production scenario, analyzed in real time. Accordingly, Section 5) describes the characteristics of production and the change of KPIs, obtained with and without the solution. Besides, it was highlighted that not only the system is scalable, reconfigurable, adaptable, and re-deployable but also offers the great performance in the computational and accuracy production metrics. Overall, it was shown how AI technologies can be applied in a real manufacturing environment for Industry 4.0.

Neither the presented framework and the system cover completely all requirements of the Industry 4.0, nor this paper does describe exhaustively and completely all aspects of the framework. We believe the following issues were not covered and expect to investigate them in detail within the future research:

6.1. Runtime impact evaluation re-configurability and adaptability

While the system provides means to change the process and adapts in case the system logic or the phenomenon change, the underlying model adaptation does not take effect immediately. The gradual model adaptation which highly depends on the learning algorithms may affect the accuracy of the model as well as the system behaviour as a whole. The implications of the runtime adaptation and re-configuration must be further studied.

6.2. Scalability Case Study

The system was designed to scale according to the application logic. However, the scalability had not been measured or evaluated in the case study to validate this statement empirically.

6.3. Real Case Study

This work was tested using a realistic simulation. However, in real scenarios, unexpected issues may arise which are not covered by this solution. A real case scenario might strengthen the validation of the method.

6.4. Generalization

The solution presented in this works presents a roadmap to solve the given problem. However, it is necessary to apply this system in several similar problems and carefully evaluate it to achieve a generalization that works for all.

6.5. Semantic Interoperability

The current solution addresses the interoperability issue in two manners. Firstly, by complying to already established and existing standards. Secondly, by integrating third-party protocols using a plugin-based abstraction system. The second step is not automatic and may require much work for each target protocol. Providing automatic or semi-automatic integration may improve the usability. This could be built using semantic technologies (ontologies) in order to exploit integration technologies in other fields.

6.6. Evaluate the acceptance in the community of practices

Evaluate the acceptance of such a framework in the production, Big Data, and data science communities in order to improve the acceptance and increase its usage. Additionally, evaluating the users of the solution will provide valuable feedback for future improvements and adoption.

6.7. Security

Security is a complex topic that goes beyond simply enforcing authentication and encryption. A truly secure system needs to completely understand the context of the solution. This requires further and extensive analysis such as identification of stockholders or analysis of vulnerabilities of the system.

6.8. Operation

Currently, the task of deploying, (re)configuring, and monitoring uses a combination of APIs and SDKs that requires certain computer engineering skills. This can be solved by using configuration management tools (e.g. Saltstack, Ansible) and graphical flow-based programming environments (e.g. Apache NiFi, Node-Red). However, some of these tools require some extra integration effort. Therefore, this topic must be further studied.

Notes

1. SMT is a method for producing electronic circuits in which the components are mounted directly onto the surface of printed circuit boards.
2. Message Queuing Telemetry Transport is a standard (ISO/IEC PRF 20,922) publish-subscribe-based messaging protocol (Banks and Gupta 2014).
3. Object Linking and Embedding for Process Control.
4. REpresentational State Transfer (REST) or RESTful.
5. Complex-Event Machine Learning (see Soto et al. 2016a).
6. Application Programming Interface.
7. The metrics are the values of the calculation of the performance scores (e.g. Accuracy or RMSE (root-mean-square error)) for classification or regression problems, respectively.
8. Matthews correlation coefficient is a preferred evaluation metric for imbalanced datasets, because it takes into account the weight of all classes.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

Work for this paper was supported by the European Commission Horizon 2020 Framework Programme under grant No. 691829 (Ilie-Zudor 2016) and the COMPOSITION project No. 723145 (Jentsch 2016).

ORCID

F. Tavakolizadeh http://orcid.org/0000-0002-5415-9899
D. Gyulai http://orcid.org/0000-0003-1422-1130

References

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, et al. 2016. "Tensorflow: Large-Scale Machine Learning on Heterogeneous Distributed Systems." arXivpreprintarXiv:1603.04467

Acciani, G., G. Brunetti, and G. Fornarelli. 2006. "Application of Neural Networks in Optical Inspection and Classification of Solder Joints in Surface Mount Technology." *IEEE Transactions on Industrial Informatics* 2 (3): 200–209. doi:10.1109/TII.2006.877265.

Aggarwal, C. C. 2007. *Data Streams: Models and Algorithms*. Vol. 31. US: Springer Science & Business Media.

Apache Foundation. 2018. "Apache Kafka." https://kafka.apache.org/

Backmann, M., A. Baumgrass, N. Herzberg, A. Meyer, and M. Weske. 2013. "Model-Driven Event Query Generation for Business Process Monitoring." In *International Conference on Service-Oriented Computing*, 406–418. Berlin, Germany: Springer.

Banks, A., and R. Gupta 2014. "MQTT Version 3.1.1." Accessed March, 2016. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

Bauer, H., C. Baur, G. Camplone, et al. 2015. "Industry 4.0: How to Navigate Digitization of the Manufacturing Sector." *Technical Report of McKinsey Digital*.

Baumgraß, A., N. Herzberg, A. Meyer, and M. Weske. 2014. "BPMN Extension for Business Process Monitoring." In *EMISA*, Luxembourg, Luxemburg.

Bodnarova, A., M. Bennamoun, and S. Latham. 2002. "Optimal Gabor Filters for Textile Flaw Detection." *Pattern Recognition* 35 (12): 2973–2991. doi:10.1016/S0031-3203(02)00017-1.

Bonino, D., M. T. D. Alizo, A. Alapetite, T. Gilbert, M. Axling, H. Udsen, J. Á. C. Soto, and M. Spirito. 2015. "ALMANAC: Internet of Things for Smart Cities." In *The 3rd International Conference on Future Internet of Things and Cloud (FiCloud2015)*, Rome, Italy: IEEE.

Bonino, D., M. T. Delgado, C. Pastrone, M. Spirito, A. Alapetite, T. Gilbert, and M. Axling. 2017. "Enabling Smart Cities through IoT: The ALMANAC Way, Chap. 8." In *The Internet of Things: Foundation for Smart Cities, eHealth, and Ubiquitous Computing*, edited by R. Armentano, R. S. Bhadoria, P. Chatterjee, G. C. Deka, et al., 173–201. New York: CRC Press Press.

Bottou, L. 1991. "Stochastic Gradient Learning in Neural Networks." In *Proceedings of Neuro-Nîmes 91*. France: Nimes. http://leon.bottou.org/papers/bottou-91c

Breiman, L. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32. doi:10.1023/A:1010933404324.

Brosnan, T., and D. W. Sun. 2002. "Inspection and Grading of Agricultural and Food Products by Computer Vision Systems a Review." *Computers and Electronics in Agriculture* 36 (2): 193–213. doi:10.1016/S0168-1699(02)00101-1.

Buchmann, A., and B. Koldehofe. 2009. "Complex Event Processing." *IT-Information Technology Methoden Und Innovative Anwendungen Der Informatik Und Informationstechnik* 51 (5): 241–242.

Calantone, R. J., and C. A. Di Benedetto. 2000. "Performance and Time to Market: Accelerating Cycle Time with Overlapping Stages." *IEEE Transactions on Engineering Management* 47 (2): 232–244. doi:10.1109/17.846790.

Cheeseman, M. J., P. Swann, G. B. Hesketh, and S. Barnes. 2005. "Adaptive Manufacturing Scheduling: A Flexible and Configurable Agent-Based Prototype." *Production Planning & Control* 16 (5): 479–487. doi:10.1080/09537280500121810.

Chen, F. L., and S. F. Liu. 2000. "A Neural-Network Approach to Recognize Defect Spatial Pattern in Semiconductor Fabrication." *IEEE Transactions on Semiconductor Manufacturing* 13 (3): 366–373. doi:10.1109/66.857947.

Chiu, S., S. L. Wang, and Y. S. P. Chiu. 2007. "Determining the Optimal Run Time for EPQ Model with Scrap, Rework, and Stochastic Breakdowns." *European Journal of Operational Research* 180 (2): 664–676. doi:10.1016/j.ejor.2006.05.005.

Chiu, Y.S. P., and H.H. Chang. 2014. "Optimal Run Time for EPQ Model with Scrap, Rework and Stochastic Breakdowns: A Note." *Economic Modelling* 37: 143–148. doi:10.1016/j.econmod.2013.11.006.

Ciresan, D. C., U. Meier, and J. Schmidhuber. 2012. "Transfer Learning for Latin and Chinese Characters with Deep Neural Networks." In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, 1–6. Brisbane, Australia: IEEE.

Cugola, G., and A. Margara. 2012. "Processing Flows of Information: From Data Stream to Complex Event Processing." *ACM Computing Surveys (CSUR)* 44 (3): 15. doi:10.1145/2187671.2187677.

De Ron, A. J., and J. E. Rooda. 2006. "OEE and Equipment Effectiveness: An Evaluation." *International Journal of Production Research* 44 (23): 4987–5003. doi:10.1080/00207540600573402.

Dierks, T., and E. Rescorla. 2008. "The Transport Layer Security (TLS) Protocol Version 1.2." RFC 5246 (Proposed Standard), Aug. Updated by RFCs 5746, 5878, 6176, http://www.ietf.org/rfc/rfc5246.txt

Docker, Inc. 2018. "Docker." docs.docker.com

Escofet, J., R. Navarro, J. Pladellorens et al. 1998. "Detection of Local Defects in Textile Webs Using Gabor Filters." *Optical Engineering* 37 (8): 2297–2307. DOI:10.1117/1.601751.

Fogliatto, F. S., G. J. C. Da Silveira, and D. Borenstein. 2012. "The Mass Customization Decade: An Updated Review of the Literature." *International Journal of Production Economics* 138 (1): 14–25. doi:10.1016/j.ijpe.2012.03.002.

Gaber, M. M., S. Krishnaswamy, and A. Zaslavsky. 2005. "On-Board Mining of Data Streams in Sensor Networks." In *Advanced Methods for Knowledge Discovery from Complex Data*, edited by Ujjwal Maulik, Lawrence B. Holder, Diane J. Cook, 307–335. London: Springer.

Gaber, M. M., A. Zaslavsky, and S. Krishnaswamy. 2005. "Mining Data Streams: A Review." *SIGMOD Record* 34 (2): 18–26. doi:10.1145/1083784.1083789.

Gallarda, H. S., C. W. Lo, A. Rhoads, and C. G. Talbot. 2003. "Feature-Based Defect Detection."US Patent 6,539,106. March. 25.

Gama, J. 2010. *Knowledge Discovery from Data Streams*. New York: CRC Press.

Hameed, B., F. Durr, and K. Rothermel. 2011. *RFID Based Complex Event Processing in a Smart Real-Time Factory*, 3rd CIRP International

Conference on Industrial Product Service Systems, Braunschweig, Germany.
Hayek, P. A., and M. K. Salameh. 2001. "Production Lot Sizing with the Reworking of Imperfect Quality Items Produced." *Production Planning & Control* 12 (6): 584–590. doi:10.1080/095372801750397707.
Hermann, M., T. Pentek, and B. Otto. 2016. "Design Principles for Industrie 4.0 Scenarios." In *System Sciences (HICSS), 2016 49th Hawaii International Conference on*. 3928–3937. Kauai, Hawaii: IEEE.
Huang, S. H., J. P. Dismukes, J. Shi, Q. I. Su, M. A. Razzak, R. Bodhale, and D. Eugene Robinson. 2003. "Manufacturing Productivity Improvement Using Effectiveness Metrics and Simulation Analysis." *International Journal of Production Research* 41 (3): 513–527. doi:10.1080/0020754021000042391.
Hui, T. W., and G. K. H. Pang. 2009. "Solder Paste Inspection Using Region-Based Defect Detection." *The International Journal of Advanced Manufacturing Technology* 42 (7–8): 725–734. doi:10.1007/s00170-008-1639-6.
Ilie-Zudor, E. 2016. "EXCELL Project." http://excell-project.eu/
Izaguirre, M., J. A. Garcia, A. Lobov, and J. L. Martinez Lastra. 2011. "OPC-UA and DPWS Interoperability for Factory Floor Monitoring Using Complex Event Processing." In *2011 9th IEEE International Conference on Industrial Informatics*, 205–211. Lisbon, Portugal: IEEE.
Jentsch, M. 2016. "Composition Project." https://www.composition-project.eu/
Jin, X., X. Lee, N. Kong, and B. Yan. 2008. "Efficient Complex Event Processing over RFID Data Stream." In *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*, 75–81. Portland, Oregon, USA: IEEE.
Jones, E., T. Oliphant, P. Peterson, et al. 2001. "SciPy: Open Source Scientific Tools for Python." Accessed March 12 2017. http://www.scipy.org/
Jones, M., J. Bradley, and N. Sakimura. 2015. "JSON Web Signature (JWS)." *Technical Report*.
JSON. 2013. "*The JSON Data Interchange Format*." *Technical Report Standard ECMA-404 1st Edition/October 2013*. ECMA. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
Kagermann, H., W. Wahlster, and J. Helbig. 2013. "Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0." *Technical Report. Lyoner Strae 9 60528*. Frankfurt/Main Germany: National Academy of Science and Engineering.
Kent, K., J. Kivlin, and E. Gasmann. 2001. "Automatic Defect Detection and Generation of Control Code for Subsequent Defect Repair on an Assembly Line."US Patent 6,240,633. June. 5.
Kim, T. H., T. H. Cho, Y. S. Moon, and S. H. Park. 1999. "Visual Inspection System for the Classification of Solder Joints." *Pattern Recognition* 32 (4): 565–575. doi:10.1016/S0031-3203(98)00103-4.
Ko, K. W., Y. J. Roh, H. S. Cho, and H. C. Kimn. 2000. "A Neural Network Approach to the Inspection of Ball Grid Array Solder Joints on Printed Circuit Boards." In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNIEEES-ENNS International Joint Conference on Neural Networks*, Vol. 5, 233–238, Como, Italy.
Kumar, A. 2003. "Neural Network Based Detection of Local Textile Defects." *Pattern Recognition* 36 (7): 1645–1659. doi:10.1016/S0031-3203(03)00005-0.
Kunz, S., T. Fickinger, J. Prescher, and K. Spengler. 2010. "Managing Complex Event Processes with Business Process Modeling Notation." In *International Workshop on Business Process Modeling Notation*, 78–90. Potsdam, Germany: Springer.
Lee, J., B. Bagheri, and H. A. Kao. 2015. "A Cyber-Physical Systems Architecture for Industry 4.0-Based Manufacturing Systems." *Manufacturing Letters* 3: 18–23. doi:10.1016/j.mfglet.2014.12.001.
Leitão, P., and F. Restivo. 2006. "ADACOR: A Holonic Architecture for Agile and Adaptive Manufacturing Control." *Computers in Industry* 57 (2): 121–130. doi:10.1016/j.compind.2005.05.005.
Li, M. H. C., A. Al-Refaie, and C. Y. Yang. 2008. "DMAIC Approach to Improve the Capability of SMT Solder Printing Process." *IEEE Transactions on Electronics Packaging Manufacturing* 31 (2): 126–133. doi:10.1109/TEPM.2008.919342.
Li, Q., M. Wang, and G. Weikang. 2002. "Computer Vision Based System for Apple Surface Defect Detection." *Computers and Electronics in Agriculture* 36 (2): 215–223. doi:10.1016/S0168-1699(02)00093-5.
Liang, S. H. L., C. Y. Huang, and T. Khalafbeigi. 2016. "OGC SensorThings API Part I:Sensing OGC Implementation Standard." http://docs.opengeospatial.org/is/15-078r6/15-078r6.html
Light, R. A. 2017. "Mosquitto: Server and Client Implementation of the MQTT Protocol." *The Journal of Open Source Software* 2 (13): 265. doi:10.21105/joss.00265.
LinkSmart. 2016. "LinkSmart Learning Agent." may. Accessed December 01 2017. https://docs.LinkSmart.eu/display/LA/
LinkSmart. 2017. "LinkSmart Platform." may. Accessed January 10 2018. https://docs.LinkSmart.eu
Lotter, B., and H. P. Wiendahl. 2009. "Changeable and Reconfigurable Assembly Systems." In *Changeable and Reconfigurable Manufacturing Systems*, edted by Hoda A. ElMaraghy, 127–142. London: Springer.
Luckham, D. C. 2011. *Event Processing for Business: Organizing the Real-Time Enterprise*. Chichester, United Kingdom: John Wiley & Sons.
Mahnke, W., S.-H. Leitner, and M. Damm. 2009. *OPC Unified Architecture*. Berlin, Heidelberg: Springer Science & Business Media.
Maier, A., A. Vodencarevic, O. Niggemann, R. Just, and M. Jaeger. 2011. "Anomaly Detection in Production Plants Using Timed Automata." In *8th International Conference on Informatics in Control*, Automation *and* Robotics (ICINCO), 363–369, Noordwijkerhout, The Netherlands.
Michael, M., J. E. Moreira, D. Shiloach, and R. W. Wisniewski. 2007. "Scale-Up X Scale-Out: A Case Study Using Nutch/Lucene." In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, 1–8. Long Beach, California, USA: IEEE.
Monostori, L. 2014. "Cyber-Physical Production Systems: Roots, Expectations and R&D Challenges." *Procedia Cirp* 17: 9–13. doi:10.1016/j.procir.2014.03.115.
Mont, O. K. 2002. "Clarifying the Concept of Product–Service System." *Journal of Cleaner Production* 10 (3): 237–245. doi:10.1016/S0959-6526(01)00039-7.
Muchiri, P., and L. Pintelon. 2008. "Performance Measurement Using Overall Equipment Effectiveness (OEE): Literature Review and Practical Application Discussion." *International Journal of Production Research* 46 (13): 3517–3535. doi:10.1080/00207540601142645.
Nassehi, A., S. T. Newman, and R. D. Allen. 2006. "STEP-NC Compliant Process Planning as an Enabler for Adaptive Global Manufacturing." *Robotics and Computer-Integrated Manufacturing* 22 (5): 456–467. doi:10.1016/j.rcim.2005.11.003.
Newman, S. 2015. *Building Microservices: Designing Fine-Grained Systems*. Sebastopol, California, USA: O'Reilly Media.
OPC Foundation. 2017. "OPC Data Access." Accessed December 13 2017. https://opcfoundation.org/developer-tools/specifications-classic/data-access/
Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12 (Oct): 2825–2830.
Pellicciari, M., A. O. Andrisano, F. Leali, and A. Vergnano. 2009. "Engineering Method for Adaptive Manufacturing Systems Design." *International Journal on Interactive Design and Manufacturing (Ijidem)* 3 (2): 81–91. doi:10.1007/s12008-009-0065-9.
Pham, D. T., and A. A. Afify. 2005. "Machine-Learning Techniques and Their Applications in Manufacturing." *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 219 (5): 395–412. doi:10.1243/095440505X32274.
Piller, F., E. Lindgens, and F. Steiner. 2012. *Mass Customization at Adidas: Three Strategic Capabilities to Implement Mass Customization*, SSRN Electronic Journal.
Richardson, L., and S. Ruby. 2008. *RESTful Web Services*. Sebastopol, California, USA: O'Reilly Media.
Scipy. 2017. "Scipy: Logistic Probability Density Function." Accessed August 02 2017. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.logistic.html
Siemens. 2016. "Plant Simulation." https://www.plm.automation.siemens.com
Siemens, A. G. 2018. "Siemens s7 SIMATIC."
Sola, J., and J. Sevilla. 1997. "Importance of Input Data Normalization for the Application of Neural Networks to Complex Industrial Problems." *IEEE Transactions on Nuclear Science* 44 (3): 1464–1468. doi:10.1109/23.589532.

Soto, C., J. Ángel, M. Jentsch, D. Preuveneers, and E. Ilie-Zudor. 2016a. "CEML: Mixing and Moving Complex Event Processing and Machine Learning to the Edge of the Network for IoT Applications." In *Proceedings of the 6th International Conference on the Internet of Things*, IoT'16, New York, NY, USA, 103–110. ACM. doi:10.1145/2991561.2991575.

Soto, C., J. Angel, O. Werner-Kytölä, M. Jahn, J. Pullmann, D. Bonino, C. Pastrone, and M. Spirito. 2016b. "Towards a Federation of Smart City Services." In *International Conference on Recent Advances in Computer* Systems *(RACS 2015)*, 163–168, Hail, Saudi Arabia.

Stéfan van der, W., S. C. Colbert, and G. Varoquaux. 2011. "The NumPy Array: A Structure for Efficient Numerical Computation." *Computing in Science & Engineering* 13 (2): 22–30. doi:10.1109/MCSE.2011.37.

Taleizadeh, A. A., H. M. Wee, and S. J. Sadjadi. 2010. "Multi-Product Production Quantity Model with Repair Failure and Partial Backordering." *Computers & Industrial Engineering* 59 (1): 45–54. doi:10.1016/j.cie.2010.02.015.

Tavakolizadeh, F., J. N. C. Soto, D. Gyulai, and C. Beecks. 2017. "Industry 4.0: Mining Physical Defects in Production of Surface-Mount Devices." In *Proceedings of the 17th Industrial Conference, Advances in Data Mining, ICDM 2017*, New York, NY, 146–151. http://www.data-mining-forum.de/books/icdmposter2017.pdf

Theano Development Team. 2016. "Theano: A Python Framework for Fast Computation of Mathematical Expressions." *arXiv e-prints* abs/1605.02688. http://arxiv.org/abs/1605.02688

Toth, D., and T. Aach. 2001. "Improved Minimum Distance Classification with Gaussian Outlier Detection for Industrial Inspection." In *Image Analysis and Processing, 2001. Proceedings. 11th International Conference on*, 584–588. Palermo, Italy: IEEE.

Tsai, D. M., and R. H. Yang. 2005. "An Eigenvalue-Based Similarity Measure and Its Application in Defect Detection." *Image and Vision Computing* 23 (12): 1094–1101. doi:10.1016/j.imavis.2005.07.014.

Vachtsevanos, G. J., I. M. Dar, K. E. Newman, and E. Sahinci. 1999. "Inspection System and Method for Bond Detection and Validation of Surface Mount Devices."US Patent 5,963,662. October. 5.

Viharos, F., J. Zs, S. D. Chetverikov, T. A. Háry, F. R. Sághegyi, F. A. Barta, S. L. Zalányi, S. I. Pomozi, E. S. Soós, N. Z. Kövér, and T. B. Varjú. 2016. "Vision Based, Statistical Learning System for Fault Recognition in Industrial Assembly Environment." In *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*, 1–6. IEEE.

Vilella, J. L. 2009. "Electronic Assembly Video Inspection System.". US Patent 7,486,813. Feburary 3.

Vyatkin, V., Z. Salcic, P. S. Roop, and J. Fitzgerald. 2007. "Now That's Smart!" *IEEE Industrial Electronics Magazine* 1 (4): 17–29.

Zliobaite, I., A. Bifet, B. Pfahringer, and G. Holmes. 2014. "Active Learning with Drifting Streaming Data." *IEEE Transactions on Neural Networks and Learning Systems* 25 (1): 27–39. doi:10.1109/TNNLS.2012.2236570.