Chapter 5
Tools and Procedures to Embed and Retrieve Product-Service Lifecycle Knowledge

Jacopo Cassina, Ida Critelli, Lara Binotti, Eva Coscia and Stefano Borgia

Abstract The cross-disciplinary collaborative management environment developed in Manutelligence project for Product-Service (P-S) engineering has the primary objective of managing all data, information and knowledge related to the P-S and its lifecycle in manufacturing. The present chapter presents tools and procedures to embed and retrieve this lifecycle knowledge. As extracting feedback from customers can make the P-S more suitable and attractive to the customers themselves, improving the P-S design and the development of new services, first of all an analysis of the approaches for customer feedback based on the Manutelligence industrial cases (automotive, ship, smart house, 3D-printing) has been performed. The customer feedback is characterized through different viewpoints: customer types, benefits, feedback content and methods for feedback collection, also focusing on how the Manutelligence platform can support the feedback collection. Concerning the use cases, practical examples on some users experience performed using the modules of Manutelligence platform are described. In particular, the application developed for Ferrari data retrieval and the visualization tool of Mayer are introduced as significant modules supporting interaction and feedback between the customer and designer. Taking into account the information coming from different sources (e.g. PLCs, sensorial IoT nodes, etc.) in the context of production system, a methodology to present coherently field data is proposed, with the idea to offer interoperability and integration also from the point of view of the different devices used to collect these data. For this purposed, a specific application—part of i-LiKe suite and comprising a specific dashboard for manufacturing data visualization—has been conceived. In order to explain the related capabilities and functionalities, a simulator related to 3D printing is presented as an extended demonstration of the proposed approach.

J. Cassina (✉) · I. Critelli · L. Binotti · E. Coscia · S. Borgia
Holonix s.r.l., Corso Italia, 8, 20821 Meda, MB, Italy
e-mail: Jacopo.cassina@holonix.it

© The Author(s) 2019

L. Cattaneo and S. Terzi (eds.), *Models, Methods and Tools for Product Service Design*, PoliMI SpringerBriefs, https://doi.org/10.1007/978-3-319-95849-1_5

5.1 Methodologies for Customer Feedback

5.1.1 Introduction

Extracting feedback from P-S customers can make the P-S more suitable and attractive to the customers themselves. However, it is always not easy to get the feedback. The objective in this paragraph is to analyse the approaches for customer feedback based on the Manutelligence industrial cases: automotive, ship, smart house, fablab/3D-printing. The customer feedback is analysed through different viewpoints: customer types, benefits, feedback content and methods for feedback collection. Additionally it is discussed, how the Manutelligence platform can support the feedback collection.

5.1.2 Customer Feedback Analysis Approach

The Manutelligence platform aims to enable designers and engineers access data from the traditional enterprise IT systems, but also from the IoT enabled systems. The objective is to manage all data, information and knowledge related to the P-S and its lifecycle in manufacturing. In the beginning of the project each of the four use case pilots specified the use scenarios, including descriptions of processes or process parts which could be supported by a P-S engineering platform. These scenarios were the main source for this feedback analysis. The descriptions included the objectives, challenges, lifecycle stage and the use cases included. Each scenario could have more than one use case. Each use case again was described with a common format including for example actors, precondition, post-condition, systems involved, diagram and the main steps. The number of scenarios and use cases defined were 14 scenarios and 34 use cases.

Here the Manutelligence use cases have been used as the information source to analyse real world needs, opportunities and methods for customer feedback. The analysis included the following steps:

- Identification of the use scenarios and use cases in which customer is involved as an active or passive source of feedback. As a whole in 7 of the 14 scenarios and 9 of the 34 use cases some kind of customer feedback was included.
- Analysis of the identified scenarios and use cases using a common approach and template was done.
- Consolidation of the analysis results. This was performed for the items considered most important, like the customer/user type, objectives and benefits, lifecycle stage, feedback type and channels or tools, how the feedback is used and what could be the role of the Manutelligence platform.
- Identification of similarities and differences; developing a mapping framework and conclusions.

5.1.3 Customer Types

When talking about customer feedback, it is important to identify that there are different types of customers. In Manutelligence the following definitions are given:

- Customer: "Actors (typically a person or organization) who request and pay for a value representation".
- User: "Actors who take immediate benefit from a value representation (e.g. ownership of a product)".

Sometimes the customers are the same as users but in other cases the customer is not necessarily the user: the user may be the customer of the customer, or even the customer of the customer of the customer. As an example: the customer of a shipyard is the ship owner and the final end users are the travellers. Additionally, the ship owner is also the user in the operation phase but with a different focus than the travellers. More, in addition to the owner and end user, there may be also actors who operate the P-S.

In principle the feedback from customers may come from different customer/user levels and the P-S providers would be interested to get the feedback as early as possible in the lifecycle. However, typically it is not as easy to get feedback from the user level (if not the first customer) in the lifecycle phase when the P-S is still in the design or manufacturing/implementation phase. On the other hand, in the usage phase it is often more simple to get the feedback from the user.

5.1.4 Objectives and Benefits

Even if the Manutelligence use cases significantly differ from each other in size and complexity, they all express, in different ways, one common objective for the collection of the customer feedback. It is the improvement of the P-S, either of the P-S instantiation (a specific P-S) or the future P-Ss. The improved P-S is expected to influence the customer satisfaction and thus to improve the competitiveness of the company. Additionally it is expected that the interaction including the feedback and the potential to influence the P-S design strengthens the customer relationship and enables better understanding about the future needs.

The availability of life cycle analysis (LCA) and life cycle cost analysis (LCC) on the platform allows the integration of LCA and LCC analysis into the design process. The results can be offered to the customer and the customer is able to give feedback if the P-S performance is sufficient. The end user can make the decision based on sustainability assessment and long term costs. For example, the customer can predict future energy consumption and ask for changes if the performance is not good enough.

In addition to high quality P-S, one expected benefit is to speed up the design, P-S specification and implementation processes, and to decrease the costs. Efficient

feedback tools enabled by the platform allow faster fixing of the design decisions, but also avoiding errors in the design. Thus there is decreased need to waste time for the correction of errors and the subsequent manufacturing/implementation phases may be more efficient.

5.1.5 Mapping Feedback Scenarios

As described above, the main reasons for collecting customer feedback is to improve the P-S design and the development of new services. Thus the content of the feedback is mainly customer opinions and data about the P-S performance. The customer feedback could be categorized with two main dimensions:

- The P-S lifecycle phase in which the feedback is collected and analysed. Main phases identified are P-S design/implementation and P-S operation/use phases. The end-of-life phase was not visible in the scenarios.
- The type and method of feedback: type meaning the information type (unstructured information, structured information, data) and method meaning how the feedback is given (customer manual input, customer selection from predefined options, automatic (for example) sensor data). It seems clear that in most cases type and method are not independent but interlinked: the unstructured information requires some customer activity while the bigger amounts of data are coming automatically from sensors, for example via IoT. The structured feedback (for example selection between options) can be derived either from the customer or from automatic devices.

Thus the main dimensions against which the use cases can be compared and analysed are: P-S lifecycle phase and the feedback type and method; customer activity with different types of data/automatic retrieval with structured data. The customer activity may mean feedback given through the platform, email or discussions in a meeting etc. In the project has been shown that the different use cases had a different focus in their scenarios and collaboration with customers.

5.1.6 Role of Manutelligence Platform

Analysis of the Manutelligence case scenarios brings out that there is a need for retrieving customer feedback in all lifecycle phases even if no end of life scenarios were available in the current project. It is clear that an IT platform managing the design and feedback information and utilizing IoT (Internet of Things) is needed to collect and manage the large amounts of data given by automatic sensors. The data can be used for use phase services and for further design. Often there is also a need to compare the real data against designed performance (for example energy consumption models).

To receive feedback from the customer through customer actions the P-S provider needs to make the feedback action attractive for the customer. This means that it should be easy and interesting for the customer, for example to understand the current P-S version in the design phase, and also easy to give the feedback. In the use cases this was implemented through visualization, even experimenting through gamification, supported by the platform. The feedback could be directly appointed to the visual models or given by more traditional means, like in meetings. Thus also here the platform is needed both to present the P-S for which feedback is needed but also to save and analyse the collected, often heterogeneous information.

In the Fablab-case there is a user community, which is interested to interact and give feedback to the P-S provider but also to share experiences. Thus the platform needs to offer tools also for this communication and collaboration.

As an important function of the platform in relation to customer feedback is to take care that the customer feedback is handled and used for the current P-S instantiation or for future P-Ss. Thus systematic change management process, supported by the platform, is needed. The change process also takes care about informing the customer about the results of the feedback process: what changes were made.

As a conclusion, a P-S platform may have different roles in the customer feedback process:

- The platform should manage the rich P-S data and information throughout the lifecycle.
- The platform should offer the P-S information to the customer in an understandable interface.
- The platform should offer the customer a possibility to give different types of comments.
- The platform should integrate to IoT to collect data from different types of sensors.
- It should be possible to analyse the feedback data and information using the platform, for example to compare real and designed data.
- The platform should support the change management process.
- The platform should allow communication between different users or different actors.
- The platform should support organizational change management by enabling dynamic changes required by changes in roles and tasks.

5.2 Tools for Customer Involvement: Manutelligence Web Application User Interfaces

The objective of this paragraph is to provide some practical examples, related to the industrial use cases, on some users experience performed using modules of the Manutelligenc**e** platform. Not all the scenarios are presented for the sake of brevity. The next paragraphs present also different tools adopted to involve different customers inside the platform.

5.2.1 Ferrari Use Case User Experience

The objective of this paragraph is to illustrate the application developed for Ferrari data retrieval from the point of view of the user stream. These data can be used to provide feedbacks to two types of customers. More specifically, in this scenario, it is possible to provide feedback to the Ferrari designer who can use the data acquisition to improve the design of the future cars and also to the driver of the car, who can visualize the progress of different variables of the test drive performed on the track. After the login inside the application, the user can access the list of cars that have been tracked inside the platform as illustrated in Fig. 5.1. By clicking new car, it is also possible to add manually a new record belonging to the list of cars.

In the same way, also a list of gateways (Fig. 5.2), used to acquire data and associated to a specific car, is made available to the user.

The user can select a specific track associated to a specific car and then can visualize the GPS position of the car and also the variables acquired as illustrated in Fig. 5.3.

The user can create a placeholder for a specific event, e.g. it is possible to report if at some point of the test drive something relevant has happened as shown in Fig. 5.4.

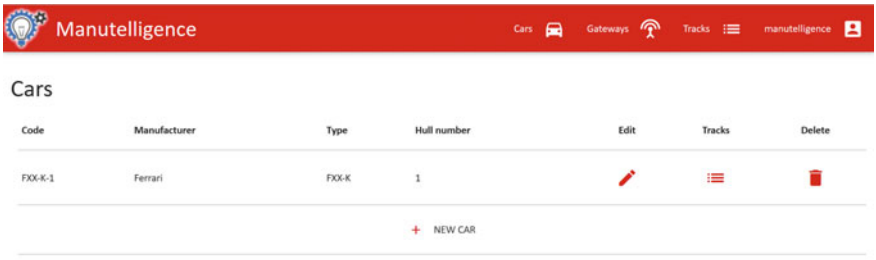

Fig. 5.1 List of cars

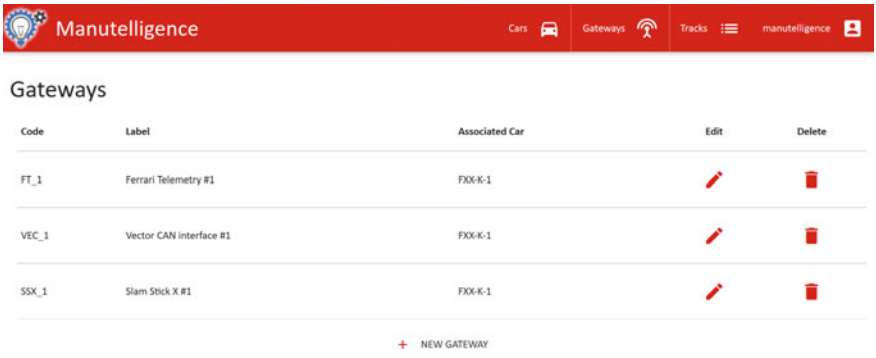

Fig. 5.2 List of gateways

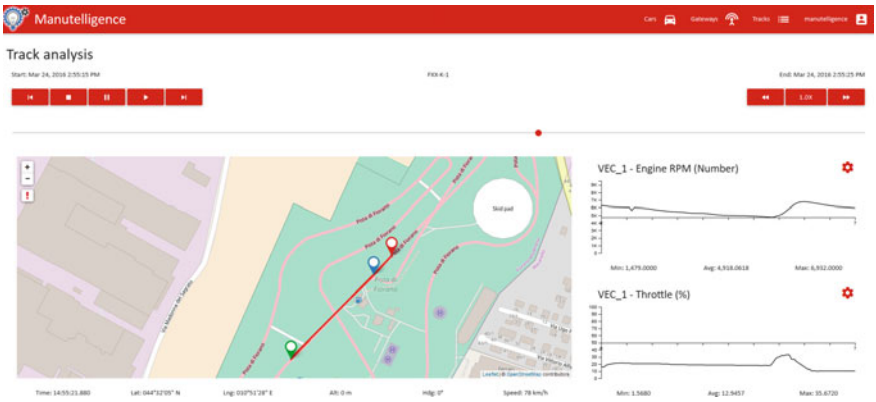

Fig. 5.3 Track data visualization

Fig. 5.4 Visualization on the map of the placeholder related to a specific event

As a conclusion, in this module of the ***Manutelligence*** platform, the roles in the customer feedback process exploited are:

- Offer the information to the customer in an understandable interface.
- Offer the customer a possibility to give different types of comments.
- Integrate the IoT to collect data from different types of sensors.
- Analyse the feedback data and information using the platform, for example to compare real and designed data.

5.2.2 *Meyer Use Case: Customer Feedback in Sales and Design*

This use case aims to support the interaction and feedback between the customer and design through ship visualization. The interaction may happen in different lifecycle phases, like sales, design and production. The customer here is the ship owner organization which may include about 200 experts in different fields (e.g. architects). The objective may be to win a sales order, to keep the customer (ship owner) satisfaction high, to get acceptance for the design solutions and to avoid change needs later in the design or in the manufacturing phase.

There are different means to modelling the ship for the customer. The engineering department can make the visual 3D ship model available to the customer and the customer can view the ship model from the engineering platform, rotate it from different angles, move in the model, and zoom in/out, but not change the model as such. A more advanced Virtual Reality demonstration using gaming technologies has been further developed and is presented here. The aim is to use it in the sales phase to make the customer convinced about the offered solution. The scenario "Immersive 3D Walking Experience" with a networked VR platform can be used with different devices, as presented in Fig. 5.5. Clients can connect to the same global VR-pool. The data security is high in the solution. All the parties can see their own personal VR experience and the communication can happen via VOIP and Avatars (Multiplayer concept). This means that the users may walk in the model and "meet" there other users.

Figure 5.6 presents a print-screen from the VR demonstration. The prototype contains a conversion of the ship model to a virtual reality environment, which is compatible with all platforms also in the future market.

The visualization can be considered as an additional knowledge-based service to the customer, which helps to sell more ships and keep the customer satisfied. The knowledge based service may include importing, cleaning and optimizing the CAD data for the game engine use, adding UI components for movements, interactive objects, creating VR-solution for various environments HTC-Vive, Hololens, advanced lighting, textures and sounds.

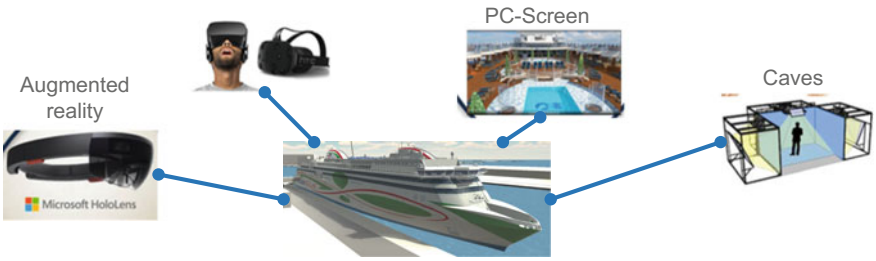

Fig. 5.5 Mock-up of networked VR platform

Fig. 5.6 A printed screen from the ship VR demo

Further development can be made towards new better people in the model (more realistic, better graphics and animations), taking into account the ship environments on where the ship will be travelling + day&night, and upgraded ship models to look good from closer distance (some modelling work will be done, for example deck furniture can be replaced with more detailed models etc.).

As a conclusion, in this module of the Manutelligence platform, the roles in the customer feedback process exploited are:

- Offer the information to the customer in an understandable interface.
- Offer the customer a possibility to give different types of comments.
- Allow communication between different users or different actors.

5.3 Manufacturing Context Driven Intelligence Layer Development and Integration

5.3.1 Introduction

The objective of this paragraph is to present a methodology to present coherently data coming from different devices. What has been designed can aggregate data coming from different devices, like PLCs or sensorial IoT nodes, which are meant to collect data mostly from industrial machines in the factory and present them in a modular and fully configurable dashboard. In order to do so, the application developed for managing different devices, which is also part of the i-LiKe suite, is described in the following. With the purpose to explain what the capabilities and functionalities are, a simulator related to FabLab machines is presented as an extended demonstration of this approach.

5.3.2 Device Abstraction Layer

The device abstraction layer (D.A.L.) is an application developed specifically to manage the devices in the environment of a given project, providing a unified point of access to device data for the other applications.

In particular, it is capable of providing an "image" of the device status built upon the data collected, applying rules to the incoming stream of data, transforming raw low level data into a more meaningful form, recording the data to long term storage and providing a way to retrieve the data recorded. It also provides access control for the devices connected, by distinguishing access for devices, gateways (e.g. a single board PC publishing data for multiple devices) and applications (which will consume the data).

Basically, D.A.L. represents a developed stand-alone application. Considering the Manutelligence platform, it is integrated inside the I-Like module, representing a middle layer between the IoT devices and the component supporting data collecting and preliminary analysis. So D.A.L. can be seen as the interchange data point between devices and the overall SW platform: it exposes a set of REST APIs that allow the dashboard to access data.

Generic implementation

To model the devices, and the rules applied onto them, three main entities are used:

- *DeviceType*: they group devices of the same type (e.g. a 3D printer model) and related rules sets;
- *DeviceTypeVersion*: the version is where the rules for a device type are really contained; it is useful as in development stage different rules sets for the same device type might be needed, so that old devices can continue working on established rules set while new ones can be experimented;
- *Device*: they are the instances of a DeviceType (e.g. the single 3D printers sold by a manufacturer), each one with its status and data.

To provide flexibility required to manage data originating from various different devices, a collection of rule "blocks" is available. The blocks can be combined with each other into the DeviceTypeVersion described above. Each rule block follows the event emitter pattern, using the base classes present in Node.JS runtime environment, so the data it process can directly contribute to the device status "image" and eventually it can be passed on to other rules for further processing.

The rules sets mapped into DeviceTypeVersions are then instantiated for single devices, each one operating on its encapsulated data set.

The DeviceType-DeviceTypeVersion-Device hierarchy, with related device and applications access control metadata, are related to a Context. Several contexts can be accommodated on the same instance of D.A.L., so that multitenancy can be achieved: in certain scenarios this is important as several smaller projects can be served by a rather complex setup of multiple application and databases management systems.

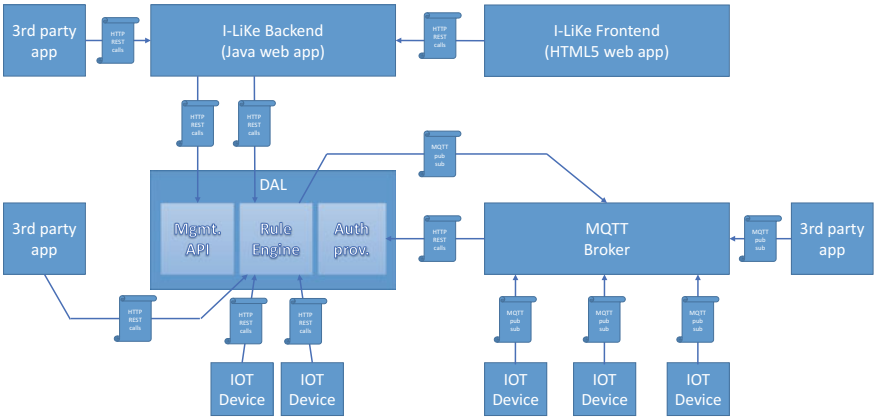

Fig. 5.7 A printed screen from the ship VR demo

The interaction and the interoperability with the system are provided through a REST API (Fig. 5.7). Basically, two sets of REST APIs are used: the first set provides REST HTTP API with JSON data encoding for the management of the devices inside the rule engine (device registry, creation of *DeviceType* and *Device*, activation/deactivation of data acquisition and elaboration, generation of new credentials, etc.); the second one makes available APIs to interact and visualize time series monitored data, alarm list, etc.

Data sources

The first main task of the D.A.L. is to provide contact points where the device can send its data. Two protocols are supported to bring the data from devices onto the system:

- MQTT: (Message Queue Telemetry Transport) is a lightweight protocol based on the publish/subscribe pattern over TCP/IP protocol. MQTT requires a broker to be present, distributing messages to interested subscribers based on the topic of the message. D.A.L. provides a rule block that supports connecting to a broker and subscribe onto a topic.

Also D.A.L. has specific support for Mosquitto MQTT broker (the most widespread open source MQTT broker), providing an interface to enable access control on it.

- HTTP: (Hypertext Transfer Protocol) is the most widespread protocol of the World Wide Web. D.A.L. provides a rule block with the capability to dynamically register a HTTP endpoint where data can be sent with common POST or PUT HTTP requests.

Data collected by the data sources rule blocks is then emitted to the subsequent rules for manipulation.

Closely related to data sources rule blocks, D.A.L. provides buffering rule blocks, enabling time sorting of the data and cadenced emission to the subsequent elaboration pipeline.

Data manipulation

Data incoming from devices sometimes is in a ready to be stored form, but often it needs an elaboration step: simpler devices tend to provide data that greatly benefits from a stream processing approach to aggregate them into a more usable form. Other frequent use cases are the generation of a notification (event or alarm) upon some condition recognized on the data.

D.A.L. provides a set of rules to carry on these tasks in a simple manner, reducing the burden of specific implementations.

To cope with unexpected elaboration steps, the possibility to evaluate a generic Javascript function block onto the data is provided through a dedicated rule.

Data storage and streaming

For data storage several rule blocks are provided, supporting different databases and storage forms:

- Time-JSON: a generic JSON payload associated with a timestamp.
- Time-Decimal: a decimal value associated with a timestamp.
- Interval: an event data structure, identified by a UUID v4 and presenting a start date, an end date and a string value.

For long term data storage, mainly Apache Cassandra and MySQL are supported. Support to other DBMS is planned to be added by providing dedicated rule blocks.

Similarly to data storage, data streaming is also supported. Data can be streamed or stored when it is received onto the elaboration pipeline. Scheduled data extractions can also be defined.

Streaming supports sending data in the following manners:

- MQTT: by connecting to a broker and publishing on a topic.
- HTTP: by sending a POST or PUT request to a URL.
- Redis PUB: by issuing a PUBLISH command onto a Redis instance, using it as a system bus.

Data streaming is a powerful concept, potentially enabling the binding of several instances of D.A.L. in various environments: for example, a D.A.L. instance could process low level data near the devices (edge computing) and then send them for further processing and long term storage to another instance of D.A.L. on an internet accessible server (cloud computing).

Data retrieval

The last important task carried on by D.A.L. is providing a way to access the data it collected so that other applications can consume them. Similarly to HTTP data

sources, dedicated rule blocks are present to dynamically register HTTP endpoints where data can be accessed through HTTP REST API call requests.

Rule blocks supports accessing the current device "image" or accessing the data stored by the data storage rule blocks.

Streaming techniques have been implemented to access the data stored in an efficient way, avoiding abnormal resource consumption with high amount of data.

5.3.3 Manufacturing Data Visualization Through Application Dashboard

To accompany the great flexibility achieved in device data management with D.A.L., an application has been developed to consume the data collected into specific domains. It is mainly a HTML5 application supported by Java based backends exposing REST APIs.

The application maps machine types (e.g. a 3D printer model) and machines (e.g. a single 3D printer); machines access device data exposed by D.A.L. through HTTP interface. Alarms are notified immediately by subscribing on a Redis channel, where D.A.L. streams alarm data.

For example, Fig. 5.8 illustrates the four typical FabLab machines that have been mapped into the application by simulating their ehavior via software as explained in the following.

Machine monitoring

A monitoring section has been implemented, focused on building a dashboard to show the relevant device data. The dashboard is built around a machine type, by composing into a web based editor, a series of panes and widgets. Widgets access

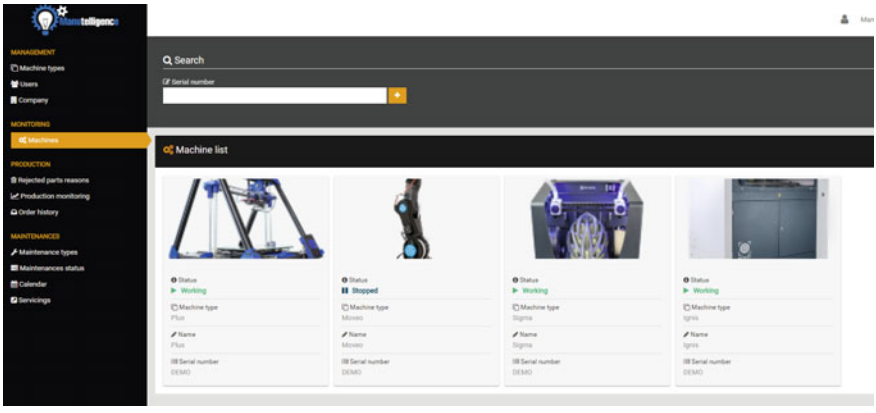

Fig. 5.8 Monitoring—machine list page

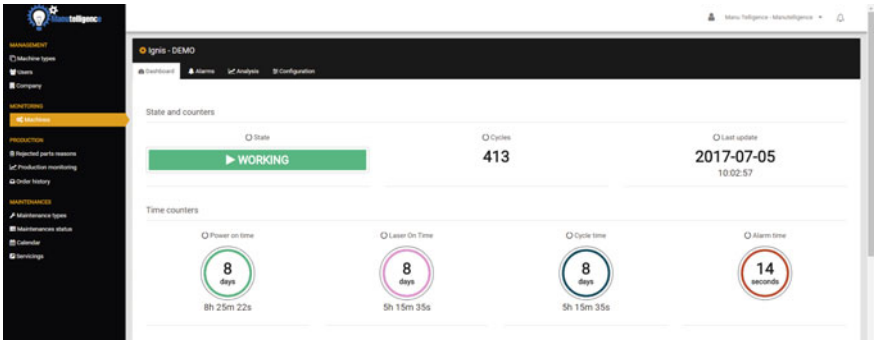

Fig. 5.9 Monitoring—machine dashboard

data from data sources, deriving by how the machine type was defined. In fact, some requirements are posed on which endpoints are exposed by D.A.L. to have a minimum set of available functionality. For example, it is possible to fully create and configure the machine dashboard.

Once the dashboard has been created and configured for the given machine types, all the machines, belonging to the same machine types and registered into the application, will be able to use them and to show their own data on it. For instance, it is possible to see the status (working, stopped, etc.) of the machine, number of cycles, power on time, laser on time and other relevant available parameters, as illustrated in Fig. 5.9. Moreover, dedicated user interfaces are provided to browse the machine alarms history and to plot and compare decimal time series charts.

Maintenance

When a machine type is registered, a maintenance plan can also be defined. The plan is composed of maintenance types, each characterized by several parameters (elapsed time, or counters on the machine status, which is mapped onto D.A.L. device "image") with scheduling parameters.

Typical scenarios on which the schedule is based are:

- Maintenances based on the time elapsed and effective working time of the machine.
- Maintenances based on the number of working cycles of the machine or on the amount of material processed by the machine.
- Maintenances based on the number of triggered events of a certain class.

The user interface provide an immediate indication of the maintenance status by showing the "health" of the machine as the lowest indicator of the remaining life according to the maintenance table. For example, as captured in Fig. 5.10, a maintenance related to the laser head replacement has been created, based on the time elapsed.

As the "health" associated to the various maintenances decreases, it is possible to schedule servicing interventions (Fig. 5.11).

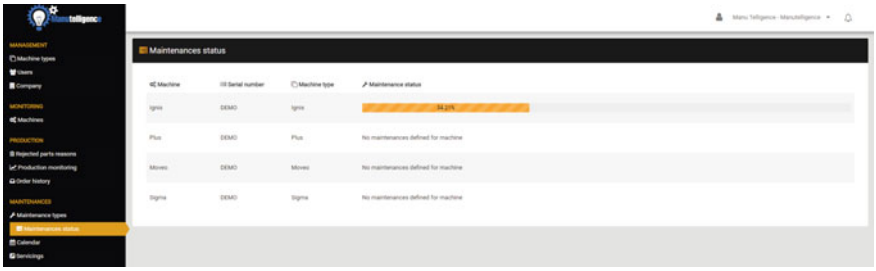

Fig. 5.10 Maintenance—maintenance status

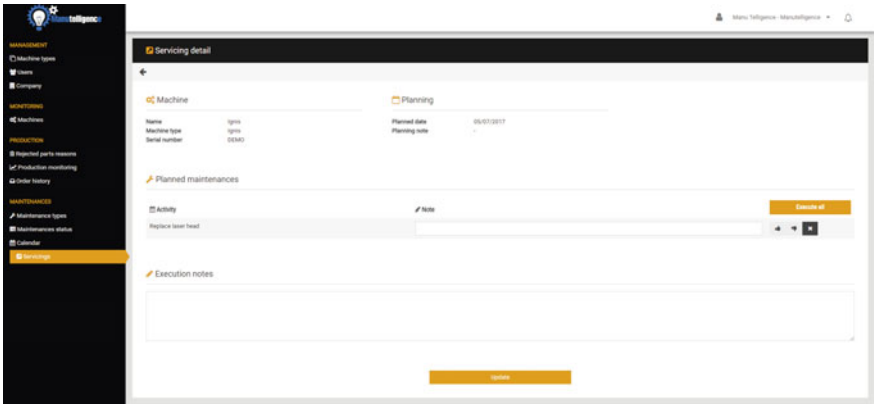

Fig. 5.11 Maintenance—servicing scheduling

It is then possible for the maintainer to clear them out, writing down notes associated to the actions performed on the machine. As the servicing is completed, the health indicators for the associated maintenance are reset.

5.3.4 Industrial Scenario: 3D Printing Monitoring

The aim of this paragraph is to present tools used for the implemented demo scenario, which was meant to describe the possibilities of IoT within the manufacturing scenario. To enable practical tests, we choose a "demo" based on 3D printers and other production machines developed by CIM-UPC, considering also the fact that CIM-UPC 3D printers are manufactured by other 3D-printers (Fig. 5.12).

Another important concept emerged working with 3D Printers, which proved to be true also for many industrial machines, as investigated during the industrial visits, is the unavailability of data from the PLC because there is no PLC (like in the scenario

Fig. 5.12 3D-printers making 3D-printers components in fundacio CIM

of 3D printers), or because it doesn't allow the reading of data (old or proprietary PLC). From this need, the development of a more advanced sensor node was decided.

The requirement were to have a standalone system to equip machines without a PLC or revamping older/not connected machines and that this system was going to work through an independent infrastructure, therefore not requiring a Wi-Fi in customers facilities, but only a single network cable.

From these specifications, the IoT sensor node was developed, being able to read analog and digital inputs and transmit them through MiWi, so through a different infrastructure than the WIFI, to a single data gateway, which connects all the sensor nodes to the cloud.

In the Fig. 5.13, it is possible to see a typical scenario of usage of the IoT sensor node, with 3 machines without possibility of connection to the PLC (as most of the currently on the market machines, due more than to technical issues, to cost of licenses to acquire PLC schemas).

From the machines data are read through analog connections; the IoT sensor node transforms them and sends them in MiWi to the IoT gateway, which, through a cabled connection transfers them to the cloud.

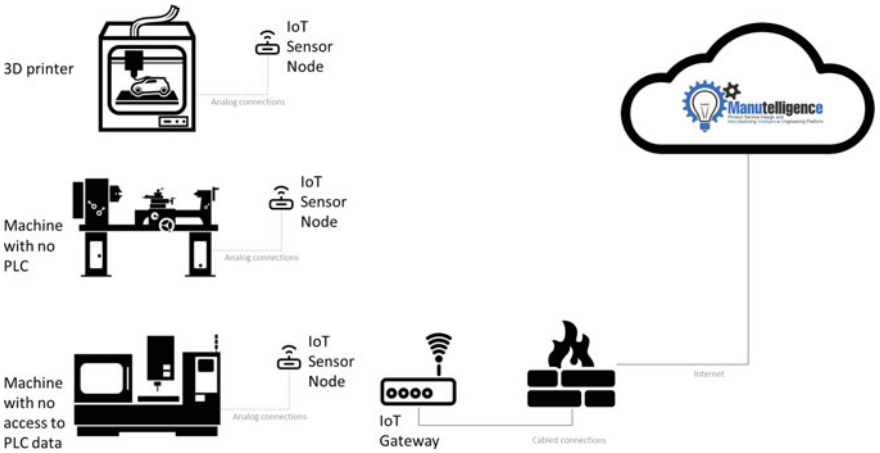

Fig. 5.13 Schema of the IoT sensor node usage

Fig. 5.14 IoT sensor node

IoT sensor node

For interfacing with machines normally unprovided of a PLC/controller or where interfacing with the machine on board controller is not sustainable, an IoT sensor node, named KISS, has been implemented to interact with the machine electrical signals, such as the ones of status lights, or by equipping simple sensors.

The sensor node is illustrated in Fig. 5.14 and it is based on a 16 bit MCU. Five isolated digital input are available (up to eight). Different signals can be identified: ON/OFF, pulses and counter. Three analog input with a 12 bit resolution and ± 10 V range are available. They can be converted into digital inputs.

Fig. 5.15 Machines simulator

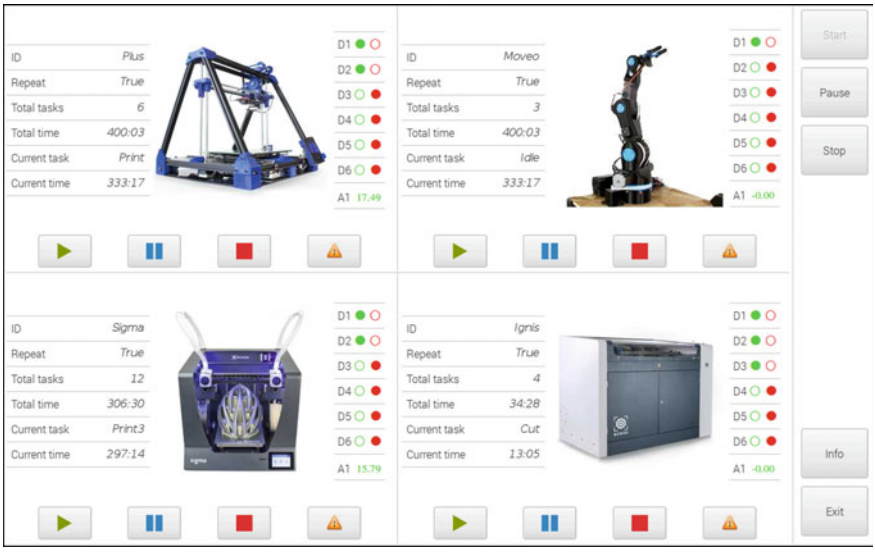

Fig. 5.16 Simulator interface

The communication is enabled by a MiWi (802.15.4) link; sensor-nodes communicates to a concentrator (hub) node, which in turn route the data collected to a gateway, usually a single board PC, where the data is finally published to MQTT through the provided MQTT client (available for both Windows and Linux operating systems).

3D printers simulation

For simulating 3D-printers behaviour, but also other types of machines, a device capable of generating electrical signals as the ones that would be available on a 3D

printer has been engineered (Fig. 5.15). This device is based on a Raspberry Pi 3 single board PC, where a custom made I/O board has been interface to the SPI (Serial Peripheral Interface) bus.

An application has been then developed to control the I/O board, capable of pulling up and down digital outputs and of generating simple patterns over the analogic output. The patterns are combined into a program, which is defined into a JSON file.

The user interface (Fig. 5.16) presents up to four machines, with the ability to start, pause and stop their program or putting them in alarm state. All these events and data are available in the dashboard application and they can be easily consultable.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.