# 8

# *Application system design - Maintenance*

**Erkki Jantunen**
*VTT*

**Mika Karaila**
*Valmet*

**David Hästbacka**
*Tampere University of Technology*

**Antti Koistinen**
*Oulu University*

**Laurentiu Barna**
*Wapice*

**Esko Juuso**
*Oulu University*

**Pablo Puñal Pereira**
*Luleå University of Technology*

**Stéphane Besseau**
*Airbus*

**Julien Hoepffner**
*Airbus*

## CONTENTS

In this chapter the use of Arrowhead Framework is tested in two application environments. The first part of the chapter "Widening the scope: Managing maintenance data in large multifunctional plant environments" reports the challenges in the mining industry and how the challenges of this type of industry can be addressed with the Arrowhead Framework. The second part of the chapter "Aircraft Maintenance System" is from commercial aviation focusing on various examples related to maintenance.



## 8.1 Widening the scope: Managing maintenance data in large multifunctional plant environments

In this section the use and role of the Arrowhead Framework in large multifunctional industrial automation environments is discussed. The basic scenario is related to enabling industrial IoT and connectivity of devices in a process industry setting. The basic motivation for the Arrowhead Framework - to facilitate the interoperability of IoT devices - is discussed in the light of practical pilot examples. The pilots are from the mining industry focusing on various examples related to maintenance. These examples were chosen due to the frequent wear in mining production equipment and the difficulties associated with traditional condition monitoring for production equipment scattered over long distances. Consequently, the new IoT solutions for gathering and analysing data in the mining industry have high potential for lowering maintenance costs and increasing the availability of production equipment.

The chapter describes how the Arrowhead Framework can be seen as an open-source proof-of-concept prototype of a self-configuring IoT sensor system framework suitable for industrial process control environments and also for supporting maintenance processes.

### 8.1.1 Introduction (scenario)

The investigation started with a number of possible targets that should be monitored at Outokumpu Chrome Oy Kemi Mine: hoisting machinery rope damage detection, conveyor pile level measurement, and wear plate condition measurement (metal plate thickness). In the first phase all these cases were studied and feasibility studies with initial hardware were carried out. After some brainstorming workshops at Kemi Mine and studies at the actual process environment, a number of feasible solutions were selected.

The main focus was on providing a system with an acceleration sensor with Bluetooth LE connectivity to monitor the wear of wear plates in the conveyor. For comparison, a high-end validation system was also installed.

Online vibration analysis methods were tested on the rod mill at Kemi Mine's concentrating plant. The rod mill is a primary refiner in the grinding

circuit, taking the maximum power of 1800 kW and so being the greatest energy consumer in mineral processing. Down time and non-optimal operation creates huge expenses in terms of lost production and energy with the mills of this magnitude. Even small improvements in operation and reductions in down time will bring significant savings [1].

In the tested measurement system, vibrations caused by the monitored event define the requirements for the logging arrangement. Vibration logging requires the measurement setup to be capable of recording data at sampling frequencies at least twice as high as the highest monitored frequency. Vibrations were recorded from trunnion bearing supports with four SKF CMSS 2200 accelerometers and data logging was performed with National Instruments cRIO-9024 controller with a cRIO-9114 chassis that includes a Xilinx Virtex-5 reconfigurable FPGA core. Vibration sensors were connected to an NI 9234 analog input module with a built-in anti-aliasing filter.

For storing measured data, remote cloud data storage was set up according to the Mimosa Open O&M information model [2]. For easy connectivity from devices and gateways a REST interface was developed, also usable by value-adding service wanting to further analyse the data as well as end-user applications. Additional control system connectivity was under discussion using OPC UA [3] and Wapice's OPC UA REST API was used as one solution.

The pilot focused on the following main points

- IoT and interoperability of IoT devices
- Arrowhead Framework developed to facilitate interoperability of IoT devices
- Automatic device detection
- Custom sensors and their use in a factory environment:
  - Hoisting machinery rope damage detection (camera & image analysis)
  - Test plate thickness measurement
  - Vibration analysis logging
  - Automatic logging and (possible) readjustment of machine settings
- Objective: To develop an open-source proof-of-concept prototype of a self-configuring IoT sensor system that utilizes the Arrowhead Framework

### 8.1.2 Motivation for monitoring

The ability to include relatively efficient processing power in small sensors and embedded devices in the field opens many possibilities for predictive maintenance and operational monitoring [4]. IoT solutions bring the information from numerous sensors together and enable interactions between distant devices and systems. The possibility of using this new refined information can

shorten the downtime in maintenance breaks by enabling prognostics through the ability to calculate the remaining useful life (RUL) of monitored components. It can also be used for signaling failure alarms to maintenance organisations and spare part providers for fast response. Relevant metadata can also be connected with the measurement information so that the maintenance operators have all the needed information even before consulting the crew at the plant. These kinds of means can enhance operator and maintenance personnel performance with facilitated access to relevant information [5].

In industry the production machinery suffers from wear and consequently maintenance is needed in order to keep the production machinery running at high efficiency. In principle the wear of components of machinery can be monitored based on time, load, or wear, but in practice various factor may have a significant influence on the analysis outcome.

#### 8.1.2.1 Time-based monitoring of wear

Time-based monitoring of wear which would support time-based maintenance is not very efficient because wear does not take place as a function of time, i.e., of the four main models of wear (abrasive, adhesive, fatigue, chemical) only the chemical one can be considered to be a function of time. The others types of wear only take place when the machines are running. In spite of the fact that it is easy to understand that if a component of the machinery is on a table beside the machine it does not wear in a similar manner as inside the running machine, time-based maintenance is widely used. The reason for the wide use is naturally the ease of use, i.e., only a calendar is needed.

#### 8.1.2.2 Load-based monitoring of wear

The loading of the components of machinery very much influences wear. In fact, doubling the load might increase the wear by an order of magnitude, i.e., making it ten times more severe. The loading of the production machinery can be either stable or varying in nature. If the load stays constant, the wear of the components can be monitored through monitoring the running hours, i.e., the time the machine has been used. When the loading of the machines varies the situation becomes more complex and both the loading and the time at a specific level of loading have to be recorded.

The higher loading of the machinery can, e.g., cause stress spikes which can start to form micro fractures in material after some time. Measuring and processing the raw data from these spikes into information can aid in estimating the developing condition [6]. The information extracted from the raw data can be treated like a process measurement such as a temperature reading. Extracted values do not include information about the severity, and, if the user does not know the meaning of these values, the measured information is of no value. This knowledge requirement can be solved by scaling the values so that the numbers are easily linked to the actual operating state. Nonlinear scaling is an effective tool for scaling these values to simple linguistic levels

[very low, low, normal, high, very high] correlating with scaled value range of [-2, 2]. Change in condition increases the vibration levels so that the scaling function does not correlate with the changed situation. The scaling function can be updated recursively according to the changed condition in order to keep it up to date [7]. The linguistic levels can be used together with the traffic light presentation indicating severity with color coding. Simple severity levels with understandable descriptions and color coding ensure that these scaled values can be inspected and understood by anyone. The value scaling can be done at the local calculation node or at the connective node collecting several maintenance measurements.



#### 8.1.2.3 Monitoring wear

Naturally, the most reliable way to define whether the machine needs maintenance is to rely on monitoring of wear as such. Unfortunately, the monitoring of wear can be technically very challenging and thus expensive. The reason for this is that the parts that wear are often moving and hidden by other parts or protective covers. Consequently, indirect methods such as vibration measurements are used to define the condition of the components of production machinery.

All of the machines with rotational or sequential operation generate vibrations according to their current state. These vibrations can be recorded and the data processed according to desired features about the machine condition or operation. Vibrations linked to certain aspect of machine operation are buried in the vibration signal with numerous other frequencies. Extracting the desired aspect of machine operation requires case-specific filtering and signal processing. Local real-time calculations can be used to extract information from the monitored vibrations without the need for demanding the transfer of the huge amount of raw vibration data to be processed elsewhere. Localised calculations can bring the extracted information to instantaneous use in automatic control or decision making. The long-term monitoring and tracking of cumulative stress gives valuable information for remaining useful life, RUL, calculations and prognostics, and short-term stress indices [8] indicate momentary high-stress periods, providing information about the causes of the wear and they can be used for control.

#### 8.1.2.4 Condition-based maintenance

Knowing the condition of the components enables the use of condition-based maintenance (CBM) strategy, which has been widely accepted as the most efficient and economical strategy for carrying out maintenance. It is not the purpose of this chapter to go into depth about the benefits of various maintenance strategies. However, one extremely important aspect needs to be brought up, i.e., the adverse influence of carrying out maintenance actions in vain. Maintenance cannot usually be carried out when the machine is running thus unnecessary maintenance causes loss of production. Another even much more

expensive result of unnecessary maintenance is the need to repair the consequences of previous maintenance actions in case something has gone wrong. From what has been explained here of carrying out maintenance in an efficient and economical way follows a great need for reliable measurements.

#### 8.1.2.5 Maintenance in mining industry

In the mining industry the wear of production equipment is usually very high. This is due to the nature of processes since they often include such subprocesses as crushing and transport of rock and chemical treatment, both of which cause high wear. Consequently, optimally organised maintenance can have a large influence on the availability and efficiency of the production equipment. As explained in the previous paragraph, it is commonly accepted that CBM is the best strategy in this kind of environment. However, the big challenge in the mining industry is that the production equipment can be scattered over a large area, making it difficult and expensive to use wired sensors. Also in mines the sensors are exposed to accidents, i.e., something can hit them, thus breaking them, or something can cut the wires. Thus the wireless Bluetooth Low Energy (BLE) System on Chip (SOC) and the new Micro Electro Mechanical System (MEMS) sensors that are low energy and relatively cheap can give a real boost to the use of condition monitoring in mining industry. The above explanations are the main reasons for choosing the mining industry as the pilot and testing the new technologies at Kemi Mine.

In the conveyor pilot the aim is to follow the development of wear of a wear plate. The wear of the wear plate reduces the thickness of the plate which in turn lowers the stiffness of the plate to a greater extent than the weight loss which in turn reduces the natural frequency of the plate which can be monitored through vibration measurements. The purpose of this kind of monitoring is to be able to predict when the wear plate has to be changed and make this information available through the Internet to the subcontractors that then make offers of new wear plates. The pilot case can be seen as a rather typical opportunity to follow CBM strategy. The components that suffer from wear are monitored on-line and the development can be followed by a number of parties such as the plant maintenance personnel and a number of subcontractors who can provide spare parts or maintenance services. A more detailed explanation of the techniques that have been used to pass the data will be presented in this chapter. However, it should be noted that the solution is generic so that it could be used for any kind of condition monitoring measurement.

### 8.1.3 Data structure

In order to be able to carry out maintenance using CBM strategy as described in the previous paragraph, it is necessary to use a Computerised Maintenance Management System (CMMS [9]) since otherwise it would not be possible to

keep track of planned and realised actions and the results of condition monitoring activities. There are a number of commercial CMMS packages available but most of them are oriented to the managing of maintenance work activities with limited support for condition monitoring and diagnosis activities that support CBM strategy. However, there exists an open solution for a plant description model and condition monitoring related data provided by MIMOSA ([10]).

#### 8.1.3.1 MIMOSA

MIMOSA defines itself as a not-for-profit trade association dedicated to developing and encouraging the adoption of open information standards for operations and maintenance in manufacturing, fleet, and facility environments. MIMOSA claims that its open standards enable collaborative asset life-cycle management in both commercial and military applications. It is not the purpose of this chapter to go to detail about what MIMOSA can offer, but further information can be obtained from the web pages of the MIMOSA association ([10]).

#### 8.1.3.2 Conveyor description

The relevant parts of the Kemi Mine conveyor have been defined in an SQL Server database using the MIMOSA Common Relational Information Schema (CRIS). It should be noted that this information schema would normally be used for the description of the whole plant but in this demonstration only the relevant parts of the conveyor and the components of the measuring system have been defined. In addition, all the measured condition monitoring and diagnosis data is saved in MIMOSA from which the developed service shows any chosen analysis values for any chosen assets (parts of production machinery).

### 8.1.4 Case-environment

To improve maintenance processes, new systems need to be developed that allow the transition to predictive maintenance based on condition monitoring data. Such systems consist of the following main components:

- Sensors
- Gateway servers
- Mediators
- Maintenance data stores
- Data-analysis services
- User interfaces

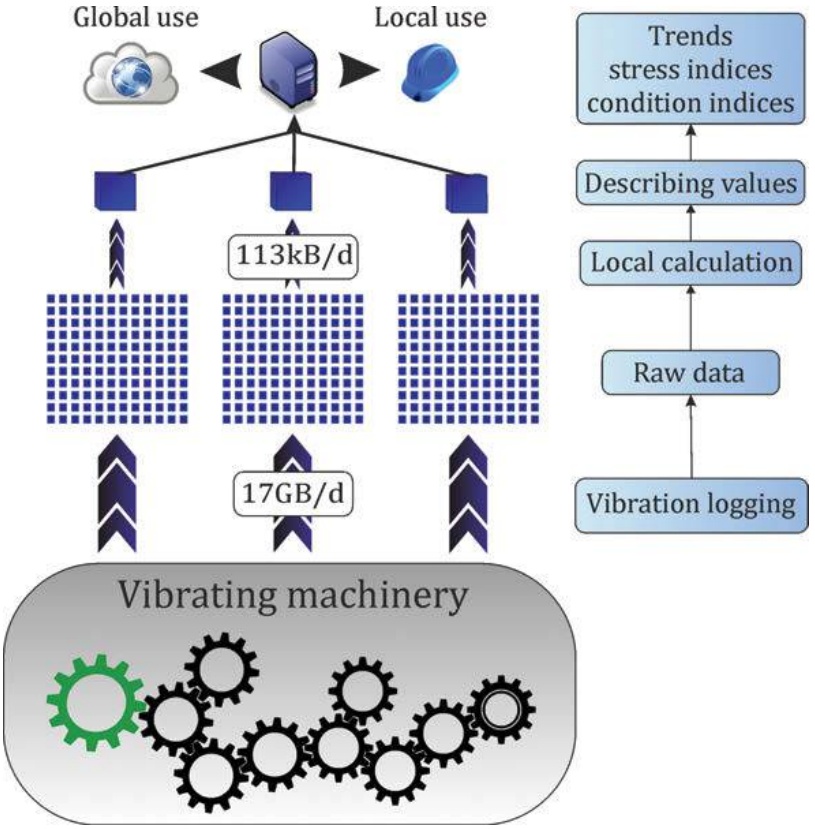


**FIGURE 8.1**
Local calculation for condition-based maintenance.

The recorded raw vibration data is needed for analysis purposes but a IoT solutions have no use for such dense data without connection to real-world events. Data collection should be planned according to possible future uses and balance between saving all of the data and discarding the data. Localised signal processing (see Figure 8.1) reduces the need for high-volume data transfer so that there can be a large number of nodes using the same transfer routes without exceeding the transfer capacity and the operation remains robust. Some applications may require the saving of the raw data from certain specific or exceptional events. In those cases there can be certain triggering for saving the raw data for some duration or until the end trigger occurs. Raw data can help in further analysis method development and historical data can act as a reference for future investigations.

### 8.1.5 Architecture

The Arrowhead Framework provides a ServiceRegistry (based on DNS-SD) that enables systems to find local services that they can consume and collaborate with. The service registry together with the other Arrowhead mechanisms of orchestration and security provide means for developing dynamic SOA applications that have a high potential for need-based scaling and configuration.

Systems providing services register an access point (e.g., REST API endpoint) automatically into the Arrowhead service registry. To keep the systems consistent it should also unregister when a service is shutting down. In the case of a crash or other unavailability it would be beneficial to use the time-to-live (TTL) parameter of the ServiceRegistry. This enables the removal of services not sending keep-alive messages during TTL. This will ensure that only services actually available are listed in the service registry.

Currently, for the applications presented in this chapter, the REST-based service registry interface is used for publishing and unpublishing services. This facilitates registry management and development because currently native DNS-SD tool support is limited in many modern runtime environments such as those based on C# and Node.js [11].

When a service instance goes online, it can first query for the REST service endpoint but currently this is provided as a parameter due to limited DNS-SD support in the platforms currently being used. The service registration is an HTTP POST request that includes the service name, type, domain, hostname, port as well as properties for version and path. For unregistering an HTTP POST to the unpublish path is performed with the name of the service as input.

To achieve dynamic behavior of service instances appearing and disappearing, a monitoring service type has been implemented that automatically removes monitored services if they disappear. It utilises the service discovery feature of the REST service registry to search for instances of specified service types. If a service instance stops responding for a given period of time, it is simply removed with the expectation that if it reappears the service will register itself. This means that when this monitoring service is also discoverable, services can use it to register themselves (or their type of services) to be monitored dynamically.

As gateways are connected and powered up, they will appear in the Arrowhead service registry. In the same way each Bluetooth sensor will connect to a gateway. The system scalability will be hierarchical as Arrowhead registry keeps gateway endpoints available and each gateway has own list of connected Bluetooth sensors. These sensors send data only when they have impact that will wake them up, otherwise sensors are in partial sleep mode and save battery. The gateway keeps all sensors and the last updated values in the buffer (cache). It can POST the latest data to the MIMOSA database (endpoint is provided by the Arrowhead Framework Orchestration system). The gateway

provides a local user interface with the same endpoints as it provides data out.

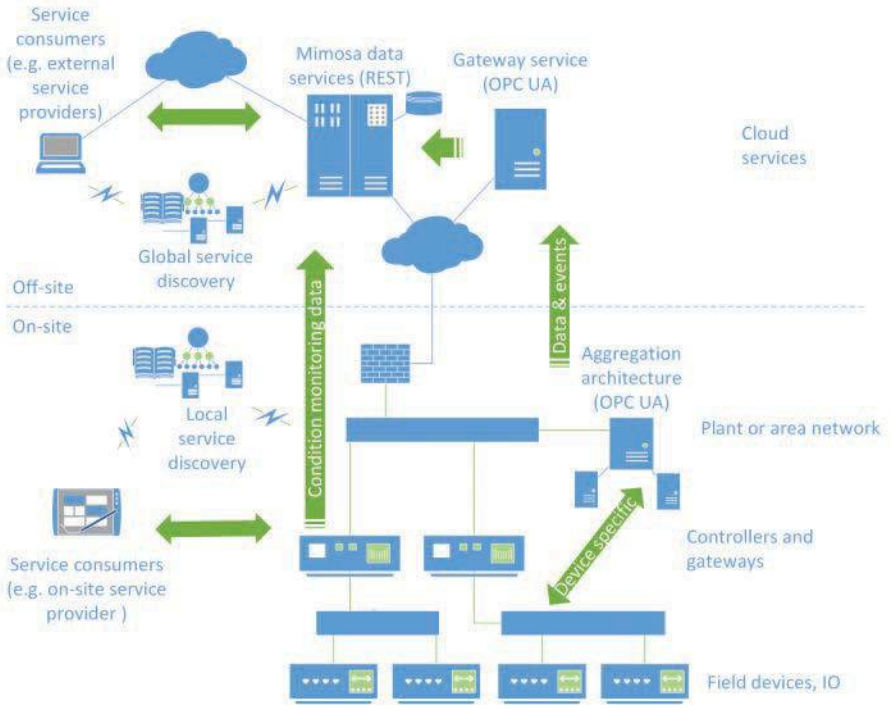


**FIGURE 8.2**
Overview of the general architecture.

#### 8.1.5.1 OPC UA - REST API aggregation

*OPC UA server interface*

The OPC-UA aggregate server interface provides access to field data using the platform independent standardised OPC-UA protocol. Clients connecting to the server can browse the address space and subscribe to data nodes from any OPC-UA server. The main benefit of the OPC-UA based interface is the standardisation of the communication and the possibility to discover the address space and use subscriptions. Compared to other protocols often used in the field, OPC-UA devices additionally offer enterprise-level security and reliability on the application level. The use of aggregating OPC-UA servers in a hierarchy discussed in [12].

At cloud level, the downside of using OPC UA from other cloud systems is the requirement for a full OPC UA communication protocol stack. Developing

an OPC UA based applications using only a protocol stack is quite a laborious task. Alternatively, commercially available software development kits and toolkits provide a more convenient starting point for a developer, but introduce additional costs. Such toolkits are available for various programming languages including Java and C#, common in cloud environments.

*RESTful interface*

The Representational State Transfer (REST) [13] is a style of software architecture for distributed systems. REST has emerged and established itself as a predominant Web service design model with key goals such as

- Scalability of component interactions
- Generality of interfaces
- Independent deployment of components
- Intermediary components to reduce latency, enforce security, and encapsulate legacy systems

An important concept in REST is the existence of resources (sources of specific information), each referenced through a global identifier (e.g., a URI in HTTP). In order to manipulate these resources, components of the network communicate via a standardised interface (e.g., HTTP) and exchange representations of these resources (the actual documents conveying the information). As a result, an application can interact with a resource by knowing two things: the identifier of the resource and the action required — it does not need to know whether there are caches, proxies, gateways, firewalls, tunnels, or anything else between it and the system actually holding the information. The application does, however, need to understand the format of the information (representation) returned, which is typically an HTML, XML, or JSON document of some kind, although it may be an image, plain text, or any other content.

In the aggregate server, cf. Figure 8.3, the REST interface provides access to the same data as the OPC-UA server interface but using a different protocol. Individual data nodes in the server are referenced based on a unique URI. The requested data is returned in a JSON or XML document. When the REST interface is used, the aggregate server operates as a bridge between these two technologies. The main benefit of using a REST-based interface is that there is no need for OPC UA SDK or Toolkit in order to access the data. Also, REST interfaces are very widely used Web technology, which is also convenient for the application developer. Therefore, a simple data access from a cloud application does not require significant development and the necessary tools and technologies are well supported in common programming languages such as C# and Java. One of the drawbacks of using a REST based interface is that the structuring of the resource URIs for the OPC UA data is not

standardised. Additionally, some may consider the lack of a publish-subscribe type of communication scheme as a negative aspect of REST interfaces since REST clients needs to poll the data from the server. However the new HTTP-5 standard will provide the publish-subscribe capability.

*Data model*

A possible data model meant to simplify the use of different server interfaces and generically and logically reflect various organisational hierarchies is presented in Figure 8.3. At the root level of this system data model visible through the REST API are enterprises (e.g., companies or customers). Each enterprise can have multiple sites or locations of interest (e.g., a factory, a ship, or a fleet). Each site can define multiple assets or resources that need to be monitored (e.g., a device, an engine, or a truck). An asset can include multiple data nodes, describing what is measured from the asset it belongs to (e.g., temperature, engine RPM). An asset can also contain other assets further extending the model by creating a tree structure of assets. At the bottom of this data model, under each data node there can be multiple process data values.

All requests to the API are made through HTTPS. With HTTPS, the HTTP protocol is protected from wiretapping and man-in-the-middle attacks, therefore data being transferred is secure. Authentication to the API will be via HTTP Basic with username and password pair and should be sent along all requests that require it. Different system user profiles can be used to define the system level access to the data requested through the REST API.

The aggregate server could also provide other interface in addition to the OPC UA server interface and REST interface. These interfaces also do not exclude each other, and a different interface may be used in different situations based on the requirements of the application that is consuming the data. Other interesting interfaces for the aggregate server could be, for instance, message queue based interfaces.

### 8.1.6 Plant monitoring system components

#### 8.1.6.1 Sensor and gateway devices

A Bluetooth low energy sensor is targeted to be low cost and energy efficient in such a way that it can be shipped with every wear part. It will work until the wear part has to be replaced. As the wear part is returned, the battery can be replaced and shipped again.

The sensor contains system on chip (SOC) that contains ARM Cortex-M0 CPU and BLE 4.0 Smart radio. The actual sensing component is an ST LIS3DH MEMS chip. These together are very low level energy consuming and energy consumption is optimised. The sensor will be active only when it gets impact that will wake it up. This threshold level can be configured locally or


**FIGURE 8.3**
Aggregated data model.

from the cloud. The actual measurement contains raw values. The sensor will broadcast values and the gateway will store them and run FFT analysis.

#### 8.1.6.2 Gateway server API

The first devices used were Raspberry B and BeagleBone Black; later versions of the gateway were tested with Intel Edison. All these are cheap and able to run Linux and small application (not enterprise size, main limitation is amount of memory). Each of these is used in many open source IoT examples and the same idea we had in using the commercial ready hardware as a gateway. Intel Edison with WiFi and Bluetooth connectivity was perfect a match for our requirements. It has a pre-built image with all software needed to run the applications.

The latest prototype fulfills basic measurement data storing locally and posting it to the MIMOSA database. It also has features to test the database connection and fetch stored data to the local user interface. The gateway application is based on Node.js and express web server. The management and orchestration were built with Node-RED [14]. It is a visual programming environment and contains a lot of ready nodes to implement application flows. The application connects to the gateway (GTW) and uses REST API to read sensors from the GTW. Each sensor can be configured from this user interface. As the sensor sends new measurement data, the application will get the data and then send it to the MIMOSA service and database. These communication

messages are all based on REST APIs. The gateway provides all messages with Swagger documented and these can be browsed and tested through a web browser.

The last prototype will contain Arrowhead service registry binding. See Figure 8.4 for a small example of how the Arrowhead REST API can be used to list services and the service types and register a new service endpoint into the registry.

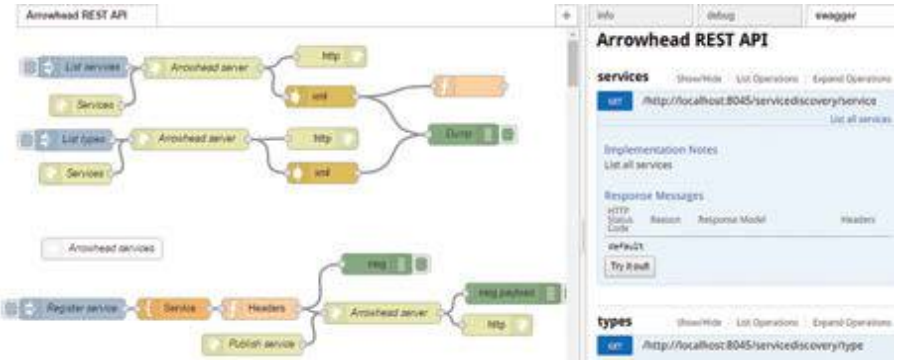


**FIGURE 8.4**
Node-RED applications contain REST API endpoints. At the same time the panel on the right shows and documents used operations. Users can test each operation immediately with the "Try it out!" button that Swagger provides if the API is documented in Swagger.

#### 8.1.6.3 Mediators

The mediator pattern is used to work as a bridge between different protocols. The GTW runs Bluetooth LE protocol to get information from the sensors and then HTTPS REST API to send measurement data to the MIMOSA database. This separates low-energy sensors from the continuous connection to on-demand short time connections. As the GTW can hold (cache) data, it again reduces energy usage of the sensor.

The mediator pattern can be used with other protocols like CoAP to connect devices to the cloud. Even old legacy systems can be connected in this way. The vendor has to provide the node that can connect to the old system and give data in payload so that it can be formatted to the other protocol message format. In Node-RED this is easy to implement and to connect different systems with different protocols.

### 8.1.7 Results and further development

There are currently still some technical issues. The automatic configuration capabilities need to be enhanced. There has to be automatic service discovery via REST API. This is currently in beta-phase with the first integrated prototype now being implemented. Another foreseen modification need is the automatic white-listing process for sensor devices. This will make the sensor discovery faster. Related to this there is a need to expand to a more complete version and stress test the system with a larger group of active sensors.

In the last phase full integration with the Arrowhead Framework should take place. The authentication and orchestration services have not been used in the earlier prototypes.

The actual business potential will be seen in the future: what does the service framework bring? Does it facilitate engineering? Does it promote open systems enabling connectivity for new applications and services? Currently it can be used to connect services within and, to some extent, between clouds.



### 8.1.8 Discussion

When the Arrowhead project was started, there were a lot of ideas (too many actually) that have now been finalized into a vision of the final solution. There is a need for a dynamic system capable of devices and systems entering and exiting the service environment, i.e., those producing, storing and refining maintenance related data. For the end user there should be a user interface that can be used to configure sensors at the customer site.

User interfaces need to be browser based. In this way users can securely access any sensor at any site (globally, any cloud). Users can also browse from the same user interface available gateways and sensors. There is a default database at the gateway that is automatically selected from the available ones (discovered from the service registry). So the gateway is autonomous and it has registered itself into the service registry. This requires some metadata, because there can be multiple data storage services registered in the service registry. For example, a data service instance can be customer specific and can contain default parameters for each sensor. As the sensor is replaced, the gateway can reconfigure a sensor with the default parameters. This way systems run automatically with self-configuring sensors and gateways.

#### 8.1.8.1 Application level issues

As already noted, the whole system scalability is not yet fully known since there are not enough sensors to run 50–100 sensors with one gateway. The gateway testing is not yet final as there is a need to stress test the system with several hundreds of gateways. Future work for expandability will cover integration with existing legacy systems. There are several standard protocols that should be integrated: OPC-UA, MODBUS-TCP, MQTT, etc. For the advanced, value creating services there should be analyzer services for end

customer/service providers. These will need a notification service that can alert/indicate with loose interface like email or REST API to CMMS/ERP interfaces, e.g., create maintenance work orders. Alternatives to DNS-SD should be considered. For example multi-cast DNS-SD distributes information in the local network and contains local cache, better performance, security to be considered, and is more robust. An especially good generalisable and reusable part is the mediator pattern (no compile needed, Javascript).

### 8.1.9 Conclusion

In this section the features that have been developed and tested in the mining industry pilot have been presented. As such, mines are very demanding environments with significant potential for improvement of availability and efficiency. This environment has provided a good platform for widening the use of the Arrowhead Framework, linking it to a number of technologies and systems. All the tests between separate systems have been successful in the selected scale. The testing that has been carried out has taken place on such a small scale that it cannot be used as proof of a working solution that could be taken into wide use. Because the small-scale case testing, in some cases with very expensive equipment, has been successful, it is clear that the work should continue on a wider scale. As the Arrowhead Framework provides local - enterprise - cloud service registry, it can be used as a "yellow book" of IoT services. Mediators are needed to build protocol gateways that can provide interoperability between different systems and technologies.

As large and different systems can be very challenging to maintain, the system documentation should be based on online up-to-date descriptions. In our case we used Swagger v2.0 to show available endpoints and parameters of the provided REST API interfaces.

The Industrial Internet will actively grow in the next years, and the principles and solutions provided and tested in the Arrowhead Framework will be needed and used.

The Arrowhead Framework can be utilised to provide measurement data for third-party maintenance service providers and to make the information about comparable components easily accessible in a standardised manner.

## 8.2 Aircraft maintenance system

### 8.2.1 Introduction

Aviation is at the moment a fast growing market foreseen to double in size in the next decade, and it has lately seen a lot of changes with the arrival of new challengers with new business models leading to a great turmoil among legacy

airlines and a higher level of competition. In this context, keeping aircraft flying is key for airlines as every unavailability can mean losing a lot of money, and aircraft maintenance is a noticeable contributor to aircraft downtime. Therefore, having better control of aircraft needs for maintenance and being able to react efficiently following a technical event is essential for airlines. That is the reason why Airbus, as a leading aircraft manufacturer, is continuously working at improving its products and at providing innovative and disruptive added value solutions for aircraft maintenance to support airlines in their day to day operations. The IoT concepts and their standardisation offers new perspectives in the way to approach system architecture and design that can help solve common issues in complex systems such as the aircraft maintenance system, and the Arrowhead framework offers to make it simple and effective.



### 8.2.2 Aircraft maintenance

#### 8.2.2.1 Aircraft maintenance in a nutshell

Aircraft maintenance is the process of ensuring that an aircraft will be available and safe to operate its flights. In general, aircraft maintenance comprises two types of activities: scheduled maintenance activities and unscheduled maintenance activities.

Scheduled maintenance activities regroup a set of tasks (inspections, part replacements, tests, etc.) defined by aircraft manufacturers and validated by the aviation authorities that shall be performed by the airline to ensure aircraft continuous airworthiness. These tasks shall be planned by the airlines to ensure that they are performed on each aircraft in a given amount of time (calendar time, number of flight hours or flight cycles – a flight cycle being a take-off and a landing).

One example of scheduled maintenance activity can be the "daily checks" which consist of some light operations such as fluid levels control and visual inspections and can be performed when the aircraft is at the line station. There are also checks like "C checks" which consist of performing some thorough inspections and running some operational and functional tests on some components. Checks can require the aircraft to be immobilized for a few days or weeks in a hangar, which represents a prominent amount of time of unavailability of the aircraft for flight operations.

Unscheduled maintenance activities are maintenance activities that shall be performed specifically following the occurrence of an unexpected event in order to return the aircraft to safe operation. An unexpected event can be anything from a sensor or computer that failed, to a vehicle bumping the aircraft or the aircraft being struck by thunder, for instance. According to the impact of the defect on the aircraft, the aircraft operations, and the time available for maintenance, an unscheduled maintenance activity can consist of simply deferring the repair by setting the failed component into an inoperative state or making an inspection, or may require repairing or replacing a component

or a structural part of the aircraft, which can have a huge impact on aircraft operations with flight delays or cancellations and everything that can result from such situation (passenger dissatisfaction, additional costs, etc.).

#### 8.2.2.2 Challenges

Aircraft maintenance in general still offers a lot of challenges. On the side of scheduled maintenance, the main inquiries are principally related to components' remaining useful life estimation, firstly to optimise scheduled maintenance planning, but also to try to move these maintenance activities toward condition based maintenance and make them therefore more transparent for operations by avoiding long immobilizations of the aircraft.

Unscheduled maintenance is still a big issue which is difficult to avoid, if not impossible, given the complexity of the aircraft themselves and of their operational environment. Therefore airlines, either by themselves or with the support of maintenance release & overhaul (MRO) organisations, have to set up an intricate and costly maintenance environment to make sure that, in the case of an unscheduled event, they will be able to quickly take action and ensure minimum disruption of operations, and this wherever their aircraft operate. That means that they may have to manage different main bases for heavy maintenance activities and outstations capable of handling most common unscheduled maintenance situations. This is in addition to spare parts provisioning, and management of tools, equipment, and skilled people required for the maintenance execution.

### 8.2.3 Managing unscheduled events

By definition, an unscheduled event can occur at any time, whether the aircraft is flying or not. While in-flight, aircraft core systems and functions are monitored by the flight warning system, which notifies the pilots whenever a malfunction has been detected. Additionally the pilots themselves can sense or notice something unfamiliar and require further analysis, or people on the ground can notice something unexpected while working around the aircraft. Based on this information, the different stages in the decision process involve assessing the impact of the defect on the aircraft operations and deciding on the maintenance to be performed in regard to operational and financial impacts.

#### 8.2.3.1 Operational impact assessment

Following the detection of a defect, cf. Figure 8.5), pilots and airlines' operation controllers have first to determine what the impact of the event is on aircraft operations.

In the case of any critical event detected when the aircraft is flying, passengers and crew safety being the primary concern, the aircraft will be landed at the closest available airport compatible with the current aircraft state with-

out any other considerations. If the event is not critical, then the outcome will depend on how the aircraft design can bear such a defect. This information is available in the aircraft minimum equipment list (MEL) and component deviation list (CDL) documents, which provide the list of equipment and components identified by the aircraft manufacturer as being required for a safe flight, and how failure of one of these elements affects this capability.

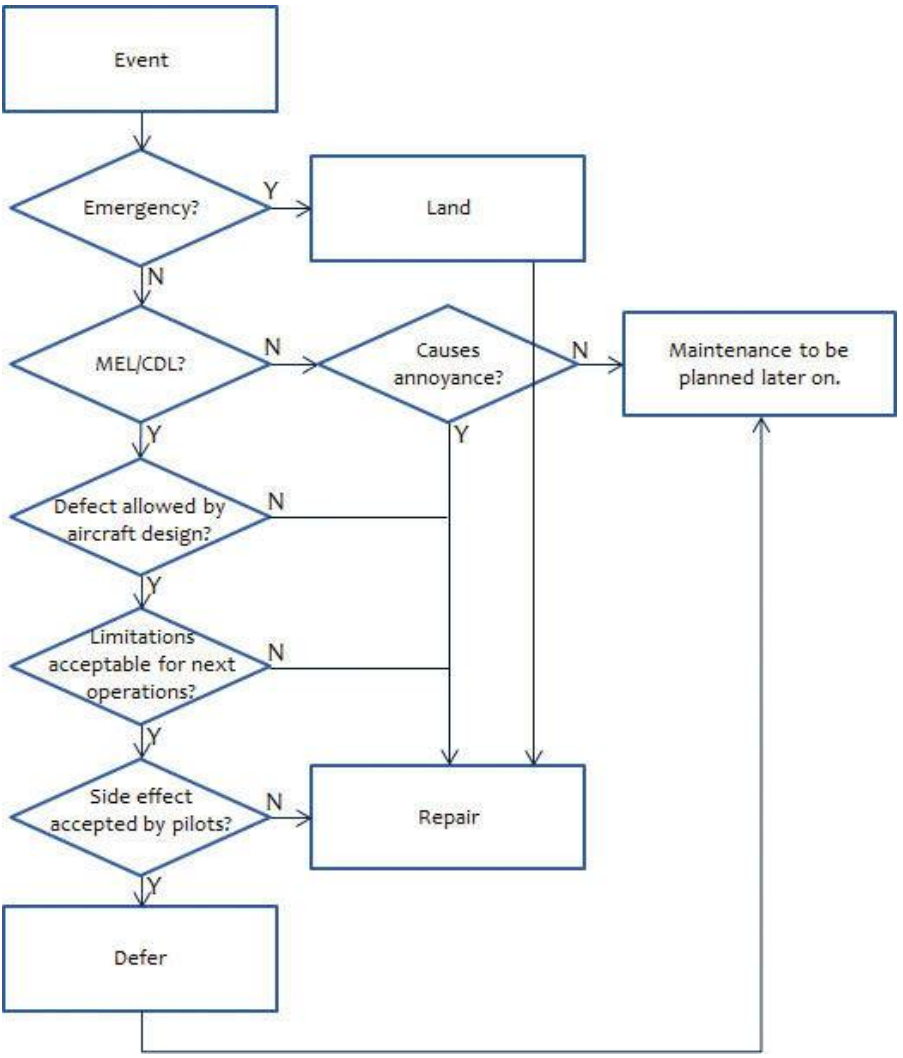


**FIGURE 8.5**
Event processing.

If the defect is stated in the MEL or CDL, then the decision will depend on

- Aircraft design accepting the loss of the component,
- Induced operational constraints that may affect the possibility for the aircraft to operate its next flights (for instance, if a flight level limitation applies whereas the aircraft will have to cross mountains),
- Induced additional burden for the pilots (having to periodically perform some visual inspections on some information to compensate the absence of a piece of equipment, for example) which the pilots may accept or not.

Additionally there may also exist some secondary technical events which may not be stated in the MEL or CDL but which can lead to some annoyance for the passengers or the flight crew (a flashing light, etc.) and that may also need to be repaired.

#### 8.2.3.2 Maintenance activities identification

When the assessment of the operational impacts is available, the maintenance controllers or operators shall then determine what the requirements are at the maintenance level and how to manage them. According to the defect's impact and as stated earlier, in this context maintenance activity can either consist of deferring the defect or making a repair or replacement if the defect has an impact, or doing nothing and planning the repair later on if the defect is secondary. Deferring the repair is generally the quickest solution as it basically consists of setting the component in an inoperative state by securing it (locking a valve open for instance) or switching off its power supply. The repair or replacement activity may be more complicated, as it may require first isolating the root cause of the problem, which may be the component or equipment itself or another component or equipment interfaced with it such as wiring or a sensor, for instance, before being able to perform the repair or replacement. Additionally some tasks may involve specifically qualified maintenance personnel, as well as appropriate tools or ground supporting equipment (ladders, etc.), and potentially also spare parts or material.

#### 8.2.3.3 Operational decision

Once the maintenance and operations controls has a clear view of the situation, they have to balance different options, taking into account the disturbances caused to passengers and other flights (delays, connections, cancellations,etc.) and the related financial impacts. For instance, they can decide to perform maintenance at the destination airport and transport required spare parts or missing equipment there if they are not available using another flight. They could also decide to move the aircraft to a location where it would be more appropriate to perform the maintenance. And to lessen impact on passengers,

they could bring in a spare aircraft to continue operations, or swap the aircraft with another one where both will fit the next flights' operational requirements.

#### 8.2.3.4 Maintenance preparation and execution

Finally, when the decision has been made, the maintenance personnel can prepare the required resources and logistical aspects so that maintenance can be performed with the least delays. Then once the aircraft has arrived, the maintenance activities – inspections, trouble shooting, repairs or replacements, software downloads, tests, etc. – can be started. These operations can be quite straightforward with simply pushing a button, or complicated with the need to access narrow areas, dismount panels to look for some components, access the cockpit to perform some tests, etc.

After completion, the maintenance operator reports the activity in the aircraft technical logs and a certificate of release to service is issued for the aircraft after a control of the operation.

### 8.2.4 Introduction to the aircraft maintenance system

The whole process described in the previous section, from the operational assessment to the maintenance decision, preparation, and execution, is generally highly time constrained with a lot of information to manage. And even though it can be thrilling to face such a situation and work to find out a solution, it also creates a lot of stress, as large amounts of money can be at stake. The purpose of the aircraft maintenance system is to help relieving this stress by providing new tools and new ways of working build around connectivity between the different parties and information digitalisation.

The aircraft maintenance system will allow people to share an overview of the operational and technical situation as well as the related operational impacts by integrating information from aircraft, flight operations, and maintenance systems with digitalised aircraft documentation. It will also provide support in maintenance preparation and maintenance execution by enabling troubleshooting activities, before the aircraft has landed, offering advanced connectivity with the aircraft during maintenance execution, and even automating some maintenance activities.

### 8.2.5 Aircraft maintenance system — System of Systems

#### 8.2.5.1 Overview

The aircraft maintenance system architecture is built around two main concepts: the IoT, with the Arrowhead Framework, and service-oriented architecture (SOA). The IoT technology is used primarily to integrate aircraft, maintenance devices and the aircraft maintenance system. SOA will be used to interconnect the aircraft maintenance system with the other information systems involved in the flight operations and maintenance decision processes.

The aircraft maintenance system's System of Systems topology is depicted in Figure 8.6.

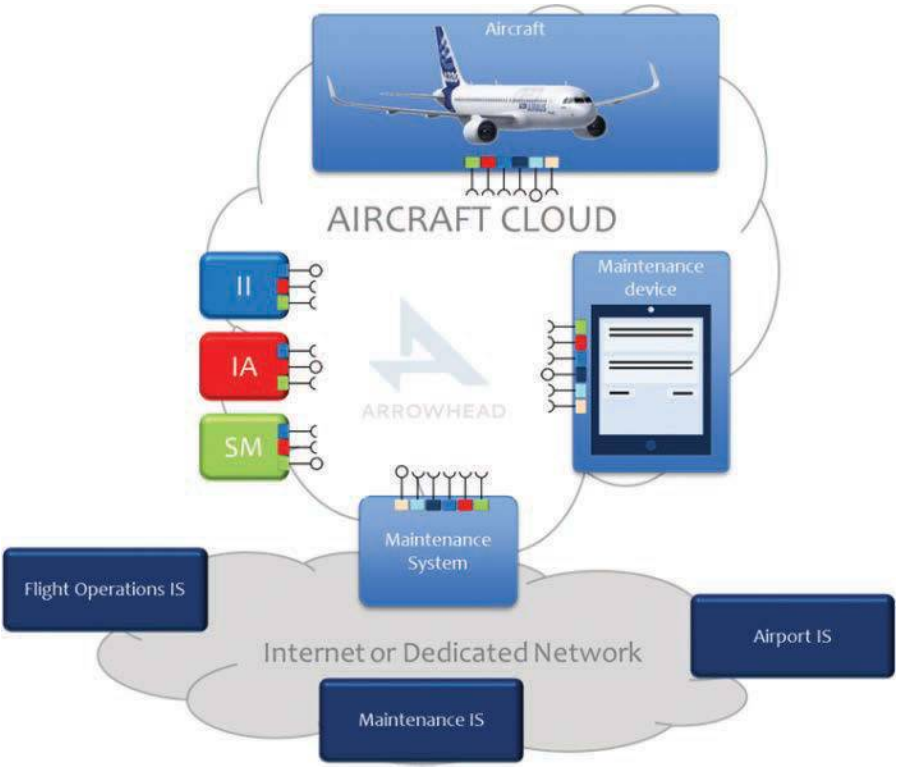


**FIGURE 8.6**
Aircraft maintenance system overview.

#### 8.2.5.2 The Arrowhead aircraft cloud

The aircraft maintenance system is connected to a dedicated Arrowhead local cloud - that is a network with a dedicated Arrowhead Framework core services instance - altogether with aircraft and maintenance devices. This Arrowhead aircraft cloud allows the publication and use of services within a secure environment, restricted either to one airline and its partners, or to a set of airlines and maintenance organisations. The use of the Arrowhead Framework provides transparent integration of new aircraft and devices into the system, making it scalable, as well as built-in security and authorization management.

The different resources within the Arrowhead aircraft cloud instance follow a specific naming convention based on airline ownership, system category (aircraft, mobile device, etc.), and system identifier. This principle enables the

setup of multi-tenancy mechanisms to support confidentiality and privacy of information and services between the different organisations sharing the cloud instance, and it also provides means to perform scale operations on a set of systems.

As an example, for a mobile device with identifier K43 managed by an airline which identifier is ABC, services will be named such as "ABC.MD.K43_notifyWorkOrder_ws_https._tcp." It is possible to look-up all the resources pertaining to the airline by requesting all the services prefixed by "ABC." to the ServiceDiscovery core service, or only the maintenance devices by requesting all the services prefixed by "ABC.MD.".

A similar principle is applied to the declaration of services by the aircraft maintenance system, supporting the management and isolation of the services contracted by the airline.

In addition the aircraft cloud makes use of certificates in combination with the Arrowhead Authorization core service allowing proper authentication of the different elements and access restriction to services.

#### 8.2.5.3 Information system integration

On the other end, the aircraft maintenance system will be interfaced with other information systems (IS) providing structuring data and services using a standard SOA approach:

- The flight operations IS provides information regarding crew roster and flight schedules
- The maintenance IS provides services to look-up maintenance personnel skills and roster; hangar facilities, spare parts, tools and ground support equipment availability; and to allow booking these elements
- The airport IS provides information on airport facilities (gates, line stations, etc.) and resources (ground supporting equipments, etc.) availability.

#### 8.2.5.4 Aircraft Maintenance System services overview

The airline maintenance system bridges the Arrowhead aircraft cloud to the airlines and partnering companies' information system and builds on these two environments to provide services to the end users.

Some examples of services that will be provided are

- "Dispatch impact assessment," which provides quick identification of the impacts and limitations induced on an aircraft by a defect and can be extended to provide information on impacts of the defect on the airlines' flight schedule
- "In-flight troubleshooting," allowing ground maintenance operators to start requesting information from the aircraft to get some insight on what

lies behind a reported event, and thus allowing better preparation for the maintenance activities to be realized and saving time for their execution

- "Remote maintenance," which allows remotely performing some maintenance activities, with the possibility of automating some of them, and therefore can avoid flight delays or cancellation when no maintenance personnel are available at the destination to perform a deferral for instance
- "Mobile maintenance," allowing a mobile device to be interfaced with the aircraft and therefore simplifying interactions with the aircraft systems to perform some maintenance operations



### 8.2.6 Arrowhead aircraft cloud systems and services insight

As mentioned earlier, the Arrowhead aircraft cloud environment allows federating services for three categories of systems: aircraft, maintenance devices, and the aircraft maintenance system.

The following sections provide the definition of these systems and give an insight into some of the services they publish.

#### 8.2.6.1 Aircraft system

The aircraft system basically represents an aircraft in itself. It provides an abstraction of the complexity of the aircraft by offering centralized and integrating access to information and services supplied on-board by the different equipment and components of the aircraft. Table 8.1 presents a sample of the services provided by the aircraft system.

#### 8.2.6.2 Aircraft Maintenance System system

The aircraft maintenance system represents the gateway between the aircraft maintenance system and the Arrowhead aircraft cloud. The purpose of this system is to allow collection of data from aircraft and maintenance devices connected to the Arrowhead aircraft cloud, as well as enabling the publication of information toward these systems, or the requesting of some information. Table 8.2 presents some of the services provided by the aircraft maintenance system.

#### 8.2.6.3 MaintenanceDevice system

The MaintenanceDevice system represents the device and applications that will be used by maintenance operators as a support for maintenance preparation, execution, and reporting. It consumes services provided by both the aircraft system and the aircraft maintenance system to render the different functions provided to the user such as the interactive job cab, remote support or the system pages. Table 8.3 presents some of the services provided by the MaintenanceDevice system.

**TABLE 8.1**
Aircraft services.

| Service | Description |
|---|---|
| connect(<credentials>) | Allows a client system to request access to the aircraft services. Call to this service is compulsory for a client system to be authorised to use other aircraft services. |
| disconnect() | Revokes the access rights granted to the client system. |
| getConfiguration() | Returns the configuration of the aircraft. |
| runTest(TID,TestParams) | Runs a test on the aircraft, providing the given parameters, and returns the result of the test execution. |
| runCommand(CID,CmdParams) | Requests the aircraft to run the command identified by the provided command identifier with the provided parameters. |
| getParameters(P1,. . . ,Pn) | Returns the current values for the list of given aircraft parameters. |
| recordParameters(P1,. . . ,Pn,dur) | Requests the aircraft to record some parameters for a given duration. Returns the recording identifier and starts the recording of the parameters. |
| getRecording(RID) | Requests the recording corresponding to the given recording identifier. If the recording is still in progress, progress status information only will be provided. |

**TABLE 8.2**
Aircraft Maintenance system services

| Service | Description |
|---|---|
| notifyEvent(ACID, EvtData) | Allows a aircraft to notifying an event to the system. |
| notifyMaintenancePhase(ACID, Location) | Allows an aircraft to indicate it has entered the phase where maintenance is allowed, and confirms its location. The aircraft maintenance system can then notify maintenance devices they can start connecting to the aircraft. |
| registerDevice(DID, Location) | Allows a maintenance device to register itself into the system with information on its location. |
| isDeviceRegistered(DID) | Allows determining if a maintenance device is registered into the system. This is used by the aircraft to accept maintenance device connection. |
| unregisterDevice(DID) | Allows a maintenance device to unregister from the system. |
| isUserAuthorised(UID) | Allows determining if a user is registered in the system and can access the aircraft. |

**TABLE 8.3**
Maintenance Device services

| Service | Description |
|---|---|
| notifyWorkOrder(location, WO) | Allows the aircraft maintenance system to forward a work order to an aircraft mechanics located at a specific location. |
| getUserInfo() | Returns information on the user currently connected on the device. |
| notifyMaintenancePhase(ACID, location) | Allows indicating that an aircraft has entered the phase where maintenance is allowed, and confirms its location. |

### 8.2.7 Use cases

The following section detail some use cases displaying the way the different systems operate within the Arrowhead aircraft cloud.

#### 8.2.7.1 Automated maintenance scenario

Automated maintenance is a specific case of remote maintenance backed up by an automated procedure. It can be used to defer the treatment of a defect, for instance cf. Figure 8.7).

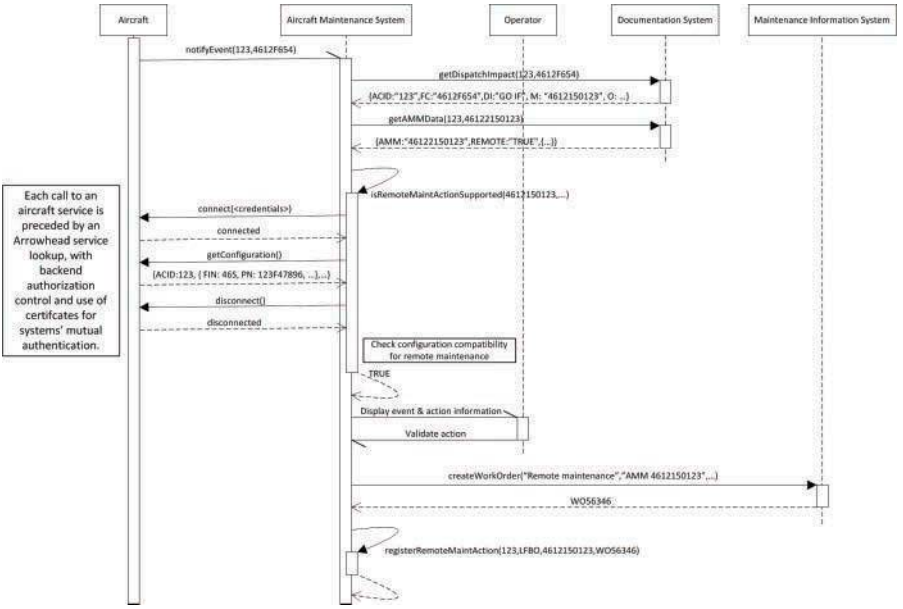


**FIGURE 8.7**
Automated maintenance decision sequence diagram.

After the aircraft notified the aircraft maintenance system, of the defect, the latter determines the related dispatch impact by referring to the aircraft MEL. The service returns information on the dispatch impact - GO IF in our example, which means that the aircraft can be dispatched under certain conditions - as well as the requirements for the aircraft dispatch - here a maintenance action to defer the repair by making the failed equipment inoperative.

The aircraft maintenance system then launches the process to determine if the action can be performed remotely. It connects to the aircraft and requests the aircraft configuration in order to validate that the versions of the software and hardware present on the aircraft are in line with the prerequisites for the remote maintenance action to be performed. If this is validated, then the aircraft maintenance system will notify maintenance control of the

event and propose the execution of an automated remote maintenance action. Once validated by the user, the maintenance action will be planned for when the aircraft arrives, and recorded into the airline's maintenance information system.

When the aircraft has arrived at its destination and is ready for maintenance, it will notify the aircraft maintenance system (refer to Figure 8.8). This notification received, the aircraft maintenance system will again connect and authenticate to the aircraft and start the maintenance procedure: First it will confirm the presence of the failure by running a confirmation test. Then, the failure being confirmed, it will request the circuit breaker that powers the failed equipment to be opened by running the specific command. And finally, a new test is run to validate that the equipment has effectively been put into an inoperative state. This achieved, the aircraft maintenance system will record the maintenance action into the airline maintenance information system and into the aircraft's electronic logbook. Finally, the aircraft maintenance system will announce that the aircraft can be released to service and the aircraft will be able to depart once ground operations are finished.

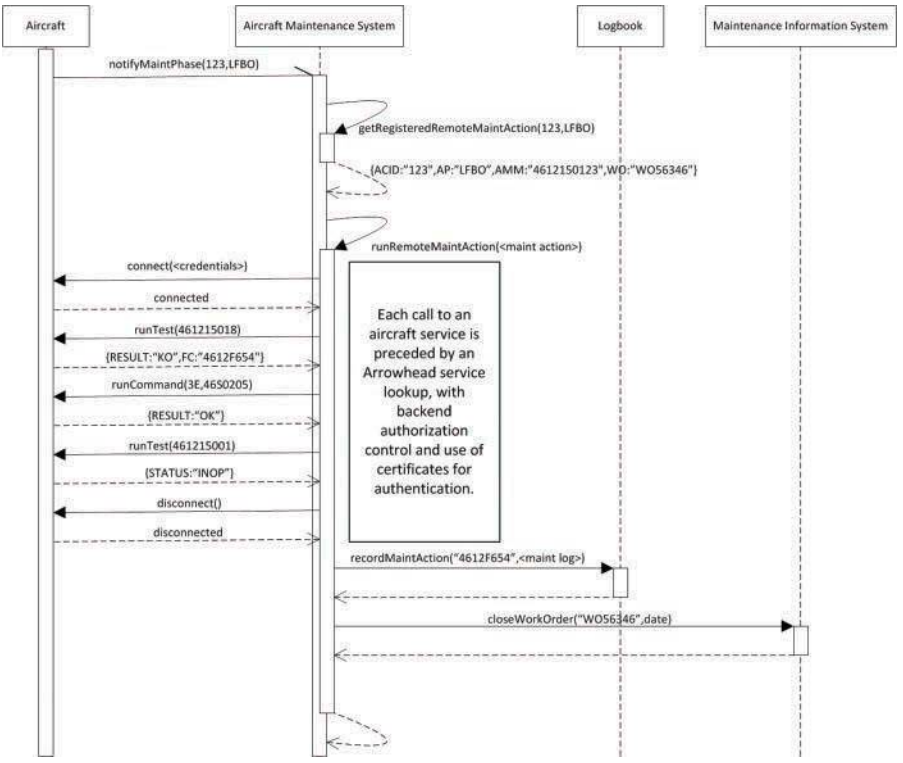


**FIGURE 8.8**
Aircraft automated remote maintenance execution sequence diagram.

#### 8.2.7.2 Mobile maintenance scenario

Mobile maintenance devices provide digital support to the maintenance operator, allowing the interaction with both the aircraft maintenance system and the aircraft, therefore streamlining exchanges between the maintenance control and the operator on the one side, and on the other side making the operator more efficient by limiting needs to go here and there around the aircraft to access some specific functions or information. Additionally, the mobile maintenance device can provide some advanced interactivity with the aircraft that is not discussed in this scenario.

Whereas the operator connects to the mobile device, the device registers to the aircraft maintenance system together with user information and its location, and becomes visible from maintenance control. Following the detection of a defect on an aircraft and the decision to perform some maintenance to repair the defect, maintenance control will send an electronic work order to the maintenance operator. The work order provides the pre-requisites and description of the activity to be performed. Before the aircraft lands, the operator can prepare for maintenance by reviewing the documentation for the different tasks to be realised and by getting the resources and material needed for the operation.

When the aircraft arrives and is ready to accept maintenance operations, it notifies the aircraft maintenance system that it entered the maintenance phase, thus authorising mobile devices to connect to the aircraft (cf. Figure 8.9). The operator at that point requires access to the aircraft, which will validate that the device and the user are authorised to perform this action by referring to the aircraft maintenance system, and once validated the operator gets access to the aircraft. At that instant the device requests the aircraft configuration to validate which functionality can be enabled or not according to the aircraft's software and hardware versions. The operator then starts the trouble shooting activity with a fault confirmation test to validate that the defect is still present. The fault being confirmed, the operator will be guided step by step in the trouble shooting process by the system after the prerequisites of each step have been validated. In the example above, for instance, after having checked that the computer providing the failed function is working correctly, it will check that the fluid supply has been cut off before allowing the user to perform the next task. Once the fluid supply cut off, it will provide the operator with instructions to test a new valve in place of the current one, which will prove to solve the issue and indicate that the previous valve was the root cause of the defect. Then the operator performs a test to validate that everything is in order and can close the work order and submit valve replacement information to the aircraft's electronic logbook. The maintenance activity being done, the release to service of the aircraft can be pronounced and the aircraft will be able to depart when ground operations are completed.

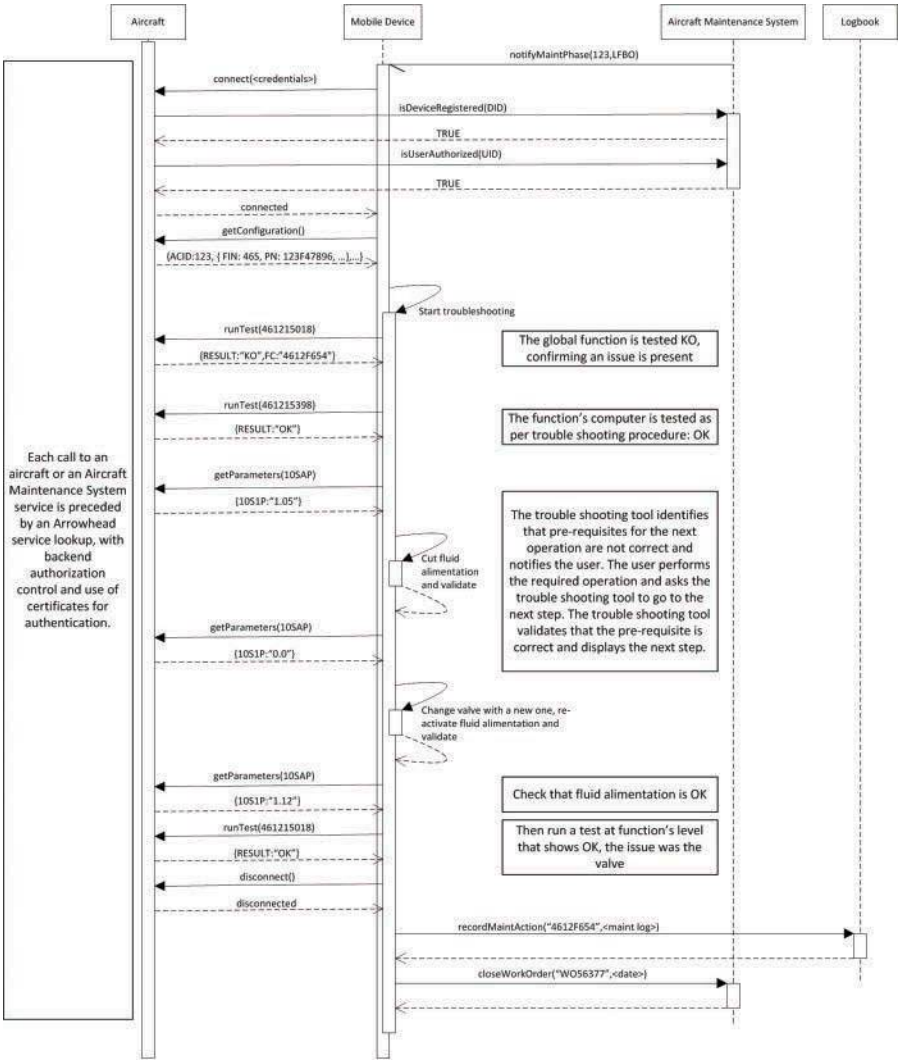


**FIGURE 8.9**
Mobile maintenance execution sequence diagram.

### 8.2.8 Conclusion

IoT principles and the Arrowhead Framework provide the means to easily set up scalable and secure systems such as the aircraft maintenance system requires to provide high-quality services in the face of a rapidly growing fleet of aircraft and higher customer expectations. The Arrowhead Framework offers the key components that make deployment and integration of systems such as new aircraft or maintenance devices within the aircraft maintenance system transparent, and enable focus on functions and information exchanges in an area where digitalisation has become the cornerstone of efficiency.



## 8.3 Glossary

**CMMS:** Computerised Maintenance Management System

**DB:** Database

**DNS:** Domain Name System

**ERP:** Enterprise Resource Planning

**mDNS:** Multicast Domain Name System

**MEMS:** Microelectromechanical systems

**MIMOSA:** Machinery Information Management Open Systems Alliance

**MODBUS-TCP:** Modbus RTU protocol with a TCP interface

**MQTT:** Message Queuing Telemetry Transport (protocol)

**REST API:** Representational State Transfer service interface

**SWAGGER:** Open API Initiative specification

**OPC UA:** OPC Unified Architecture (protocol and information model)

**XML:** Extensible Markup Language

## Bibliography

[1] J. Laurila, A. Koistinen, E. K. Juuso, and T. Liedes, "Monitoring of a rod mill using advanced feature extraction methods," in *12th International Conference on Condition Monitoring and Machinery Failure Prevention Technologies 2015, CM 2015 and MFPT 2015*, 2015, pp. 580–590.

[2] A. Mathew, K. Bever, M. Purser, and L. Ma, "Bringing the MIMOSA OSA-EAI into an object-oriented world," in *Engineering Asset Management and Infrastructure Sustainability.* Springer Science + Business Media, 2012, pp. 633–646. [Online]. Available: http://dx.doi.org/10.1007/978-0-85729-493-7_49

[3] "OPC unified architecture (OPC UA) new opportunities of system integration and information modelling in automation systems," in *9th IEEE International Conference on Industrial Informatics, INDIN 2011.* Institute of Electrical & Electronics Engineers (IEEE), July 2011. [Online]. Available: http://dx.doi.org/10.1109/indin.2011.6035016

[4] A. Koistinen, "On-site calculations of generalised norms for maintenance and operational monitoring," in *Maintenance, Condition Monitoring and Diagnostics & Maintenance Performance Measurement and Management, MCMD 2015 and MPMM 2015*, 2015, pp. 104–109.

[5] D. Hästbacka, P. Aarnio, V. Vyatkin, and S. Kuikka, "Empowering industrial maintenance personnel with situationally relevant information using semantics and context reasoning," in *Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, 2015, pp. 182–192.

[6] E. K. Juuso and M. Ruusunen, "Fatigue prediction with intelligent stress indices based on torque measurements in a rolling mill," in *10th International Conference on Condition Monitoring and Machinery Failure Prevention Technologies 2013, CM 2013 and MFPT 2013*, vol. 1, 2013, pp. 460–471.

[7] E. K. Juuso, "Recursive data analysis and modelling in prognostics," in *12th International Conference on Condition Monitoring and Machinery Failure Prevention Technologies 2015, CM 2015 and MFPT 2015*, 2015, pp. 560–567.

[8] E. K. Juuso and D. Galar, *Current Trends in Reliability, Availability, Maintainability and Safety: An Industry Perspective.* Cham: Springer International Publishing, 2016, ch. Intelligent Real-Time Risk Analysis for Machines and Process Devices, pp. 229–240. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-23597-4_17

[9] "The CMMS Benchmarking System," in *Reliable Maintenance Planning Estimating, and Scheduling.* Elsevier BV, 2015, pp. 439–452. [Online]. Available: http://dx.doi.org/10.1016/b978-0-12-397042-8.15006-5

[10] MIMOSA. Mimosa organization. [Online]. Available: http://www.mimosa.org

[11] Wikipedia, "Node.js — wikipedia, the free encyclopedia," 2016, [Online; accessed 25-October-2016]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Node.js&oldid=746166794

[12] D. Hästbacka, L. Barna, M. Karaila, Y. Liang, P. Tuominen, and S. Kuikka, "Device status information service architecture for condition monitoring using opc ua," in *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*. IEEE, 2014, pp. 1–7.

[13] R. T. Fielding, "REST: Architectural styles and the design of network based software architectures," Doctoral dissertation, University of California, Irvine, 2000. [Online]. Available: \url{http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm}

[14] "Node-red." [Online]. Available: https://nodered.org