Budapesti University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunications and Media Informatics

Interoperability and data processing solutions for current industrial IoT challenges

Scientific Students' Associations Conference paper

Author
Csaba Hegedűs

Supervisor
Pál Varga, PhD

October 27, 2017

Contents

Kivonat

A dolgozat jelenlegi és lehetséges architekturális megoldásokat tárgyal a Dolgok Internete (IoT) koncepciójának ipari esetekre való alkalmazásához. Felvezetésképpen bemutatom az Ipar 4.0 elképzeléseit, főként a régi-új elvárások begyűjtésével és értelmezésével, esetünkben a Productive4.0 [1] európai uniós projekt kapcsán. Ebben a projektben, a MANTIS projekthez [2] hasonlóan, az IoT világ megközelítéseit, koncepcióit és technikai megvalósításait kívánjuk az ipari esettanulmányok kivitelezésére igába fognia. A MANTIS-on belül számos iparág szerepelteti magát, többek között présgépek, hagyományos (földgáz alapú) energiatermelés vagy akár speciális ipari járművek (például targoncák) gyártója is.

Mindkét eddig említett európai projekt az Arrowhead [3] architektúrájára támaszkodik, így ezekben az előző munkáim eredményeit használják fel a partnerek. Továbbá, mindkét projekt esettanulmányai érdekes, esetleg eddig még nem lekezelt követelményeket támaszthatnak az Arrowheaddel szemben. Ezek rendkívül fontosak, az ipari partnerek fő üzleti folyamataihoz és például logisztikájukhoz köthető. Így például az adat létrehozás és intelligens feldolgozás vagy a piaci termékek integrálhatóságának kérdései is nyitottak. Néhány ilyen probléma már megoldott - néhány pedig még elemzésre szorul.

Ezen problémák egy részét mutatom be dolgozatomban. Így például a MANTIS architektúra referencia implementációit, melyben az Arrowhead játssza a többszereplős rendszerek integrátorának a szerepét valós üzleti demonstrátorokban (például a fentebb említett esetekre). Ezen esettanulmány implementációkban mutatom be saját, jelenlegi munkámat, illetve tárgyalom a problémákat és nyitott kérdéseket, melyek felmerültek a rendszerek tervezéseikor (és amelyek relevánsak az Arrowhead továbbfejlesztéséhez).

Végezetül bemutatom a munka következő, lehetséges fázisait, amelyek egyelőre a megvalósíthatósági tervezési fázisban vannak. Természetesen, ezen ipari igényeken alapulnak, a Productive4.0 projekt kapcsán.

Abstract

This paper discusses current and possible architecture solutions for industrial Internet of Things applications. First, a brief introduction to the new concepts of Industry 4.0 is provided. More precisely, this part of the work focuses on what are the current requirements and how they can be collected and understood. These aspects include, among others, requirements relating to real-time communications, security and other non-functional requirements (such as scalability or reliability). This is specifically presented for a relevant initiative, the ECSEL Productive4.0 [1] project. This project, similarly to ECSEL MANTIS [2], is dedicated towards utilizing IoT fundamentals, concepts and technologies to realize very specific industrial use cases. Within MANTIS, various fields of industry are represented, such as press machines, conventional energy plants or even special purpose vehicles (e.g. forklifts).

Both projects are based on the Arrowhead architecture [3], and within that, my previous works. Moreover, they both showcase very specific, industrial requirements towards the Arrowhead framework, i.e. related to data collection (or generation), or using commercial, off-the-shelves (COTS) product for further, specific subtasks. These are necessary, since they create the core business processes of the involved partner companies. Therefore, it needs to be established, how the service-oriented architecture of the Arrowhead framework can integrate these specific data processing systems - and aid their operations. Some of these requirements can be facilitated already, but some not - analysis is needed.

These are presented with examples from the MANTIS architecture instantiations (i.e. in the above mentioned use cases), where the Arrowhead framework plays the integrator role for such multi-vendor system-of-systems. In here, my major contributions to these use case implementations are presented with a discussion on the issues raised during architecting and coordinating these demonstrators (in the above mentioned use cases within MANTIS).

Finally, future work is presented on the currently ongoing design phase about the missing capabilities and approaches. The goals of these are to align with the industrial use cases of Productive4.0 and help the integration of business and logistics systems and processes.

Introduction

The concepts, ideas and fundamentals of the Internet of Things (IoT) paradigm is interesting for industrial players and domains as well. There are many initiatives, industrial working groups or even products that are targeting this world. However, in my experience within various initiatives, they all only cover certain aspects or tasks, and maybe only provide solutions suitable for a given set of application domains. Building whole systems, even for pilot use cases, still requires serious integration work and expertise combining traditional engineering and information technologies. This is "architecting" work is pressing, and I have seen state-of-the-art solutions brought by large companies, that are not feasible to implement or deploy any other place – yet interoperability would be necessary and mutually beneficial. Moreover, in such deployments, the various issues of IoT are even more pressing, that are related i.e. security, real-time communications, interoperability or reconfigurability.

My previous works also revolved around certain well-defined parts of this problem space, as part of the Arrowhead framework. [1] My efforts were related to the refining and designing the framework's service-oriented architecture, aided by certain centralized, core functionalities. Although, the resolved problems are essentials within the framework and forward pointing for its evolution, current issues are related to its wider usability and exact scope of functionality. Therefore, this work is a dedicated towards describing and designing Industry 4.0 use cases and their application systems, using the Arrowhead framework and other products.

These current changes in industry is often referred to as the fourth industrial revolution – or shortly Industry 4.0. The end goal is more or less clear (and defined in many places, such as in the Platform Industrie 4.0 – RAMI [6]). Besides the natural cost-related motivations, this turn of tide is more focused on achieving among others:

- End-user customizable products: "customer wishes for everything from the product idea through the recycling shall be taken into account".
- More flexible production: production techniques like just-in-time (JIT) requires serious logistics not to cause unexpected shutdowns.
- An easier integration possibility for small- and medium-sized enterprises to the "big league".

[1] See my Scientific Student' Associations Conference Papers of 2015 [4] and 2016 [5].

Moreover, the overall design principles are still a work in progress, but the fundamentals have been laid down [7]:

- Interoperability: The ability of devices, sensors and people to connect via the Internet of Things paradigm.
- Information transparency: The ability of information systems to create a virtual copy of the physical world by enriching digital plant models.
- Technical assistance: The ability of assistance systems to aggregate and visualize information for advanced and spot-on decision making (i.e. using big data systems).
- Decentralized decisions: Cyber-physical systems are ought to operate as autonomously as possible without human intervention.

Accommodating these expectations is not an easy challenge, since most parts are still considered green-field research areas – and therefore also rewarding ones. Many technologies and approaches are competing on how to achieve it – and partially succeed. Now, it is an interesting area how such pilot systems and COTS (commercial off-the-shelves) products can be utilized and integrated.

Arrowhead, MANTIS and Productive4.0

This current paper primarily discusses my contributions in three European Union funded projects.

Firstly, my previous works were parts of the Arrowhead Project [3], which has ended by January 2017. It had the vision of a wide-ranged industrial consortium, and aimed at nothing less than to create a full-scoped framework enabling collaborative automation by networked embedded devices. The grand challenges it faced were the creation of interoperability and integrability of almost any device. The represented stakeholders also showed heterogeneity - as opening to an integrated IIoT world is in the interest of many technological areas. Since the primary objective was to create a common platform for all sorts of future implementations, the primary pilot (use case) scenarios of the project involved several business domains:

- industrial automation and production: both the processing and manufacturing sectors,
- smart buildings and infrastructure in smart cities,
- electro mobility: smart electric cars, smart traffic control,
- energy sector: virtual market of energy, energy production and end-user services.

The MANTIS project [2] is similar in a sense to Arrowhead, however more targeted in its objective. It is aimed to utilize IoT technologies and approaches to a specific task: building proactive maintenance enabling systems. The business goal is clear here: prevent unexpected failures and have knowledge about the status of various components well in advance. However, building up such a system architecture and implementing something that is capable of facilitating such tasks is a challenge here. New data collection devices and methodologies (strategies) have to be established, since general maintenance or asset status related information is usually not collected in a well processable or aggregate-able manner. Moreover, since the collected data "has to go somewhere", a cloud-based architecture needs to be proposed - specifically tailored to the requirements of the involved use cases. Figure 1 depicts how such a MANTIS solution would embed into a business scenario.

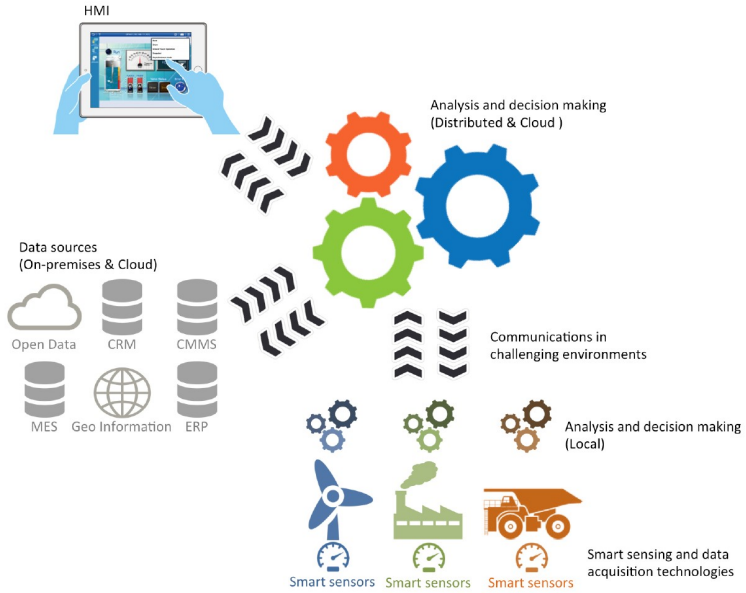

Figure 1: *Overview of the scope for a MANTIS platform*

Meanwhile, the Productive4.0 [1] project started in May 2017, and within one of the work packages it wishes to continue the development of the Arrowhead framework. This is a large initiative, with more than 110 partners coming from many European industries and sizes (from ball bearings to car manufacturers). It supposed to provide an integrated solution from sensory and actuating level up to enterprise decision making assistance. It is targeting exclusively industrial environments, namely three aspects of production and the supply chain is in scope:

- digital production,
- supply chain management and monitoring,
- and product life cycle management.

In its first year of the project and therefore current efforts are related to establishing a the common, unanimous understanding of the requirements (which is a general problem in current IIoT endeavors). This is of highest priority, and a shall result in a roadmap for the further development of the Arrowhead framework. In here, we shall build upon the lessons learned in the Arrowhead and MANTIS project, and clarify how Arrowhead can be utilized as full system integrator in broader IIoT applications targeting all aspects of production. To this end, this paper is presented in three major steps.

Outline of this work

Firstly, Industry 4.0 is discussed, with its difference compared to traditional production engineering. In here, the importance of the ISA95 architecture is emphasized and how this is being replaced by an IoT minded approach. It is of essence here to describe how the new requirements are phrased: the architecture elements that will need to be translated into implementation. For this purpose, an IIoT (Industrial IoT) modeling reference architecture is presented: the 'Industrial Internet of Things Reference Architecture' [8].

Secondly, related previous works is presented on the Arrowhead framework and currently avail-

able functionalities and modules. These include my previous contributions and additional systems from various partners. Although the Arrowhead project has ended, the developed technology is used in other initiatives. This framework is essential and used later on in this paper, as part of use case demonstrations, and also future work is drawn up for it.

Then IIoT application architectures are identified and analyzed. This is presented in two steps. Firstly, the general system architectures of IoT and big data applications are presented and discussed, which culminates in the lambda architecture. Secondly, my major contribution is presented in various applications, as use case pilot demonstrators within the MANTIS project. These implementations are utilizing Arrowhead while building upon commercial products and sometimes rapid prototyped devices, as presented in Chapter 3.

Moreover, the literature overview and lessons learned (from MANTIS solutions) are used to build a requirements collection questionnaire that is supposed to represent the use cases within the Productive4.0. The results (responses from partners) are also elaborated, in Chapter 4.

Finally, future work is presented, based on the technology gap analysis provided previously and the lessons learned from the relevant application demonstrators from the MANTIS project. This results in a road map for the developments, now as part of the Productive 4.0 project.

Chapter 1

The World of the Industrial Internet of Things

1.1 Traditional Networks within Industrial Automation

Traditional automation systems are built having the maximum security and reliability in mind. These are monolithic systems, often provided by one company as a whole, closed solution. Making changes often require a complete overhaul of the system, e.g. one replacement of a single device requires a whole team of experts specializing on the various levels of the ISA 95 [9] architecture pyramid (as shown on Fig. 1.1). Here, there are five levels to a traditional system, namely:

- Level 0: the physical process to be controlled;
- Level 1: Sensors and actuators connected to the physical process;
- Level 2: Monitoring, controlling and supervising elements of Level 1 (e.g. PLC-s);
- Level 3: Manufacturing Execution System (MES) controls the work flow, operate on machine level;
- Level 4: Enterprise Resource Planning (ERP) systems are aimed to utilize the data coming from Level 3, e.g. for material usage planning and shipping.

ISA 95 is accompanied by many related standards that define the general and communications architecture within and between the pyramid levels (e.g. IEC 62443 [11] is related to industrial network and system security). However, each level has many possible suppliers and vendors, each of them often specializing for a small number of application fields.

There are a limited number of well accepted networking technologies for industrial use, mostly proprietary solutions capable of real-time performance (millisecond range latency). These are not Internet Protocol-based (IP) usually. For the actual physical control loops, and process control mechanisms (between levels 1-2-3), closed networks have always been created.

In here, legacy communication protocols are accepted, ranging from one-way analog technologies (like 4-20 mA current loops [12]) to field buses (i.e. Profibus [13]). These are well-built robust networks, with usually one or more controller systems (i.e. a PLC-s) in charge over one or many

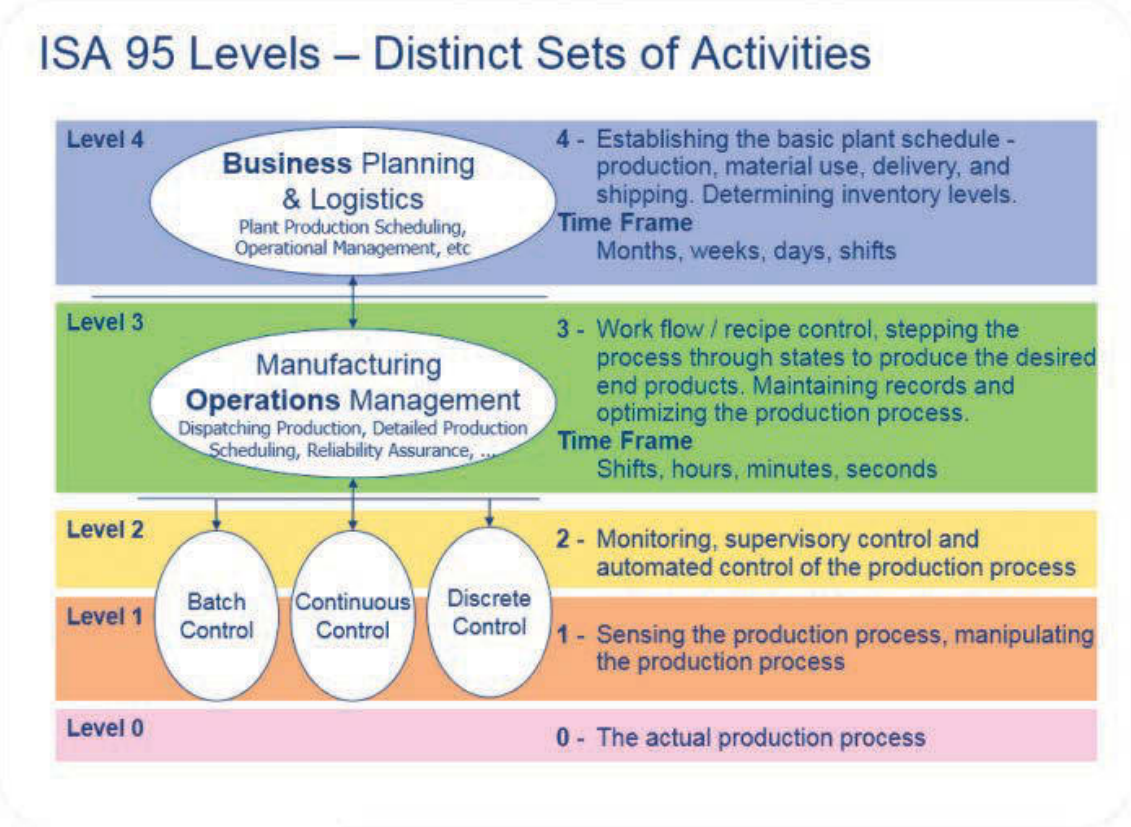

Figure 1.1: *The ISA-95 system hierarchy [10]*

"slaves" (i.e. sensors, actuators or human-machine interfaces). On the physical layer, we can usually find RS 232 or fast industrial Ethernet [14].

In here, masters "pull" data from slaves, usually at equidistant time intervals, ranging from sub millisecond to couple of seconds refresh rate (cycle time). These control loops are inherently competing for controller resources to meet their own cycle time. Therefore, communication latency and jitter has to meet serious, so-called hard real-time requirements in a deterministic way [10].

The controllers themselves then can have outbound connectivity (towards higher levels in the pyramid, towards level 3), and might pose the last point of software reconfigurability. The input and output (I/O) ports are usually fixed to sensor and actuator instances (design time binding), any modifications require physical access. All access to level 1 is centralized in level 2 – level 3 only communicates via level 2 [14].

This creates a fairly rigid network setup, both on a physical level and in an accessibility sense. Any modification to the system requires that all levels are partially shut down and entered the new setup. Moreover, if e.g. a technician wishes to see environmental variables (such as temperature of the process), it has to access it via the MES system - that will access the level 2 controller instance, and that controller will access the I/O port of its own, locking it from parallel use.

According to [10], a "number of clear trends related to such automation systems have been stated via different road maps and initiatives". These include:

1. Production flexibility and customization shall be easier.

2. Multi-stakeholder interactions shall be possible, all having proper access to the automation systems according to their ”stakes” in it.
3. The possibility to build larger automation systems (current practicality limit is around 100 000 I/O ports).
4. Maintain automation system security while moving away from closed, proprietary networking technologies.

It is a focal point of Industry 4.0 approaches: everything should be on the same network (e.g. IP-based), for easier reconfigurability. Current developments in sensing technologies enable for so-called smart sensors and actuators which themselves can be IP endpoints, addressable and accessible on the network directly. This way, every elements or device in a production plant can become a ”thing” in the IoT sense [10].

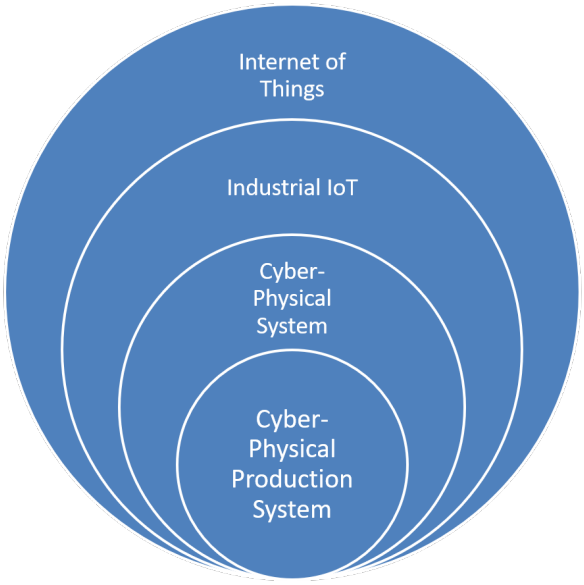

Figure 1.2: *Understanding the terminology*

Figure 1.2 depicts how the various terms should be considered regarding industrial IoT, how we get from industrial "things" to a whole new generation of enterprise systems building on IoT fundamentals. Basically, in this terminology, we are starting from production systems, that are becoming "cyber-physical", described and discussed in the next section. Then such cyber-physical systems can be part of a larger, industrial environment, where many of such systems shall cooperate, providing data and receiving commands from one another, creating the possibility for quick reconfigurability and flexibility.

1.2 Smart Shop Floors: Cyber-Physical Systems Communicating

The first aim is clear from our property list of visions: utilize current developments in embedded microprocessor systems and sensing technologies and build automation systems based on IP networks, rather than proprietary technologies. Smart sensors nodes, motes can be built, and they can connect through a whole new variety of wired or wireless networks. This way an ”old”

automation system can be monitored and even controlled, with comparatively cheaper solutions than previously. All data describing the environment (and process) can be collected, while not connected to the real-time control loops of the process at first.

Moreover, the legacy machines and systems themselves can also have an "addon" that can represent the capabilities of the machine control system towards the "modern network". These "gateways" can be connected to the PLC-s or controller PC-s of the machines, and can utilize the legacy (e.g. RS 232-based) interfaces of the machine. Here, we can access diagnostic and telemetry type of information from the machine controllers.

This concept is connecting the "physical and cyber spaces", hence such a system is called cyber-physical system (CPS). A working definition for CPS has been offered in [15], where a CPS is defined as a systems consisting of computational, communication and control components combined with physical processes.

Therefore, CPS-s are establishing the foundation for the deployment of a new generation of complex systems. Until now, embedded computing has been focused on improving on-device computational capabilities, on the other hand CPS are focusing more and more on the necessity to have some kind of communication between CPS-s, i.e. the need of autonomous CPS to dynamically interoperate in order to achieve a higher (e.g. production) goal.

Primarily, CPS-s differ from traditional centralized MES systems in one major aspect: CPS-s are intended to be autonomous to some extent. They can incorporate level 2-3 functionalities, and represent these resources to other level 3 or level 4 entities in the ISA95 sense. They shall pose (the cyber part) an interface towards other CPS-s that help with the monitoring, telemetry and even configuration of the incorporated physical resources, processes.

This concept is pointing towards Service-Oriented Architectures (SOA) [16] that use exactly such fundamentals. The novelty of migrating from a legacy process control systems into a SOA, is to transform and gradually upgrade highly integrated and vendor-locked standards into a more open structure while maintaining and extending the functionality. Decentralization and non-design time bindings are key components here. There are several definitions and descriptions available for SOA, OASIS [17] defines it the following way:

A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.

From a communication point of view, as mentioned earlier, Industry 4.0 sees IP-based networking as the cost-effective, proven and pervasive technology that – with special restrictions – can be used for the new generation of automation systems. Moving away from master-slave field buses, the IP technology offers communication in a Plug-and-Play and sometimes zero-configuration way.

It is worth noting, that the characteristics of of the network used is not in scope of this paper. We only assume that there will be endpoints that can be accessed on a local network when needed, these endpoints will be usually connected to automation systems and industrial processes.

1.3 Arrowhead [1]

The Arrowhead project was started exactly on these fundamentals: make automation based on CPS-s operating in a Service-Oriented Architecture (SOA) manner. This approach can be interpreted as an extension of the interface concept used in the object-oriented programming paradigm. A SOA based system is built upon a collection of loosely coupled components that are somehow interconnected: they provide services to one another [18]. Fig. 1.3 depicts how this can be abstracted (and presented in the Arrowhead nomenclature).

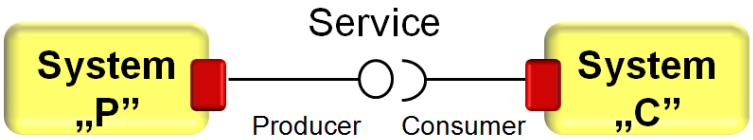

Figure 1.3: *Services produced and consumed by systems*

The building blocks of the SOA systems are these services. There is no standardization on what characteristics services have, or how they can be implemented. The authors of [19] emphasized the following characteristics of service abstraction:

- Loose coupling: hardwired connections between entities are not permitted.
- Rather services are dynamically discoverable upon need (run-time).
- Services are self-contained and modular. A service supports a set of interfaces and these interfaces are logically cohesive (they implement the same functionality).
- Modular understandability: the user has to be able to use the service without having knowledge of any other underlying implementation details.
- Modular decomposability: a complex service can be created from simpler atomic services.
- Interoperability: systems using different platforms and programming languages should able to communicate with each other using services.

SOA is primarily associated with the Web Service [2] stack. However, it is worth noting, that the SOA principles can be implemented using (nearly) any technology - might be even used outside of information technology purposes [21]. Its main advantage lies in its emphasis on a minimalistic common ground and thrive for resource reuse. Moreover, late binding essentially has the capability of providing reconfigurability since service discovery can be tailored to retrieve specially chosen service providers.

The Arrowhead Framework aims at providing interoperability (connectivity) by facilitating the service interactions within closed or at least separated automation environments, called *Local Automation Clouds* (LC-s). Hence creating central governance with a minimalistic set of core components that take over certain configuration tasks from individual systems and components (e.g. sensors). In such Arrowhead Local Clouds there can be an arbitrary number of *Systems* that

[1] This subsection is extract of Arrowhead based on my Scientific Student' Associations Conference Paper of 2016 [5].

[2] Web Services (WS) are basically a given protocol (HTTP) and design guidelines to provide (mainly) a machine-to-machine interoperability layer managed by OASIS [17] and W3C [20].

can provide and consume *Services* from one another: they create and finish *servicing instances* dynamically in run-time. Figure 1.3 depicts these relations.

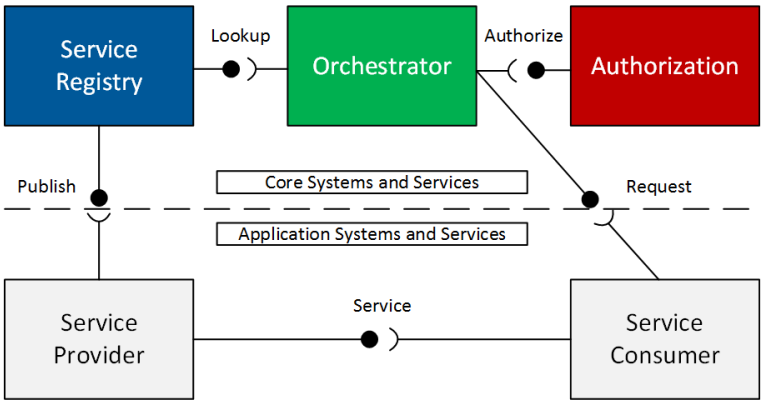

Figure 1.4: *Interactions with the Arrowhead core*

There are Currently, there are three mandatory Core Systems and other optional ones that support automation, as depicted in Figure 1.5.

- The *Service Registry* stores all the Systems (that are currently available in the network) and their service offerings. Systems have to announce their presence, and the services they can offer. The registry takes note of this information when systems come on-line, and might have to revoke them when they go off-line.
- The *Authorization System* - as its name suggests - manages authentication and authorization (AA) tasks, however, it covers some other security-related issues as well (e.g. certificate handling).
- The *Orchestrator* is responsible for instrumenting each System in the Cloud: where to connect and what to consume. It instructs Systems so by pointing towards specific Service Providers to consume specific Service(s) from.
- The QoS Manager concept aims to make the real-time constrained embedded devices integrable in a Local Cloud (i.e. actuators) [22].
- The System Configuration Store [23] provides customized storage for firmware and configuration file updates for Application Systems. This storage solution has to support a multitude of transport and firmware management protocols in a secure and safe way.
- The Plant Description system [24] aims to integrate various topologies that might be used to describe the same establishment.
- The Historian hosts application data for Systems, and stores the data in query-able form.
- The Translator System [25] can (will) provide application-level translation so that a producer and consumer implementing a different protocol of the same Service can still interact.
- Event handlers are tasked with the run-time detection of various events and errors and triggering the appropriate Systems (in a publish - subscribe manner).

My previous contributions were related to local orchestration and also inter-Cloud servicing. Arrowhead Local Clouds might be sufficient to handle certain tasks in their isolation. However, as one single Arrowhead Cloud (one SoS) cannot serve for all, there is need for inter-cloud operations to achieve true global interconnectivity within the Framework-compliant networks.

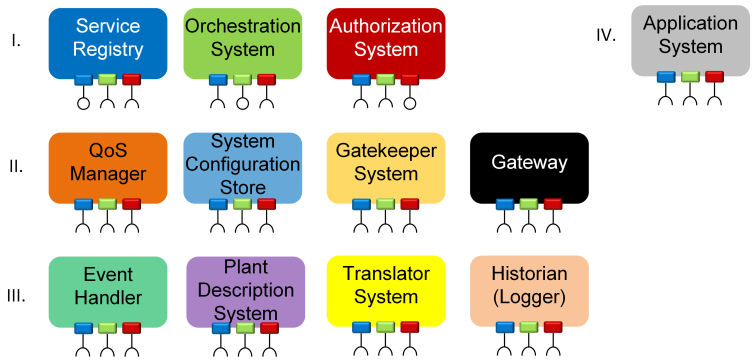

Figure 1.5: *Arrowhead Core Systems*

My inter-Cloud communication architecture builds heavily upon the Core Systems' exclusive involvement in facilitating and managing the connections (servicing instances) within their authority: in this sense, the Core Systems are responsible for all other Application Systems and their data handled in the Local Cloud. This concept will be fundamental for later analysis. Figure [?] depicts a sequential overview of the inter-Cloud orchestration process using Gatekeepers and Gateways.

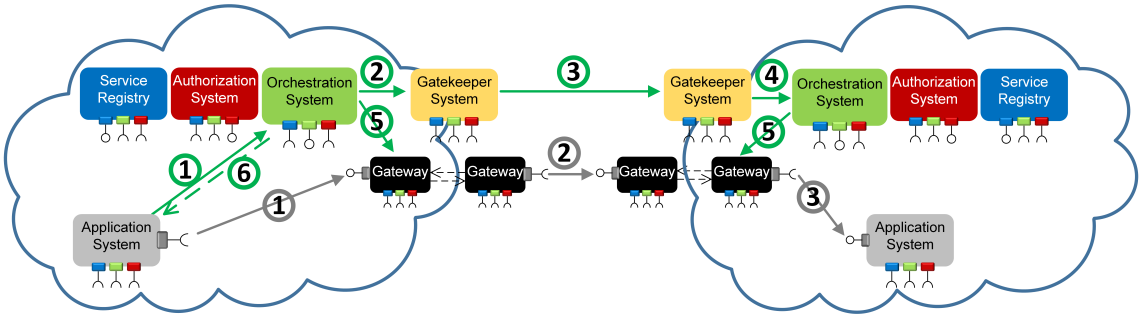

Figure 1.6: *Inter-Cloud collaboration using Gatekeepers and Gateways*

To conclude, Arrowhead primarily assumed so far that there is a physically and conceptually closed local network in automation environments, where Application Systems (CPS-s) are communicating in order to achive their operating goals. These processes are aided by the Core Systems of framework through providing reconfigurability and late binding, resource management and monitoring.

With inter-Cloud collaboration, we are opening up these closed networks (not just on a conceptual level), and allowing external stakeholders and their systems to consume some sort of Services (information) from our production network. My Gatekeeper and Gateway [26] modules are intended to restrict and see over such connections, but in principle the data that will be given out is still dependent on the Service contract (interface definition regarding functionality).

Chapter 2

Internet of Things and Big Data Processing

2.1 IoT Platforms and Components

This chapter discusses a completely different world - yet to be merged within industrial automation. A "usual" application scenario here involves *things* collecting enormous amount of data and sending them to a *cloud*. There are many competing, full or partial solutions to the problems of this world, with many communication protocols, networking technologies and cloud architectures.

It is of essence for this paper to identify and analyze the generic architecture model of IoT systems and the data flows between components. A survey of 39 IoT platforms [27] concluded in a generic architecture and common characteristics of IoT platforms. In here, all relevant "big player" of the industry are inspected and included. These systems naturally have common characteristics, such as:

1. There is usually a cloud-based solution that collects data from many type of devices in a centralized manner.
2. There is only a handful of protocols that can be used to submit data to the cloud.
3. Majority of the platforms only allow specialized protocols to be used for transmission. Usually, no add-dons can be created for the platform to receive data in a proprietary protocol.
4. The majority of the listed platforms are not incorporating mechanisms for storing or manipulating the data locally, rather the data is sent to the cloud platform in raw format, stored there, unencrypted.
5. Moreover, later on, access control to the collected data is not well established in most cases either. Data ownership is given up once the device sent the record to the cloud.
6. There are various data processing entities that can operate on the data.
7. There is an included Human-Machine Interface (HMI) that can present the data.

Figure 2.1 depicts a generalized commercial IoT framework in its fullest form, based on these characteristics. In here, there are two possibilities shown. The various IoT services can be deployed

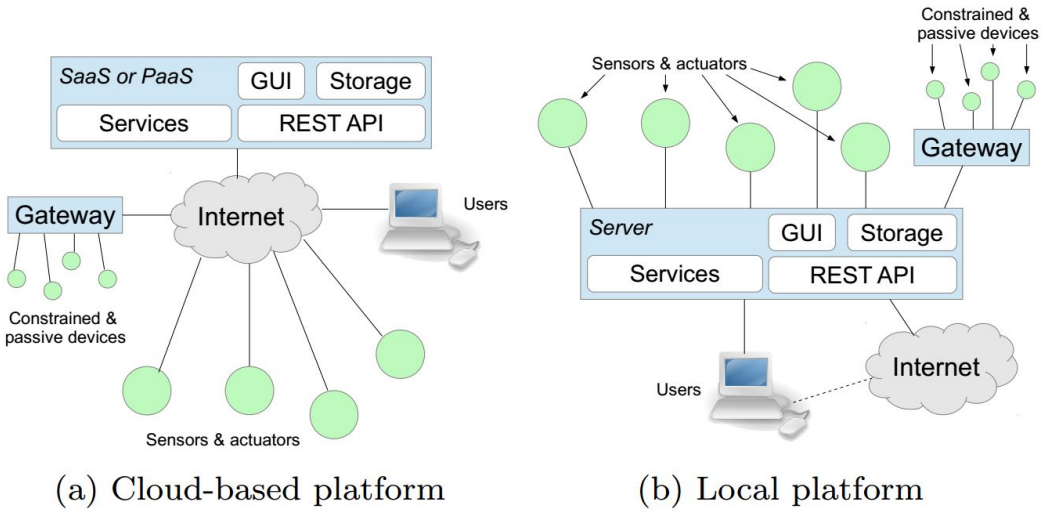

Figure 2.1: *A generalized IoT framework [27]*

on local premises or within an IT cloud. For both cases, however, there is distinct two levels: devices, "things" and a platform that receives data and processes it and provides a User Interface (UI) to view the data and the provided analytics of it. The generic modules in such a platform are:

- Sensor or actuator nodes, motes that send in the data.
- Gateways that "hide" too constrained devices that might be communicating via non-IP based networking and/or incapable of implementing the platform interface (e.g. REST or MQTT).
- A platform interface that receives the data from the devices and passes it to other modules (data distributor).
- Distributed data storage, accessible by all.
- Various service modules that can access the historical or even the current inbound data streams and provide insights and various processing possibilities (e.g. big data applications).
- Graphical interfaces (Business Insights) for operators to manage the system and validate the output.

In this generic data flow model, things are collecting the data, and the cloud processes it. The device-cloud interface is - in majority - is implemented using popular protocols. These include primarily the Representational State Transfer (REST [28]) over HTTP, sometimes uses Web Sockets or the Constrained Application Protocol (CoAP [29]). Moreover, publish-subscribe typed protocols are also gaining popularity, such as the Message Queue Telemetry Transport – MQTT [30]) protocol is also used.

The data distributor module is basically a publish-subscribe brokering pattern [31] on its own. However, it has very unique operational conditions, and the engineering of a such system is mostly dedicated towards maintaining is scalability, resilience and fault tolerance. It receives and aggregates the data streams form the devices, and passes it on to the appropriate modules in

the cloud: presistent data storage, analytic services, HMI-s, and so on. This module is usually deployed on a cluster, and exposes multiple, load-balanced IP endpoints for devices to upload their data. A popular, open source implementation of this module is the Apache Kafka [32].

In some cases, this distributor even realizes two-way communication with the devices: cloud components can also push back information (cloud to device) using the same interface (e.g. if it is implemented over MQTT or other persistent connection). These messages can be related to commands towards the devices. However, in a generic IoT application this feature is not required.

Data storages are naturally a cornerstone of such platforms, and there is wide literature on how to build and maintain distributed data storages, on both SQL- and non-SQL-based technologies as well [33, 34, 35]. The same design and operational aspects are relevant as to all cloud-based modules (i.e. with special attention to scalability, fault tolerance, robustness). One popular, open source implementation used by the majority of the players is the Apache Hadoop [36]. It has REST [28] based API (Application Programming Interface), as many other technologies in this world.

2.2 Data Processing: The Lambda Architecture

This section discusses how IoT platforms process and make use of the collected data: what are the analytic services and how they operate. This is highly important, since the whole purpose of IoT is to facilitate data analytics for various business purposes. Industrial use cases shall be built this way as well, and will use the same tool set available on these platforms.

According to the generalized lambda architecture pattern [37] defined by industry experts, data can be processed either as soon as it reaches the platform (stream processing), or later on, on demand fetched from storage (batch processing).

Figure 2.2 depicts the overview of a generic analytic platform. In here, the "speed layer" comprises of stream processing technologies, that are processing inbound data real time. It is an event driven programming paradigm, where the processing functionality receives periodically touples and executes the same function over them (e.g. creating a counter for a specific message type). This type of processing shall not take too much time and effort, and should be prepared to be heavily parallelized, over various network nodes in a cluster.

Such stream processors can usually only execute in a directed acyclical graph (DAG), as does the most popular open source implementation, Apache Storm [38]. Within Apache Storm for example, every data touple forms an immutable data structure, and operations are distributed over nodes in the cluster, in a fault tolerant manner. More design guidelines on implementing application pipelines can be found in its documentations.

The other type of processing is asynchronous to the inbound data, and can be called on batch datasets queried from the data storage. These tasks are run once at a time and might take long to complete. In here, machine learning algorithm training or other hard computation-heavy jobs can be run.

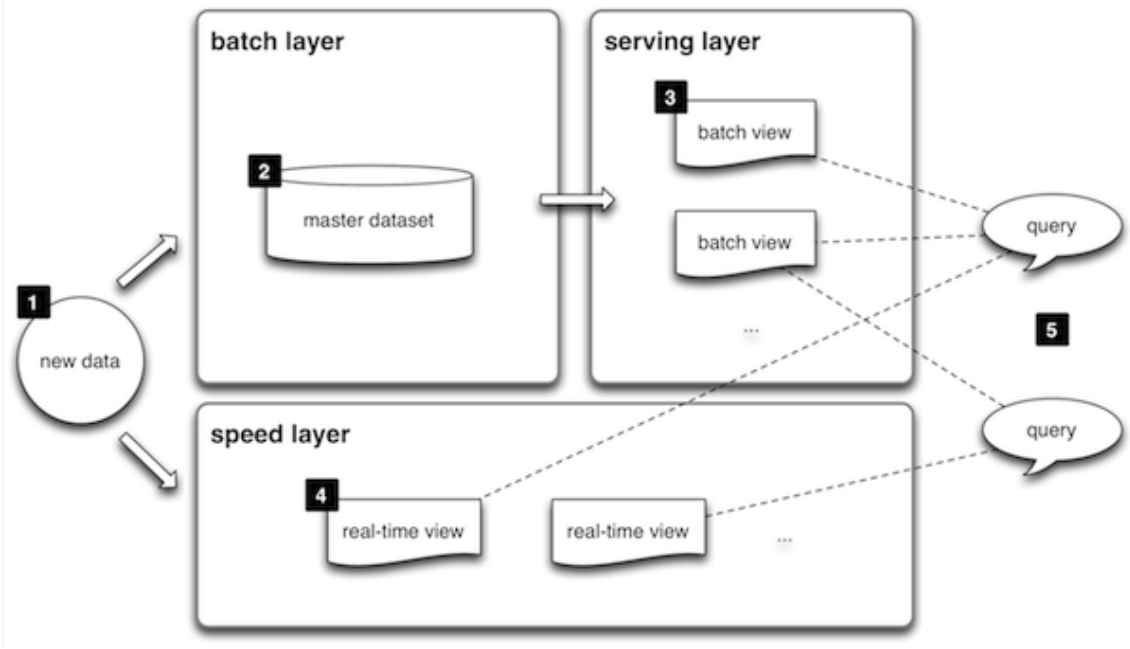

Figure 2.2: *The lambda architecture overview [37]*

An other major responsibility of the batch layer is to maintain the data storage, aided by the serving (database) layer. Primary functions of the latter include access control and enforcing the data ownership policies over the database. As discussed in [27], the authentication and data retention policies are usually resolved with access (and API) tokens that verify the sender mote. Moreover, the Public Key Infrastructure (PKI, certificates) is used, especially for the REST based implementations.

On a side note, as discussed in our article [39], these IoT platforms, systems are subject to a number threats on its every "layer", coming from the data sources up to the applications that utilize them. Figure 2.4 depicts how such a system can be abstracted as a layered solution. Appendix 2 contains the main conclusions of this research, determining the main security threats present for each layer.

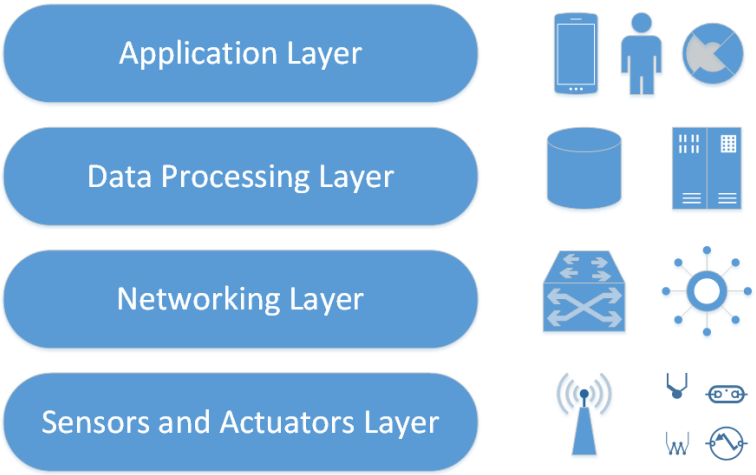

Figure 2.3: *IoT framework layers*

2.3 Edge Computing

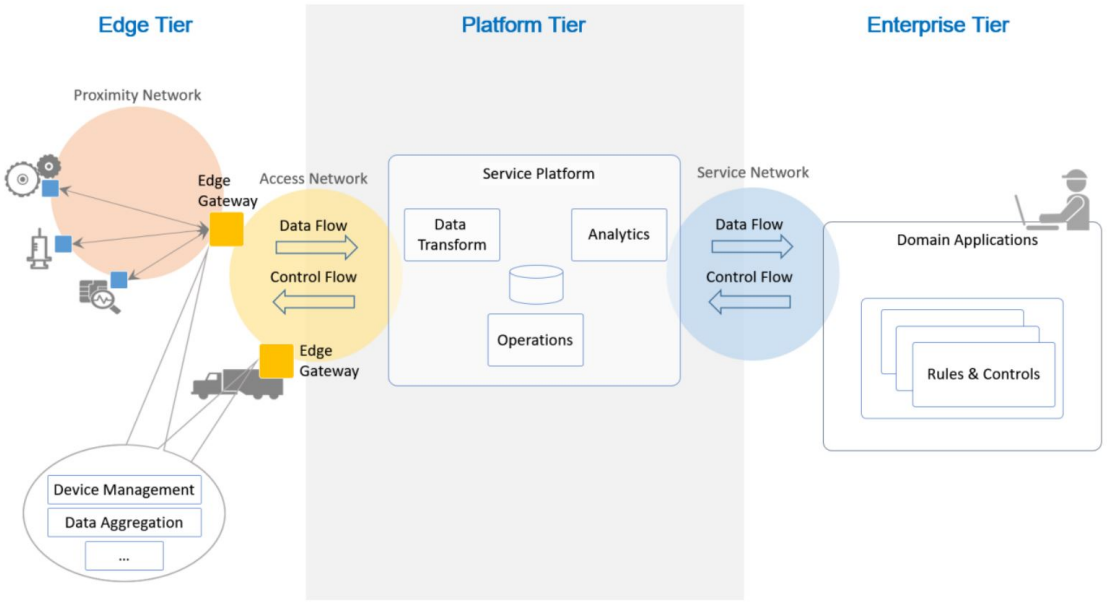

Figure 2.4: *The IIoT Reference Architecture model [8]*

Industrial use cases are more inclined to refine this general business model of IoT platforms. One of the main reasons for this is related to the security and data ownership policies (or better said their lack of) of the commercial systems. European data retention policies [40] are very restrictive on what type of business (or customer) data can be stored, and especially where. Moreover, companies are also reluctant to trust third parties with their business sensitive data (naturally, for corporate espionage reasons). However, the biggest issue with sharing all data with a remotely deployed platform is related to the data quantity. Process control or general telemetry data is of millisecond resolution, while various logs are generated in a very verbose, lengthy format.

Therefore, it is viable to place various (pre-)processing tasks locally, close to the data source. This concept is called edge computing, and here the cloud platforms do not get directly in contact with the devices themselves, but through an *edge gateway*.

The need for this is also verified in the Industrial Internet of Things Reference Architecture [8]. In here, the architecture is dissected into three tiers. The first one is the edge tier that can be located in shop floors, and might include real-time control loops (cf. ISA95). In here, we are collecting the data, aggregating it, preprocessing it and doing preliminary analytics that can be done locally (e.g. outliers detection), as seen in 2.4.

The platform tier can be cloud-based, or generally run on corporate servers.The platform tier receives, processes and forwards control commands from the enterprise tier to the edge tier. It consolidates processes and analyzes data flows from the edge tier and other tiers. It provides management functions for devices and assets. It also offers non-domain specific services such as data query and analytics.

The enterprise tier implements domain-specific applications, decision support systems and provides interfaces to end-users including operation specialists. The enterprise tier receives data flows from the edge and platform tier. It also issues control commands to the platform tier and edge tier.

Chapter 3

Building an IIoT architecture for MANTIS

3.1 General MANTIS Platform Design

This chapter details my contributions on establishing the reference architecture for the MANTIS project. MANTIS aims to provide a proactive maintenance platform for cyber pyhiscal systems. Based on the requirements submitted by partners, my job was to research and draw up a reference architecture model for a cloud-based proactive maintenance targeting IoT system that fits within the industrial use cases. This was used as a "architecture template" afterwards, and use cases can contribute on which modules are they implementing, what are the use case specific modifications they had to make. Moreover, I participated in three of the eleven pilot demonstration implementations.

Within the project, there are pilot demonstrators that were not starting from scratch (internally called "brown field use cases"), and there were partner groups whose use case were yet to be architected ("green field ones"). Therefore, my suggested architecture model is based on my research (as presented in Chapter 2) and the lessons learned from the first trial implementations done by the brown field use cases.

The purpose of the MANTIS architecture is to make proactive maintenance possible. Since large amounts of data can be collected from industrial devices, machines or vehicles, there is sense in trying to utilize them. One of the first drawbacks experienced, of course, was that although data has been collected for several years now, this data collection, aggregation and storage systems were not designed to be actually used later on. This means that the format, structure and quality of the data is not sufficient for today's analytic services to comprehend or use. Within MANTIS, analytic services are that need to be developed are related to three functions [41]:

- Remaining Useful Life (RUL) of components: continous tracking of telemetry (usage) data and estimating how much time the given device or component has left before needs to be replaced,

- Fault Prediction (FP): the system shall predict based on diagnostic data an inbound failure mode (different to wear-out to be detected by RUL),
- Root Cause Analysis (RCA): when an unpredicted, complex failure occurs, the system shall deduct the actual module (cause) that caused the break .

The MANTIS platform has designed to support these functions [ibid.]. All three functionality requires that various, preprocessed data is streamed into the platform, with appropriate device and location identification. It was decided early on within the project, that one commonly used platform instance is not in the interest of project partners (yet again due to business-related reasons). Moreover, a reference architecture in general shall not be implementation-specific, see [8]. As a first step, I have identified key differences in the requirements that are pointing away from a standard IoT system implementation (described in the literature):

1. The edge computing paradigm has to be used to reduce the data sent to the platform and to enable on-site maintance operations.
2. Therefore, a whole infrastructure within the edge has to be defined for storage, limited analytics and HMI support.
3. Various legacy data formats need to be translated to a unified MANTIS-capable one before sent to the cloud, if possible.
4. Dynamical discovery of available platforms and and attachment procedure to a platform instance is needed.
5. Emphasis must be put on a bi-directional edge-cloud interface specification so that the platform and enterprise tiers can access the edge and command it.
6. Many platform tiers shall be able to reach one edge installation for multi-stakeholder interactions.

Based on this, I have proposed the MANTIS architecture as presented on Figure 3.3. In this high level overview, I utilize the edge-platform separation and we should also allow standalone devices to send their data directly. Here, we have to emphasize that the edge level is a fully fledged system on its own. It shall have local storage, analytic tool sets and even man-machine (HMI) interface. Nevertheless, when the edge system is not a standalone device, but e.g. a shop floor, the edge gateway must be the sole module that shall be tasked with keeping the connection to a higher level, corporate platform. It needs to tap into the relevant local data sources (i.e. CPS-s), and transmit the pre-processed stream. The purpose of the higher level platform is to collect data from various sources (not just e.g. from production sites), and create analytic modules that can utilize the extra data sources, not available locally.

Since we are facing serious interoperability issues when using and trying to monitor legacy systems and custom-made devices, understanding the data and parsing it into a unified, processable format is not an easy task. For this reason, MIMOSA [42] was selected as a (SQL) database schema. This way, all partners can store their data in a unified manner, and the algorithms developed for prognostics can be reused between use cases. As presented in our joint paper [43], MIMOSA can be a standard-bases cornerstone that creates interoperability for such a maintenance platform. Even if later on the SQL nature of the database schema will have to be changed,

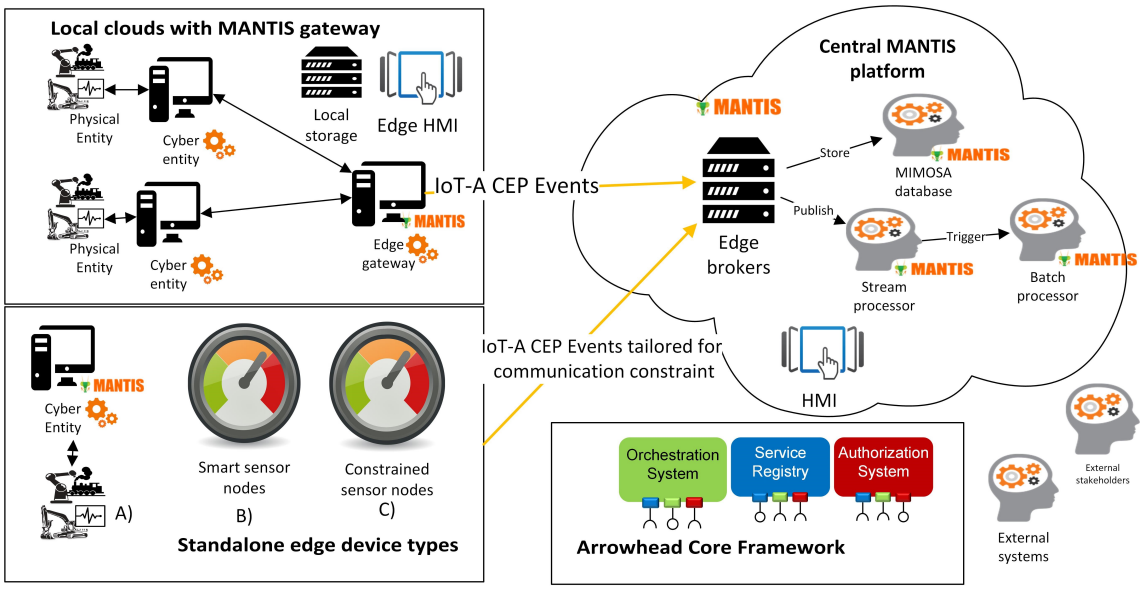

Figure 3.1: *Overview of the proposed MANTIS architecture*

the implied data ontology and semantics can still be reused. However, this design decision will have to be revisited, once the MANTIS platform reaches its scalability limit in its use (and the bottleneck turns out to be the very complex SQL nature of MIMOSA).

The standards that MIMOSA develop are compliant with the ISO-13374 Standard (Condition Monitoring and Diagnostic of Machines), being an implementation of the latter's functional specifications. According to this standard, a Condition Based Monitoring (CBM) system should be composed of various functional blocks: 1) Data Acquisition (DA), 2) Data Manipulation (DM), 3) State Detection (SD), 4) Health Assessment (HA), 5) Prognostics Assessment (PA) 6) Advisory Generator (AG) [42]. This corresponds well to a general IoT system architecture with edge computing.

Arrowhead is intended to resolve not just the edge computing interoperability and connectivity issues, especially if real time control loops are kept. In my general concept, Arrowhead can be used to resolve issues #4 and #6, regarding multi-stakeholder applications. Multiple "cloud levels" shall be enabled to access one single production site or edge device in order to get the necessary information. The same goes vice versa, one edge instance shall be able to locate and connect to additional services, once local decision making algorithms are deployed, e.g. the production plant shall be submit replacement part orders if it detects the need for it (based on RUL). Arrowhead's metadata-based service discovery and orchestration could be utilized for this exact purpose.

This can be achieved, if the edge level is abstracted and implemented as an Arrowhead Local Cloud on its own, and the same is done for the global IT cloud implementation. Since, inter-cloud collaboration in the Arrowhead framework assumes that the two Core System sets are negotiating the connection. This way, one edge level is accessible by multiple stakeholder clouds, if those need additional data (e.g. one from the operator corporate, and one from the ball bearing supplier). Moreover, this way we are also enabling local decision making, and such algorithms can utilize the Arrowhead architecture to collect the ad hoc, on demand information the algorithm needs,

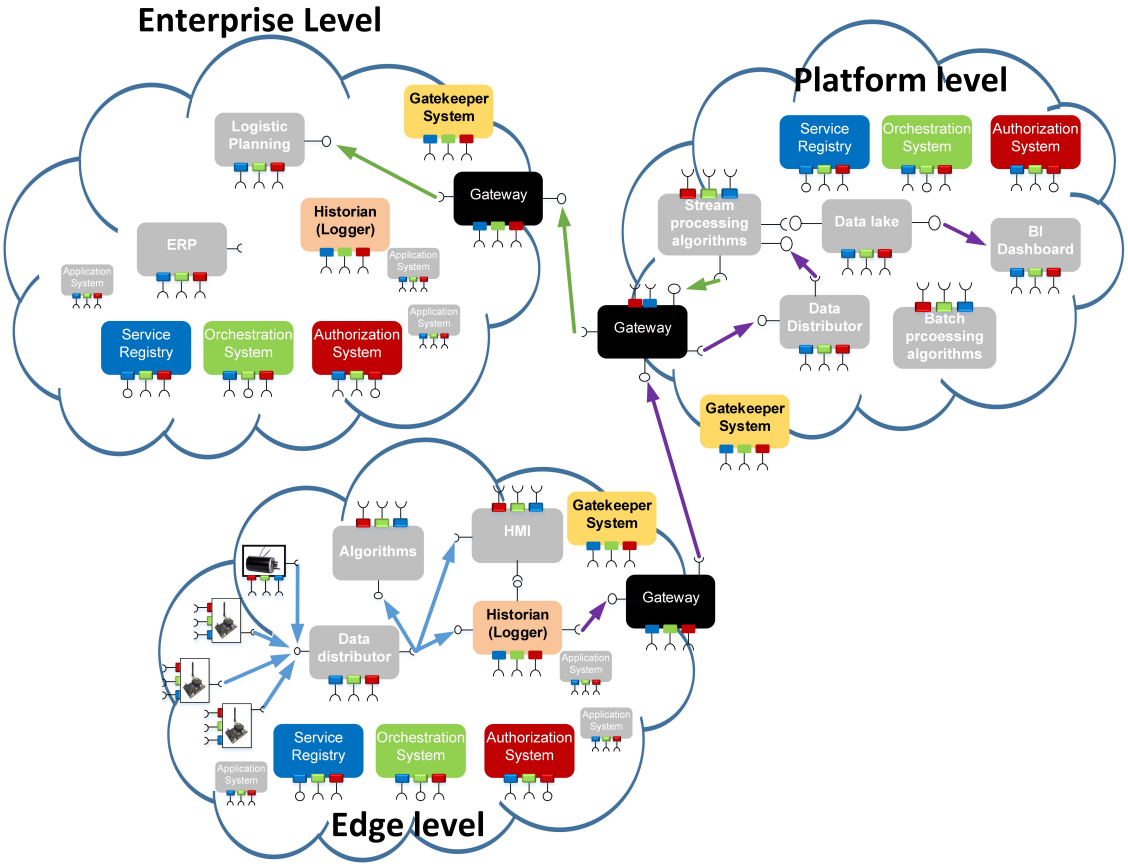

Figure 3.2: *MANTIS elements abstracted as Arrowhead objects*

fetched from its original source. Figure 3.2 depicts how the edge, platform and enterprise tiers can be abstracted as Arrowhead objects.

In here, we can utilize the Arrowhead framework for three example data flows. Firstly, motes, industrial "things" are collecting data, let those be sensor nodes, data coming from industrial PC-s or even PLC-s logging their activity. They shall utilize the Arrowhead framework to establish the connections they need, and start their operations. This usually shall end in a local (and sometimes temporary) storage of the (sometimes pre-processed) data. Secondly, the platform level receives batch uploads from edge instances, that can be done via the Arrowhead core framework's lookup and (inter-cloud) orchestration features. This way, not just one platform instance can receive the data from the edge, but many, in an ad hoc, on demand manner. Thirdly, analytic algorithms (e.g. remaining useful life calculation in our case) can trigger the enterprise level when a certain corporate action is needed (event handling, in this case e.g. ordering a new replacement part, since our current component is breaking down soon).

All this can be aided, so that multiple stakeholders can in run-time receive and request information they need, in an asynchronous way. More Arrowhead supporting core systems can be used as well, implemented on any platform that can facilitate the official service interfaces that belong the given system.

3.2 Demonstration of the MANTIS Platform for Various Use Cases

In this project, there are eleven pilot demonstrations to be created. I have participated in four of them in the architecting phase, and within those, in two as a developer-contributor. This section shortly reviews the differences in approach and design deviations from the generally established reference model. The first two use cases are examples for edge computing, and the latter two are standalone devices with special communication requirements for the edge-cloud interface.

3.2.1 Conventional Energy Production

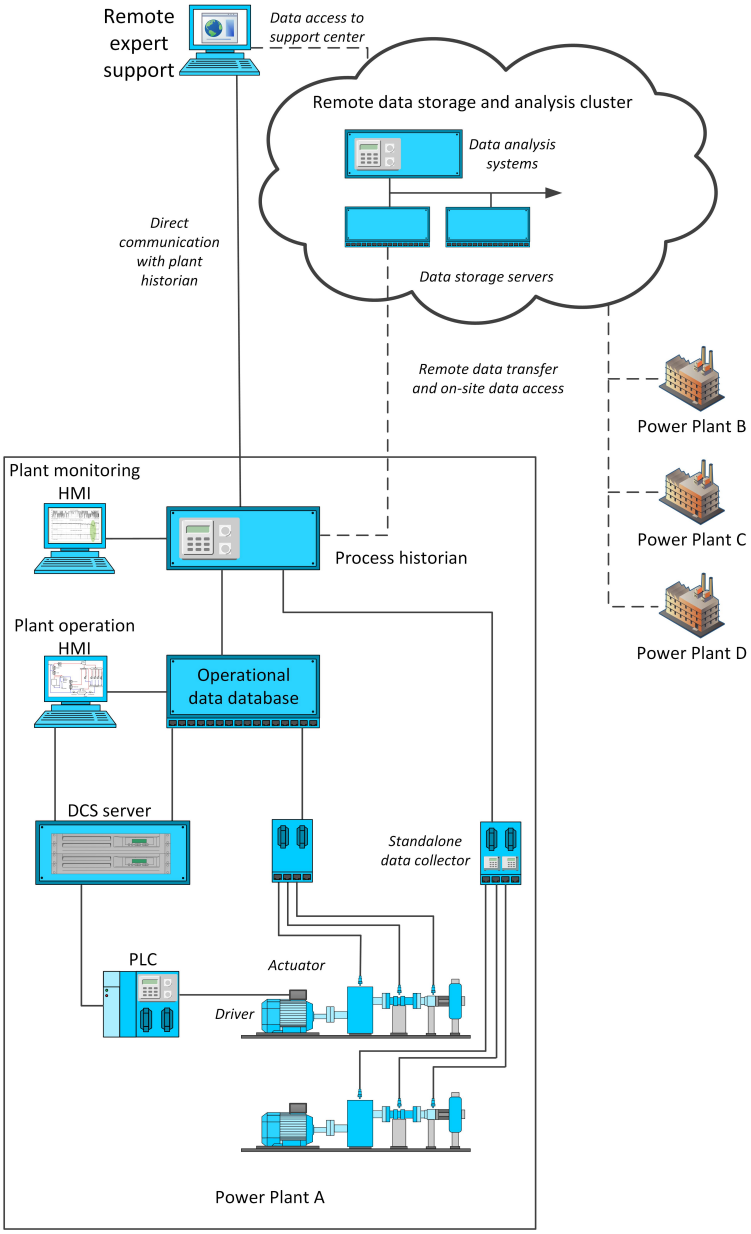

Figure 3.3: *Instantiation of MANTIS for conventional energy production [43]*

In this use case, we are monitoring a power plant [44] that operates with a wide range of utilizable fuels, mainly wood biomass, peat and waste based fuels in Finland. The generator is the

device under scope, and most importantly the ball bearings inside. Failure and remaining lifetime estimation is highly important for this machine, since unexpected maintenance shutdowns are endangering the power supply of the region. To this end, accelerometers are placed in the generator, and commercial sensor motes (developed by a local Finnish project partner) are sampling the vibration with high frequency. This data is sent to a local storage, in addition to the telemetry data coming from the PLC system via traditional 4-20 mA analoge output signal, through a MANTIS developed gateway mote. The overall architecture is compliant with my MANTIS platform proposal, depicted in Figure 3.3.

As presented in our paper [43], the primary analytic tool is based on the spectrum of the vibration emitted by the generator. This tool is running in Microsoft Azure, and can be presented on site, in the local machine HMI. The vibration data is collected in a central MIMOSA database, with specific measurement site identification. Various other stakeholders are also interested in the prognostics of the system, such as the local supplier of the bearings. The presented RUL and fault prediction output is invaluable for proper stocking of replacement parts.

In this use case, I have participated in the design of the MANTIS platform instantiation, and the cooperations with the various partners. Moreover, I have adapted the MIMOSA REST interface implemented in Java (deployed on a Tomcat servlet container), by adding two new SQL tables to be accessible via this interface.

It is worth noting, that this use case does not implement stream distribution or processing, and the central data distribution method is via the MIMOSA database, in an asyncronous manner. Each data source puts its information into MIMOSA using the REST interface, and all other "subscribers" are fetching the data via the database as well. This design decision was taken by the Finnish partners for rapid prototyping, but it is a known risk of being a bottleneck for further developments.

Future work within this use case includes the integration into the Arrowhead framework as the final demonstration at the end of the project. This use case will be the first that will adapt inter-cloud communication once the Gateway module is fully available for public use [1].

3.2.2 Industrial Press Machines

My second contribution was in making an the industrial sheet metal press machine "smart" residing at ADIRA [46], Portugal. In here, the overall MANTIS solution was realized in an open source manner. This is also an edge computing scenario, where the edge-cloud interface is realized by using RabbitMQ. This architecture implementation is worth examining, since all popular, quick prototyping technologies for IoT projects are present, see Figure 3.4[2]:

- All legacy machines have a gateway mote (Raspberry Pi 3B [47]) representing it as a CPS on the network.
- New sensor motes were developed using Arduino and Bluetooth 4.2 (BLE) [48].

[1] This addition is developed and presented by Nikolett Szeles in this section of the conference [45].
[2] This figure is presented in an internal MANTIS deliverable D7.16 Use case integration report UC1.3.

- These CPS-s are connected in a local OPC-UA [49] network.
- The edge gateway is implemented using Node-Red [50] and uses the Advanced Message Queuing Protocol (AMQP) towards the cloud.
- The RabbitMQ [51] broker is utilized as the cloud gateway.
- Apache Kafka [32] is used to deliver the data for the various cloud modules.
- Binary objects (e.g. csv files) are stored in Hadoop [36].
- An R server [52] is used to do offline analytics on the data.

My contribution to this use case was supporting the integration of the local automation network into Arrowhead. All local modules (CPS-s, the MANTIS-PC and edge gateway) are behaving as an Arrowhead Application System, and the SOA paradigm of late binding is taken advantage of. Although, the Open Platform Communications - Unified Architecture (OPC-UA, [49]) enables in theory for runtime binding using the OPC Discovery feature, open source implementations are not supporting it - or not in an interoperable way. This is a shortcoming of the OPC framework that even though there are many implementations (commercial or open source), modules are not always interoperable with each other. Nevertheless, one's shortcoming, is Arrowhead's opportunity.

As a lesson learned, future work will have to include the full implementation of the Arrowhead framework over the popular, publish-subscribe protocols, since currently it is only implemented in REST. Such message-oriented middlewares (MOM) are gaining popularity for various reasons. These scenarios will have to be properly analyzed, since various automation supporting core system functionalities can and should be provided by the brokering mechanism itself. I have identified the following major pros and cons of using MOM-s while working on this use case with the partners (this list is incomplete, full analysis can be found in e.g. [53]):

+ Easy to get started, available broker implementations and client skeletons in many programming languages.
+ The client initiates the connection towards the broker, so clients can be behind firewalls and Network Address Translation (NAT).
+ There can be broker clusters, hence no single point of failure (even with load balancing).
+ The client-broker communication can be secured with Secure Socket Layer (SSL) variations.
+ The publisher is not overloaded when it has to deliver to multiple subscribers.
+ Clients can even go to sleep, the broker can retain messages and deliver upon reconnection (asyncronous communication).
+ The broker can store the communications, easy to create accountable systems.

– Centralized architecture, the broker is an additional entity.
– End-to-end security (i.e. payload encryption) is hard to achieve.
– The broker can read everything, and store everything.
– If the brokering system is down, the whole network falls apart.

It is worth noting, that all IoT platforms are utilizing some sort of brokering mechanism. However, they are not interoperable mostly, even if they implement the same protocol theoretically. Therefore, we need establish how this can be handled: via translators at cloud level.

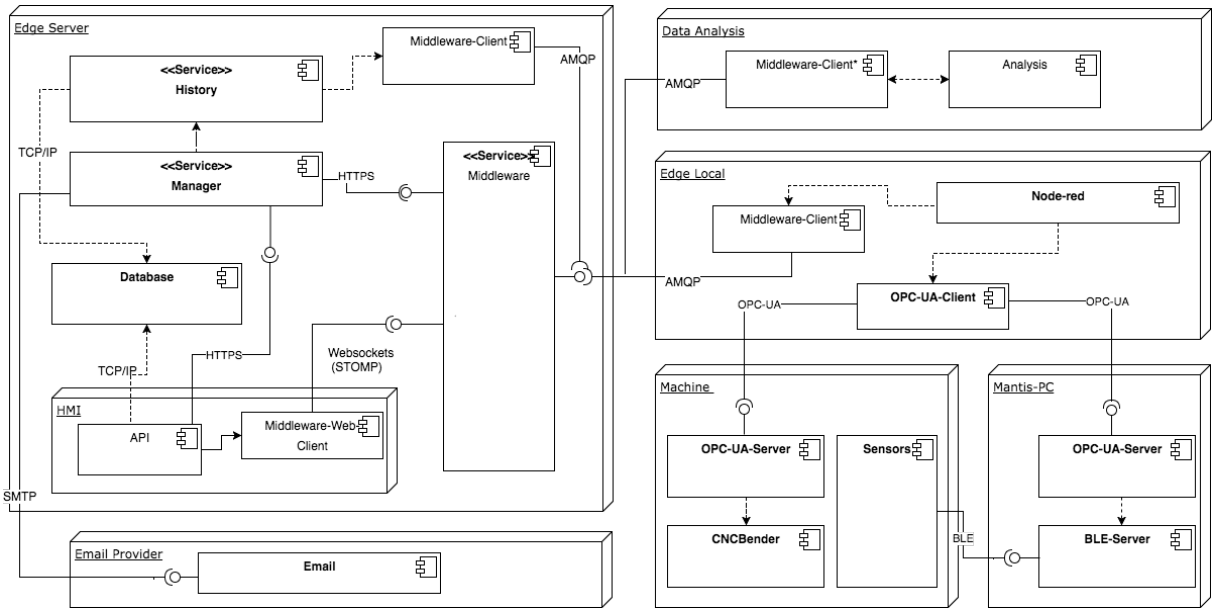

Figure 3.4: *Instantiation of MANTIS for industrial press machines1*

3.2.3 Special Purpose Vehicles

Our department (BME-VIK TMIT) is primarily involved with these two use cases. In here, we are developing the full solution for the two use cases, from sensing and data collection, up to analytic services provided for the three main targets: RUL, fault prediction and RCA. Our contributions made here are a team effort, and presented in two other papers submitted by Zoltán Umlauf [54] and Attila Frankó [55] to this conference, to different sections. My involvement here is the design and specification of the architecture with the modules and analytic services to be developed by my fellow students.

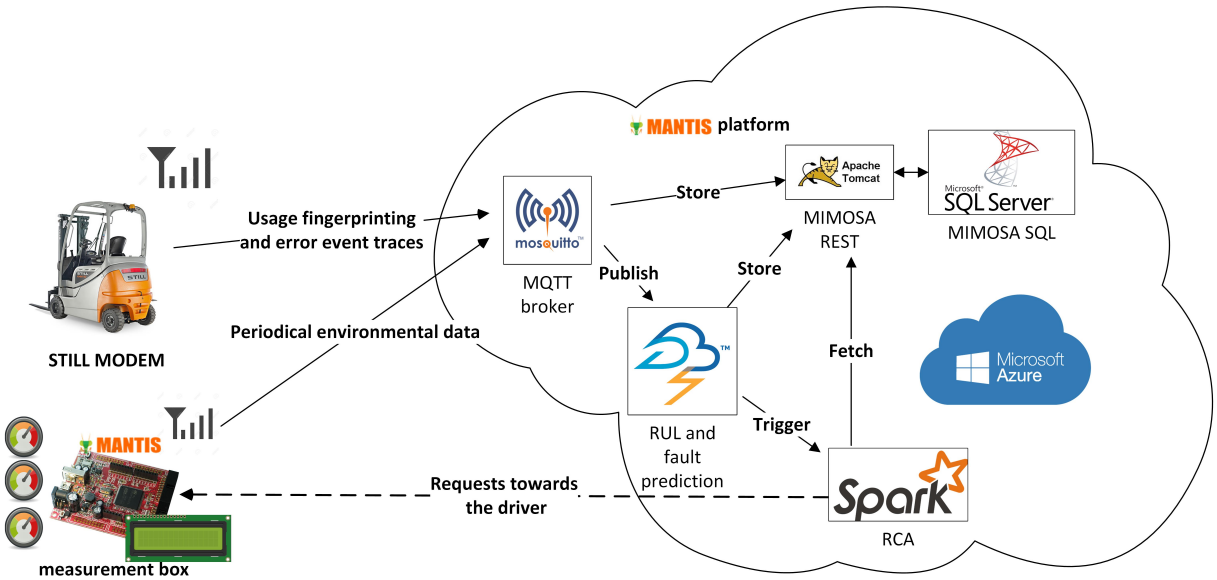

Figure 3.5: *Instantiation of MANTIS for forklifts*

Special purpose vehicles here are translating into forklifts, since STILL GmbH [56] is a German

producer of such fine machinery. The business value is there for a proactive maintenance platform, since majority of the clients are merely renting the forklifts from STILL. Therefore, the maintenance is STILL's to do, and the client expects high availability for the renting fee (as settled in the Service Level Agreement - SLA).

Since these are vehicles with their own onboard systems providing all sorts of data, it might seem easy to create a platform that collects everything (usage characteristics, environmental conditions, system telemetry) and then assumes e.g. remaining useful life for components. However, current fleet management platforms are limited, historical data is very scarce and not processable mostly (the data collection for service reports and fleet manager data has been done for legal purposes only so far). Moreover, these forklifts can only communicate via mobile networks (2G-3G-4G), and network subscription fees are very volatile and still expensive. Hence, we cannot "send everything all the time", as from a production plant.

Nevertheless, we are making headway - despite the initial hardships. Our work here is separated into three phases. Firstly, offline data mining and analysis has been done for the provided historical data sets from STILL. These datasets are related to the fleet manager system, service technician reports and specially recorded CAN bus error message traces. This concluded on what type of algorithms we can develop for fault prediction and root cause analysis, and that we need to conduct new controllered data gathering for assessing tire wearout (RUL). This phase is described by Zoltán's paper, with regard to the second phase, special attention to RUL.

Secondly, we are going to implement the algorithms on the Apache Storm and Spark systems, and firstly replay parts of the historical data (emulating as if it was coming from the trucks). Thirdly, for the final demonstration part, three test trucks at STILL's test facility will be equipped with special modem firmwares to send us online, revised set of telemetry data. The data flow architecture is presented in Figure 3.5, and the proposed stream processing nodes to be developed for Apache Storm is depicted in 3.6. This stream processing entity shall receive periodical updates on the usage and environmental conditions from the forklifts, and calculate RUL expectations based on Zoltán's developed algorithm (boosted decision tree).

In here, besides the overall architecture design and planning works, I have lead the discussions with STILL experts directly, and researched the Microsoft Azure [57] platform, on how we can implement the three cloud modules: RUL, fault prediction and RCA. As a result of this lengthy process, Figure 3.5 is established. My current works are related to MIMOSA: how to use it, what fields we can utilize for mesurements, warnings and forecast value storage. Moreover, since MIMOSA is connected to the reference MANTIS HMI instance, I am working on establishing the usage scenarios with the appropriate (Slovenian) partners. Future works is now to implement the modules both at edge device (forklift) and cloud levels.

3.2.4 Railway Switches

For the ANSALDO STS [58] rail switches use case, we are deploying the same infrastructure, but the analytic services are provided by a different, Italian partner. The endgoal is the same,

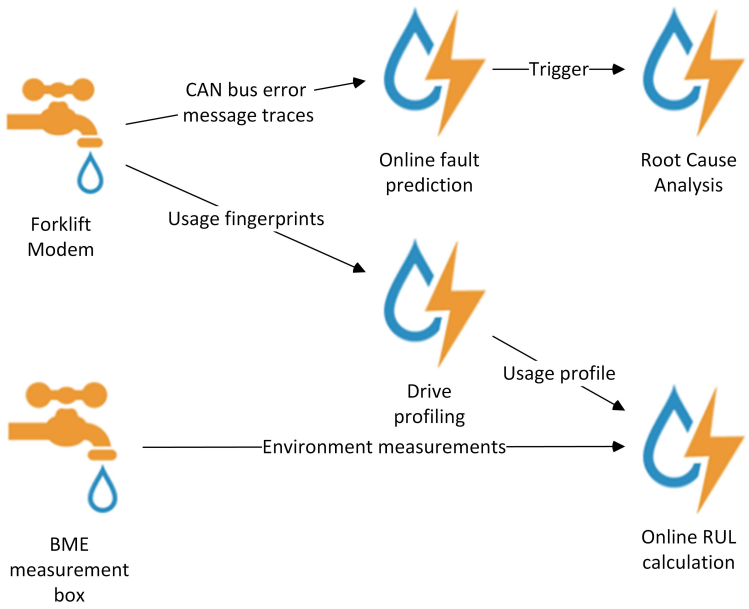

Figure 3.6: *Overview on stream processing data flow for RUL and online fault prediction engines*

create prognostic models for RUL, fault prediction and RCA. However, our main contribution here is a measurement device that will measure how a rail switch works over time, and also in what environmental conditions it is used. This device is communicating using my interoperable message ontology, and the data arrives in MIMOSA. Details of this device can be found in Attila'a paper [55].

To this end, I have written a translator module in Java, that is receiving the message over MQTT and posts it in the database, using the Tomcat 8 [59] servlet container-based REST interface (that I have extended to support other types of measurement posts as well). This translator can be deployed at the broker, and automatically detects if a new measurement is sent in. Later work might include migrating the whole edge-cloud broker into Azure, maybe using the IoT Hub implementation of MQTT, by Microsoft. Currently, the following messages are implemented both on the measurement box and the server-side as well:

- Periodical environmental measurements (every half an hour or so): rail temperature, ambient humidity and temperature;
- Movement measurements: four analog industrial-grade displacement sensors;
- Warning for error detection (switch blade stuck before reaching endpoint);
- Request on-the-spot measurements from server side.

Moreover, I have adapted the interoperable data models (to be used between the various modules). These are based on MIMOSA embedded in the IoT-A message ontology [60] as suggested by work package leadership, and submitted to the consortium for consideration, as part of my reference model proposal. The IoT-A was another European project, marked as a technical cornerstone to MANTIS. All four message types are formatted now in JSON, using this ontology.

As Figure 3.7 depicts, the overall architecture of this use case is a fairly simple one, not utilizing

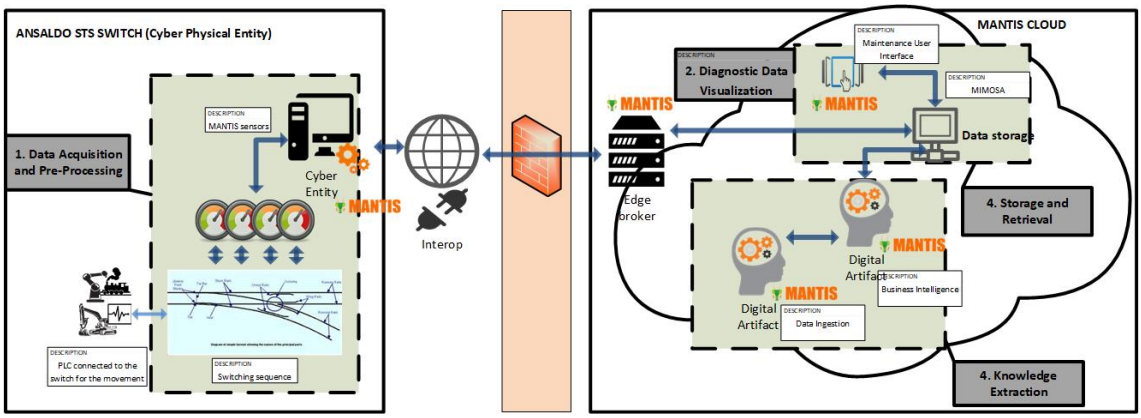

Figure 3.7: *Instantiation of the MANTIS architecture for railway switches*

advanced edge-cloud communication, as the special purpose vehicle use case will. In here, only generic IoT pattern is followed:

1. The measurement box [55] is measuring and sending in the data over MQTT [30]. This box uses a simple 2G modem, and will never even go to sleep mode over the demonstration.
2. A generic MQTT broker is receiving the interoperable messages.
3. My MQTT client is parsing the messages and posting the data in MIMOSA [42].
4. The MANTIS reference HMI is fetching the data from MIMOSA and can send the spot measurement command to the box.
5. Additional analytic services will be developed, once sufficient amount of measurement have been done over a longer time period. The final demonstrator will hopefully showcase online version of the services as well.

Chapter 4

Requirements Collection for IIoT

This chapter details my work within a freshly started project, Productive4.0. This project is aimed to utilize the results of its predecessor initiatives, such as Arrowhead or MANTIS. The architecture baseline is offered by the current Arrowhead framework (and in there my previous contributions), however, the currently targeted scope, maturity level and use case scenarios are far more extending than previously. Our department (BME-VIK TMIT) is task leader with the reference architecture design and modeling, and therefore has high stakes in the requirements collection phase as well. This chapter is presenting my work done within this, where I have developed a questionnaire to try and cluster the ten new use case visions of the project. Hopefully, we shall find commonalities and they shall be clear enough to start developing a road map for the Arrowhead framework. Use cases here are now related to:

- Tracing the product in its life cycle: during production, transport, end user usage;
- Process simulation framework for semiconductor and other chemical production plants;
- Machine and fleet management services;
- Production network reconfigurability.

4.1 Questionnaire

Since Arrowhead was primarily focusing on interoperability issues for industrial scenarios, partially delt with the shop floor issues (however, only the last entry in the list above). Therefore, it was also in focus how the ISA95 design and implementation paradigm can be shifted towards a service oriented thinking (see Chapter 1), other aspects and requirements of Industry4.0 visions were not considered. Therefore, Arrowhead in its core is a closed-network connectivity platform that aimed to establish a level of connection reconfigurability and component reuse for closed automation networks.

The current, reference implementation is based on REST. However, based on the lessons learned in MANTIS (see chapter 3) and the new use case visions within Productive4.0, Arrowhead has to reconsider its scope and purpose. Since our department (BME-VIK TMIT) is responsible for the

reference architecture modeling within this project (that is supposed to be Arrowhead-based), therefore it is necessary to analyze what is needed.

To this end, I created a questionnaire to assess three aspects mainly: (i) what type of Application Systems will be there, (ii) what are the resource management and Quality of Service (QoS) expectations, and (iii) on what level of high security is regarded. This questionnaire is formed primarily using the Arrowhead nomenclature, and assumed that that workpackage participating partners are familiar with the core framework, presented in Section 1.2. It was an interactive Google Forms form, and we have received 34 answers, both from the industrial and academic partners. After cleaning and aggregation, sadly, not all use cases have provided at least one partner's response (or at least not put the affiliation).

4.2 Results and assessment

Exactly as expected, the automation-related use cases are voting for a local network, consisting of traditional elements (like PLC-s) and some additional sensing elements (IoT motes) to be connected to monitor an already established process. In here, device replacements are not in scope, once the deployment was made, it should stay like that (1/3 of the replies). Inside such a local cloud (in the Arrowhead sense), majority of the partners indicated using the OPC framework (60%).

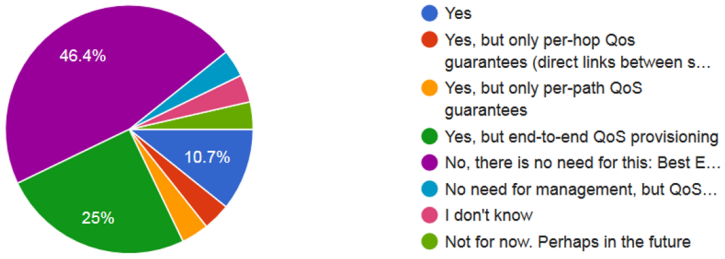

Figure 4.1: *Questionnaire answers related to QoS enforcement*

Regarding QoS, real time networks are to be established during design time: hard delays have to be engineered at network design level. For other cases, best effort is sufficient, only delivery has to be guaranteed. Some not-so-closed control loops can require on-the-fly (statistical) QoS provisioning, only indicated by one partner. However, they will also use a proprietary technology to achieve that (time sensitive networking way). This is depicted in Figure 4.1.

Responses given for data quantities is also interesting,as shown in Figure 4.2. For fixed setups, constant connections are made, and therefore data flows are almost all the time there. However, the other usage of the network is related to system logging and telemetry. These connections are small traffic, and occasional. It is a question here, whether they are also deterministic (e.g. following a scheduling) or on demand. If the latter were true, it would be necessary to verify whether the QoS of other flows are violated while this new traffic is generated or not. Follow-up questions are necessary.

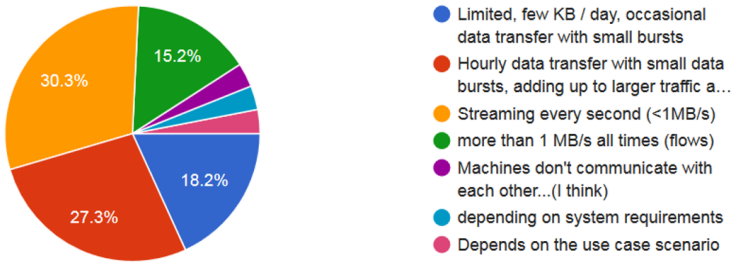

Figure 4.2: *Questionnaire answers related to data quantities*

Almost 2/3 of partners indicated a connection to a cloud platform, which, similarly to MANTIS, provide analytic services, asyncronous to the production. Here, data processing in some cases (2 answers) would have a feedback possibility to new batches in the production (quality control). For other purposes (4 answers), the tracking and tracing of products or other entities (e.g. vehicles) is the scope. This might be connecting to the previous statement about ad hoc, small traffic bursts indicated.

This also relates to the overwhelmingly indicated utilization of system logs, telemetry or the monitoring of logging interfaces (more than 2/3). Such information shall be first transported to a local storage and processed locally (1 case), and then submitted to a distributed processing platform. The transmission shall happen rarely, in a batched way. It might be even possible that the closed "gates" of the production network would be lifted for the data streaming.

Interestingly, automation partners are still overconfident in their networks being physically seggregated. Hightened security is expected, however, no security assessments are to be taken (except for 3 responses, from the proposing security expert researchers). Moreover, one use case is expecting to use block chains and distributed ledger technologies to be used for its cyber-physical production system pilot.

One major aspect for a local automation system is the visualization of data. Half of the relevant responses indicate that advanced HMI is necessary, maybe even both at edge level, and within the cloud. However, the need for user roles and access rights management could not be indicated, because the questionnaire form didn't have space for it. Therefore, follow up interviews will be necessary to conduct here.

Regarding orchestration capabilities, as Appendix 3's evaluation suggests, the partners either didn't fully understand this Arrowhead terminology, or didn't see the applicability to their use cases in most (here, only the industrial partner responses shall be validated, since they are the pilot owners). In there, I have assessed the partners' answers to the orchestration capabilities and security on a subjective scale of 1-5. Also, the raw answers are depicted on Figure 4.3. These two diagrams show that devices will not be replaced or newly introduced often, and their role is also fixed most of the time. As indicated here, there might be need for rearrangement possibility, that can be provided by late binding (and new system configuration pushes).

On a sidenote, there are two complete project tasks related to process simulation. Here, Arrowhead is not considered yet, because a proprietary simulation framework will be used. The

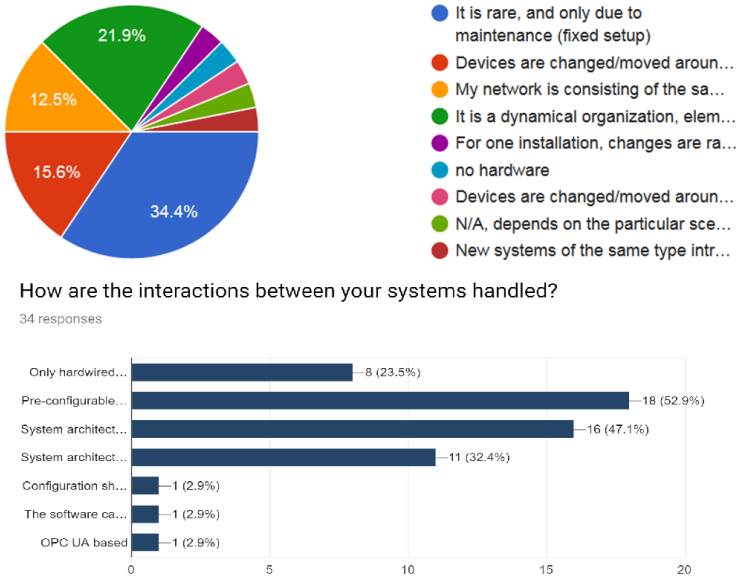

Figure 4.3: *Questionnaire answers related to orchestration capabilities*

integration of this is not in scope for this paper.

4.3 Roadmap for Future Developments: Arrowhead 4.0

Based on these responses, my assessment is that there are a couple of urgent matters need attending to be resolved as Arrowhead develops its fourth iteration. This section explores my proposals for augmenting the Arrowhead framework (described in Chapter 2) on these six topics. These are related to:

1. Legacy adapters and translator motes
2. Arrowhead core implementations using publish-subscribe protocols (OPC-UA)
3. A Logger core System
4. Arrowhead Local Clouds connecting to global clouds
5. Arrowhead core implementations for platforms (IoT gateways, cloud platforms)
6. Arrowhead's AAA features (authentication, authorization and accountability)

The first two cases are related to edge computing, and making CPPS-s. Arrowhead assumed so far, that all systems are capable of (i) communicating with the Arrowhead core, (ii) providing its services in one of the protocols related to IoT and (iii) consume various other services. This can be achieved by either (i) writing the whole software in an Arrowhead-compliant manner, (ii) modifying the software on a legacy device or (iii) adding an extra hardware adapter that is facing the legacy device in its primary interface, e.g. via serial port.

This idea is representing itself in the responses as well. Having an OPC-UA local network assumes the integration via translators, or better said: adapters. Arrowhead should provide a couple of adapter implementations, ready to be used for the pilot projects. This was not a focal point before, every use case expected to develop its own. However, with this wider range of partners,

if Arrowhead wishes to be a viable platform, it should invest in these. Hardware targets can be ARM-based embeddable devices, such as the Raspberry Pi 3B (as we used it in MANTIS, a first prototyping tool). There are many alternatives in hardware, some even capable of industrial grade resilience, yet the software would still be the same. The platform architecture work package should issue follow up questions to the use cases on what the target environment should be. This would also require, that the Arrowhead client skeletons (Arrowhead Provider and Consumer interfaces) are reproduced in not just Java, but other languages as well.

A more pressing design issue is related to publish-subscribe mechanisms, and their popularity. Arrowhead was primarily imagined for a request-response protocol (i.e. REST). Surely, REST (HTTP) servers and requests are easy to code (even on embedded), but the overwhelming advantages of publish-subscribe protocols (that were also expressed here, in Chapter 3) are pointing towards their more widepsread use. Next steps shall include the careful examination of the OPC-UA stack, and deciding on what and how needs to be re-implemented there, from an Arrowhead core point of view. Service Discovery and Authorization System functionalities are already provided by this protocol stack, orchestration needs to be solved – and runtime central governance as well.

Thirdly, in here (and also in MANTIS) we are facing a very prevalent scenario: we want to utilize logs from various machines. These might be syslogs [61], or any custom logging format. We need a new supporting Core System that shall take the logs from Application Systems in two ways: the systems themselves submit their entries over time or the logger periodically fetches it from the device. In the ADIRA use case (Section 3.2.2) for example, industrial press machine logs is fetched via simple File Transfer Protocol (FTP) periodically and the result is a Comma Separated Values (csv) file, stored in Hadoop.

This is necessary for Arrowhead as well. Arrowhead shall provide a new supporting Core System that provides this feature, or change the primary purpose for the Historian. Furthermore, this logger functionality has to support various formats, yet all log information should be stored in one way, in one storage. To this end, this Core System should be modular in a sense that log format templates could be added to help parsing the inbound message types. These addons should be easy to create, and should use an appropriate descriptive language. Also, the integration with popular big data storage technologies – such as Hadoop – might be wise. Nevertheless, this System and its interfaces should support legacy devices, as well as publish-subscribe protocols used. Moreover, push and pull has to be supported: for the latter case, a handshake procedure is needed as well. It is also worth exploring, how this logging might be connected to the orchestration process.

Regarding the fourth (and with that the fifth) item on this list, Arrowhead has to adapt to the a general use case architecture: edge and cloud-based computing. My inter-cloud architecture, and the new Gateway implementation is pointing towards this concept. As depicted in 3.2, Arrowhead should facilitate periodical or ad hoc connections outside of its closed network (”Local Cloud”). Arrowhead (and within that the inter-cloud collaboration mechanisms introduced by me) assumes that the Local Cloud is governed by the Core Systems, and all connection attempts happen with

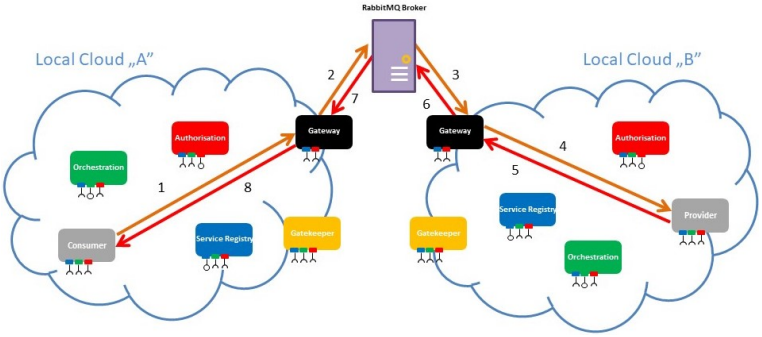

Figure 4.4: *Data path realization inbetween Arrowhead Local Clouds [45]*

the sanction of these Core Systems: this is necessary for security reasons, as discussed in our previous article [62]. Therefore, when an Application System "wishes" to connect somewhere, it should request orchestration: this orchestration process then negotiates with another Cloud, if necessary, to establish the connection. The data path is also built up during this inter-cloud orchestration process, as Figure 4.4 [45] presents using a generic, RabbitMQ broker.

However, in order for this mechanism to work, the Arrowhead core has to be available for IT cloud platforms. In here, we can think of virtual machine images, Docker or Tomcat containers, or even some other modular, easily deployable implementation. This needs to be put up as follow up questions for the pilots. The generic idea is, however, that we need have the support for an IT cloud platform to become Arrowhead Local Cloud as is.

Moreover, the Gateway module has to be adapted, so that it would be somewhat related to the brokering mechanism used by the cloud platform itself. For example in Azure [57], the gateway towards devices is called "IoT Hub", a special implementation of MQTT over SSL. This technology has to be connected to my Gatekeeper module and the inter-cloud orchestration process. The IoT Hub can be abstracted as an implementation of the Gateway module – with some additional security mechanisms.

Finally, Arrowhead's Authentication-Authorization-Accounting (AAA) capabilities have been somewhat limited, especially regarding the last 'A'. So far, authentication was tasked to the transport layer using the Public Key Infrastructure (X.509 certificates, using SSL connections [63]). Authorization was built upon it in my development version, as discussed in my last conference paper [5]. Accounting, on the other hand, requires that we are aware of any use case specific requirements. I have suggested a separate working group to be established for this purpose.

Conclusion

This work was presented in three parts. Firstly, the generic understanding of industrial IoT and cyber-physical (production) system is established. The motivation and how-to is given, in the context of the Arrowhead framework. Additionally, as a fundamental research, I have given an brief overview on IoT solutions: "things" and "clouds". This work was part of the result of our paper regarding IoT security. Moreover, I presented what different the extra I in IIoT makes, how are we moving away from standard engineering solutions towards something new.

Secondly, a specific IIoT project was shown in MANTIS, in which we are using the concepts and popular technologies of IoT to realize very a specific and unique industrial scenario: enable proactive maintenance for a handful of industries. In here, my work is related to a generalized architecture design, and research on possible implementation directions. Moreover, I have also participated in the high level design and negotiations of four use case pilot demonstrators, and within those, in two as major contributor. These included German forklifts, Italian railway switches, Finnish energy production plant and Portuguese press machines. In the first two use cases, I am working together with two other MSc students [55, 54], and we are developing a full IoT solution with devices, a cloud architecture and various big data analytic services. My involvement here is the design and specification of the architecture with the modules and analytic services to be developed by my fellow students. Moreover, my work in the latter two use cases has already resulted in one conference paper describing the approach taken in a Portuguese pilot [43]. This paper is being re-submitted for a journal edition.

Thirdly, I have presented my requirements collection work done in the Productive4.0 project, pointing towards the development of a full stacked IIoT solution to become of Arrowhead. As a first general, orientating trial, I have created a questionnaire to make the project consortium start thinking in a service-oriented, Arrowhead way. The results are also analyzed in this paper, along with a possible road map given for the architecture working group.

To conclude, the IoT paradigm is also interesting for industrial partners as well. However, their generic skepticism towards IoT technologies is more or less understandable, there are still many problems to solve. It is easy to create a sensor mote based on rapid prototyping, such as Arduino, however making it industrial grade is a huge challenge. Not just on a networking level (i.e. wireless in challenging environments), but on application level as well. These new solutions have to be cost effective, yet be able to replace traditional proven technologies – and even provide new functionalities as well. Cloud technologies are also promising, however their business and legal aspects are still a hold-back for many cases.

Also, the current maturity level of IIoT solutions, in my experience and based on literature, is still in the early stages: partners are assessing what is even possible to do, and how. There are many ideas floating around, from dynamically re-orchestrated production plants with individually customized products, up to assembly line machines utilizing blockchains to measure the exact cost of making for the product itself. Resolutions of the arising issues and contradictions is still ahead this time: managing the expectations and maintaining "industrial grade".

Regarding future endeavors, I intend to start executing the road map detailed in Chapter 4, and also finishing the remaining tasks in my four use cases within MANTIS (as discussed in Chapter 3) as my MSc thesis work – or even further.

Bibliography

[1] The Productive4.0 project consortia, "The productive4.0 project website," http://productive40.eu/.

[2] The MANTIS consortium , "The mantis project website," http://www.mantis-project.eu/.

[3] The Arrowhead consortia, "The arrowhead framework wiki," https://forge.soa4d.org/plugins/mediawiki/wiki/arrowhead-f/index.php/Arrowhead_Framework_Wiki.

[4] C. Hegedus, "Making things collaborate," http://tdk.bme.hu/VIK/Intelligens1/Felhok-kozti-egyuttmukodes-megteremtese-az, 2015.

[5] ——, "Cyber-physical systems collaborating in a service oriented architecture," http://tdk.bme.hu/VIK/Halozatok1/Kiberfizikai-berendezesek-egyuttmukodese-egy, 2016.

[6] Industrial Internet Consortium, "Reference architecture model for industrie 4.0," https://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/rami40-an-introduction.pdf?__blob=publicationFile&v=4, 2015.

[7] M. Hermann, T. Pentek, and B. Otto, "Design principles for industrie 4.0 scenarios," in *49th International Conference on System Sciences, USA*, 2016.

[8] Industrial Internet Consortium, "The industrial internet of things reference architecture," 2017.

[9] B. Scholten, "The road to integration: A guide to applying the isa-95 standard in manufacturing," International Society of Automation, 2007.

[10] J. Delsing, *IoT Automation: Arrowhead Framework*. CRC Press, 2017.

[11] I. E. Commission, *Security for Industrial Automation and Control Systems*, Std. IEC 62 443, 2003–2007.

[12] N. Instruments, *Fundamentals, System Design, and Setup for the 4 to 20 mA Current Loop*, Std., 2017.

[13] E. Tovar and F. Vasques, "Real-time fieldbus communications using profibus networks," *IEEE Transactions on Industrial Electronics,*, vol. 46, pp. 1241–1251, 1999.

[14] J. Powell, "Profibus and modbus: a comparison," https://www.automation.com/automation-news/article/profibus-and-modbus-a-comparison.

[15] M. Cengarle, S. Bensalen, J. McDermid, R. Passerone, A. Sangiovanni-Vincetelli, and M. Torngren, "Characteristics, capabilities, potential applications of cyber-physical systems: a preliminary analysis," 2013.

[16] T. Erl, *Service-Oriented Architecture (SOA) Concepts, Technology and Design*, 2005.

[17] OASIS, "Organization for the advancement of structured information standards." http://www.oasis-open.org.

[18] M. Bell, *Introduction to Service-Oriented Modeling.* Wiley and Sons., 2008.

[19] M. H. Valipour, B. Amirzafari, K. N. Maleki, and N. Daneshpour, "A Brief Survey of Software Architecture Concepts and Service Oriented Architecture," in *2nd IEEE International Conference on Computer Science and Information Technology*, 2009, pp. 34–38.

[20] W3C, "The World Wide Web Consortium," https://www.w3.org/.

[21] C. H. Crawford and G. P. Bate, "Towards an on demand service-oriented architecture," in . *IBM Systems Journal*, vol. 44, 2005, pp. 81–107.

[22] L. L. Ferreira, M. Albano, and J. Delsing, "QoS-as-a-Service in the Local Cloud," in *21th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'2016)*, 2016.

[23] O. Carlsson, P. P. Pereira, J. Eliasson, and J. Delsing, "Configuration service in cloud based automation systems," in *21th IEEE Conference on Emerging Technologies and Factory Automation*, 2016.

[24] O. Carlsson, D. Vera, J. Delsing, and B. Ahmad, "Plant descriptions for engineering tool interoperability," in *14th International Conference on Industrial Informatics*, 2016.

[25] H. Derhamy, P. Varga, J. Eliasson, J. Delsing, and P. Punal, "Translation error handling for multi-protocol soa systems," in *20th IEEE International Conference on Emerging Technologies and Factory Automation (EFTA)*, 2015.

[26] C. Hegedűs and P. Varga, "Service interaction through gateways for inter-cloud collaboration within the arrowhead framework," in *5th International Conference on Wireless Communications, Vehicular Technology, Information Theory, Aerospace and Electronic Systems (VITAE)*, Hyderabad, India, 2015.

[27] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, "A gap analysis of internet-of-things platforms," *Computer Communications*, vol. 89, pp. 5–16, 2016.

[28] R. T. Fielding and R. N. Taylor, *Architectural styles and the design of network-based software architectures.* University of California, Irvine Doctoral dissertation, 2000.

[29] Z. Shelby, K. Hartke, and C. Bormann, *The Constrained Application Protocol (CoAP)*, IETF RFC 7252, 2014.

[30] OASIS, "Message queue telemetry transport," http://mqtt.org/, 2017.

[31] R. Baldoni, M. Contenti, and A. Virgillito, "25. the evolution of publish/subscribe communication systems."

[32] Apache, "The kafka distributed streaming platform," http://kafka.apache.org/, 2017.

[33] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid.* IEEE Computer Society, 2009, pp. 124–131.

[34] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci *et al.*, "Windows azure storage: a highly available cloud storage service with strong consistency," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 2011, pp. 143–157.

[35] J. Wu, L. Ping, X. Ge, Y. Wang, and J. Fu, "Cloud storage as the infrastructure of cloud computing," in *Intelligent Computing and Cognitive Informatics (ICICCI), 2010 International Conference on*. IEEE, 2010, pp. 380–383.

[36] Apache, "Hadoop distributed file system," http://hadoop.apache.org/, 2017.

[37] M. Hausenblas and N. Bijnens, "The lambda architecture website," http://lambda-architecture.net/, 2017.

[38] Apache, "Storm stream processor," http://storm.apache.org/, 2017.

[39] P. Varga, S. Plosz, G. Soos, and C. Hegedus, "Security threats and issues in automation iot," in *Factory Communication Systems (WFCS), 2017 IEEE 13th International Workshop on*. IEEE, 2017, pp. 1–6.

[40] C. Donnelly, "Eu data protection regulation: What the ec legislation means for cloud providers," http://www.computerweekly.com/feature/EU-Data-Protection-Regulation-What-the-EC-legislation-means-for-cloud-providers, 2017.

[41] E. Jantunen, U. Zurutuza, L. L. Ferreira, and P. Varga, "Optimising maintenance: What are the expectations for cyber physical systems," in *Emerging Ideas and Trends in Engineering of Cyber-Physical Systems (EITEC), 2016 3rd International Workshop on*. IEEE, 2016, pp. 53–58.

[42] M. consortium, "The mimosa project site," http://www.mimosa.org/, 2016.

[43] E. Jantunen, U. Zurutuza, M. Albano, G. di Orio, P. Maló, and C. Hegedus, "The way cyber physical systems will revolutionise maintenance," in *30th Conference on Condition Monitoring and Diagnostic Engineering Management*, 2017.

[44] Fortum, "Company website," https://www.fortum.com/frontpage/com/en/?from=irene, 2017.

[45] N. Szeles, "Secure, service-oriented data communication inbetween local iot clouds," http://tdk.bme.hu/VIK/Halozatok2/Lokalis-IoT-felhok-kozotti-biztonsagos1, 2017.

[46] Adira, "Company website," http://www.adira.pt/default.aspx?, 2017.

[47] Raspberry, "Raspberry pi website," https://www.raspberrypi.org/.

[48] Arduino, "Arduino website," https://www.arduino.cc/.

[49] OPC Foundation, *OPC Unified Architecture Specification*, Standard IEC 62 541, 2010.

[50] Node-Red, "Website," https://nodered.org/.

[51] RabbitMQ, "Website," https://www.rabbitmq.com/.

[52] R, "Website," https://www.r-project.org/, 2017.

[53] E. Curry, "Message-oriented middleware," *Middleware for communications*, 2004.

[54] Z. Umlauf, "Data processing methodologies and solutions for proactive maintenance of cyber-phsical systems," http://tdk.bme.hu/VIK/Jelfeld/Adatfeldolgozasi-modszerek-es-megoldasok, 2017.

[55] A. Frankó, "Information-driven proactive maintenance strategies," http://tdk.bme.hu/VIK/Info/Informaciovezerelt-proaktiv-karbantartasi, 2017.

[56] STILL, "Company website," http://www.still.hu/, 2017.

[57] Microsoft, "The azure platform," https://azure.microsoft.com/, 2017.

[58] Ansaldo-STS, "Company website," http://www.ansaldo-sts.com/en, 2017.

[59] Apache, "Tomcat servlet container website," http://tomcat.apache.org/, 2017.

[60] IoT-A, "Project website," http://www.meet-iot.eu/Meet-iot-a.html, 2017.

[61] R. 5424, "The syslog protocol," https://tools.ietf.org/html/rfc5424, 2009.

[62] S. Plosz, C. Hegedus, and P. Varga, "Advanced Security Considerations in the Arrowhead Framework," in *DECSoS*, September 2016, pp. 1–13.

[63] T. S. S. I.-T. International Telecommunication Union, "X.509: Public-key and attribute certificate frameworks," http://www.itu.int/rec/T-REC-X.509-201210-I/en, 2012.

Appendices

.1 Appendix 1

Productive4.0 Service Platform Questionnaire

Think of your system or system-of-system, that you **aim to create**, further develop, or enhance during Productive4.0. Think about what is **your desired outcome** for your systems during Productive4.0.

Try to answer these questions with that in mind. These questions are aimed to discover the focal points of the project use cases and in what scenarios the service platforms are to be deployed in.

Questions marked with red * are mandatory, although if possible, elaborate on those that are not mandatory, as well.
Note that Squares represent MULTIPLE choices - whereas Circles represent SINGLE choices.

This questionnaire will take about 15 minutes of your time.
Thank you!

*Required

1. **Email address** *

2. **Your company** *

3. **Your name** *

4. **What is your position in the company?** *

5. **Please describe your use case's objective and/or demonstration system scenario plans *briefly*. (Optional)**

Skip to question 5.

Your Application Systems' capabilities

The purpose of this section is to assess what systems are already there - and what can we build upon later on in the project.
Here, we shall collect information connected to the to-be-used systems in the use cases.

NOTE that a "system" here can be any addressable entity (through IP or other protocol), may it be hardware, software embedded in hardware, or software module(s).

NOTE that "system-of-systems" is a set of systems collaborating with each other to reach greater goals.

6. **What sort of elements are in your own Productive4.0 target system-of-systems in mind? (More than one answer can be ticked.)** *

Tick all that apply.

- [ ] Sensors, (wireless) sensor networks
- [ ] Controlled actuators
- [ ] IoT gateways
- [ ] Data processing elements locally
- [ ] Data processing elements in a private (or local) cloud / server cluster
- [ ] Distributed (e.g. big data analytic) systems in a public cloud (e.g. Hadoop, Mapreduce, Azure, etc.)
- [ ] Advanced Human Machine Interface (Web browser, Mobile Application, or even more modern...)
- [ ] Other: _____

7. **If possible, please give a short description about the *purpose*, *processing* and *communication capabilities* of your application systems.**

_____

_____

_____

_____

_____

8. **Please detail your targeted deployment scenarios and use case objectives, if possible.**

_____

_____

_____

_____

_____

9. **How do you rate your system-of-systems in terms of resource constrained devices? There are ...** *

Tick all that apply.

- [ ] Resource constrained, embedded devices (e.g. low-power 8- or 32-bit CPU and running on battery)
- [ ] Highly capable embedded devices (e.g. multi-core processors but still battery operated, like smart phone strength)
- [ ] Legacy (industrial) PC-s (fixed power supply, e.g. old x86 architecture, Windows NT, etc.)
- [ ] Systems with no resource constraints at all (e.g. PC-s, multi-core, multi network interface, etc.)
- [ ] Other: _____

10. **You may elaborate on your resource constraints here.**

11. **How are the interactions between your systems handled?** *
Tick all that apply.

- [ ] Only hardwired connections between my systems
- [ ] Pre-configurable connections that almost never change
- [ ] System architecture and elements can change regularly - through human interaction
- [ ] System architecture and elements can change regularly - dynamically
- [ ] Other:

12. **System management and diagnostic capabilities (Tick if 'yes')** *
Tick all that apply.

- [ ] Do you utilize their logs? Do you transmit reports over the network to another location (e.g. database)?
- [ ] Do your systems have system management interfaces?
- [ ] Can you remotely push new firmware/system configuration onto them?
- [ ] Do you have a central configuration server which provides the config files?
- [ ] Do you have a secure boot image loading mechanism over the network?
- [ ] Other:

13. **Please detail how system management and diagnostics is handled, if possible.**

14. **How regular is to introduce/change a device in your system-of-systems (network)?**
Mark only one oval.

- ◯ It is rare, and only due to maintenance (fixed setup)
- ◯ Devices are changed/moved around for re-organization of the network, but rarely
- ◯ My network is consisting of the same devices in general, but their places/roles are changed frequently.
- ◯ It is a dynamical organization, elements come and go (e.g. market-like scenarios)
- ◯ Other:

15. **How are system elements and components identified and described? Describe *roughly* what type of identifiers and metadata are stored per a system. (e.g. ID-s, location, network addresses, MAC address, textual description, etc.)**

16. **Do you follow any metadata or data descriptive ontology schema/standard? (e.g. SenML)**

Skip to question 16.

Network, QoS and resource management

This section is aimed at assessing network and resource management conditions in the use cases.

17. **Are there low-latency requirements in your system?** *
Mark only one oval.

- ◯ Real time, absolute delay < 10 ms
- ◯ Soft real time, e.g. mean delay < 100 ms
- ◯ On-the-fly, 100- 500 ms mean delay
- ◯ Best effort, sec-range response
- ◯ Other:

18. **You may describe your real-time requirements in more detail here.**

19. **How much data is transmitted between elements (one-to-one) in general?** *
Mark only one oval.

- ◯ Limited, few KB / day, occasional data transfer with small bursts
- ◯ Hourly data transfer with small data bursts, adding up to larger traffic a day (e.g. couple MB-s)
- ◯ Streaming every second (<1MB/s)
- ◯ more than 1 MB/s all times (flows)
- ◯ Other:

20. **You may describe your data flows here, briefly.**

21. **What other QoS guarantees do you need in your network?** *
Tick all that apply.

- [ ] guarantees for message delivery (e.g. 100%, at least once/at most once)
- [ ] guarantees for given (range of) latency
- [ ] guarantees for given throughput
- [ ] Other: _____

22. **You may detail your Quality of Service needs here.**

_____

23. **Do you need centralized QoS enabling and enforcing governance? (QoS management system)**
Mark only one oval.

- ◯ Yes
- ◯ Yes, but only per-hop Qos guarantees (direct links between systems)
- ◯ Yes, but only per-path QoS guarantees
- ◯ Yes, but end-to-end QoS provisioning
- ◯ No, there is no need for this: Best Effort is enough in all communications
- ◯ Other: _____

24. **What is/will be your network infrastructure made of? (More than one option)** *
Tick all that apply.

- [ ] Devices access through wireless, IP-based network (e.g. Wi-Fi)
- [ ] Devices access through wired, IP-based infrastructure (e.g. Ethernet)
- [ ] Switches and routers are part of the infrastructure
- [ ] There are/will be redundancies (multiple, redundant links, connections).
- [ ] There are redundant networking elements (or load balancing between elements)
- [ ] It partially uses ungoverned infrastructure (e.g. goes through Internet)
- [ ] There are elements in a private cloud/server cluster
- [ ] There are elements in a public cloud
- [ ] I use proprietary/legacy networking technology.
- [ ] Other: _____

25. **What kind of application protocols do you intend to use between your systems?** *
Tick all that apply.

- [ ] Request-response: e.g. REST, FTP, CoAP, SOAP
- [ ] Publish-subscribe: e.g. MQTT, AMQP, OPC-UA
- [ ] Push model: e.g. WebSocket, Comet
- [ ] Non IP-based: e.g. elements communicating on Modbus, Profibus, 4-20 mA
- [ ] Other: _____

26. **What protocols and networking technologies do you intend to use precisely? Please, elaborate.**

Skip to question 26.

Security

27. **How strict are your security and safety requirements for this system of systems?** *
Mark only one oval.

	1	2	3	4	
Very Low: my deployment is within a safe and guarded environment	◯	◯	◯	◯	Very High: potential attack is probable and rewarding

28. **How strict is (going to be) your network security in the project use case?** *
Mark only one oval.

- ◯ Physically segregated network, no connection outbound
- ◯ Closed network, but with occasional short connections to the outside world (e.g. for backing up)
- ◯ Business sensitive network with connection to the Internet, but secured (e.g. firewall or VPN)
- ◯ Services and interfaces are provided towards the open Internet (maybe even unsecured)
- ◯ Other: _____

29. **Do you have resource constrained devices that you would like to secure?**

30. **Do you force users to follow a security procedure when they bootstrap/introduce a new device in the network?**

31. **What security aspects do you need?** *

Tick all that apply.

- [ ] Parties mutually authenticated
- [ ] Message integrity assured (data integrity verified)
- [ ] Authorization checked and guaranteed (access control, data confidentiality)
- [ ] The communication encrypted
- [ ] Data availability at all times
- [ ] Other: _____

32. **You may provide additional comments, questions or ideas regarding the previous question.**

_____

_____

_____

_____

_____

33. **Do you already have any specific *Authentication-Authorization-Accounting* (access control) technologies already implemented?**

_____

_____

_____

_____

_____

34. **Do you plan to carry out a complete security assessment of your deployment (use case scenario)? (e.g. based on STRIDE)** *

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Maybe
- ◯ Other: _____

Layer	Threat type	Mitigation
Physical	Tampering	tamper-resistant packaging
	Denial of Service	spread-spectrum techniques
Networking	Denial of Service	active firewalls, passive monitoring (probing), traffic admission control, bi-directional link authentication
	Eavesdropping	encryption, authorization
Data processing	Back door attack	properly configured firewalls on all system entry point
	Social Engineering	educating employees to security awareness
	Exhaustion	traffic monitoring
	Malware	malware detection
Application	Client app.	anti-virus filtering
	Comm. channel	proper authentication, authorization, integrity verification
	Integrity	testing
	Modifications	validation
	Multi-user access	process planning and design
	Data access	Traceability

Figure 5: *Security threats within IoT systems [39]*

.2 Appendix 2

.3 Appendix 3

1		Real-time (1: best effort, 5: hard RT)	Device capability (1: resource constrained 5: no constraints)	Security (1: no security req 4: highest security)		Orchestration dynamics (1: hardwired 5: fully dynamic, a: config push also)
2	univeristy of applied sciences burgenland	1	3	4	134	5
3	Agileo Automation	4	2	2	422	2
4	CEA	4	3	3	433	3
5	TWT GmbH	1	3	1	131	1
6	Eindhoven University of Technology	1	5	1	151	1
7	Unger Fabrikker	3	4	3	343	1
8	MGEP	5	3	4	534	1
9	Philips Lighting	3	4	1	341	4
10	Bosch	3	5	4	354	5
11	SimPlan AG	1	5	1	151	2
12	Universität Klagenfurt	1	5	3	153	1
13	AIT	3	2	4	324	5
14	Konecranes	4	2	3	423	1
15	Infineon Technologies Austria AG	1	2	1	121	1
16	TTTech Computertechnik AG	5	5	4	554	1
17	TU Braunschweig	4	3	1	431	1
18	IMA s.r.o.	3	2	3	323	4
19	Institut fuer Automation und Kommunikation e.V.	5	2	4	524	3
20	NXP Semiconductors GmbH Germany	5	1	3	513	5
21	BnearIT AB	3	2	2	322	4
22	tno	5	4	1	541	3
23	Bosonit S.L.	4	4	2	442	5
24	Tampere University of Technology	3	3	3	333	3
25	Aalborg University	4	2	3	423	4
26	MSI Grupo	3	4	3	343	1
27	GUEP Software GmbH	1	5	4	154	1
28	fortiss GmbH	5	1	1	511	5
29	evopro Innovation Ltd.	1	2	3	123	2
30	Philips CL (05A)	5	4	3	543	5
31	Metso	1	1	3	113	4
32	Wapice Oy	5	2	4	524	5
33	CrossControl Oy	4	2	3	423	3
34	Trimek	3	4	3	343	1

Figure 6: *Responses regarding orchestration capabilities*