TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

MEHDI MAHMOODPOUR

ROLE-BASED DATA VISUALIZATION FOR INDUSTRIAL IOT

Master of Science Thesis

Examiners: Prof. Jose L. Martinez Lastra, Assoc. Prof. Andrei Lobov
Examiners and topic approved on 01 November 2017

ABSTRACT

MEHDI MAHMOODPOUR: ROLE-BASED DATA VISUALIZATION FOR INDUSTRIAL IOT
Tampere University of Technology
Master of Science Thesis, 65 pages
August 2018
Master's Degree Program in Automation Engineering
Major: Factory Automation and Industrial Informatics
Examiners: Prof. Jose L. Martinez Lastra, Assoc. Prof. Andrei Lobov

Keywords: Internet of Things, IoT platform, data collection, data visualization, factory automation, ontology

The competition among manufacturers in the global markets calls for the enhancement of the agility and performance of the production process and the quality of products. As a result, the production systems should be designed in a way to provide decision-makers with visibility and analytics. To fulfill these objectives, the development of factory information systems in manufacturing industries has been introduced as a practical solution in the past few years. On the other hand, the volume of data generated on the factory floor is rising. To improve the efficiency of manufacturing process, this amount of data should be analyzed by decision-makers. To cope with this challenge, visualization assists decision-makers to gain insight into data. To give a better perspective of collected data to decision-makers, effective visualization techniques should be employed. Adequate data visualization allows the end user to have better understanding of data and make effective decisions faster. Meanwhile, the adoption of the Service-Oriented Architecture (SOA) and Internet of Things (IoT) as state-of-the-art technologies are among the most prominent trends in industrial automation. IoT technology is expected to generate and collect data from various sensors and devices within the production system, and enables enterprises to have real-time visibility into the flow of production process. Moreover, data received from factory floor should be transmitted from back-end side to the front-end side for future analysis. To implement the exchange of data efficiently, the solution should support different communication protocols to make interoperability among heterogeneous devices on shop floor.

This study describes an approach for building a role-based visualization of industrial IoT. An extensible architecture was provided by which the future growth of data and emerging new protocols has been anticipated. By using the IoT platform introduced in this thesis, selected KPIs can be monitored by different levels of enterprise. Three prototype IoT dashboards have been implemented for a pilot production line, "Festo didactic training line" located in Seinäjoki University of Applied Sciences (SeAMK) and results have been validated.

PREFACE

The work done throughout this study was carried out at Mechanical Engineering and Industrial Management (MEI) Department at Tampere University of Technology. Firstly, I would like to thank Professor Lastra under whose direction this thesis was conducted. Thanks are also extended to Professor Minna Lanz for giving me the opportunity to work on my thesis in MEI Department. Moreover, I should express my deep gratitude to my supervisor Professor Andrei Lobov for his generous support, guidance and encouragement throughout the thesis. Last but not least, thanks to my other colleagues in MEI Department for creating a delightful atmosphere to work conveniently.

I want to thank my father, brother and specially my mother who has been my supporter for every stage I have embarked on in my life. Words are not enough to express my gratefulness for my mother's endless sacrifice through my whole life. I would like to thank my wife for all the love she gave me. I should confess from the bottom of my heart that without her support this success would not have happened. We took each single step together all along. She was always beside me in ups and downs of our life. Also, I would like to mention my son's impressive effect on my work. Although he is not old enough to read this text at the moment, I am writing for the future that he was my motivation to carry on my journey. He was the one who always boosted my energy for the next day.

I like to finish this script with the poem from Omar Khayyam a great Persian poet:

"Be happy for this moment. This moment is your life."

Tampere, 15.08.2018

Mehdi Mahmoodpour

CONTENTS

LIST OF FIGURESS

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

API	Application Programming Interface
AI	Artificial Intelligence
CPS	Cyber-Physical Systems
DPWS	Device Profile for Web Services
ERP	Enterprise Resource Planning
HMI	Human Machine Interface
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
I/O	Input/Output
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology
KPI	Key Performance Indicator
JSON	JavaScript Object Notation
M2M	Machine to Machine
MES	Manufacturing Execution System
MQTT	Message Queuing Telemetry Transport
OPC	Object Linking and Embedding for Process Control
OPC UA	OPC Unified Architecture
OWL	Web Ontology Language
PLC	Programmable Logic Controller
REST	Representational State Transfer
RDF	Resource Description Framework
RFID	Radio Frequency Identification
RTU	Remote Terminal Unit
SCADA	Supervisory Control And Data Acquisition
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
TCP	Transmission Control Protocol
UI	User Interface
UDP	User Datagram Protocol
URI	User Resource Identifier
URL	Uniform Resource Locator
WS	Web Service
WSDL	Web Service Description Language
XML	Extendable Markup Language
XSD	XML Schema Definition

1. INTRODUCTION

This chapter outlines a presentation to the work presented in this thesis. The background of the topic followed by the problem definition and work description of the thesis is dicussed. Furthermore methodology is depicted in this chapter. Finally, the outline of the thesis is represented in the last section.

1.1 Background

The volume of data which is being generated in the current era has increased dramatically. Due to IBM Marketing Cloud report, 90% of all data generated has been produced in the last two years and 2.5 quintillion bytes of data is being generated every day [1]. As a result, analyzing and exploring this amount of data has become one of the critical challenges for data analysts and decision-makers. Information visualization can be employed as a helpful tool to deal with a large volume of data. Visualization enables the end user to have a better understanding and gain insight into data, conclude results and make an efficient decision [2]. Visualization should be designed in a way that can convey all needed information for analysis. On the other hand, it should not overwhelm the end user by the complexity of the data and mask the actual content.

The appearance of the cyber-physical system (CPS) and the Internet of Things (IoT) technologies have affected the industrial automation significantly. Efficient information exchange plays a pivotal role to implement vertical data integration across the entire enterprise entities. As a result, many efforts have been put in the evolution of industrial communication networks as a framework to facilitate the flow of data over the years [3]. Moreover, the emergence of new concepts, such as Industrie 4.0 has led to the tremendous changes in industrial automation domain. The term Industrie 4.0 refers to the fourth industrial solution concerning the digitization of physical assets of the system and employing internet technology to manufacture products smartly [4]. By implementing Industrie 4.0, machines, devices and sensors on the factory floor level will be connected to the Internet, which makes them analyzable and explorable [5]. The collection of data from devices can be stored in the cloud for further processes and analysis to make the system more optimized and efficient. IoT technology has introduced another facility for monitoring the selected KPIs of the system which is known as dashboard that enables the end user to intercat with the system. Dashboards are tools to display the visual information using charts, graphics and diagrams. Currently, many IoT service providers have offered

IoT platforms that can be employed to visualize the data gathered from plants, factories, and devices.

1.2 Problem Definition

In the prevailing industrial environments, the volume of data created in production systems is increasing enormously. On the other hand, it is a necessity to collect the generated data for further analysis. As a result, the data should be collected in a structural form and delivered to the user securely so that any distortion or loss of data can be prevented. In addition, data visualization as a powerful tool that allows decision-makers to obtain insight into the data has a pivotal role in today's enterprises. Moreover, several users from different levels of enterprise need to access the data reagrding their responbilities in the organization. Thus, developing a proper architecture to collect data from heterogeneous data sources can be seen as an ongoing challenge. It should be noted that proper visualization techniques should be employed to display the collected data for end users based on their role within the enterprise not to mention that from a visualization perspective, the flexibility and dynamicity of visualization solution should be taken into consideration to apply configuration changes at run time actively. Moreover, the growth of data should be anticipated for the proposed solution to simplify the addition of new data sources in the future.

Regarding the aforementioned discussion, the following questions are expected to be answered in this study:

- How to identify and efficiently access various data sources in production systems?
- How to introduce extensible and scalable solution to consider future growth of data and new protocols?
- How to provide an adequate visualization of data to give the best insight to decision-makers?

1.3 Work Description

The primary objective of this study is to introduce a comprehensive and extendible solution for the visualization of data regarding the role of the end user by utilizing IoT technology. To collect data from data sources, the extendible architecture should be proposed to consider the growth of data amount in the future. The case study of this thesis is Festo Didactic training factory, which is located at Seinäjoki University of Applied Sciences (SeAMK). This production line, which consists of four stations, including Storage, Drilling, Assembly, and Inspection can manufacture cell phones in 3 different colors. The line

is capable of processing various orders coming from different customers. During production, the system generates data which is stored in Microsoft Access Database. The database contains multiple tables through which the preferred KPIs are expected to retrieve by creating proper queries.

1.3.1 Objectives

According to the above discussion, the goals of the thesis are defined as follows:

- Design an extensible architecture to cope with heterogeneity problem of different data sources and future growth of data.
- Design data collection framework to establish communication and render desired data from backend to frontend.
- Design required services to monitor real-time and historical data of the production system.
- Select, design and implement role-based IoT dashboard platform for visualization of data.

1.4 Methodology

Before starting the implementation phase of the study, the state-of-the-art tools and technologies were studied in theoretical background. To meet the scope of this thesis, the literature review for following issues was selected and fulfilled.

- The importance of visualization for different roles of enterprise.
- Industrie 4.0 technology.
- The evaluation and comparison of various Internet of Things platforms.
- Standard communication protocols to interconnect IoT objects.
- The inclusion of knowledge-based approach in industrial automation.

In the next step, the approach for developing an architecture of the proposed solution is provided. The proposed solution was implemented using “Festo didactic training line” as the test bed.

For the implementation part of study, there are:

- Designing KPIs for Strategic, Tactical and Operational planning of enterprise.
- Designing appropriate queries to crawl the database for retrieving selected KPIs.
- Modeling the entities of the system using knowledge-based approach in ontology.
- Designing and accomplishing the modular and extendible visualization dashboard for monitoring the selected KPIs.

- Rendering required data from backend to frontend.
- Testing the implemented dashboard for monitoring the selected KPIs.

1.5 Thesis Outline

The thesis consists of 5 chapters. The first chapter includes introduction that was discussed above. The second chapter contains the theoretical background and literature review related to the technology and concepts used in this study. In chapter three, the methodologies used to implement the objectives of the thesis are presented. The fourth chapter presents the use case and implementation phase. Finally, in chapter five the conclusions and future work are discussed.

2. THEORETICAL BACKGROUND

In this chapter, the state-of-the-art technologies which are related to the scope of this study are examined.

2.1 Internet of Things (IoT)

The Internet of Things has become a hot topic of conversation in diverse fields, including scientific researches and industrial environments. In future industrial environments, it is expected to leverage IoT paradigm to monitor the activities of assets to enhance the productivity and agility of the enterprise's performance. Despite all accomplished researches in IoT field in last years, there is no unique definition for the term IoT. To address this issue, a number of researches have been carried out to study the different definitions of IoT within various domains [6][7][8]. The term "Internet of Things" was firstly introduced by Kevin Ashton, who used IoT phrase in his article to explain the usage of RFID tags in supply chains processes [9]. However, during the last decades the utilization of Internet of Things paradigm has extended to a wide range of applications in various fields, such as transportation systems, smart manufacturing, healthcare, building and home automation, automotive industry, etc. In a nutshell, regarding the concept of IoT, any object at physical layer that is equipped by embedded system and is able to connect to the Internet, can be a part of IoT ecosystem. Hence a thing in IoT can be referred to the entity that is capable of sending and receiving information over a network e.g. humans, animals, home appliances, vehicles, etc. Figure 1 shows a typical scenario for the IoT ecosystem in which different applications of IoT are illustrated.

Figure 1. *The typical scenario for the IoT ecosystem* [10]

Despite the boundless opportunities that IoT paradigm provides for businesses, there are still some considerable challenges for IoT to be leveraged. For instance, the IoT will comprise a wide range of heterogeneous devices, each using its own protocol for the exchange of data. As a result, making interoperability among IoT devices to bridge the entities of physical layer to the higher levels of IoT architecture becomes a challenge. To tackle this problem, several solutions have been introduced among which Service Oriented Architecture (SOA) is one of the most successful responses to this critical challenge. In SOA, the functionalities of an entity are represented as service, thus facilitating the integration of data from different data sources with various characteristics. SOA allows the dynamic discovery of devices and services and peer to peer (P2P) communication among devices [11]. As a result, SOA can be seen as a middleware that abstracts the functionalities of devices and establishes communication between different components of the system regardless of their underlying platform, with the aid of powerful interfaces such as Web Services (WS). Figure 2 illustrates a generic SOA-based architecture for IOT middleware.

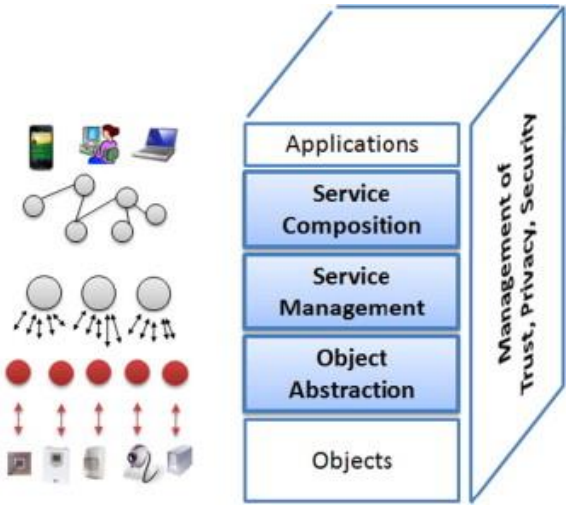


Figure 2. *SOA-based architecture for the IoT middleware*[12].

2.1.1 Industrie 4.0

Industrie 4.0 or Industrial IoT (IIoT) refers to the current phase of industrial revolution, and represents the fourth industrial revolution concept. In Industrie 4.0, the latest technologies such as IoT, Cloud Computing, Data Analytics, Cyber Physical Systems (CPS), etc. gather on one umbrella to enable the implementation of manufacturing processes smartly. The Federal Ministry of Education and Research in Germany coined term Industrie 4.0 in 2011, which was a project aimed to apply high-tech trends to industrial manufacturing through digitalization of assets. Regarding the Industrie 4.0 concepts, the history of industry divides into four stages. The first stage refers to the mechanization of production with the aid of water- and steam-powered machines; the second stage mentions the emergence of electrical energy which resulted in mass production; the third

phase refers to automated production using IT technologies and electronic devices and the fourth involves smart manufacturing incorporated with advanced technologies such as CPS. Figure 3 shows the stages of industrial development over time.

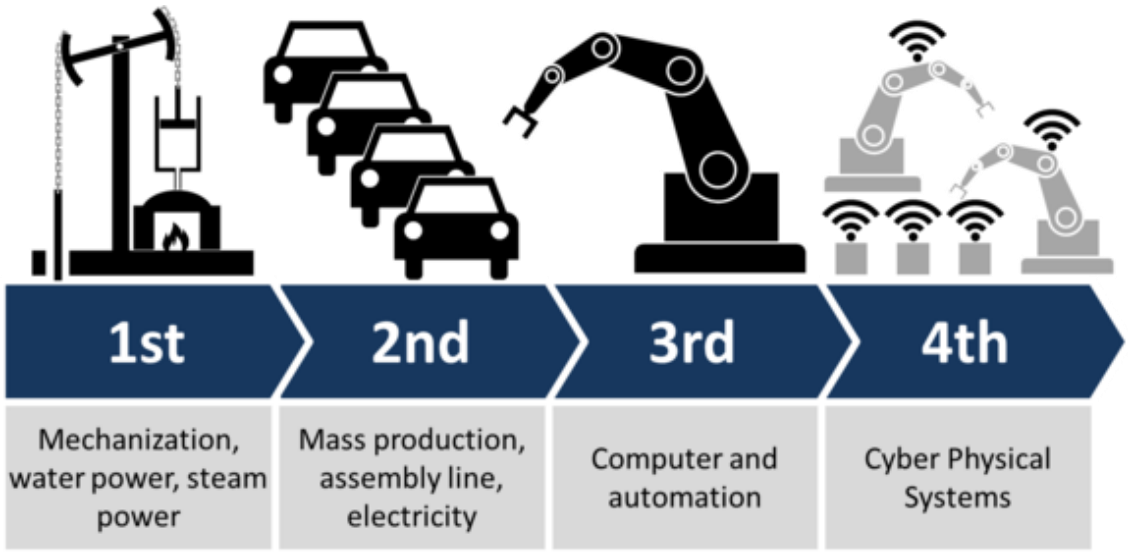


Figure 3. *The stages of industrial advancement* [13].

Moreover, Industrie 4.0 provides the framework for manufacturers to leverage decentralized production system, where entities of the system can interact with each other to fulfill assigned tasks accordingly. Exchanging information among machines in Industrie 4.0 is known as machine-to-machine communication (M2M). According to Hermann et al. [14], four design principles for Industrie 4.0 are interconnection(the ability of sharing information among the components of system), information transparency (the capability of presenting contextual information by duplicating the physical properties virtually), decentralized decisions (CPSs are capable of making decisions autonomously), technical assistance (the ability of assisting humans with dangerous and difficult tasks). The goal of Industrie 4.0 is that any product throughout the manufacturing process can be handled based on its current state and needs individually in a smart way by establishing communication between machines and products[15].

2.1.2 Cloud Computing

Cloud computing is the paradigm that allows organizations to access the computing services (e.g. storage, database, application, analytics, etc.) through the internet. In service, resources are configurable in cloud computing pattern and cloud providing companies charge the customers based on pay-as-you-go pricing method [16][17].

Cloud computing not only helps the enterprises to minimize the cost of investment on IT infrastructure, but also allows them to run their application efficiently with lower maintenance costs. Moreover, the demand for resources can be managed dynamically in terms of software or hardware.

Cloud computing services can be represented in three service types: infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS). Figure 4 shows the pyramid of services on three levels.

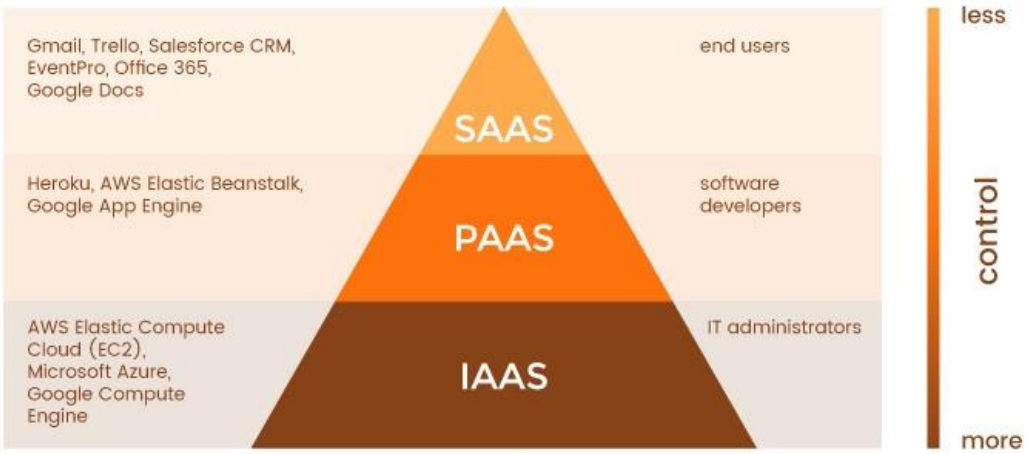


Figure 4. *Cloud computing services pyramid*[18].

Infrastructure as a service (IaaS)

In this model, a third-party provider allocates raw computing resources (e.g. storage space, host servers, virtual machines, networks, etc.) for users, thus, allowing businesses to pay resources on demand and delegate physical maintenance and support of resources to the cloud vendors. Moreover, the resources are configurable and scalable so that enterprises can manage the needed resources due to the actual demand of company.

IaaS examples: Amazon Web Services (AWS), Microsoft Azure, Google Compute Engine (GCE)

Platform as a service (PaaS)

In PaaS model, a third-party provider delivers the on-demand framework for the development of applications on top of vendor's infrastructure. PaaS users leverage the scalability feature of business, thus, enabling the users to focus on development of their applications with no concern about the lack of required resources.

PaaS examples: AWS Elastic Beanstalk, Heroku, Apache Stratos, Google App Engine

Software as a service (SaaS)

In SaaS service, a third-party provider hosts and delivers the application through web browsers to the enterprise. Since the software runs on vendors' servers, the maintenance, management and development of software is carried out by third party so the client not only can take advantage of removing the cost of technical supports for software (e.g. install, manage, update), but also just pays for the subscription price instead of spending higher fees on licensing software.

SaaS examples: Gmail, Salesforce, NetSuite

Moreover, cloud computing can be implemented in three forms: public clouds, private clouds, and hybrid clouds.

Public Clouds

A public cloud is the most frequent form of cloud computing deployment. The cloud resources are provided by third-party over the internet. In terms of economic issues, public cloud is the most reasonable for small enterprises since cloud servers are in charge of the delivery and maintenance of infrastructures. Moreover, most providers follow pay-per-usage model in which users only pay according to the usage of resources or services. Although it should be taken into consideration that for the companies that interact with sensitive data, public cloud is not an ideal solution regarding the lack of security in this model.

Private Clouds

A private cloud contains computing resources owned by one enterprise exclusively. The physical resources can be located at organization's site or hosted by a third party. This model of cloud provides the highest level of security for the company. Although it can be an expensive solution since the company should take a lead in purchasing and maintenance of all software and infrastructure.

Hybrid Clouds

The architecture of hybrid cloud is the combination of private and public clouds. In this model, organizations can take advantage of security nature of private cloud model while using scalability feature of the public cloud services. As an example, during peak demand, specific applications can be pushed to a public cloud to reduce the load of hosted data on internal infrastructure. This approach also can be followed during outages or periodic maintenances of private resources.

2.1.3 IoT platforms for industrial systems

IoT platforms can be seen as the backbone for the development of IoT applications that allows the connection between physical objects and the application layer. Moreover, IoT platforms can help to overcome the lack of capability to implement distributed manufacturing by current ERP and MES solutions [19].

Currently, there are hundreds of IoT platforms provided by different vendors. In the following, a number of most prominent IoT platforms are studied.

AWS IoT Platform[1]

The AWS IoT platform offers broad device SDK, which consists of a set of client libraries to establish the connection of IoT devices enabling data exchange with Device Gateway using MQTT, HTTP, or WebSockets protocols. Device Gateway forwards messages to AWS IoT core using publication/subscription model securely and efficiently. Using this approach makes one-to-one and one-to-many communication applicable.

The scalability of Device Gateway is considerable in a way that it is capable of supporting the connection of over a billion devices. The security of platform is accomplished by using credential and access control for the authentication of devices. With embedded Rules Engine feature in AWS IoT Core, incoming messages can be processed to define customized business rules. On top of that, other AWS services (e.g., AWS Lambda, Amazon Kinesis, Amazon S3, Amazon Machine Learning, Amazon DynamoDB, Amazon CloudWatch, and Amazon Elasticsearch Service) can be called by IoT Core to enrich and transform data into valuable information. Moreover, Device Shadow service within the AWS IoT Core can be utilized to create applications for reading messages and monitoring the latest state of devices, even when the device is offline. In addition, the user can set a desirable state of the device in the future through REST API whenever the device reconnects. Registry component takes a lead in creating/maintaining resources associated with each device in the AWS Cloud.

The key components of AWS IoT are shown in Figure 5. In this architecture, each device publishes the information about its current state on MQTT topics and in JSON format. The message broker is in charge to send published messages to clients who subscribe for the interesting topics. The state information encompasses two items: the last state of device and the desired state requested by an application.

[1] https://aws.amazon.com/

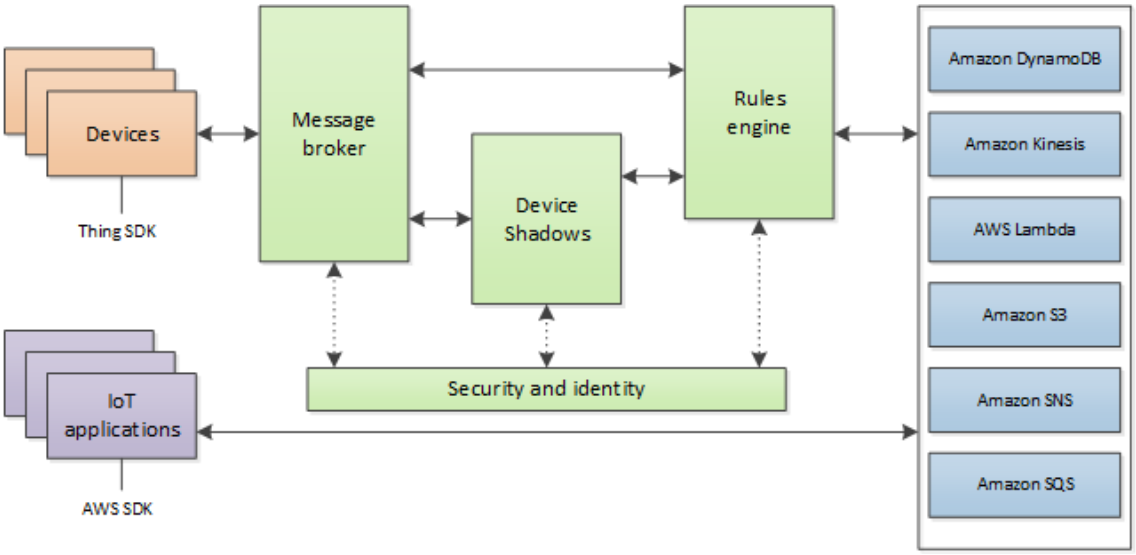


Figure 5. *AWS IoT architecture*[20].

IBM Watson IoT[2]

IBM Watson IoT is a cloud-based Platform-as-a-Service (PaaS) solution that enables the connection of sensors and devices to the cloud via MQTT protocol. Devices can publish data either directly or through gateways. IoT applications can be created by IBM Bluemix and secure REST API is utilized by IBM to make the connection between platform and the application layer in real-time. Figure 6 illustrates the architecture of the Watson IoT Platform. In Watson IoT platform architecture, a device can be any entity that is able to connect to the Internet and send data to the cloud. Although the devices can communicate indirectly with each other through applications by sending/receiving the events/commands. Any device in ecosystem is identified by a unique authentication token and should be registered to be able to connect to platform.

The IBM Watson IoT Platform now supports the use of IBM Blockchain services for IoT applications to enable the customers to add selected data to a private blockchain. Moreover, using blockchain technology allows the decentralized use of IoT data. Decentralized blockchain networks add scalability feature to IoT platform enabling the addition of a large number of devices to the system. Watson Cognitive Analytics is another service from Watson IoT platform that facilitate the transformation of unstructured data into understandable data to provide insight from data for enterprise using machine learning, machine reasoning, image analysis and natural language processing. The Watson IoT platform dashboard plays the user interface role at front-end side to enable the customer to

[2] https://www.ibm.com/internet-of-things

operate the platform simply. Moreover, the data can be visualized in dashboard to enhance the understanding of data.

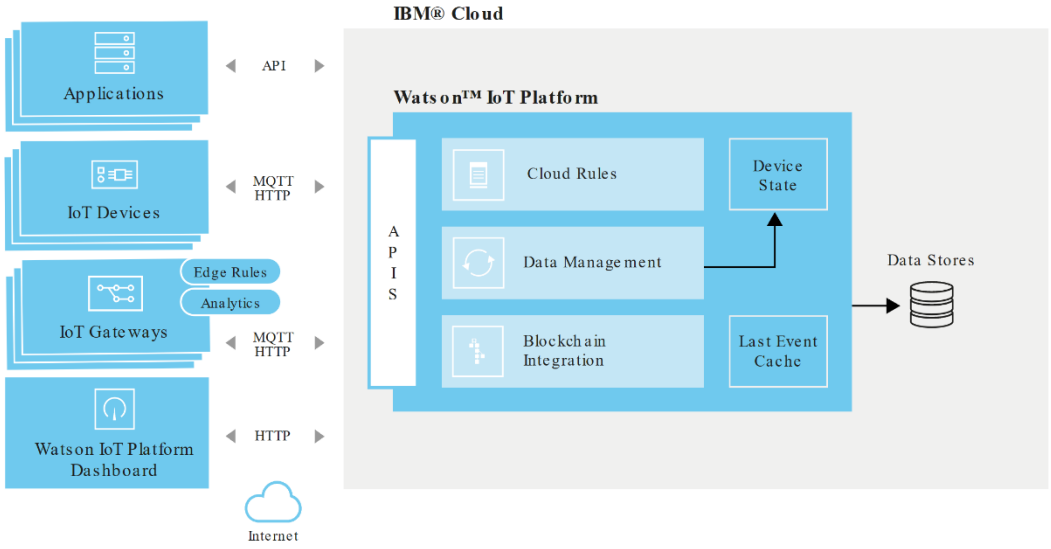


Figure 6. *Architecture of the Watson IoT platform*[21]

Microsoft Azure IoT suite[3]

Microsoft introduced Microsoft Azure IoT suite to allow enterprises to get insight and real-time comprehension of existing and future devices and assets. It empowers the customers with preconfigured solutions built on Azure Platform-as-a-Service (PaaS), such as remote monitoring, connected factory and predictive maintenance to carry out remote diagnostics, increase the performance of the system and predict the devices' failures. Regarding the architecture of Azure IoT suite which is shown in Figure 7, the collected data from IoT devices is sent to cloud gateway, which provides the data to stream processing block for processing and integrating with business processes. Then the processed data is delivered to the presentation layer to be visualized for human operators. In the presentation layer, historical data or near real-time data can be displayed by designing dashboards or BI reports.

The scalability of Microsoft IoT solution has been accomplished by making the interoperability across different sensors and devices with diverse characteristics. To achieve this, connection to the cloud can be done either directly or by utilizing an intermediate gateway. The subsystems of architecture are independent from each other and can be deployed

[3] https://azure.microsoft.com/en-us/suites/iot-suite/

separately to increase the flexibility of the entire system. The subsystems of architecture communicate over REST/HTTPS in JSON format. The architecture is designed in a way to support hybrid cloud and edge computing technologies. The horizontal scalability of subsystems can be achieved using an orchestrator offered by Microsoft, such as Azure Managed Kubernetes or Service Fabric. Additionally, some optional subsystems can be integrated with core subsystems. Data transformation can be employed for transforming and restructuring retrieved telemetry data from devices. Machine-learning block can be used for predictive maintenance by applying algorithms to historical data. User management allows to manage the accessible functionalities for different roles.

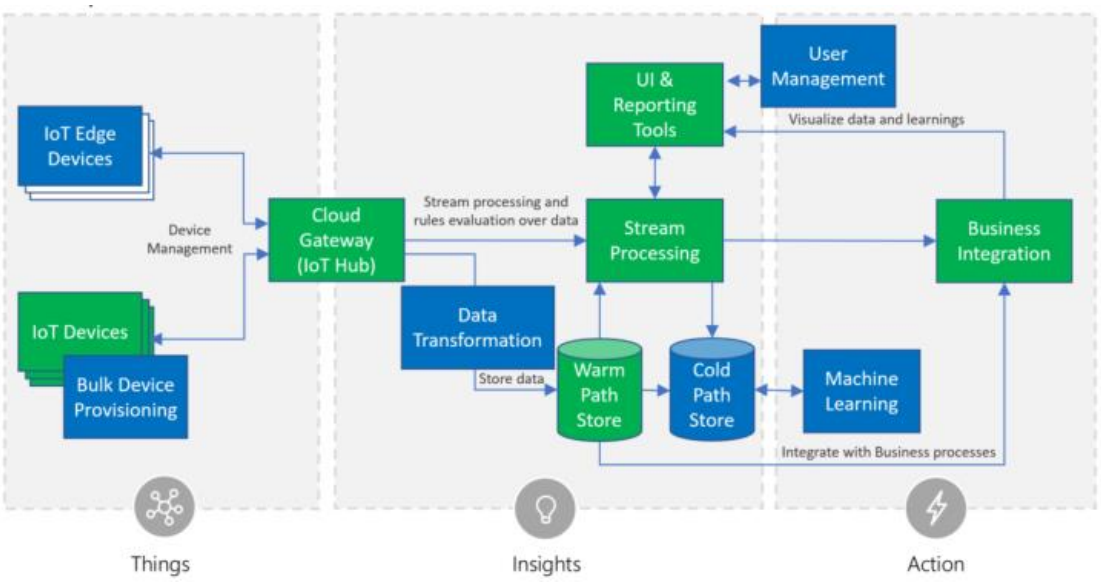


Figure 7. *Microsoft Azure IoT Reference Architecture*[22]

ThingWorx IoT platform[4]

ThingWorx is one of the first software cloud-based platforms i.e. a product of PTC Company. REST API and other protocols, such as MQTT and sockets make the connection between IoT devices and the platform. The ThingWorx platform supports the connectivity, event processing, analytics, storage and presentation of IoT and M2M data. 3D storage engine of the platform allows big data analytics and supports scale requirements for millions of devices. For the visualization of data, objects can be added to the GUI development IDE, which is called Mashup Builder to create a dynamic application. Mashup Builder allows to place Widgets on a web page by drag and drop. ThingWorx Data services are in charge to bind data into a Mashup to be displayed on dashboard using visualization widget. Mashup builder also manages the access to the user Extensions.

[4] https://www.ptc.com/en/products/iot

The designed application can be extended with 3rd-party plug-ins. REST API is the communication interface to 3rd-party systems as well as other sub components of the architecture. Furthermore, ThingWorx provides an event-driven execution engine within the platform to allow enterprises to apply business logics and rules to the applications. 3D storage engine enables the platform to make connectivity with a massive number of devices and store retrieved data in a fast and optimized way.

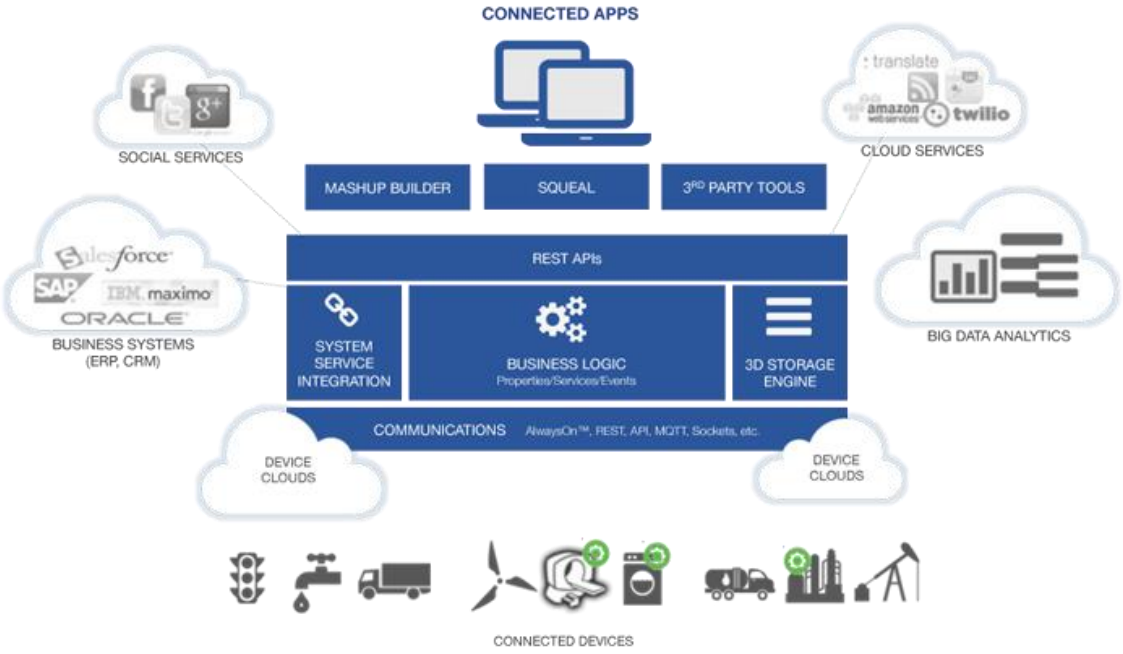


Figure 8. *ThinxWorx IoT platform*[23]

Wapice IoT-Ticket platform[5]

IoT-Ticket is an end-to-end IoT solution, which supports data acquisition and analytics, besides remote monitoring and supervisory control of devices via Dashboard User Interface (UI). Moreover, it provides powerful reporting features, capable of creating regulatory reports. The dashboard is a web-based solution by which users can design interface by adding new elements into dashboard by drag and drop without having programming knowledge. The Interface Designer and Dataflow Editor are two graphical web-based tools that are employed to create customized dashboard for end users. The data gathered from devices can be sent to Wapice's Big Data Analytics Server by REST API or OPC UA protocols. Also, WRM 247+ manufactured by Wapice is hardware that supports a broad range of protocols and acts as a sensor gateway to collect data from heterogeneous

[5] editor

devices within the environment. IoT-Ticket provides user management to handle the access and permissions of different roles of enterprise. A set of APIs for programming languages such as Java, C++, Python, C# are supplied by Wapice to allow developers to deploy their solution.

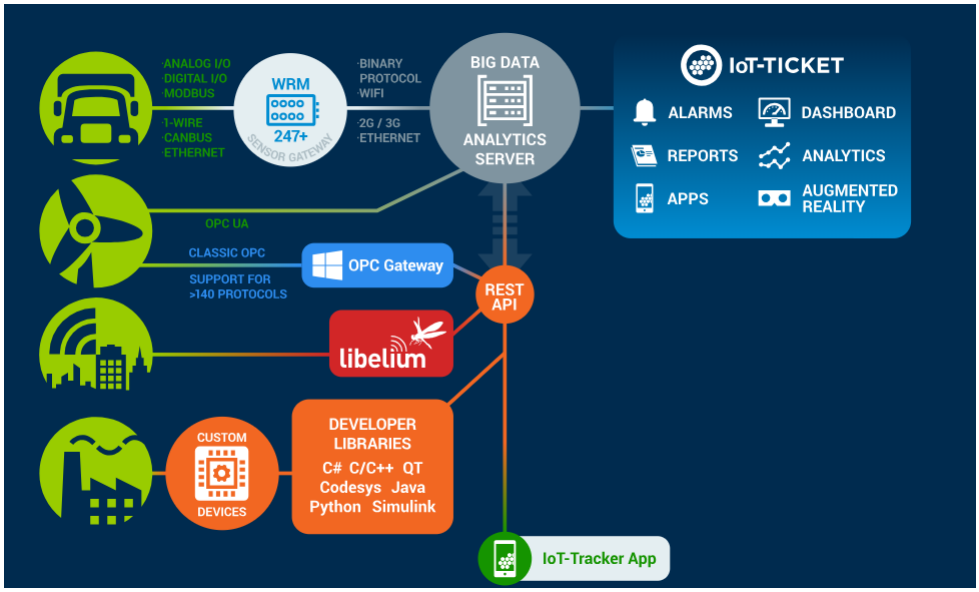


Figure 9. *Wapice IoT-Ticket architecture, adapted from* [24]

VersaSense IoT Fabric[6]

VersaSense IoT Fabric allows the implementation of the Industrie 4.0 properly as well as beneficially. The solution is composed of software, networking and hardware components that work together to forward sensor data to IT platforms. One of the remarkable features of IoT Fabric is its modular and plug & play nature which enables data collection in an adaptable and scalable fashion. Moreover, IoT cloud Services proposed by VersaSense allows the enterprises to leverage scalable data storage, rich data visualization, customizable business dashboards and flexible alerts. All these can be achieved through a user-friendly interface with no programming knowledge.

[6] https://www.versasense.com/

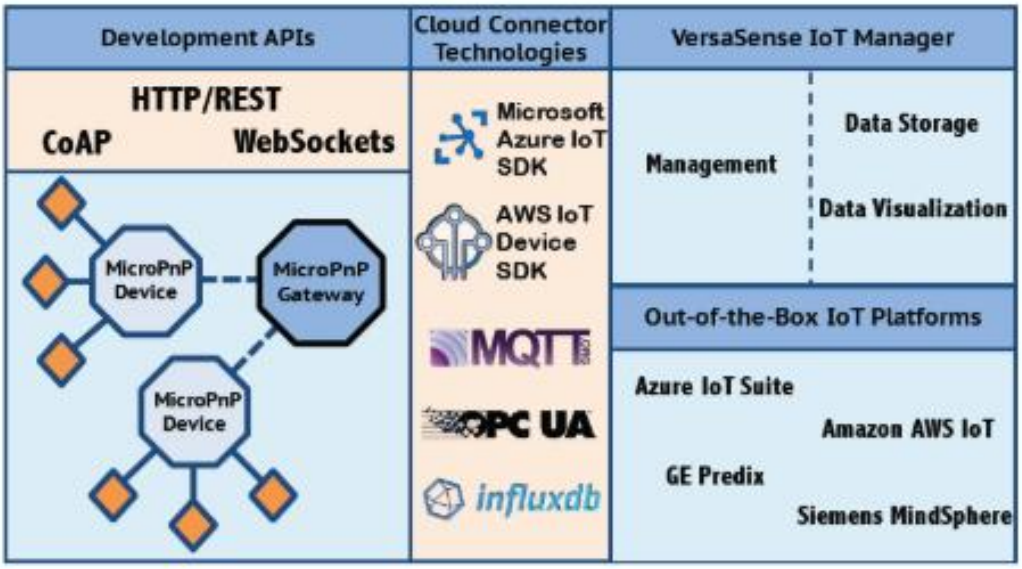


Figure 10. *VersaSense IoT overview*[25]

2.1.4 Dominant IoT communication protocol

HTTP (Hypertext Transfer Protocol) is an application layer protocol by which the exchange of date between client (e.g. a web browser) and server runs over TCP connection. In this protocol, the streaming of data is accomplished based on request and response pattern. A client sends a request to the server via HTTP methods supported by REST (e.g. POST, GET, PUT, and DELETE). Then, the server sends back the response including requested content along with status information about the request.

MQTT (Message Queuing Telemetry Transport), is a publish/subscribe messaging mechanism that enables lightweight communication among entities. The aim of developing MQTT was leveraging from its efficient band-width besides low energy consumption nature specifically to establish the communication over resource-constrained devices. Its routing mechanism (one-to-one, one-to-many, many-to-many) makes it suitable for IoT and machine-to-machine (M2M) communication [26]. Subscriber, publisher and broker are three components of MQTT. Publisher provides the messages through broker to the subscribers that have registered for the interested topics.

CoAP (Constrained Application Protocol), is another infrastructure-independent protocol designed for "use with constrained nodes and constrained (e.g., low-power, lossy) networks." [27]. The request/response model of this protocol makes it suitable for M2M application communication specifically in energy and building automation domain. CoAP is often known as a simplified version of HTTP which supports REST. Unlike HTTP,

CoAP is UDP-based rather than TCP-based, thus making the size of messages extremely light.

AMQP (Advanced Message Queuing Protocol), is an open standard protocol which is designed to exchange information between applications and organizations through reliable communication channels [28]. It is a message-oriented middleware protocol which makes interoperability among different vendors within IoT ecosystem. AMQP architecture is based on publish/subscribe messaging model in which the messages are routed between clients and servers through a message broker. Exchange and Queue are two subcomponents of Message broker. The published messages are sent to Exchange. Afterwards, Exchange distributes the messages to Queue using a binding module. Next, the messages either will be delivered to the subscriber by broker or the subscriber itself pulls messages from Queue.

2.2 Key Performance Indicators

A Key Performance Indicator (KPI) is a selected quantifiable measure that reflects the progress toward the strategic objectives of the organization. Also, according to Lohman et al. [29], KPIs can be defined as "a variable that quantitatively expresses the effectiveness or efficiency, or both, of a part of or a whole process, or system, against a given norm or target". KPIs give insight to decision-makers to evaluate the performance of system and acknowledge to the needs for the improvement of system effectively.

In manufacturing domain, KPIs play a key role since they are used to monitor, evaluate and analyze the activities within the processes. The KPIs should be designed in a way to cover all needs for different levels of enterprise. For instance, at higher levels, the KPIs echo the overall performance of business while at low levels, KPIs involve the metrics of processes in detail. As a result, for each layer of industrial manufacturing systems (i.e. ERP, MES, SCADA, shop floor machines), specific KPIs should be selected and provided. On the other hand, Identifying appropriate KPIs that can fully provide the context which echoes the objectives of enterprise can be sophisticated. Thus, employing an effective model for identifying applicable KPIs is a necessity. Veleva and Ellenbecker proposed in their study [30] 8-step iterative methodology to select KPIs of the production system which is shown in Figure 11.

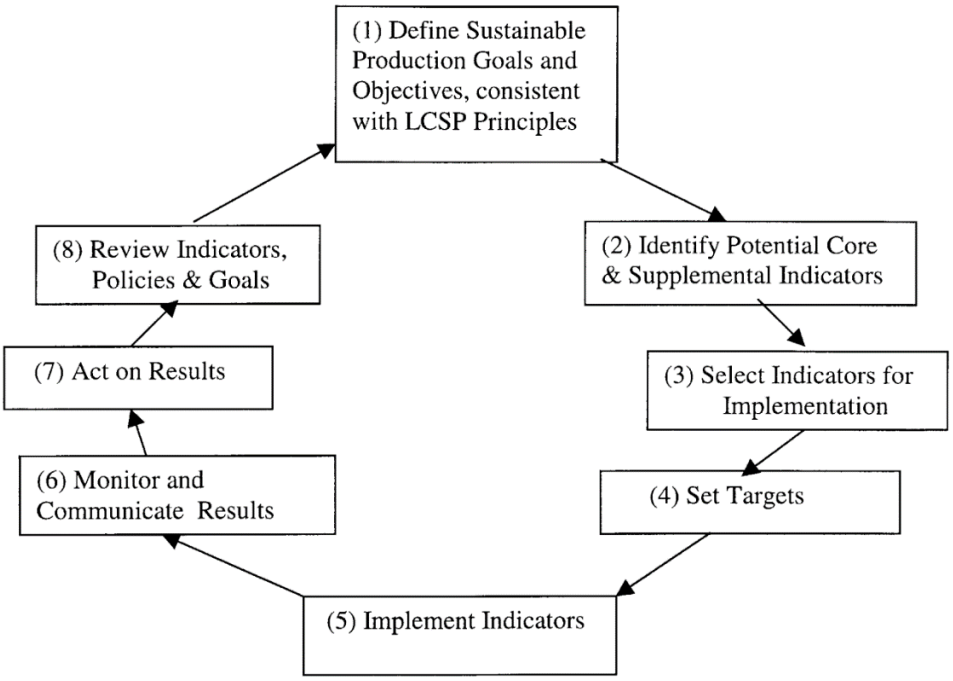


Figure 11. *Closed-loop iterative methodology for identification of production KPIs*[30]

Moreover, Eckerson discussed in his article [31] that selecting proper KPIs that can perfectly mirror the key information of performance will lead to the enhancement of organization operation eventually. Thus, he proposed 10 key characteristics of KPIs that should be taken into account when KPIs are being identified:

- KPIs echo the key drivers of business value.
- KPIs are constructed by "executives".
- KPIs are cascadable all through the enterprise.
- KPIs are on the basis of organization-established standards.
- KPIs are on the basis of valid data.
- KPIs must be coherent for users.
- KPIs must be effective in improving performance.
- KPIs give the context about thresholds, targets and benchmarks.
- KPIs enable individuals to make decisions effectively.
- KPIs allows to take actions positively.

Besides the mentioned characteristics, Lake[32] discussed that KPIs must be 1- quantitative, i.e. represented by numbers; 2- practical to be convenient to integrate with existing

processes; 3-directional to determine whether the organization on the right track; 4- actionable to be used to make the desired modifications achievable by taking disciplinary actions.

In the manufacturing arena, several researches have been conducted in terms of selecting effective KPIs for the production processes. According to Tokola et al. [33], KPIs for visualization were selected with regard to the exploratory survey they carried out at manufacturing companies in machining industry domain. Kibira et al. [34] proposed a selection method for KPIs by utilizing the combination of human judgment and quantitative analysis, thus the final KPIs can be extracted from candidate ones. Amrina and Yusof [35] presented a set of KPIs of sustainable manufacturing in the automotive industry. Fukuda and Patzke [36] proposed 35 indexes for the performance measurement in production process.

2.3 Role-based visualization

Data acquisition from shop floor devices and vertical integration of data among all the hierarchy levels of enterprise plays a pivotal role in today's manufacturing companies' success. Utilizing production information systems enhances the agility and efficiency of the industrial manufacturing systems and facilitates the exchange of data between different entities that range from the shop floor to business level.

Since the amount of data provision in the system can be massive, it is needed to emphasize the most important information for data analyzers [37]. To achieve this goal, the visualization comes to picture to assist the decision-makers to obtain meaningful data and respond to the changes in the system effectively [38]. By using visualization, the data can be summarized and represented in an easy-to-understand form to provide insightful information for users. Moreover, utilizing an effective visualization technique helps to reduce the complexity of collected data, allowing the end user to focus on the key parameters of manufacturing processes and make proper decisions efficiently. As a result, the usability of data visualization for users should be taken into consideration to improve the performance and productivity of systems.

On the other hand, the visualization of information for different users of the system should be different, because they have diverse responsibilities and need a distinct data type to take action according to their job description. Eckerson [39] discussed that selecting effective KPIs for different types of users in the organization leads to the improvement of business performance. Moreover, the best dashboards for visualization are the ones that parse out the data and KPIs regarding the role, task, and level of decision-makers. This discussion is shown in Figure 12. On shop floor level, workers interact with data, such as

the status of devices in detail while analysts and mid-level managers deal with the summarized data at a higher level to optimize the performance. At the highest level, managers and executives intend to have the overall view of the strategic objectives to set targets and plans for company's future progress.

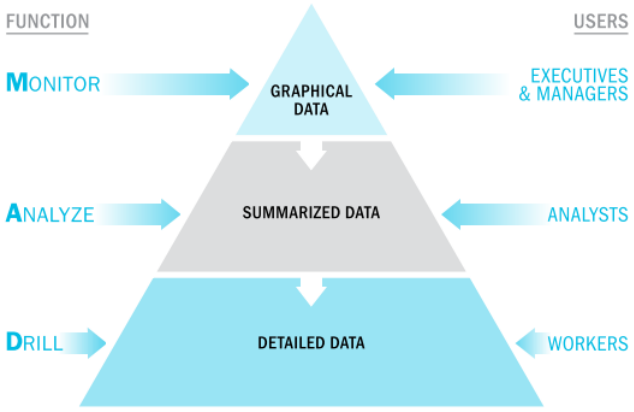


Figure 12. *MAD (Monitor, Analyze, Drill) framework illustrating the needs of different users to different types of data*[39].

Tokola et al. designed three dashboards for end users on different hierarchy levels in their research [33] based on the selected KPIs : Strategy dashboard, tactical dashboard, and operational dashboard. To fulfill the dashboard they have used Dashing framework [40], a Sinatra based framework that enables the designer to build a web-based dashboard using premade widgets or create a customized dashboard by using SCSS, HTML, and CoffeeScript technologies.

An implementation of a personalized adaptive HMI for different users of a Service-Oriented Production Control systems was carried out in [38]. They considered the humans that run and control the organization as a context of the system besides production systems and environments. Moreover, humans were classified into different categories regarding their roles, skills and other characteristics of the enterprise, such as maintenance personnel, supervisors, managers, and operators. Thus, the multi-modal information was represented and visualized regarding the state of the user, the production system, and environment. Mobile devices, such as smartphones and tablets were utilized for visualization purpose to increase the flexibility for users to have access to information anywhere and anytime.

One of the international projects that have been launched by the European Commission in visualization domain is PLANTCockpit[41].The PLANTCockpit aimed to provide a

central environment for monitoring and controlling of the production system to allow different users of the system (e.g., supervisors, line managers, and foremen) to make effective decisions for optimizing the system performance.

2.4 Knowledge-Based Systems

In the contemporary manufacturing systems, the reconfigurability and adaptability are the most imperative characteristics due to the high demand for agility implementation and rapid response to the changes in the system. As a result, in the past few years, the deployment of knowledge-based approach solutions in industrial environments has increased significantly regarding its promising potential for the successful fulfillment of the lean manufacturing concept. Thanks to the KB technology, the complex structured data of the system (e.g. object descriptions, rules) can be stored in a semantic repository. Moreover, with the help of reasoning mechanism of inference engine as part of Artificial Intelligence (AI), the contents of KB can be processed by applying logical rules to the KB and solutions for a specific problem will be deducted. Figure 13 illustrates the typical architecture of knowledge-based system.

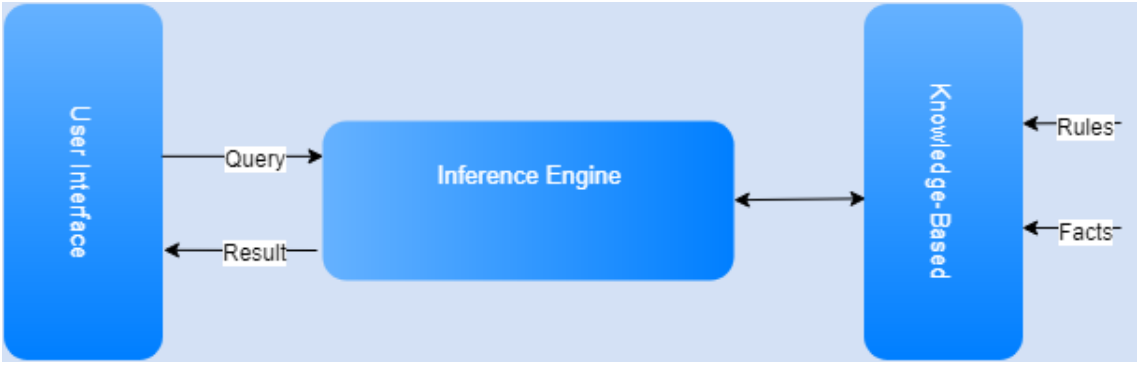


Figure 13. *The typical architecture of knowledge-based system*

In the industrial automation domain, the KB approach allows for the easiest description of system, thus the functionalities and relations between different entities of the system can be abstracted and represented as knowledge. The advantage of utilizing KB for describing the entities of the system, is that the stored data can be provided in machine/human readable format. Moreover, with the help of KB approach the dynamic adaptation of system is achievable, thus if any changes happen to the objects, the new information can be updated without the interruption of system. Employing the semantic approach for modelling the information of system components as knowledge has been done in several researches [42][43][44][45].

Ramis et al. in [46] proposed an architecture for utilizing KB in industrial automation domain, in which the knowledge-based system is composed of four main components: a KB service which stores the information of the entities of the system; Physical equipment which represents the real components on shop floor (e.g. sensors, actuators and controllers); Orchestration service that takes a lead in controlling the processes to execute the requested works; user interface for the visualization of the current state of production process and requesting an order to be executed.

2.4.1 Ontology

Ontologies are the most widely used methods for representing knowledge in systems that take advantage of knowledge-based approach. The term "ontology" is derived from the Greek language and in a philosophical context, it refers to the study of the existence of things.

In the realm of Artificial Intelligence, different definitions for the concept of ontology have been provided. One of the most cited definitions was proposed by T.R. Gruber who described ontology as "an explicit specification of conceptualization" [47]. Borst in his PhD thesis [48] defined ontology as the "formal specification of shared conceptualization". According to Guarino et al. [49], "the backbone of an ontology consists of a generalization/specialization hierarchy of concepts, that is, a taxonomy".

In general, in ontology, data is modeled in a way that represents the knowledge as a series of concepts associated to a domain and their relationships. Classes, attributes, relations and individuals are the fundamental components of ontology to model a domain.

- **Classes**: The terms that represent concepts.
- **Relations**: describe the interaction between classes.
- **Attributes**: the properties of objects.
- **Individuals**: instances of objects.

The ontologies enable us to model the concepts within the specific domain and relations between them in an understandable way both for machines and humans. As a result, ontologies provide a common language among different applications. In addition, according to Noy and McGuinness [50] there are further motivations for developing ontologies:

- Ontologies enable people to share common understanding of a domain with each other.
- Ontologies make the domain knowledge reusable.
- Ontologies make the assumptions of domain explicit.
- Ontologies separate domain knowledge from the operational knowledge.
- Ontologies make domain knowledge analyzable.

Moreover, one of the most significant aspects of ontologies is that ontologies provide a mechanism to model the entities of the system in a flexible and extendible way, thus any new relationship can simply be added to the existing ontology.

The construction of ontology is carried out by utilizing ontology languages. There are many languages available for the implementation of ontology [51] but two of them are more commonly used: Resource Description framework (RDF) [52] and Web Ontology Language (OWL) [53]. In order to achieve these standards, classes that represent real-thing concept are linked together with relationships, which are defined as properties. The information of classes and their relationships is represented by triples. A triple has three components: subject, predicate and object. A set of triples can be combined together to represent all the components of the specific system within the ontology thoroughly.

OWL is an RDF-based XML language, which in comparison with RDF, it supports more semantic features, such as concept intersection, atomic negation, universal restrictions and more reasoning and inferring facilities. These features make OWL the excellent solution for the implementation of ontology in industrial automation domain.

2.4.2 SPARQL query language

The SPARQL Query Language for RDF [54] can be utilized for retrieving and manipulating the data stored in RDF-based ontology models. As a query language, it has been the designed and endorsed by World Wide Web Consortium (W3C). Unlike SQL, SPARQL supports the querying multiple data sources.

A SPARQL query, comprised of three elements (subject, predicate and object) can consists of variables. Query results are found by corresponding the patterns in the query to triples in the dataset.

There are four types of SPARQL queries:

- **ASK**: Returns a Boolean result whether data matches the query pattern.
- **SELECT**: Returns matches of variables of query pattern in a tabular form.
- **CONSTRUCT**: Returns the matched data as an RDF graph by a specific template.
- **DESCRIBE**: Returns the resource matched by the given variables of query pattern.

2.5 Integration technologies

The lowest layer of IoT ecosystem is composed of heterogeneous physical devices, which are expected to collect data from the environment. One of the challenging needs for processing the huge amount of generated data within the manufacturing systems is the data integration of different layers of the enterprise with various protocols [55]. To achieve this, diverse communication protocols have been introduced for exchanging data among different entities from different levels within the industrial environments. By increasing the adoption of emerging Service Oriented Architecture (SOA) paradigm in manufacturing systems, the capabilities of factory floor devices have been encapsulated and represented as service [56]. The SOA paradigm facilitates the integration of factory automation systems and components with the aid of the Web Service (WS) technology [46], which can be employed for implementing a gateway between factory shop floor devices and higher-level systems, such as SCADA, MES and ERP [57].

Currently, one of the most prominent data exchange standards in the industrial automation domain is OPC-UA which is the successor version to "classic OPC". OPC-UA allows the data communication between different data sources (e.g. factory floor devices, test system fixtures, databases) in a safe, reliable way. Moreover, it enables the data transaction among different manufacturers and across operating systems. Its extensibility feature supports the vertical and horizontal integration of entities within the organization, and new components can be added without affecting on the current status of the system. OPC Foundation recently has launched the publish/subscribe mechanism for the sake of bandwidth optimization by utilizing MQTT protocol. The aim of IoT can only be achieved if the adoption of global communication standard is acknowledged for networking the physical devices of the system. Besides a publish/subscribe model, a "secure connection oriented client/server communication model" is needed to fulfill the supervisory control that enables management to send commands to devices. OPC-UA supports all the complex requirements for implementing the data integration on all vertical layers for remote device access and machine to machine (M2M) communication [58].

Devices Profile for Web Services (DPWS) [59] is another standard that employs Web Service technology to expose resource-constrained devices on shop floor as services for seamless integration. DPWS allows devices to be discoverable through Web services dynamically. Moreover, secure messaging to/from Web service, description of Web service and subscription/receiving events to/from Web service are built-in services that are specified by DPWS. The messaging mechanism of DPWS uses IP, TCP, HTTP, UPD, SOAP, XSD and WSDL to establish cross-platform communication between different vendors.

Izaguirre et al. [55], studied the interoperability of OPC-UA and Device Profile for Web Services (DPWS) by utilizing a middleware which is capable of processing generated

events and notifications. They concluded that the combination of these technologies brings about the complete implementation of SOA within the shop floor level.

3. METHODOLGY

This chapter represents the technologies and tools that are employed to implement the proposed solution for the data collection during manufacturing processes and role-based visualization of generated data regarding the responsibilities of different roles of organization. Moreover, to propose a comprehensive solution for the given research problems of this study, the architecture of the system is designed and presented.

3.1 System architecture

As it was discussed earlier, the number of connected devices to the internet is growing on a daily basis, which leads to the profusion of data. Therefore, collecting and analysing this amount of data will be sophisticated. Moreover, devices on shop floor have different characteristics, thus one of ongoing challenge in IoT ecosystem is the heterogeneity of diverse devices on factory-floor layer. To cope with these challenges, the proposed architecture should be designed in a way to make interoperability between heterogeneous devices with various protocols and architectures so that the connectivity across them can be accomplished. In addition, the new devices and protocols which will be introduced in the future should be anticipated. The developed architecture in this research follows the loosely-coupled approach where changes in components have the least effect on the operation of the system. Moreover, the design of architecture should consider the IoT characteristics, such as interoperability, scalability, extensibility, security, etc. Consequently, a well-designed architecture should provide dynamic and scalable approaches for the deployment of solutions. Figure 14 illustrates the overall architecture of the solution, which consists of three layers.

At data retrieval layer, different data sources such as databases and knowledge bases, IoT devices, and legacy devices are considered. The communication layer consists of different modules that interact with the Gateway. Finally, at application layer, the visualization of data as well as other cloud services (e.g. data analytics, machine learning, etc.) is handled. The right of architecture represents the examples of tools and services that are used by actors within each layer for development purposes.

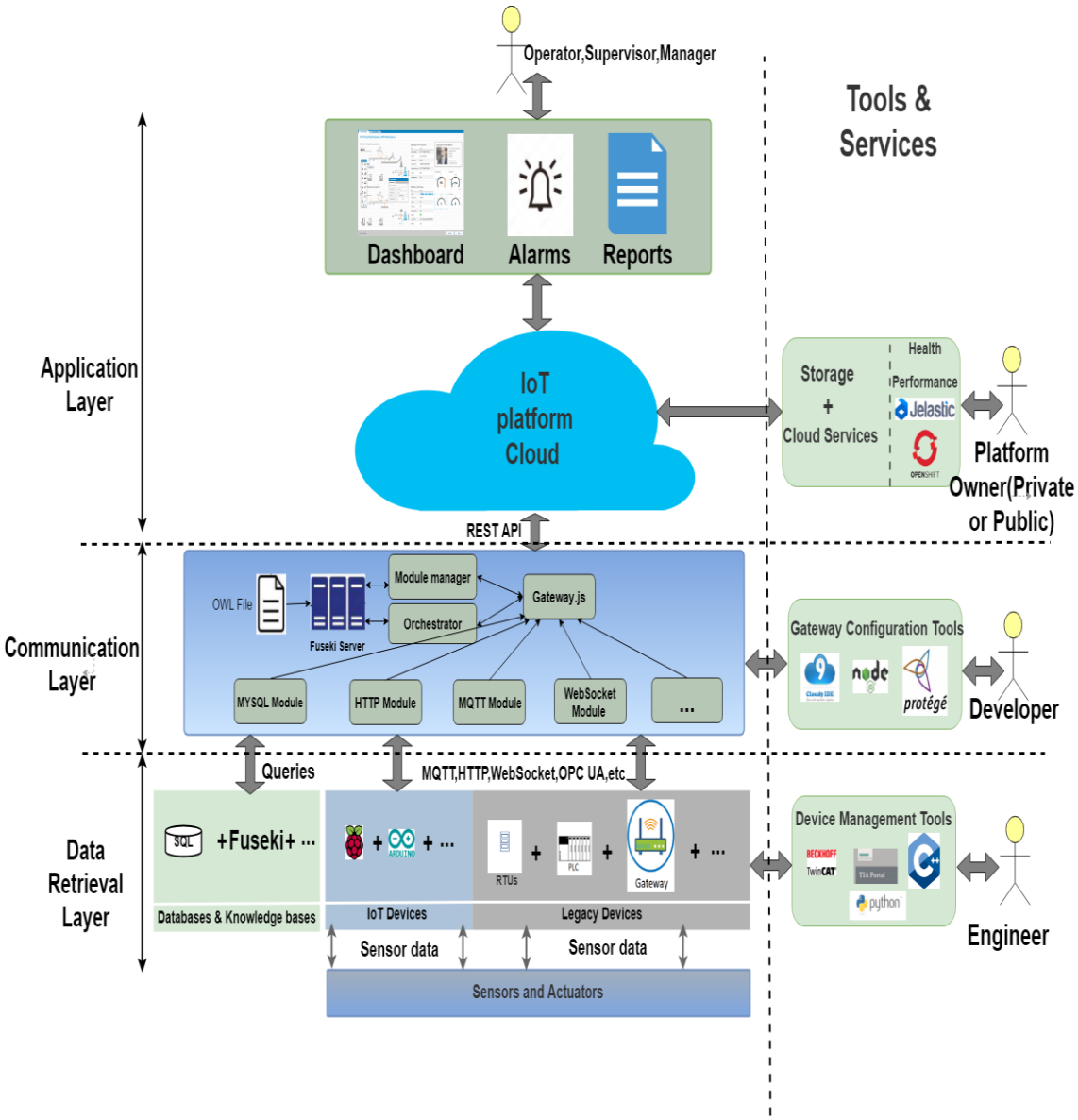


Figure 14. *System architecture. Adapted from* [60]

3.1.1 Data retrieval layer

On the factory floor, data retrieval is a critical need to support the effective control of production processes. The obtained data from manufacturing environments can be analyzed to enhance the performance of the system according to the desired objectives of the organization. To achieve this, the collected data should be delivered to higher levels of the system to identify weaknesses and strengths of the production process by analyzing the data. Recognizing problems (e.g. bottlenecks, delays, etc.), analyzing root causes and finding the proper solutions can be managed only when the process of gathering data is carried out accurately.

Sensors and actuators at the lowest level are the sources of data and remote triggers in IoT scope respectively. Also, databases and knowledge bases can be seen as historical information sources where data can be retrieved using queries. Moreover, within the factory field, the messages are exchanged using different standards such as Profibus, Ethernet, Profinet, etc. Also, wireless communications on the shop floor include standards like ZigBee, Wi-Fi, Bluetooth, etc. At this level of organization, mainly engineers and technicians are in charge of managing the devices with the help of available tools and technologies. For instance, in the domain of industrial automation, the Programmable Logic Controllers (PLCs) are widely used to automate processes and functions on shop floors. In case any change is needed at device level of PLC, the engineer can use PLC programming software to adapt the configuration of logic and then the new logic can be executed.

3.1.2 Communication Layer

As previously stated, in order to tackle the heterogeneity of IoT devices in terms of communication, multiple communication protocols should be supported by the system to enable the connectivity between devices and the cloud. The communication layer takes the lead to aggregate the retrieved data from shop floor. Also, it is responsible for delivering data to application layer using standard communication protocols such as REST API. In fact, the communication level bridges the physical layer and the application layer. It can be seen as an interface that transforms retrieved data from heterogeneous data sources to a common format for transferring to a higher level.

According to the proposed architecture, each module supports the specified protocol that interacts with device level to retrieve sensor data. The gateway has a pivotal role in the system as it interacts with different modules handling various protocols to achieve data and conduct protocol translation into a common protocol for delivering the data to the cloud.

Data sources in the proposed approach are modeled using ontology and triples are stored in OWL file. Gateway queries all create instances through the orchestrator to collect their associated data for each data source. Moreover, for the historical data, instances for the queries are created using the same way to receive data from databases. Figure 15 shows the sequence diagram for the configuration of gateway.

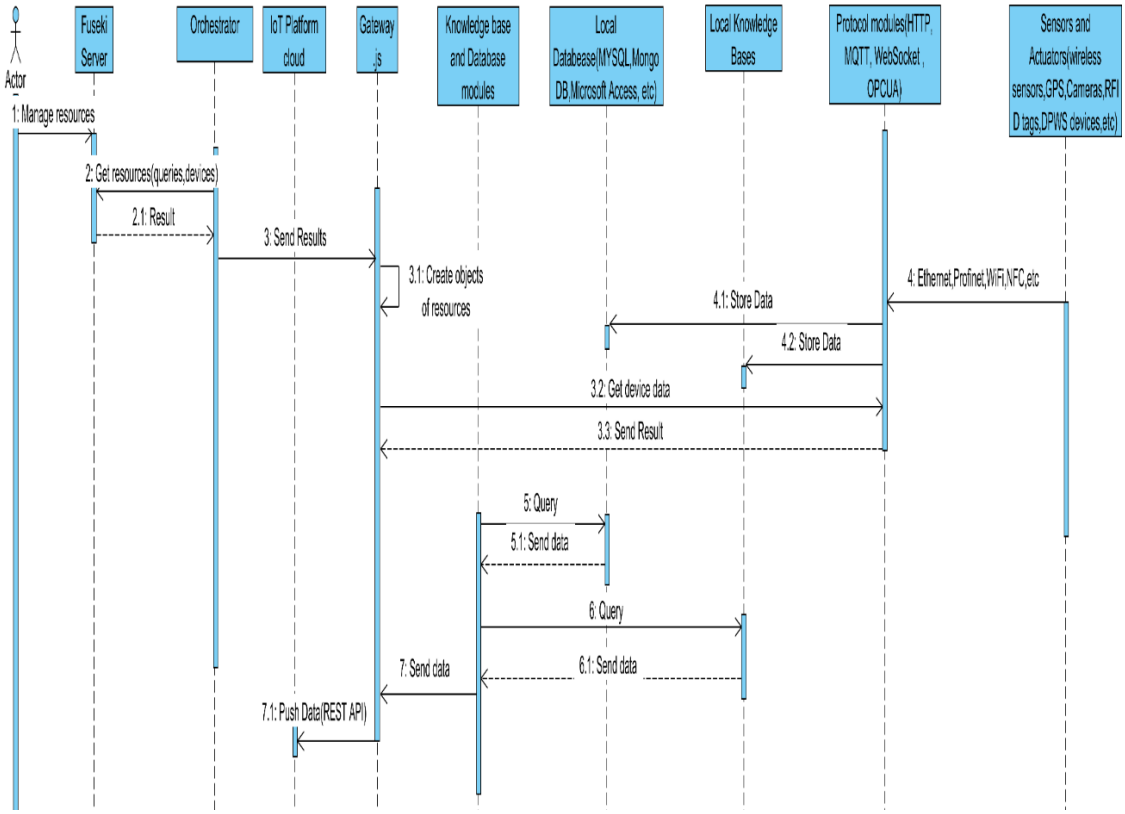


Figure 15. *The sequence diagram for the gateway configuration* [60]

This approach not only covers the interoperability across different protocols, but also makes the communication layer configurable and extensible so that new protocols can be supported by designing a new module and adding it to the system. In addition, the modules are isolated enough from each other, which makes the structure reusable. Furthermore, the loosely-coupled design of architecture allows creating/adding modules with no effect on the performance of other modules. This provides flexibility and extensibility because the modification of components can be accomplished without interrupting the operation of other components.

Each module in this architecture consists of two parts: core and configuration. The core is responsible for making connectivity to the device to get the data while configuration part sets the variables of protocol's attributes such as host and port. Module manager takes the lead for getting the configuration information for different protocols from ontology server and applying retrieved information to modules. The new configuration of modules can be registered into ontology via ontology editor. Figure 16 shows the sequence diagram for initiating a new module within the system. As it can be seen, module manager retrieves the all configuration needed information from ontology server. The module is initiated with the received configuration information (i.e. host, port, username, and password) from module manager.

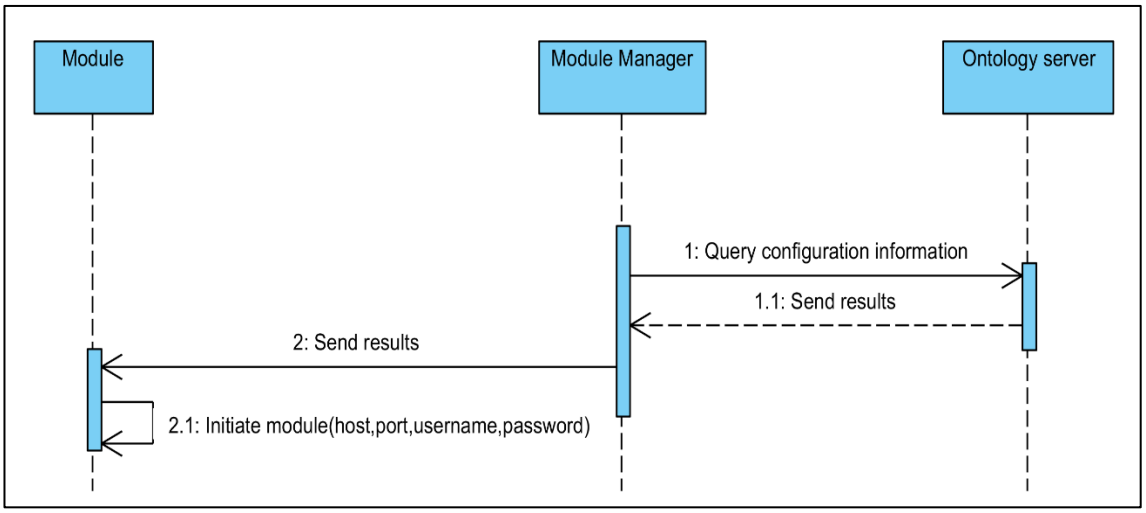


Figure 16. *Sequence diagram of initiating a module* [61]

Furthermore, the proposed architecture takes advantage of distributing the load of data by running multiple modules for a specific protocol in parallel. For instance, two separate HTTP modules can be run simultaneously to address the load-handling of network devices. This provides the horizontal extensibility and allows the connection of a large number of devices to push their data to cloud at the same time and consistently.

The development and change in the components of communication layer can be carried out by developers using available tools. For instance, if any change happens in data sources, such as adding a new device at run time, the ontology can be updated and the information of new device will be available dynamically by using existing ontology editor interfaces, such as Protégé.

3.1.3 Application layer

The highest level of the architecture is known as application layer, in which the retrieved data from downstream will be stored, analyzed and processed. With the aid of IoT platform services, such as big data analysis, machine learning, visualization and reporting tools, decision-makers gain insight into the data. Indeed, the application layer can be seen as front-end of the IoT architecture by which end users can interact with the system.

For the IoT architecture, the dashboard is an important component that transforms the gathered data from physical layer into comprehensible form for individuals using powerful visualization widgets, such as gauges, charts, tables, graphs, etc. Dashboards empower end users to monitor and control the physical assets of organizations.

In order to accomplish a complete solution for the visualization of data, the dynamicity and modularity of User Interface (UI) should be taken into account. In other words, the dashboard should be designed in a way that the designer can apply changes of content during run-time. For instance, adding, replacing and deleting elements within the existing dashboard should be possible in run-time without any interruption of visualization for users who are monitoring the data at the same time. Having a dynamic and modular visualization dashboard facilitates the modification and maintenance of user interface. Moreover, this loosely coupled approach allows us to achieve flexibility and develop UI in an extensible and reusable fashion. In addition, UI should be platform-independent, meaning that users can run the UI and monitor the data not only on any hardware (e.g. PC, smart phone, tablet, etc.), but also on any OS (e.g. Windows, Linux, Android, IOS, etc.). In this case, a user who can connect to the internet is able to access all the functionalities of a web-based user interface.

The proposed solution can be deployed in both public and private scenarios. For the health performance of cloud infrastructure, the platform owner is responsible. In the proposed architecture some tools for maintaining the cloud resources, such as Jelastic[7] and OpenShift[8] are introduced.

3.2 Ontology design

As mentioned previously, the functionalities of data sources are constructed using ontology-driven approach. Using ontology for modeling the system enables the (re)configuration of system components at run time. Knowledge-based systems provide a repository for storing the semantic data where multiple ontology models can be added or deleted. Web Ontology Language (OWL) is used for implementing ontology in this thesis. Ontology modelling can be implemented for each data source and its related attributes using ontology editor interface and data is stored in OWL files. The Protégé software, which is introduced in the programming technologies and tools chapter was employed to create the ontology model. The class diagram of data sources for ontology modeling is shown in Figure 17. The data sources are divided into two sub-classes. Devices represent the sensors/actuators on shop floor and queries indicate historical data that are being retrieved from databases. It should be pointed out that since the Fuseki server supports the HTTP REST methods, a web interface can be utilized to execute SPARQL update queries through RESTful WSs for modifying the current information of data sources.

Figure 18 illustrates the sequence diagram of interaction between different components of the proposed ontological approach. The user is able to create ontologies through ontology editors, such as protégé and the created OWL file is loaded to SPARQL servers such

[7] https://www.jelastic.com/
[8] https://www.openshift.com/

as Fuseki Server. The orchestrator queries to SPARQL server to get the list of existing data source and sends the results to the gateway. Then gateway creates the objects of two defined classes (i.e. device and query) to gain their associated data using the modules that are responsible for making the connectivity of their supporting data sources to the gateway. This approach allows for the dynamic adaptation of data sources without the interruption of the system. If any change happens to data sources, such as adding a new device at run time, the ontology can be updated and the information of new device will be available dynamically.

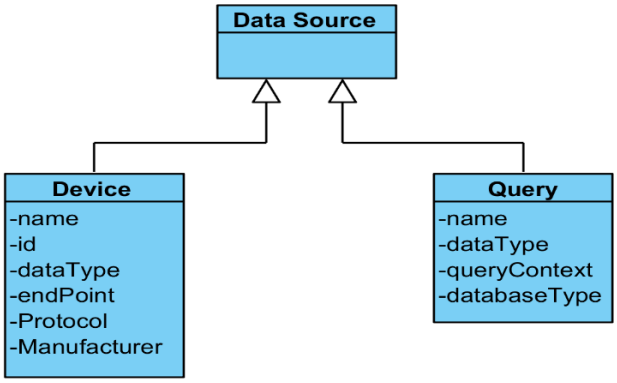


Figure 17. *The class diagram of data sources*

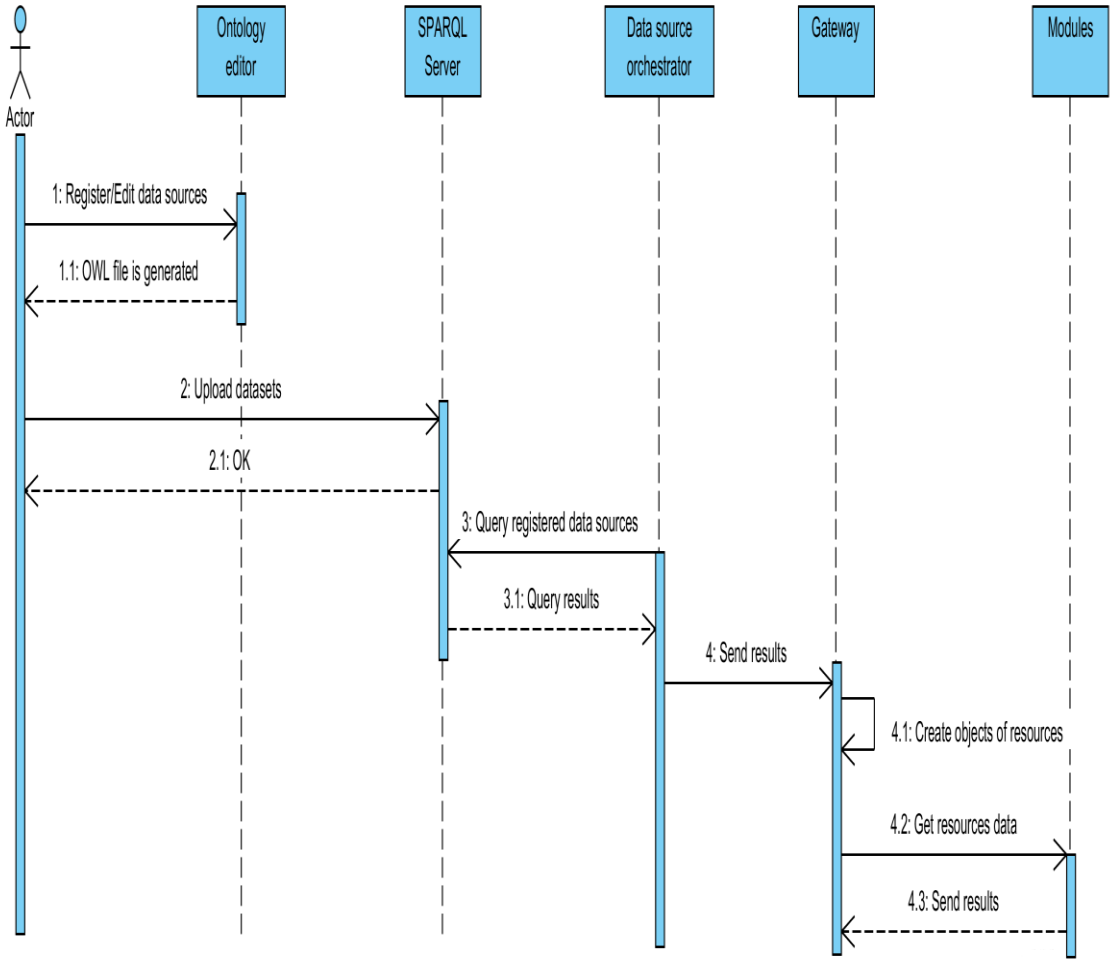


Figure 18. *Sequence diagram for the interaction of different components of proposed ontological system.*

The queries can be used to manage the stored information in the knowledge base. SPARQL queries are executed using Fuseki server through HTTP request. Query patterns include SELECT, DELETE and INSERT clauses to either get results or change the stored triples. Figure 19 illustrates a basic query pattern. It returns all stored RDF-triples in dataset. In retrieved triples, *predicate* defines the relationship between the *subject* and *object*.

```
SELECT ?subject ?predicate ?object
}WHERE{
 ?subject ?predicate ?object
}
```

Figure 19. *Example of basic SELECT query pattern*

Another query example for updating the information of data sources is shown in figure 20. This query pattern is designed to delete and change the endpoint of a device named D1. The query has PREFIX declaration in order to abbreviate the URIs. Once the query is applied to the dataset, the current endpoint of D1 device will be deleted and changed to "http://192.168.1.1/".

```
PREFIX iii:<http://www.semanticweb.org/MEI-
TUT/ontologies/2018/1/Resources#>
DELETE{
 iii:D1 iii:endpoint ?endpoint
}INSERT{
 iii:D1 iii:endpoint "http://192.168.1.1/"
}WHERE{
 iii:D1 iii:endpoint ?endpoint
}
```

Figure 20. *SPARQL query pattern for updating information*

SPARQL query language is powerful enough to support more complicated query patterns. In this study, SPARQL is used to manipulate ontologies throughout the implementation of solution.

Also, on the front-end side, the ontology design can be used to implement the visualization of data based on the role of users. The users' information and the associated visualization elements can be modeled using ontologies. The stored data can be requested by the application layer to render the visualization elements of dashboard in web browser. In this way, the reconfiguration of visualization dashboard can be carried out during runtime by adding the information of new visualization element to ontology model so that the ontology is updated and the dashboard is regenerated by requesting the information from ontology.

3.3 KPIs selection

Today, company employees need to have a proper perspective to the performance metrics according to the role and responsibilities they have within the organization chart. To visualize the data, different roles of the system should be identified. In addition, selecting appropriate KPIs that match the defined roles should be taken into consideration. In this thesis, to select efficient KPIs, the results of the survey which is conducted in [33] have been used. Regarding the results of survey, three sets of dashboards were designed according to the hierarchical structure of the organization. The roles that are identified to monitor the data are: manager, supervisor, operator and the designed dashboards are: operational dashboard, tactical dashboard and strategic dashboard for the specified roles respectively. KPIs for different roles are represented as following:

1. Operator: States of machines (i.e. idle and busy), errors, running hour, pending orders

2. Supervisor: Utilization of resources, OEE, number of manufactured products, number of rejected parts, failure time.

3. Manager: manufacturing costs, energy consumption

3.4 Programming technologies and tools

3.4.1 Node.js

Node.js is a platform based on the Chrome's V8 JavaScript engine designed to create fast and expandable applications. Node.js uses an event-driven and non-blocking I/O model. Thus, Node.js is a lightweight and efficient environment for implementing real-time and data-driven web applications that can easily run on decentralized servers[62]. While most existing server-side frameworks use synchronous architecture, Node.js uses asynchronous architecture. Node.js manages the asynchronous events using callback functions. For instance, if the server sends a request to read a file, the execution of program continues without blocking the server, thus the server can respond to other requests and callback can be invocated later when the content of the requested file is available. This style of programming is very different from the synchronous style, and it can be difficult to use in other languages.

Node.js is open-source platform that uses Node Package Manger (NPM) [63] to manage modules for the development of projects. NPM is the largest package registry in the world, including JavaScript packages. Developers from all over the world share and publish their modules to implement applications. In this study, the Node.js is used to develop

the back-end side. In particular, Express as a powerful Node.js framework was employed to handle data collection and interacting with cloud servers.

3.4.2 Protégé

Protégé is an open-source tool for constructing ontologies for knowledge-based systems. It was introduced by the Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine. It is based on Java and provides extensibility that enables designers to develop their application flexibly. It can be run on multiple platforms and support diverse storage formats including relational databases, XML and RDF.

Protégé takes the advantage of owning a user friendly user interface that allows user to create/edit ontologies in OWL conveniently. The Graphical User Interface (GUI) consists of a series of tabs, in which each tab provides a specific functionality of tool for users. In addition, Protégé allows to load multiple ontologies into a single workspace and the user can switch between ontologies dynamically.

3.4.3 Fuseki Server

Apache Jena Fuseki [64] s an open-source framework for running the standalone SPARQL servers. The Fuseki Server offers API support to allow user to extract/write data from/to RDF graphs. Having the feature of hosting ontologies, it provides the HTTP user-friendly interface for managing and updating RDF data by executing SPARQL queries. Datasets can be uploaded to the server through web interface running on a local host.

The triples stored in ontology files can be added, updated and deleted via Fuseki. In this thesis, the Fuseki server is used to retrieve/manipulate the data stored in ontologies.

3.4.4 Front-end dashboard selection

Dashboard has the key role for the implementation of the proposed solution. Dashboards are in front-end side to visualize metric information for decision-makers to gain insight into the most critical aspects of organization. As a result, selecting the proper dashboard technology is very crucial to fulfill the perfect solution. Based on the studies of the number of dominant IoT platforms in theoretical background section, Table 1 summarizes some features of the studied IoT platforms. These characteristics are selected due to their key role to meet the fundamental requirements of a successful solution.

Table 1. *Summary of different IoT Platforms*

Plat-forms	Type/Ar-chitec-ture	Support-ability (lan-guage support.)	Scala-bility	Interoperabil-ity(protocol support)	Reusabil-ity(Docu-mentation)	Price
AWS IoT Platform	Iaas/Cloud-based	Node.js, .NET, JAVA, PHP, Python, Ruby, Go, C++	millions of devices	HTTP, MQTT, Web-socket	Online documen-tation and tutori-als	Free (limited)/Paid depending on the number of mes-sages.
IBM Watson IoT	PaaS/Cloud-based	Python, Node.js, JAVA, C#	millions of devices	HTTP, MQTT, Web-sock-et	Online Tutorials and courses for customers	Trial free/ Paid de-pending on the size of messages
Microsoft Azure IoT Suite	PaaS/Cloud-based	.NET, Java, Python, Go, Node.js	millions of devices	HTTP, MQTT, Web-sock-et	Online documen-tation and tutorial videos, Commu-nity support	Free (limited)/Paid depending on the number of mes-sages.
ThingWorx	M2M PaaS/Cloud-based	.NET,C, JAVA	millions of devices	MQTT, HTTP	Online eLearning libraries, Courses and tu-torials. Commu-nity support.	Available on re-quest
Wapice IoT-Ticket	PaaS SaaS/Cloud-based	JAVA,C#, C++, Python	millions of devices	HTTP, OPC-UA	Online documen-tation, Built-in user manual	Available on re-quest
VersaSense IoT Fabric	PaaS/Cloud-based	Supported languages by Altizon Dato-nis, AWS and Azure	millions of devices	HTTP, MQTT, CoAP, Web-sock-et	Online documen-tation(white pa-per, datasheets, brochures)	Available on re-quest. Starter kit is about 850 euros.

As the table shows, all studied IoT platforms support the main features that are essential to carry out the IoT solution. Among them, Wapice IoT-ticket can run over SaaS or PaaS. Therefore, while the required resources are hosted by Wapice, the dashboard is a web-based solution, thus it is not required to install software on users' personal computers. Moreover, IoT Ticket provides an end-to-end solution, meaning that Wapice also supplies hardware such as WRM247+ to collect data from back-end side. WRM247+ is a robust device designed by Wapice to implement remote management, measurement and control-ling the devices on shop floor. It is utilized to integrate IoT platforms with legacy devices which use different interface modules or protocols such as CAN, Modbus, OBD and etc. Furthermore, IoT-Ticket provides the reporting feature which enables the users to create reports regularly for different roles of organization. Moreover, the access to dashboard

features can be managed by admin users so that it is possible to grant users permissions to the designed dashboards and their elements. The access rights are controlled by profiles e linked to users' accounts. There are four different profile roles: managers, engineers, operators, and viewers. Viewers can only see dashboards and reports, but cannot interact with widgets such as buttons, date pickers, charts, etc. Operators can interact with widgets, but have no permission to modify dashboards. Engineers and managers are able to edit dashboards.To implement and design dashboards, IoT-Ticket provides two tools: Interface Designer and Dataflow Editor.

The Interface Designer, which is illustrated in Figure 21, is a web-based tool that is used to create dashboard's user interface Within the Interface Designer, user can add visualization widgets, such as gauges, charts, buttons, images and so forth. To visualize the data values, data tags should be connected to widgets. Widgets can be added to the dashboard simply by dragging and dropping.

Another powerful tool which is included in IoT-Ticket is Dataflow Editor. Dataflow Editor allows to manage the flow of retrieved data from sources using graphical block programming. It is based on IEC 61131-3 standard [24]. Two main components of Dataflow Editor are *blocks* and *connections.* Blocks are connected to each other via connections to control the flow of data. Figure 22 shows a simple example of data flow within Dataflow Editor in which a button triggers the fetching of pressure values. The values are represented using gauge widget.

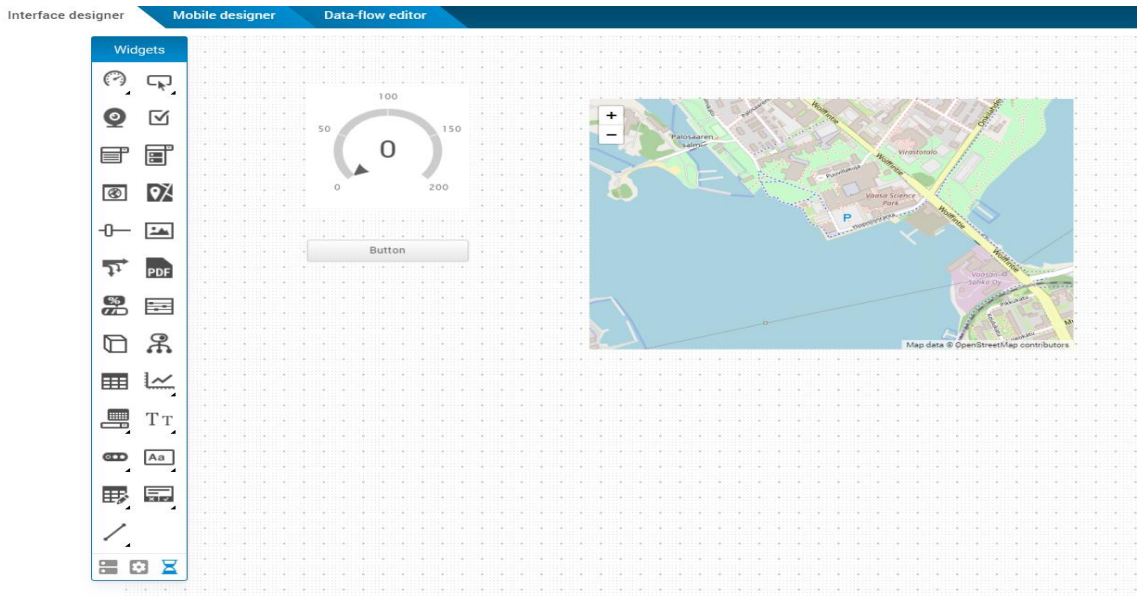


Figure 21. *IoT-Ticket Interface Designer*

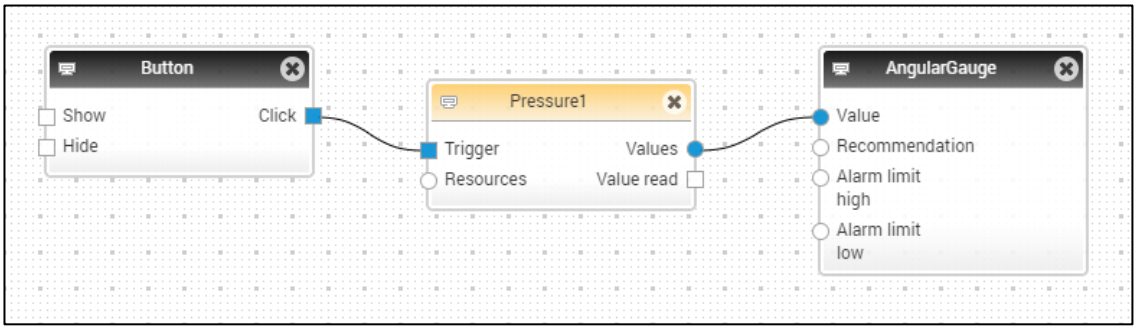


Figure 22. *Simple example of Dataflow Editor components*

IoT-Ticket supports REST API, which corresponds with the proposed architecture in this study. In addition, a comprehensive documentation is provided by Wapice to explain the methods to manage devices, create data nodes and write/read the values of data nodes. The security of IoT-Ticket is achieved by making requests through HTTPS protocol. The request should convey username and password to meet the reliable authentication. According to aforementioned factors, IoT-Ticket was selected as front-end technology in this study to accomplish the visualization of the received data from back-end side.

The provided API development guide by Wapice explains the guidelines for managing devices and data nodes using REST API. Figure 23 shows the sequence diagram for device registration and sending measurements to IoT-Ticket API server.

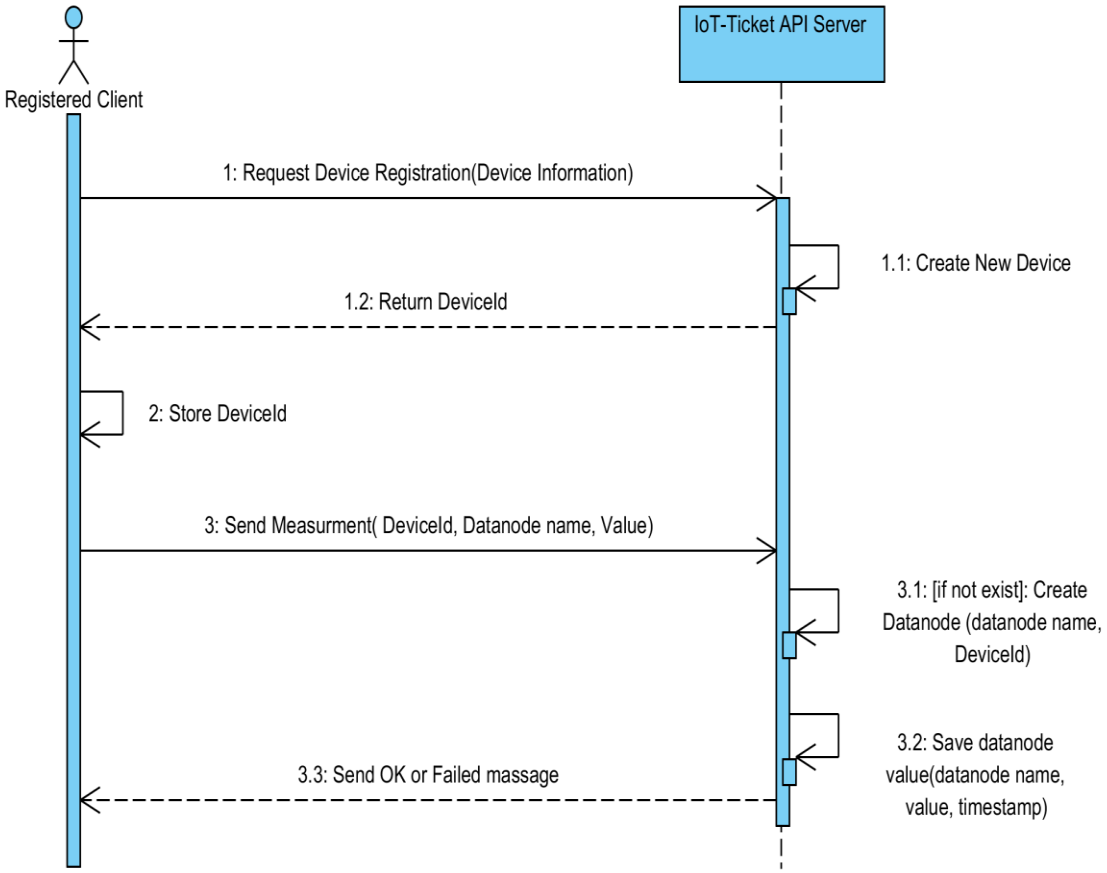


Figure 23. *Sequence diagram for device registration and sending measurements, adapted from* [65].

Different operations can be carried out through HTTP methods and using specific Uniform Resource Locators (URLs) that have been defined within API. These services and their associated URLs are summarized in Table 2.

Table 2. *IoT-Ticket API services* [65]

Service Description	Service URL	Service method	Authentication required
Register a new device	/devices/	POST	Yes
Get the list of registered devices	/devices	GET	Yes
Get the information Of a specific device	/devices/deviceID	GET	Yes
Get the list of device data nodes	/devices/deviceID/datanodes	GET	Yes
Write the values to data nodes	/process/write/deviceID/	POST	Yes
Read device data node values	/process/read/deviceID	GET	Yes
Get a client quota information	/quota/all	GET	Yes
Get quota in-formation for a specific device	/quota/deviceID	GET	Yes

Figure 24 shows an example the invocation of a Post service for a provided URL to write a value into data nodes of a specific device.

```
POST:  /process/write/deviceID
body:
[{  "name": " Temperature ",
     "path":"station1/main",
      "v": 14}]
```

Figure 24. *An example of POST method invocation to write a value to data node*

4. IMPLEMENTATION AND RESULTS

In this chapter, the implementation of the proposed approach in chapter three is presented. The implementation involves the utilization of a test-bed production line "Festo didactic training line" located in Seinäjoki University of Applied Sciences (SeAMK) and results have been validated. The chapter is divided into four parts so that the first section introduces the test-bed production line and its characteristics. The second section discusses the design of queries to crawl the database. The third section explain the implementation of back-end side to collect and deliver data to the application layer and the last section discusses the front-end side implementation, including designing the user interfaces and visualization of data for end users using IoT dashboard.

4.1 Festo didactic training factory case-study

The proposed solution is implemented using "Festo didactic training line" as the test bed. The line is composed of four stations: Buffer Storage, Machining, Robot cell and Quality assurance along with a conveyor for the transportation of products. The stations have separate control systems that use Siemens and Beckhoff PLCs to control the process of unit. The PLCs are connected to production control systems through Ethernet network. The line is capable of receiving orders from different customers and manufactures' diverse products. Each product consists of three components: base circuit board, two circuit board fuses and cover plate that can be assembled in three different colors: red, blue and green. The conveyor is in charge of moving pallets between stations. The pallets are equipped with RFID tags to store the information of pallets, the task that should be done for each pallet throughout the process. Thus, when the pallet reaches each station, the work cell carries out the tasks according to the stored information contained in RFID tag. Once, the tasks are done, the station asks the production control system for the next step of work and writes the received information to the memory. Finally, the station releases the pallet to move to the next station for the remaining tasks.

The buffer storage stores the finished products and raw materials that are used for the production. The Siemens's PLC is used to control the buffer station. The machining station makes holes on the base plate pneumatically and is equipped by Beckhoff PLC. The robot cell employs ABB robot to fit the fuses on the circuit board and fix the cover plate to accomplish the assembly task. The robot station is controlled via Siemens PLC. The quality assurance station inspects the product that is released from robot work cell. Beckhoff's PLC is in charge to control this station. Figure 25 illustrates the arrangement of different stations of production line.

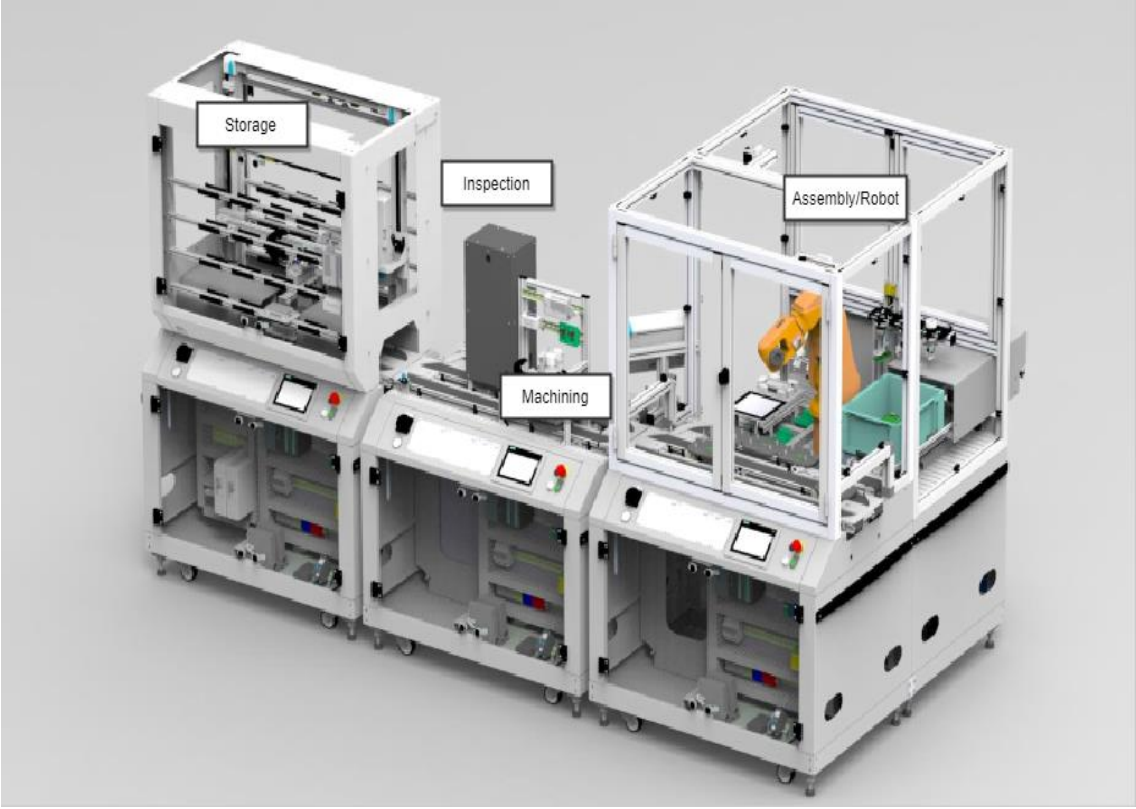


Figure 25. *The layout of Festo production line case-study* [66]

The entire system is integrated with the industrial information systems, including Enterprise Resource Planning (ERP) and Manufacturing Execution System (MES). MES4 is Festo's proposed MES (Manufacturing Execution System) for the implementation of Industrie 4.0 which is integrated with an Access database. During production, the database is updated by generated data from the line. MES4 enables us to read orders and update status, create warehouse data and material buffer, create and manage the information of customers, etc. The Microsoft Access database consists of different tables including orders, machines status report, operations information, parts, work plans definitions, stations' activity information, etc. Figure 26 illustrates the architecture of MES4 system for the cyber-physical factory of Festo. The stations are connected to PLCs via Fieldbus connections. The PLCs communicate with VB.net Program over TCP/IP protocol while the database is connected to Microsoft Access database through Open Database Connectivity (OBCD). Designed queries to crawl the database

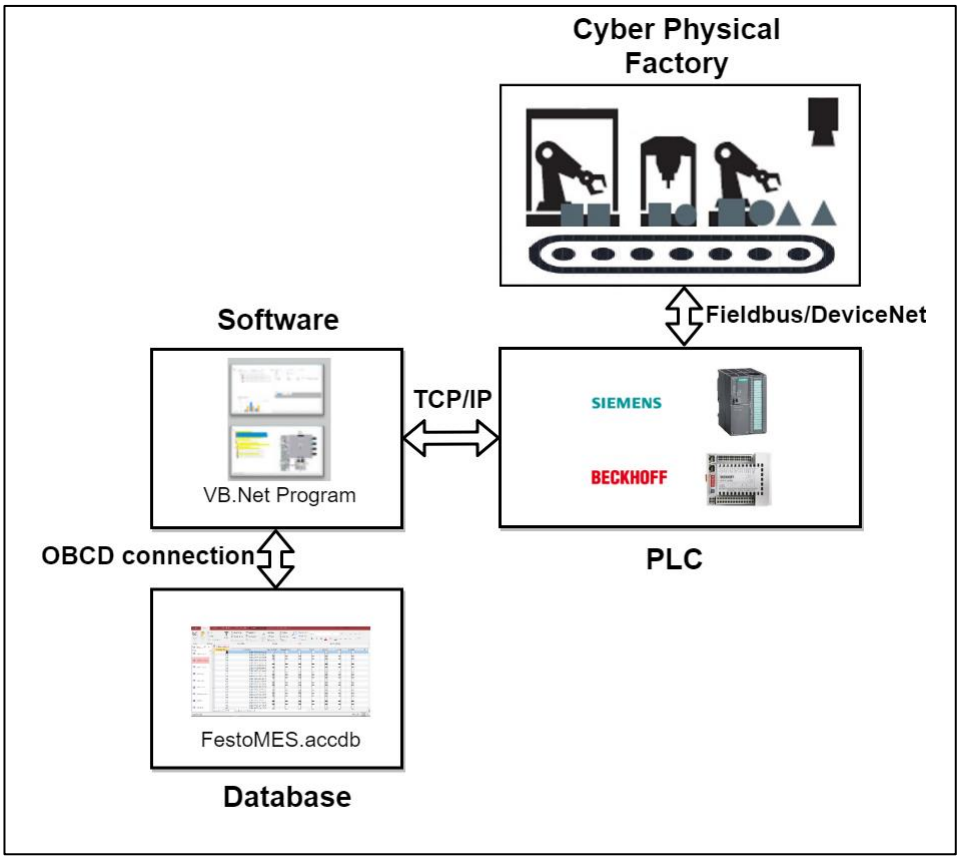


Figure 26. *The architecture of MES4 of Festo smart factory*

4.2 Design queries to crawl the database

To achieve the data of selected KPIs from Microsoft Access database for the visualization at front-end side, the set of queries was designed. Table 3 lists the designed queries.

Table 3. *Designed queries to crawl the database*

Query name and description	Query
MachinesStatus Query to get the status of each station.	`SELECT * FROM qrMAchineReport WHERE ID IN (SELECT MAX(ID) FROM qrMAchineReport GROUP BY ResourceID)`
TotalNoProducts	`SELECT SUM(MaxOfOPos) AS TotalNoOfProducts FROM qrMaxAndSum`

Query to get the total number of produced products.	
TodayNoProducts Query to get the number of produced products on the current day.	`SELECT SUM(MaxOfOPos) AS TodayNoOfProducts FROM qrMaxAndSum WHERE qrMaxAndSum.Date = Date()`
NumberOfDoneOrders Query to get the number of finished orders	`SELECT COUNT(ONo) AS TotalNumberOfOrders FROM tblFinOrder`
TodayNumberOfDoneOrders Query to get the number of finished orders on the current day	`SELECT COUNT(ONo) AS TodayNumberOfOrders FROM tblFinOrder WHERE (((CVDate(Int([tblFinOrder.End]))) = Date()))`
RunTime Query to get the run time of production line	`SELECT SUM(TimeDifference) AS TotalRunTime FROM qrFinStep WHERE ResourceID=64`
FailureTime Query to get the failure time of production line	`SELECT SUM(TimeDifference) AS FailureTime FROM qrFinStep WHERE StepNo=999`
FailureNumbers Query to get the number of failures during production.	`SELECT COUNT(TimeDifference) AS NumberOfFailures FROM qrFinStep WHERE StepNo=999`
CycleTimeAssemble Query to get the cycle time of assembly operation	`SELECT AVG(TimeDifference) AS cycleTime FROM qrFinStep WHERE ResourceID=64 AND OpNo=311`
CycleTimeDisassemble	`SELECT AVG(TimeDifference) AS cycleTimeDisassemble FROM qrFinStep WHERE ResourceID=64 AND OpNo=316`

Query to get the cycle time of disassembly operation	
StorageUtilization Query to get the time that storage station has been in the operation during production.	`Select SUM(TimeDifference) AS StorageTime FROM qrFinStep WHERE ResourceID=61`
DrillUtilization Query to get the time that machining station has been in the operation during production.	`Select SUM(TimeDifference) AS DrillTime FROM qrFinStep WHERE ResourceID=63`
RobotUtilization Query to get the time that robot station has been in the operation during production.	`Select SUM(TimeDifference) AS RobotTime FROM qrFinStep WHERE ResourceID=64`
InspectUtilization Query to get the time that inspection station has been in the operation during production.	`Select SUM(TimeDifference) AS InspectTime FROM qrFinStep WHERE ResourceID=68`
ManualUtilization Query to get the time of manual operation during production.	`Select SUM(TimeDifference) AS ManualTime FROM qrFinStep WHERE ResourceID=70`
PendingOrders Query to get the list of pending orders.	`SELECT tblOrderPos.ONo, tblOrderPos.OPos, tblCustomer.Company, tblStepDef.Description AS Task, tblWorkPlanDef.Description AS [Production Order], tblStates.Short AS State, tblResource.ResourceName AS Resource, tblOperation.Description AS Function FROM (tblCustomer INNER JOIN tblOrder ON tblCustomer.CNo = tblOrder.CNo) INNER JOIN (tblOperation INNER JOIN (tblResource INNER JOIN (((tblStepDef INNER JOIN tblOrderPos ON (tblStepDef.StepNo = tblOrderPos.StepNo) AND (tblStepDef.WPNo`

	= tblOrderPos.WPNo)) INNER JOIN tblWork-PlanDef ON tblOrderPos.WPNo = tblWork-PlanDef.WPNo) INNER JOIN tblStates ON tblOrderPos.State = tblStates.State) ON tblResource.ResourceID = tblOrderPos.Re-sourceID) ON tblOperation.OpNo = tblOr-derPos.OpNo) ON tblOrder.ONo = tblOr-derPos.ONo WHERE (((tblOrderPos.State)<90) AND ((tblOrder.Enabled)=True)) ORDER BY tblOrderPos.PlanedStart, tblOr-derPos.ONo, tblOrderPos.OPos;
ElecEnergyConsumption Query to get the amount of consumed energy for the pro-duction	Select SUM(ElectricEnergyCalc) AS Elec-tricEnergy FROM tblFinStep

4.3 Back-end Ontological approach implementation

As it was mentioned previously in the approach part, the data sources are considered *de-vices* for real-time data collection and *queries* to get historical data from databases. Figure 27 shows the sequence diagram for registering a new data source to the knowledge base system. After registration, data sources are discoverable by the gateway. For the case-study of this thesis, the data from shop floor is collected and stored in Microsoft Access database. As a result, to implement the ontology, the data sources that are needed to de-velop are queries. Thus, the *query* class is defined in ontology. The associated attributes with *query* class are *name*, *query*, *dataType* and *databaseType*. In this study, the protégé software was used to instantiate the queries in ontology as it is shown in Figure 28. Each instance represents a query along with its attributes. Using ontology enables the dynamic adaptation of data sources without the interruption of the system. If any change happens in data sources, such as the addition of a new device at runtime, the RDF ontology can be updated and the information of the new device will be available dynamically. This can be achieved by SPARQL Update language.

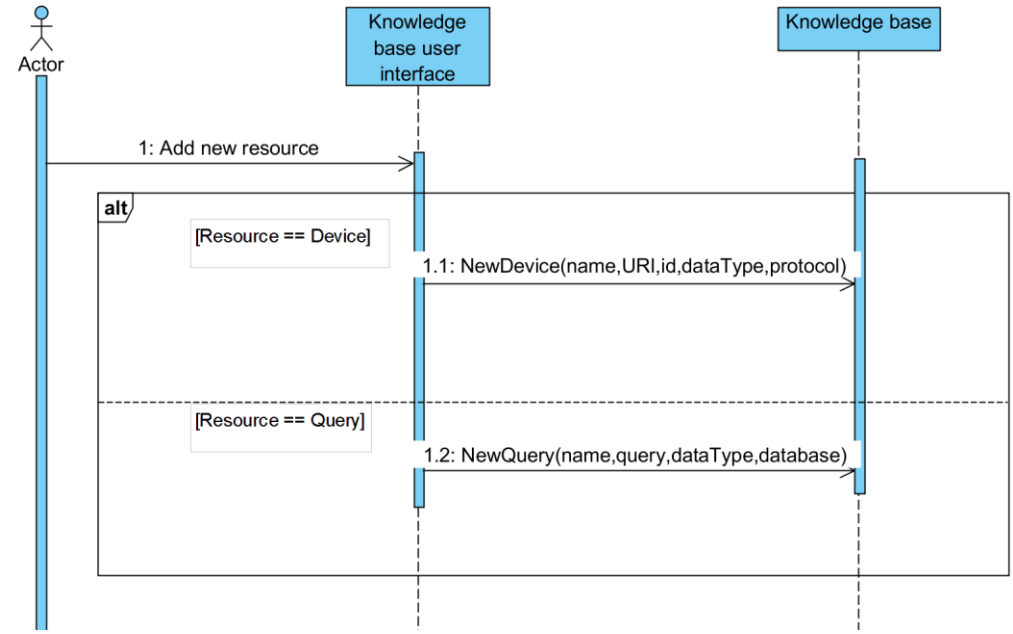


Figure 27. *Sequence diagram for adding a new data source* [60]

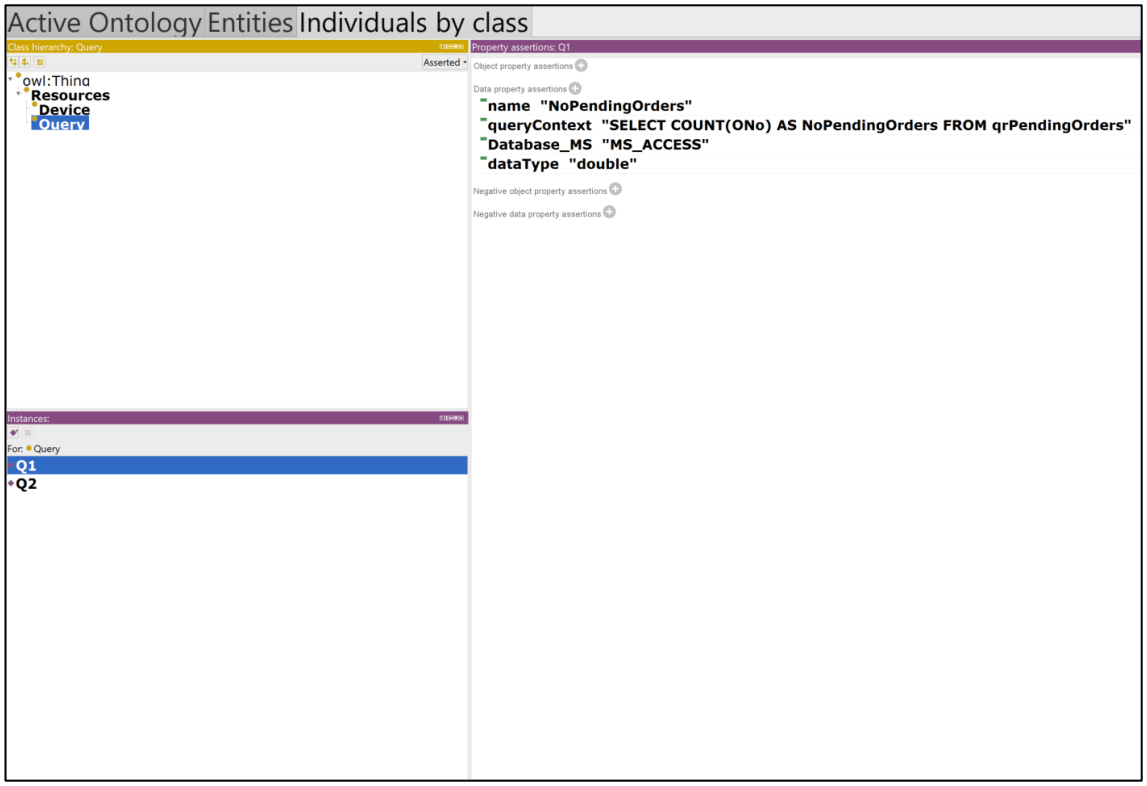


Figure 28. *Instantiation of queries using Protégé*

As it was already stated in the methodology part, in this thesis Node.js is selected to fulfill data collection at the back-end side and deliver collected data to the IoT platform. To implement the proposed solution, firstly the OWL file is uploaded to Fuseki server. Then

the orchestrator sends a SPARQL query to Fuseki server over HTTP request. Once the query is executed within Fuseki Server, the results will be returned to gateway using orchestrator callback function. The class for query is already created in Node.js environment. When the gateway receives the instances of queries through orchestrator, it creates the objects of the query class. Next, the gateway uses "MsAccess-module" callback function in order to query the database and retrieve the data from database. The parameter of callback function is the query context, which was designed and represented previously in Table 3. Next, the retrieved data from database is pushed to IoT-Ticket platform using POST method.

Figure 29 shows the sequence diagram for the implementation of data collection using the ontological approach. Also, Figure 30 shows the query pattern to get the information of the queries that are stored in ontology and the results including the information of each instance and its attributes.

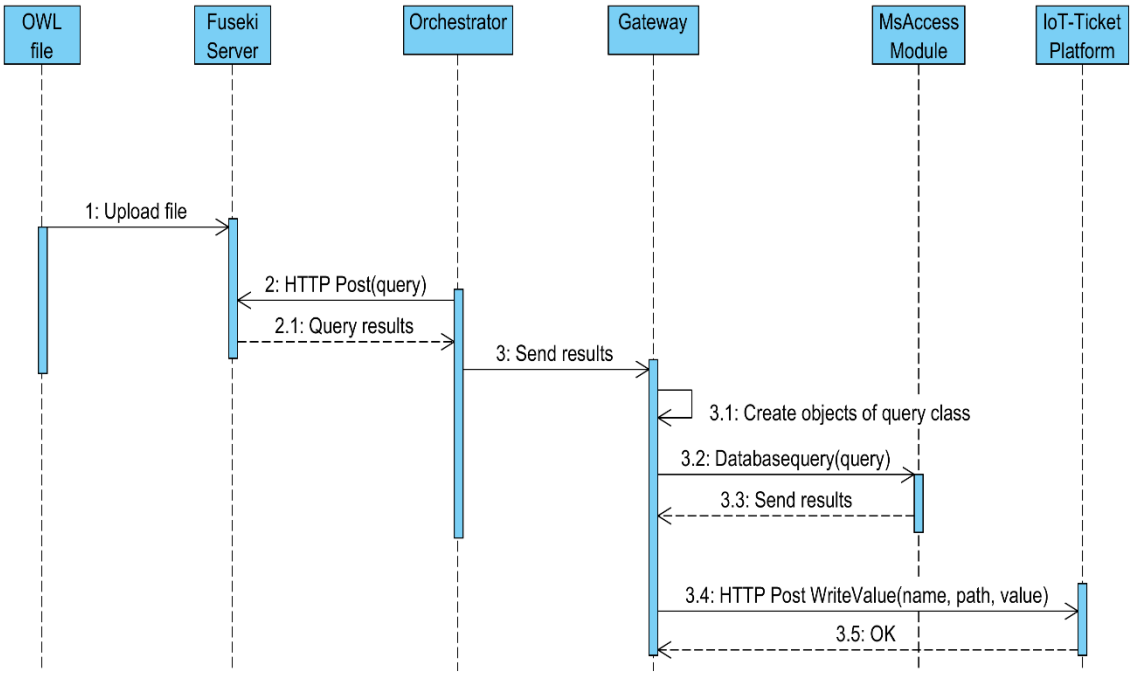
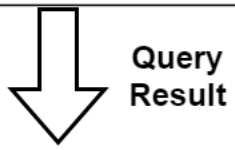


Figure 29. *The sequence diagram for the implementation of back-end ontological solution*

```
PREFIX iii:<http://www.semanticweb.org/MEI-TUT/ontologies/2018/1/Resources#>
SELECT *
WHERE { ?inst a iii:Query.
        ?inst iii:name ?name.
        ?inst iii:queryContext ?queryContext.
        ?inst iii:dataType ?dataType.
        ?inst iii:Database_MS ?Database_MS. }
```



inst	name	queryContext	dataType	Database_MS
iii:Q2	"TodayNoFinishedOrders"	"Select COUNT(ONo) AS NoFinishedOrders FROM qrFinishedOrders"	"double"	"MS_ACCESS"
iii:Q1	"NoPendingOrders"	"SELECT COUNT(ONo) AS NoPendingOrders FROM qrPendingOrders"	"double"	"MS_ACCESS"

Figure 30. *The query pattern to get query instances from ontology and example result*

4.4 Front-end Implementation

After delivering data to IoT-Ticket, the next step is the front-end side development to allow the end users to interact with the system. This part consists of two sub-sections. The first section presents the designing process of the user interface to communicate with IoT-Ticket server for the device management. The second part discusses the design and implementation of three dashboards for data visualization based on the role of the end user.

4.4.1 User interface for IoT-Ticket

In order to enable the end user to manage the devices registered in IoT-Ticket server, a user interface was designed. This user interface can be used to add new devices along with its attributes as it is shown in Figure 31. The name field defines the name of the device. The name of device's manufacturer should be filled in the next field of form. Type

field describes the associated category of the device, such as sensor, mobile phone, etc. The third field offers the description of devices. Finally, the last field includes the additional attributes of devices that can be stored on the server. All fields are required to be filled expect the attribute, which is optional. Once the *submit button* is clicked, the list of existing devices will be retrieved by Get method using API provided by Wapice. If the name of the device does not exist in the list, then the device will be registered on the server using Post method. After the registration of the device, the measurement values can be sent to the server for further operations.

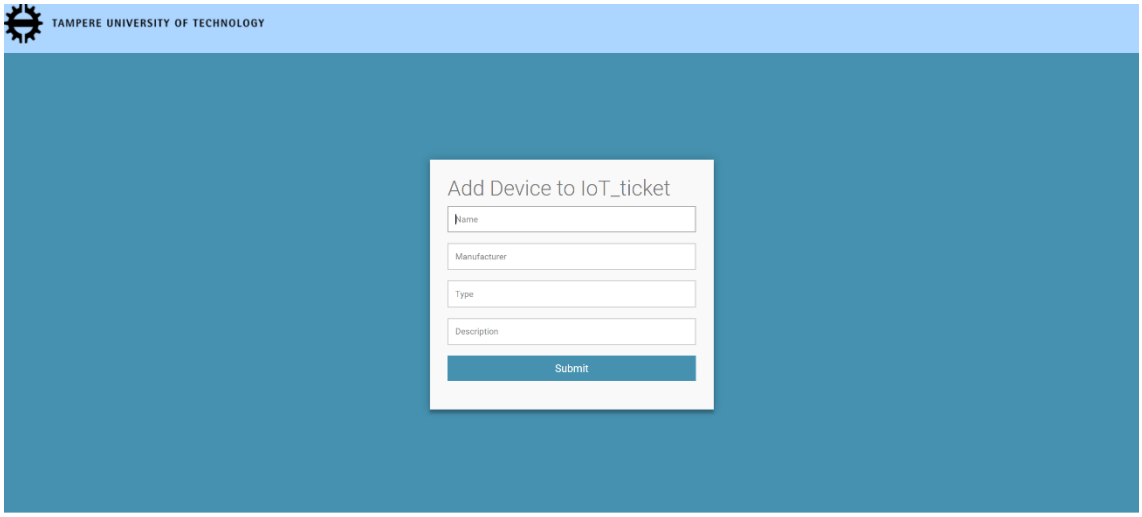


Figure 31. *User interface to add new device on IoT-Ticket's server*

4.4.2 Dynamicity of User Interface

As it was discussed in approach section, it is a crucial aspect for decision-makers of the enterprise that the visualization elements of user interface can be reconfigured dynamically at run-time. In particular, for monitoring the real-time data, the visualization dashboard should be operational continuously to enable the end users to monitor critical activities within enterprise without any interruption. The advantage of employing the IoT-Ticket for the implementation of monitoring dashboards is its capability for changing the interface dynamically. New visualization elements such as gauges or charts can be added to the existing dashboard at the run-time. This feature, prevents the monitoring disruption for decision-makers. Figure 32 shows an example of adding new widget to dashboard at run-time. As it can be seen, on the left side one visualization widget i.e. the gauge is represented for end user. While the end user monitors the dashboard, one new widget is added to the dashboard by developer behind the scenes. The new element will be shown for the end user by refreshing the web browser which can be seen on the right side.

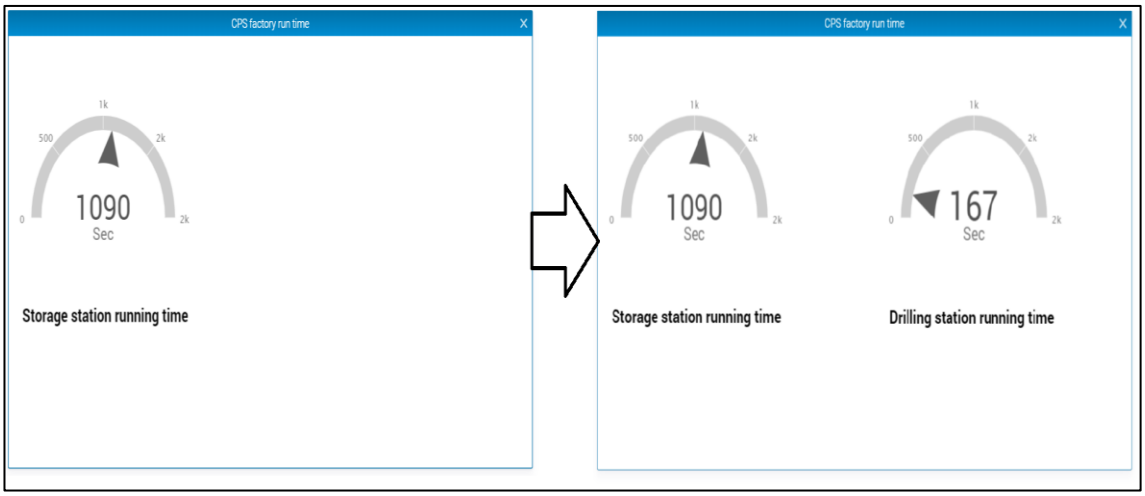


Figure 32. *The dynamicity feature of IoT-Ticket dashboard*

4.4.3 Designing and implementing dashboards

After delivering data to IoT-Ticket server, the design and creation of dashboards can be carried out. IoT-Ticket allows to create multiple dashboards and users can navigate through different dashboards using the navigation pane as it is shown in Figure 33. To deploy dashboards, online Interface Designer is used along with Data-Flow Editor to manage the flow of data which is displayed on the dashboard. The Interface Designer enables the designer to include desired visualization elements within dashboards by drag and drop operation.

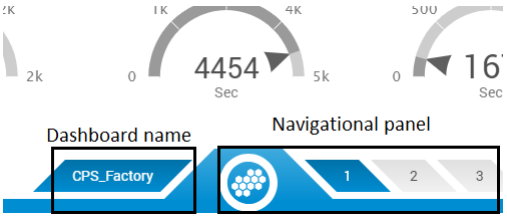


Figure 33. *IoT- Ticket Navigational panel to switch between different dashboard*

The first dashboard is designed for the operator to monitor the performance of units on shop floor. The KPIs that are visualized within operational dashboards are discussed in the following. One of the most important indicators of the operator is the status of each unit to observe whether the unit is *on/off* or is on *idle* mode. To achieve this, the retrieved

data for each station is managed in Data-Flow editor. Figure 34 illustrates the aforementioned data management for Storage unit as an example using logic blocks such as If-then-else. The timer interval is set to 1000 ms to trigger value every one second.

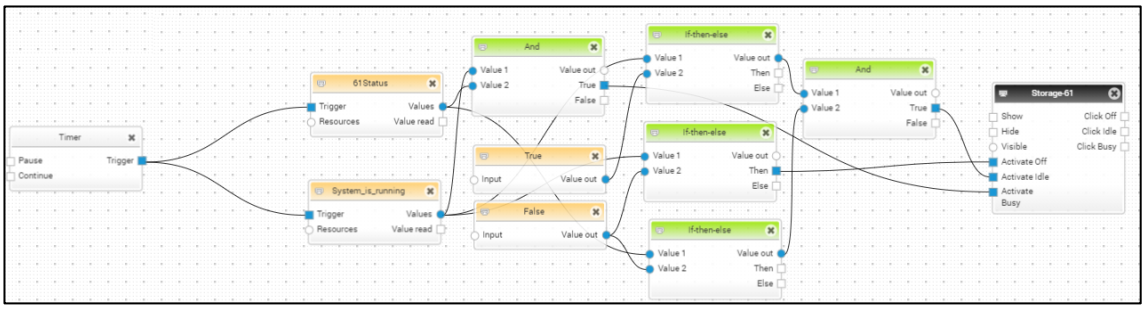


Figure 34. *Data flow management for status of Storage unit using Data-Flow editor*

Moreover, the Operational dashboard throws an error sign if any error occurs during operation. As Figure 35 shows, the Boolean value of error data nodes are compared to a constant Boolean value using logic blocks to show whether each station is running normally or a problem has occurred during the manufacturing process.

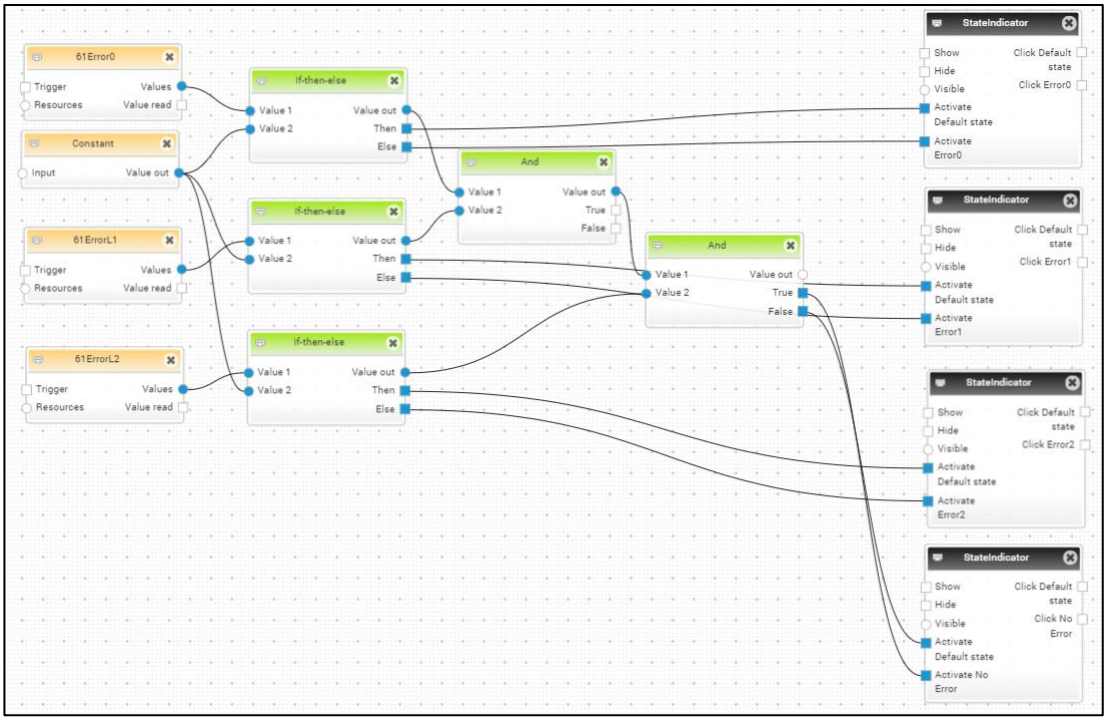


Figure 35. *Data flow management for error displaying on the dashboard*

Also, Figure 36 shows the data flow management for representing the running hour of each unit using gauges. This information will be used for the maintenance and control of

machines. In addition, the pending orders during the manufacturing process are represented using table.

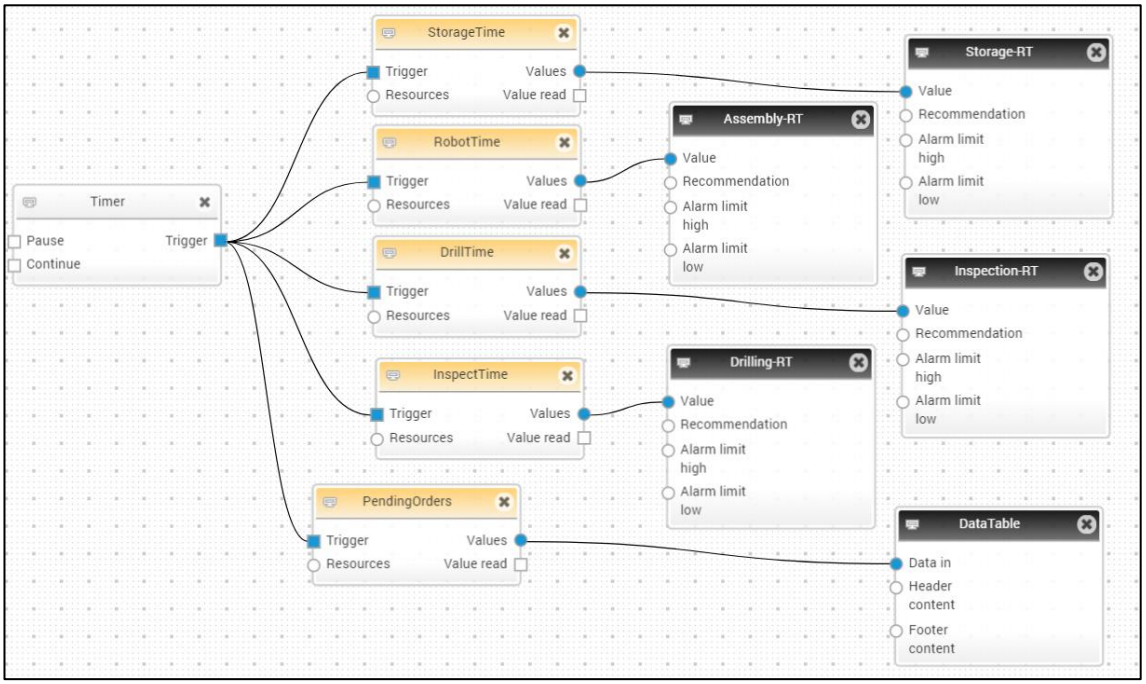


Figure 36. *Data management flow for working time of units and pending orders*

Within the Tactical dashboard, chart block is used to connect data tags to visualize the utilization of resources in a chart format. As it can be seen in Figure 37, *Series* are the inputs of chart element that can be added or removed with the *Series configuration* property. Also combined Operational Equipment Effectiveness (OEE) calculation of station is carried out by utilizing calculation blocks, such as Subtraction, Multiply and Division. Again, in the Strategic dashboard the calculation blocks are used to compute the cost of production.

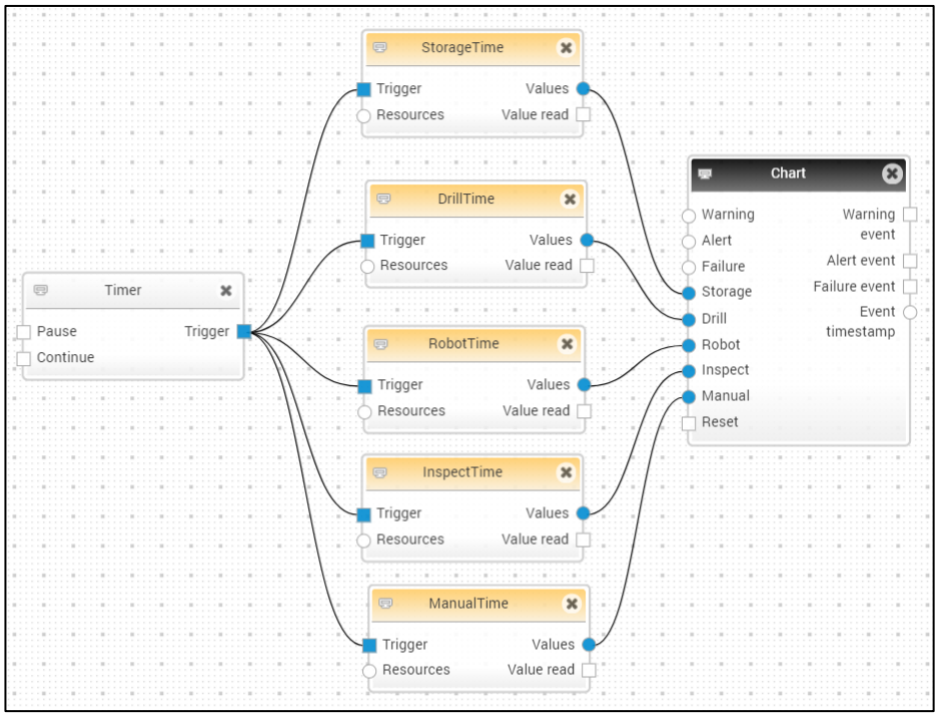


Figure 37. *Data flow management using chart element to visualize the utilization of resources in chart format.*

By managing the data flow in IoT-Ticket server, the visualization widgets can be arranged within dashboards to display the data for end users. Regarding the studies in this thesis, data should be visualized for different users of enterprises according to their responsibilities. Figure 38 shows the sequence diagram for the interaction of different roles of the system and the flow of data. Gateway delivers collected data from Festo CP factory and MES4 to IoT-Ticket Platform. The measurement values are stored on IoT-Ticket server for further operations. Different users are entitled to login to the platform according to their roles. The admin user is in charge to manage the user accounts and their permissions to the platform. IoT-Ticket allows specific access to users. For in-stance, the admin can allow a user to access the read-only dashboard without permission to edit it.

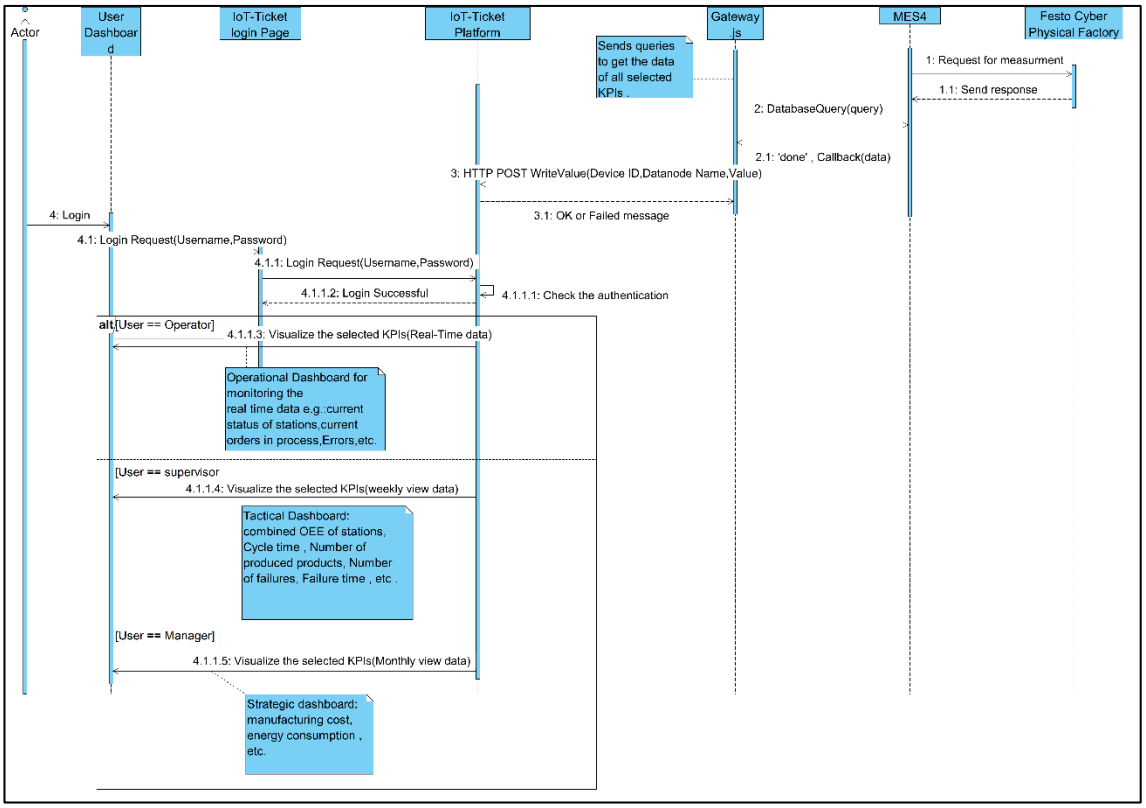


Figure 38. *Sequence diagram for the interaction of different roles*[60]

Figure 39 shows the implementation of Operational dashboard to represent the critically data for a worker on shop floor, including the status and running time of each work cell. The layout of CPS factory containing all stations was included in the dashboard to give a more intuitive perception to the operator for the associated data of each station. On top of each station symbol, the status of unit is represented to show the current state, such as *Off*, *Idle* and *Busy*. Below the blue rectangle which represents conveyor, the indicators of errors are placed to notify the operator about any abnormal problem during manufacturing process. The red colour signs are considered for errors to help the operator to spot errors readily among other elements of dashboard to act as soon as an error occurs. At the bottom of Operational dashboard, the working hour of units are provided using gauge widget. Since the production line runs only for educational purposes, the working hour of the whole system is less than real production lines. As a result, the time unit for the running hour of stations was selected to be second. Finally, on the top right side of the dashboard, a table widget is placed to show the job queue of the manufacturing process. It consists of four columns in which the order number and its associated task and state are represented. Also, each row conveys the name of the responsible station to carry out the task.

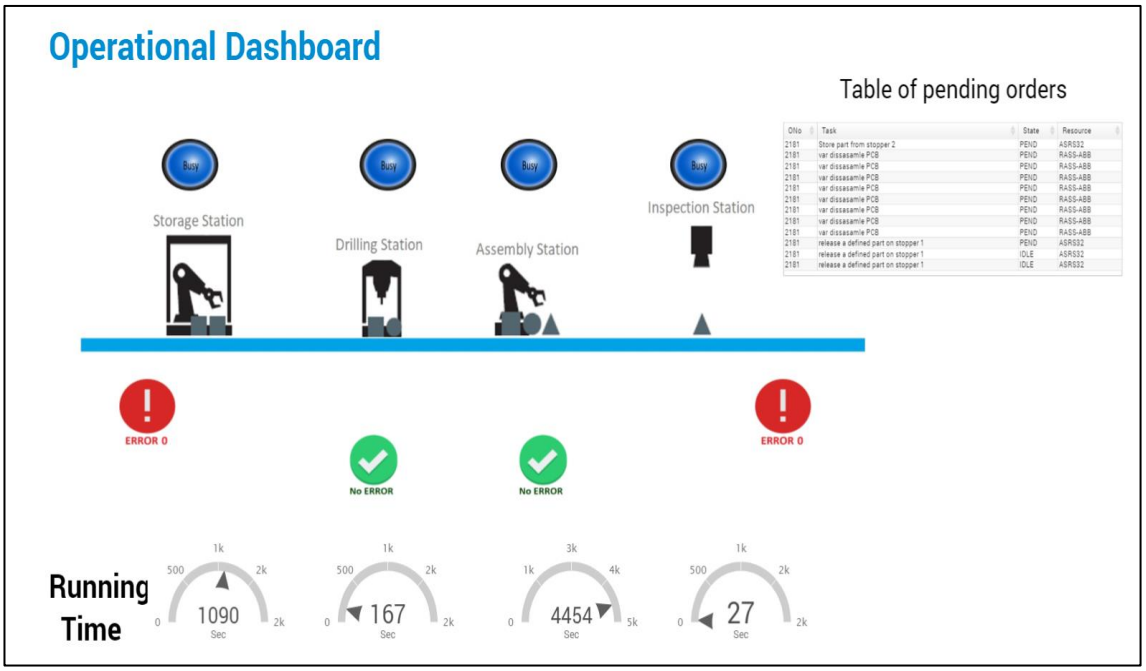


Figure 39. *Operational dashboard for operator at factory floor*

The tactical dashboard which is shown in Figure 40 provides the data for mid-level managers or supervisors to enable them to analyze the performance of processes according to the objectives of the enterprise. On top of dashboard, a number of important KPIs such as the number of products, rejected products, cycle time and failure time are shown using a linear gauge. Moreover, the total time of running time of the production line is displayed. Also, combined OEE calculation as a key KPI and its three main components (i.e. availability, performance and quality) are represented to give the mid-manager insight into the effectiveness of production to be used for improving manufacturing productivity. Also, at the bottom of dashboard, the pie chart is employed to show the utilization of different resources during manufacturing processes. Pie chart is an effective widget for comparing the percentage of stations that have been in use during production time.

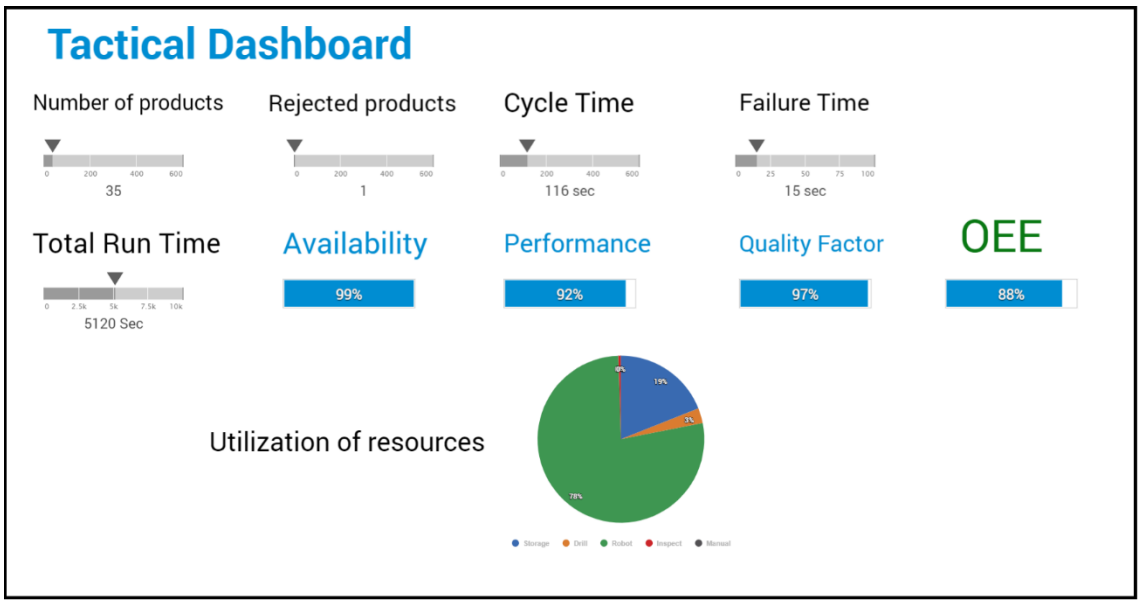


Figure 40. *Tactical dashboard for mid-level supervisor*

Finally, the Strategic dashboard is designed for senior managers who need insight into the performance metrics of the system to find the optimal strategy for the business. The energy consumption during production is shown using the gauge. Also the total production cost was calculated according to the hypothetical raw material and energy consumption cost. The total cost of production and manufacturing cost per product are visualized by the gauge. The Strategic dashboard is shown in Figure 41.

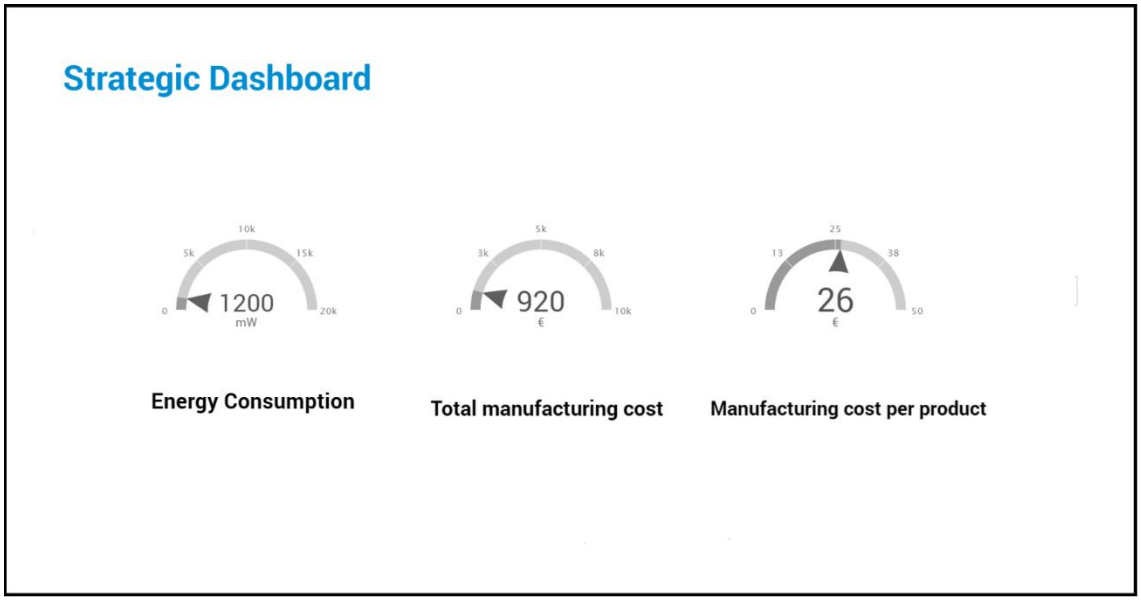


Figure 41. *Strategic dashboard for high-level manager*

5. CONCLUSIONS

5.1 Conclusion of Implementation and Results

The competitive market among modern manufacturing enterprises has resulted in utilizing new technologies to meet the customers' demands. In the current era, Internet of Things has opened new horizons for organizations to leverage provided IoT services to monitor the activity of assets to thereby enhance the productivity and agility of an enterprise's performance. To achieve this, the data from enterprise's assets should be collected to be processed and analyzed to add value to business. Also, it should be taken into consideration that in the IoT concept, any physical object can be incorporated into the IoT and connected to the Internet. Thus, the IoT will comprise a wide range of heterogeneous devices, each using its own protocol for the exchange of data. To solve the heterogeneity problem in the IoT domain, an architecture must be developed that bridges the gap between the physical devices and the higher levels of IoT services. Meanwhile, since the number of devices on shop floor is growing rapidly, the volume of generated data will be enormous. By using visualization the data can be summarized and represented in an easy-to-understand form to provide insightful information for users.

This thesis introduces the design of role-based visualization for industrial automation systems employing IoT technologies. A number of prominent IoT platforms available in the market were studied, and some characteristics were compared to each other. Also, an approach was introduced to implement a modular and extensible architecture where future growth of data sources and communication protocols in backend side was considered. The proposed architecture in this thesis provides the mechanism for discovering IoT devices using ontology-driven approach while different IoT protocols can be integrated into a gateway. It was discussed that the ontological approach allows the run-time reconfiguration of entities which increases the agility and flexibility of the system as high demand in the current manufacturing environments. Furthermore, a set of KPIs was defined and visualized for different users of enterprise regarding their role and responsibilities. For the visualization, three different dashboards were designed to monitor the performance of organization using the visualization tools of an IoT platform.

5.2 Future Work

According to the objectives of thesis, for the designed architecture the heterogeneity of data sources was considered while the proposed solution supports the extensibility and flexibility features in IoT domain. However, to improve the proposed solution, the possible future works are discussed in the following.

Future work will consider the development of dashboards so that more KPIs can be specified to evaluate the organization objectives more accurately. Also, for the development of the ontological approach, design and implementation of web user interface to support the RESTful operations for CRUD (Create/Read/Update/Delete) management can be a part of future work.

REFERENCES

[1] IBM, “10 key marketing trends for 2017,” p. 18, 2016.

[2] D. A. . b c Keim, “Information visualization and visual data mining,” *IEEE Trans. Vis. Comput. Graph.*, vol. 8, no. 1, pp. 1–8, 2002.

[3] M. Wollschlaeger, T. Sauter, and J. Jasperneite, “The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0,” *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 17–27, 2017.

[4] G. Reinhard, V. Jesper, and S. Stefan, “Industry 4.0: Building the digital enterprise,” *2016 Glob. Ind. 4.0 Surv.*, pp. 1–39, 2016.

[5] R. Drath and A. Horch, “Industrie 4.0: Hit or hype?,” *IEEE Industrial Electronics Magazine*, vol. 8, no. 2. pp. 56–58, 2014.

[6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[7] L. Atzori, A. Iera, and G. Morabito, “Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm,” *Ad Hoc Networks*, vol. 56, pp. 122–140, 2017.

[8] E. Borgia, “The internet of things vision: Key features, applications and open issues,” *Comput. Commun.*, vol. 54, pp. 1–31, 2014.

[9] K. Ashton and others, “That ‘internet of things’ thing,” *RFID J.*, vol. 22, no. 7, pp. 97–114, 2009.

[10] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, “Future internet: The internet of things architecture, possible applications and key challenges,” *Proc. - 10th Int. Conf. Front. Inf. Technol. FIT 2012*, pp. 257–260, 2012.

[11] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, “Interacting with the SOA-based internet of things: Discovery, query, selection, and on-demand provisioning of web services,” *IEEE Trans. Serv. Comput.*, vol. 3, no. 3, pp. 223–235, 2010.

[12] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” *Comput. networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[13] C. Roser, “A critical look on Industry 4.0 | AllAboutLean.com,” 2015. [Online]. Available: https://www.allaboutlean.com/industry-4-0/. [Accessed: 18-Sep-2018].

[14] M. Hermann, T. Pentek, and B. Otto, “Design principles for industrie 4.0 scenarios,” *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, vol. 2016–March, pp. 3928–

3937, 2016.

[15] H. Kagermann, J. Helbig, A. Hellinger, and W. Wahlster, *Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry; final report of the Industrie 4.0 Working Group*. Forschungsunion, 2013.

[16] "What is Cloud Computing? - Amazon Web Services." [Online]. Available: https://aws.amazon.com/what-is-cloud-computing/. [Accessed: 01-Mar-2018].

[17] "What is cloud computing? A beginner's guide | Microsoft Azure." [Online]. Available: https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/. [Accessed: 02-Mar-2018].

[18] "IaaS vs PaaS vs SaaS: A Clear Explanation of Cloud Services." [Online]. Available: https://rubygarage.org/blog/iaas-vs-paas-vs-saas. [Accessed: 02-Mar-2018].

[19] P. Helo, M. Suorsa, Y. Hao, and P. Anussornnitisarn, "Toward a cloud-based manufacturing execution system for distributed manufacturing," *Comput. Ind.*, vol. 65, no. 4, pp. 646–656, 2014.

[20] "How AWS IoT Works - AWS IoT." [Online]. Available: https://docs.aws.amazon.com/iot/latest/developerguide/aws-iot-how-it-works.html. [Accessed: 05-Apr-2018].

[21] "About Watson IoT Platform." [Online]. Available: https://console.bluemix.net/docs/services/IoT/iotplatform_overview.html#about_iotplatform. [Accessed: 24-May-2018].

[22] "Microsoft Azure IoT Refrence Architecture." [Online]. Available: https://aka.ms/iotrefarchitecture.

[23] D. Canty, "Case Study: IoT Technology Platform – ThingWorx." [Online]. Available: https://deniscanty.com/2015/04/21/case-study-iot-technology-platform-thingworx-10/. [Accessed: 25-May-2018].

[24] "IoT-Ticket, the Office Suite for Internet of things." [Online]. Available: https://iot-ticket.com/. [Accessed: 08-Jan-2018].

[25] "VersaSense MicroPnP Development/Starter kit - Wireless - eewiki." [Online]. Available: https://eewiki.net/pages/viewpage.action?pageId=61047016#VersaSenseMicroPnPDevelopment/Starterkit-VersaSenseIoTManager. [Accessed: 25-May-2018].

[26] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[27] Z. Shelby, K. Hartke, and C. Bormann, “The Constrained Application Protocol (CoAP),” no. 7252. RFC Editor, Jun-2014.

[28] “OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0,” 2012. [Online]. Available: http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf. [Accessed: 25-May-2018].

[29] C. Lohman, L. Fortuin, and M. Wouters, “Designing a performance measurement system: A case study,” *Eur. J. Oper. Res.*, vol. 156, no. 2, pp. 267–286, 2004.

[30] V. Veleva and M. Ellenbecker, “Indicators of sustainable production: Framework and methodology,” *J. Clean. Prod.*, vol. 9, no. 6, pp. 519–549, 2001.

[31] W. Eckerson, “Ten Characteristics of a Good KPI,” *Data Warehous. Inst. (TDWI). Retrievd June*, vol. 14, p. 2009, 2006.

[32] L. Lake, “What Are Key Performance Indicators?,” 2017. [Online]. Available: https://www.thebalancesmb.com/what-are-key-performance-indicators-2296142. [Accessed: 20-Apr-2018].

[33] H. Tokola, C. Gröger, E. Järvenpää, and E. Niemi, “Designing Manufacturing Dashboards on the Basis of a Key Performance Indicator Survey,” *Procedia CIRP*, vol. 57, pp. 619–624, 2016.

[34] D. Kibira, M. P. Brundage, S. Feng, and K. C. Morris, “Procedure for selecting key performance indicators for sustainable manufacturing,” *J. Manuf. Sci. Eng.*, vol. 140, no. 1, p. 11005, 2018.

[35] E. Amrina and S. M. M. Yusof, “Key performance indicators for sustainable manufacturing evaluation in automotive companies,” *Ind. Eng. Eng. Manag. (IEEM), 2011 IEEE Int. Conf.*, pp. 1093–1097, 2011.

[36] Y. Fukuda, “Standardization of Key Performance Indicator for Manufacturing Execution System,” *SICE Annu. Conf. 2010, Proc.*, pp. 263–265, 2010.

[37] B. Fry, *Visualizing data: Exploring and explaining data with the processing environment*. “ O’Reilly Media, Inc.,” 2007.

[38] N. A. N. Lee, L. E. G. Moctezuma, and J. L. M. Lastra, “Visualization of information in a service-oriented production control system,” *IECON Proc. (Industrial Electron. Conf.*, pp. 4422–4428, 2013.

[39] W. W. Eckerson, “Performance Dashboards : Measuring , Monitoring , and Managing Your Business,” *John Wiley Sons*, 2010.

[40] “Dashing-The exceptionally handsome dashboard framework.” [Online]. Available: http://dashing.io/.

[41] “PLANTCockpit Open Source.” [Online]. Available: http://www.tut.fi/plantcockpit-os/. [Accessed: 05-Jan-2018].

[42] B. Ramis Ferrer, S. Iarovyi, L. Gonzalez, A. Lobov, and J. L. Martinez Lastra, "Management of distributed knowledge encapsulated in embedded devices," *Int. J. Prod. Res.*, vol. 54, no. 18, pp. 5434–5451, 2016.

[43] D. Hastbacka and A. Zoitl, "Towards semantic self-description of industrial devices and control system interfaces," *Proc. IEEE Int. Conf. Ind. Technol.*, vol. 2016–May, pp. 879–884, 2016.

[44] D. Schachinger, W. Kastner, and S. Gaida, "Ontology-based abstraction layer for smart grid interaction in building energy management systems," *2016 IEEE Int. Energy Conf. ENERGYCON 2016*, 2016.

[45] J. Puttonen, A. Lobov, M. A. C. Soto, and J. L. M. Lastra, "A Semantic Web Services-based approach for production systems control," *Adv. Eng. Informatics*, vol. 24, no. 3, pp. 285–299, 2010.

[46] B. Ramis *et al.*, "Knowledge-based web service integration for industrial automation," *IEEE Int. Conf. Ind. Informatics*, pp. 733–739, 2014.

[47] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowl. Acquis.*, vol. 5, no. 2, pp. 199–220, 1993.

[48] W. N. Borst and W. N. Borst, "Construction of Engineering Ontologies for Knowledge Sharing and Reuse." Centre for Telematics and Information Technology (CTIT), Netherlands, 1997.

[49] N. Guarino, D. Oberle, and S. Staab, "What is an ontology?," in *Handbook on ontologies*, Springer, 2009, pp. 1–17.

[50] N. F. Noy, D. L. McGuinness, and others, "Ontology development 101: A guide to creating your first ontology." Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880, Stanford, CA, 2001.

[51] X. Su and L. Ilebrekke, "A comparative study of ontology languages and tools," *Adv. Inf. Syst. Eng. 14th Int. Conf. CAiSE 2002*, pp. 761–765, 2002.

[52] "RDF - Semantic Web Standards." [Online]. Available: https://www.w3.org/RDF/. [Accessed: 11-Apr-2018].

[53] "OWL Web Ontology Language Reference." [Online]. Available: https://www.w3.org/TR/owl-ref/. [Accessed: 08-Apr-2018].

[54] "SPARQL Query Language for RDF," 2008. [Online]. Available: https://www.w3.org/TR/rdf-sparql-query/. [Accessed: 08-Apr-2018].

[55] M. J. A. G. Izaguirre, A. Lobov, and J. L. M. Lastra, "OPC-UA and DPWS interoperability for factory floor monitoring using complex event processing," *IEEE Int. Conf. Ind. Informatics*, pp. 205–210, 2011.

[56] A. Lobov, J. Puttonen, V. V. Herrera, R. Andiappan, and J. L. M. Lastra, “Service oriented architecture in developing of Loosely-coupled manufacturing systems,” *IEEE Int. Conf. Ind. Informatics*, pp. 791–796, 2008.

[57] S. Iarovyi, W. M. Mohammed, A. Lobov, B. R. Ferrer, and J. L. M. Lastra, “Cyber-Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems,” *Proc. IEEE*, vol. 104, no. 5, pp. 1142–1154, 2016.

[58] OPC Foundation, “OPC Unified Architecture,” pp. 1–44.

[59] “Devices Profile for Web Services Version 1.1,” 2009. [Online]. Available: http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.pdf. [Accessed: 25-May-2018].

[60] M. Mahmoodpour, A. Lobov, M. Lanz, P. Mäkelä, and N. Rundas, “Role-based visualization of industrial IoT-based systems,” *14th IEEE/ASME Int. Conf. Mechatron. Embed. Syst. Appl.*, 2018.

[61] M. Mahmoodpour, A. Lobov, and M. Lanz, “Configurator module to integrate different protocols for IoT solution,” *IEEE 16TH Int. Conf. Ind. INFORMATICS*, 2018.

[62] “Node.js.” [Online]. Available: https://nodejs.org/en/. [Accessed: 07-May-2018].

[63] “NPM.” [Online]. Available: https://www.npmjs.com/. [Accessed: 07-May-2018].

[64] “Apache Jena - Apache Jena Fuseki.” [Online]. Available: https://jena.apache.org/documentation/fuseki2/. [Accessed: 08-May-2018].

[65] “IOT API USER MANUAL.” [Online]. Available: https://www.iot-ticket.com/images/Files/IoT-Ticket.com_IoT_API.pdf. [Accessed: 11-May-2018].

[66] P. Mäkelä and N. Ristimäki, “SEINÄJOEN AMMATTIKORKEAKOULUN TEOLLISEN INTERNETIN LABORATORIO.” [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/125235/Teollisen internetin opetus SeAMKissa.pdf?sequence=1. [Accessed: 07-Aug-2018].