TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

MUHAMMAD USMAN

AN IMPLEMENTATION OF KPI-ML TO A MULTI-ROBOT LINE SIMULATOR

Master of Science thesis

Examiner: Prof. Jose L. Martinez Lastra
Examiner and topic approved by the Council meeting of the Faculty of Engineering Sciences on the 29th March 2017

ABSTRACT

MUHAMMAD USMAN: AN IMPLEMENTATION OF KPI-ML TO A MULTI-ROBOT LINE SIMULATOR
TAMPERE UNIVERSITY OF TECHNOLOGY
Master of Science Thesis, 66 pages, 5 Appendix pages
December 2017
Master's Degree Programme in Automation Engineering
Major: Factory Automation and Industrial Informatics
Examiner: Professor Jose L. Martinez Lastra
Supervisor: Borja Ramis Ferrer and Wael M. Mohammed

Keywords: key performance indicators markup language, ISO 22400 standards, knowledge-based systems, service oriented architecture, web services, manufacturing systems.

Emergence of highly competitive markets have led to more deep and thorough evaluation of performances across the manufacturing industry to enhance the efficiency of production processes. Manufacturing industry across the globe have been using different performance indicators and measuring terminologies for performance evaluation. This diversity deters evaluating and comparing manufacturing industries performance on a global scale and thus limiting industry collaboration.

To define key performance indicators and general terminologies that are applicable in manufacturing operations management level of manufacturing industries, the International Standards Organization (ISO) developed the ISO 22400 standard. The Manufacturing Enterprise Solutions Association (MESA) international, an international association for manufacturing solutions, takes forward the work done of defining Key Performance Indicators (KPIs) by developing a Markup Language (ML) that represents the data models for the KPIs defined in ISO 22400 standards in an Extensible Markup Language (XML) schemas format. This language is formally known as KPIML.

This thesis implements several the key performance defined in ISO 22400 standards to monitor the performance and efficiency of a real-world production line. In addition, this research work demonstrates the visualization of the implemented performance indicators in the form of different graphs. This visualization aids the management to analyze and evaluate the performance of production line in run time. A knowledge-based system is designed on data models present in KPIML for the implemented KPIs, which can easily be extended. Moreover, keeping in view the varying nature of manufacturing industry, the implementation of this research work allows users to model their own KPIs, which are specifically applicable to their use case and able to visualize them in run time.

PREFACE

‘In The Name of Allah, The Most Gracious and The Most Merciful’

I will start by thanking my family for their continuous support and motivation at every step of my studies. I am thankful to my supervisors Borja Ramis and Wael Mohammed from FAST-Lab for their valuable guidance. I would specially like to thank Professor Jose Lastra and Dr.Andrei Labov for giving me an opportunity to work under their supervision in such a multi-cultural environment.

I would like to mention some of my friends who supported me a lot during the course of my degree. I would start by expressing my gratitude to Umer, Adnan, Abdul manan, Shahbaz and Farooq for their incredible support through their expertise during my research. I am especially grateful to my flat mates Zeeshan, Shadman, Ihtisham and Ahsan for their help and encouragement during the writing phase of my thesis.

Last but not the least I am very thankful to my fiancé for her continuous motivation and standing beside me during my tough times and encouraging me whenever I felt disheartened.

Thanks to everyone who helped me in achieving this milestone.

Muhammad Usman

18th December 2017

Tampere, Finland

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

APT	Actual Production Time
AUBT	Actual Unit Busy Time
BEEP	Blocks Extensible Exchange Protocol
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheet
DCOM	Distributed Component Object Model Object
DCS	Distributed Control System
DOM	Document Object Model
ERP	Enterprise Resource Planning
eScop	Embedded systems Service-based Control for Open manufacturing and Process automation
FAST	Factory Automation System and Technology
FASTory	Factory Automation System and Technology Laboratory
FTP	File Transfer Protocol
HATEOAS	Hypermedia as the Engine of Application State
HMI	Human-Machine Interface
HTML	HyperText Mark-up Language
HTTP	HyperText Transfer Protocol
ID	Identification
ISA	International Society of Automation
ISO	International Standards Organization
JSON	JavaScript Object Notation
KB	Knowledge Base
KBS	Knowledge Based System
KPI	Key Performance Indicators
KPI-ML	Key Performance Indicators Markup Language
KR	Knowledge Representation
LIMS	Laboratory Information Management System
MOM	Manufacturing Operations Management
MES	Manufacturing Execution System
MESA	Manufacturing Enterprise Solutions Association
ORB	Object Request Broker
OWL	Web Ontology Language
RDF	Resource Description Framework
REST	Representational State Transfer
RFID	Radio Frequency Identification Device
SCADA	Supervisory Control and Data Acquisition
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol

SPARQL	Simple Protocol and RDF Query Language
SVG	Scalable Vector Graphics
TCP	Transmission Control Protocol
TUT	Tampere University of Technology
URI	Uniform Resource Identifier
UML	Unified Modelling Language
URL	Uniform Resource Locator
WCS	Warehouse Control System
WS	Web Services
WSDL	Web Service Definition Language
XML	Extensible Markup Language
XSD	XML Schema Definition

1. INTRODUCTION

This chapter highlights the core purpose and objective of this thesis. The scope of the thesis along with hypothesis is also briefly described. In addition, it explains the basic need or problem that is to be solved in this thesis. This chapter also enlists certain challenges are faced during the implementation. Moreover, this chapter presents the limitation that are assumed during the course of this thesis. Finally, a thesis outline is described in the last section of this chapter.

1.1 Thesis Scope

This thesis is carried out at the Factory Automation Systems and Technology (FAST) laboratory, which belongs to the Automation and Hydraulics Engineering laboratory at Tampere University of Technology. This thesis serves as a paving step towards the work done by MESA international in implementation of the standard ISO 22400-Automation systems and integration, KPIs for manufacturing operations management. This research work implements its solution for the use case of assembly line present at FAST laboratory.

1.2 Problem Definition

Performance measurement and assessment has constantly been a critical factor for the management to assess the performances at various levels and departments of an organization. Previously, almost every industry has individually researched, hired people and used different performance evaluating tools for monitoring the performance according to their own parameters. However, as the competition between industries become more intense and the phenomenon of globalization has evolved, the search for those performance indicators have started that can be critical in the success of an industry in a competitive marketplace [46]. The immense interest from organizations in finding the KPIs got the attention of international organizations such as International Standards Organization (ISO), International Society of Automation (ISA) and MESA, which shifted their focus towards making a standard performance measurement system. Although there has been different set of standards and performance indicators on different levels of manufacturing system, this research work will emphasis only on KPIs at manufacturing operation management level and production level.

The major concern of this thesis work is to find a strategy for evaluating and assessing the performance of production line on the basis KPIs defined in ISO 22400 standards. A

multi-robot line simulator has been used as a testbed. Previously no standardized method was available to evaluate and asses the performance of the testbed in run time. This research work provides a standardized solution with help of ISO 22400 standard and KPIML.

1.3 Hypothesis

In order to identify and define the right set of KPIs for manufacturing industry, ISO has developed the standard under the banner "ISO 22400-Automation systems and integration, Key performance indicators (KPIs) for manufacturing operations management". The ISO 22400 standard define 34 KPIs along with its description, including formula, audience, scope, range, etc. which are applicable at manufacturing operation level of automation pyramid in any organization. The implementation of these KPIs will allow the supervisors to monitor and evaluate the performance of manufacturing system in runtime for different set of production orders. This monitoring will lead the management to take critical decisions related to production operations and will help them in planning different production activities. Furthermore, adding visual graphs and charts will help in better visualization of these KPIs.

1.4 Objectives

This thesis work aims two major objectives, which are to i) implement and ii) visualize the selected set of KPIs from the ISO 22400 standard that are relevant and applicable, to monitor the production process in the manufacturing systems. In this research work, the set objectives were achieved on the multi-robot line simulator test bed.

The first objective is to implement the selected set of KPIs, which means to compute the values for these selected KPIs from the manufacturing system in run time. The first objective can be achieved by computing the values of the KPIs with the help of the formulas given in the ISO 22400-2 standard [25]. The data acquired from the manufacturing system is stored and managed with help of a Knowledge Based System (KBS). Moreover, a subtask in the first objective is to give an additional feature to the user for making its own KPI and gets its visualization. The subtask is that any user should have an option to create its own KPI as per its own requirements and system functionality. The user will have an option to enter all the details of its own KPI such as name of the KPI, formula, audience, range and description etc. and will be able to get the measurements and visualization for its KPI in the desired format.

Each performance indicator measurement is viewed and analyzed at different levels of organization with different visual graphs in different periods of time. Some KPIs are required to be analyzed daily, some weekly and some monthly. Therefore, the second objective of this thesis is to visualize those selected set of implemented KPIs in a form of scatter plots, pie charts, line charts or histograms in run time. Visualization of these

KPIs will help the technical as well non-technical supervisors such as people in upper hierarchy of the organization to use the simulator efficiently. With visualizations, the technical staff will be able to assess the future state of the production line and will be able to take the necessary preventive steps. Whereas, it will enable managers to evaluate the performance of the manufacturing system as whole in the sense of production capacity, personnel performance and quality and will be able to position them self in a best way in marketplace.

1.5 Challenges, Limitations and Assumptions

There were certain challenges encountered during the course of this research work, which are the following.

- The first challenge was to get continuous data from the testbed in order to test the validity of the solution. To counter this challenge an orchestrator was designed to process production orders, which generated continuous events for the testing the solution.
- Secondly, designing a dynamic user interface for visualizing the KPIs was a challenge, however that was solved by selecting that technology for front end development, which ensures two-way data binding.
- Thirdly, creating a new customized KPI by the user in run time requires complex computation.
- Fourthly, Data acquisition and data analysis requires a thorough and tedious examination in solving research problems. The implementation of KPIs requires the computation of different variables from a large set of raw data that is retrieved from the production line, which creates the challenge of managing this large data. Moreover, designing an efficient knowledge based system to counter the aforementioned difficulty of managing large data is a challenging task.

Although this research work fulfills the desired objectives, there are certain limitations associated with the implemented solution, which are the following:

- The implemented solution is not a standalone tool that can be used for every other production line by just ‘plug and play’. However, it can be adapted to other production lines by slight modifications specific to that line.
- The user interface of the solution is embedded with the user interface of the testbed FASTory Simulator.
- The creation of user defined customized KPIs is also bounded by certain limitations. For example, the formula can only have the variables that already have the data available in our model, the formula can have only two variables.

1.6 Thesis Outline

Thesis is structure in the following way. Chapter 2 describes the theoretical background and tools required to understand the concept. Chapter 3 presents the approach that is adopted in this thesis work to reach the desired results. Chapter 4 explains in detail the implementation of this research work. Chapter 5 illustrates the results of implemented KPIs in form of visual graphs and discuss them. Chapter 6 concludes the work done and presents the future prospect in this research field.

2. STATE OF THE ART

This chapter presents the research background. The chapter starts with the broader topic of manufacturing systems and the international standards ISA 95 and IS0-22400. Moreover, it also highlights the Service Oriented Architecture (SOA) in manufacturing systems. Furthermore, web services and REST (Representational State Transfer) are also explained in this chapter. Afterwards, this chapter briefly introduces the programming tools used in the implementation process. Finally, this chapter explains the knowledge based systems and the FASTory simulator, which is the testbed used in this thesis.

2.1 Manufacturing Systems

Manufacturing industry has flourished in the recent years with the use of automation, computer systems and software [48]. Automated production lines and high-tech software have helped manufacturing systems to progress and become more profitable. In [1], Manufacturing systems are defined as a collection of different equipment such as machines, computers, people, transportation items and other elements that are utilized together for manufacturing. This equipment transforms inputs such as raw material and energy into desired products with help of manufacturing processes.

The most generic aim or responsibility of a manufacturing system is to produce a product by utilizing the available resources. The primary objectives of any manufacturing system are to decrease the time to market, enhance the quality for its customer at lower costs.

In a manufacturing system, making a product is often related to several dynamics of the market, which may include the demand of the product over a specific period of the time. Moreover, when to produce a specific product, how fast to produce it, and the variance of the product from other products in the market are other dynamics of making a product. Manufacturing systems can be custom made designed for a specific product or it can be assembly manufacturing system or may be flexible one which can easily be modified for customized products. Other manufacturing systems include reconfigurable, just-in-time and lean manufacturing systems [2].

In [2], the author presents the diagram presented in below Figure 1 to explain the aforementioned definition of manufacturing systems. Manufacturing systems have the inputs, that are fed in to the system, such as raw material, energy and demand. Then, the disturbance or transformation process the manufacturing system executes on the inputs

and add an additional value to it. Finally, the output can be produced in the shape of different services and products for the end user.

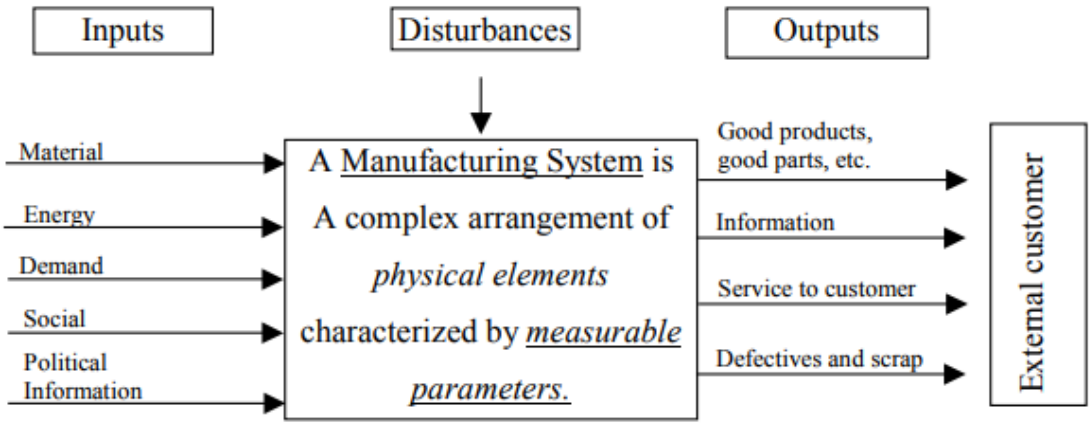

Figure 1: *Definition of Manufacturing System [2]*

In some cases, manufacturing systems are also defined as the combination of manufacturing facilities and manufacturing support systems [4]. As manufacturing systems is the collection of resources such as equipment, people and set of plans to manage the manufacturing process and operation. The equipment used in the production line of manufacturing system may include conveyors, robots, sensors and buffers. Cells, work centers, or work stations, which are themselves systems are considered as the subset of manufacturing systems.

2.2 ISA 95 Standard

At the end of 20th century as industries were progressing, the need to integrate business with production system had also increased [47]. There was a clear gap between the shop floor, and the processes related to business in a Manufacturing Execution Systems (MES). This gap lead companies to define different MES functionalities according to their own specific needs [49]. The need of common terminologies and technical language for organizations on which they can operate and communicate in another end of the world was also one of the challenge [7]. Therefore, to fill in the aforementioned needs of integrating business processes with production systems and defining different MES function, the international standard ISA-95 has been introduced. ISA-95 is an American National Standards Institute (ANSI) standard developed by an ISA Committee of experts. The ISA-95 provides a common ground for integration between the enterprise level and control system level.

The major working principles and operation performed in a manufacturing organization usually follows the same common principles. The functional hierarchal architecture of the manufacturing industry has been defined in the standard ISA 95 and is shown in

Figure 2. The functional hierarchy gives a very basic architectural model of any manufacturing system.

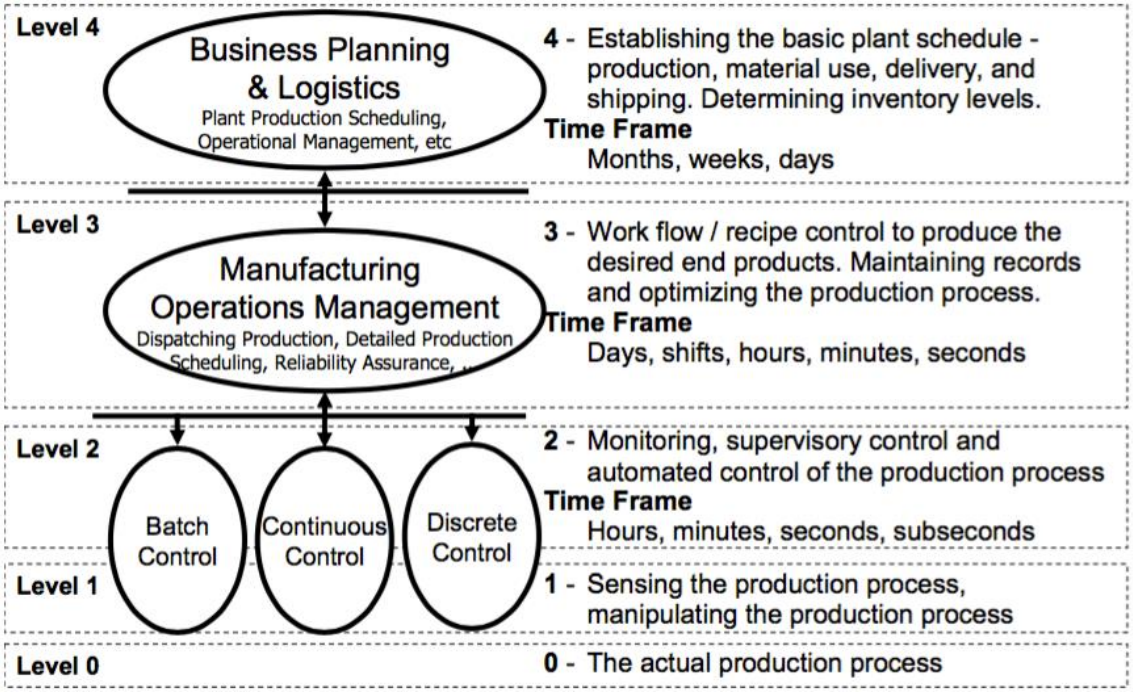

Figure 2: *Functional Hierarchy of Automation systems per ISA 95 [8]*

It is one of the very first model that laid the foundation for automation system in process industries. This model defines the functionality and information flow between different levels of MES for integration.

The levels presented in this model clearly define the main activities in a MES such as production planning, maintenance activities, quality, and inventory control [50].. The top levels in the functional hierarchy of Figure 2 had broader domain in terms of spatial scale and time scale then the lower levels such as level 0 or 1 [10].

The lower levels of Figure 2, Level 0, 1, and 2 represent the layer at shop floor level such as sensors and process control layer. The process control layer can mainly be Distributed Control System (DCS) and Supervisory Control and Data Acquisition (SCADA) [8]. The lowest three levels are generally hardware oriented, usually there are sensing elements, electronic circuits or microprocessors. A lot of embedded technology is also used at this level, as software elements operate in very proximity to the hardware base at this level.

Manufacturing operations management (MOM) level work as an integrating layer between ERP and Control system. It works as a kind of medium for communication and flow of information between ERP and control system layer. There are several other level 3 applications examples which are very common in industry such as Manufacturing

execution system (MES), Laboratory information management systems (LIMS), asset management system, batch management and Warehouse Control Systems (WCS) [12].

The top layer of the pyramid i-e Level 4: Business planning and logistics, outlines business-related activities that occur in a manufacturing organization. Level 4 includes activities such as plant scheduling, keeping check on inventory levels of different materials in use, keeping track of the logistic services for enabling in time delivery of all the raw material for the production. The operations in level 4 mostly ranges form days to months.

ISA-95 were developed with the objectives that it will provide a common terminology for with in the organizations as well for the communication between manufacturers and suppliers. It gives clear norms and standards for how information should be interchanged between different levels and how information can be processed. ISA-95 is introduced for worldwide manufacturing environments.

ISA-95 provides a standardize way of describing the flow and exchange of information between MOM and ERP systems. There are 5 parts of the ISA-95 standard. The Part 1, Models and Terminology of ISA 95 standards consist of all the common terminology and object models, which are used in organization or manufacturing systems from top management level to shop floor for exchange of information. Part 2: Object Attributes consists of attributes for every object that is defined in part 1. It further defines formal object models for exchange information described in Part 1 with help of Unified Modelling Language(UML) object models, tables of attributes, and examples. Part 1 and Part 2 together gives the direction of how to exchange information between different automation systems or levels. Part 3 focuses on the functions and activities at level 3: Manufacturing operations management/ Manufacturing Execution System (MES layer) in the Figure 2. Part 3 further divides layer 3 into production, maintenance, quality and inventory, thus helping its users in identifying and comparing production activities and control systems in a standardize way by applying a standard language. Part 4 of the standards emphases on level 3 of the automation pyramid by defining detailed models of the information flows among the activities called out in Part 3. Part 5 of the standards is about business to manufacturing transaction. It defines transactions activities carried out at level 4 of automation pyramid in perspective of exchanging information among applications executing business and manufacturing activities related to levels 3 and 4 of automation pyramid, here the standard emphasizes on how to best standardize manufacturing with organization productivity and profitability.

2.3 Service Oriented Architecture (SOA)

2.3.1 Service Oriented Architecture in Manufacturing Systems

Over the last few decades, Service Oriented Architecture (SOA) has been one of the focus area for researchers in manufacturing systems as well as in industrial automation [57]. SOA provides the complete functionality of applications by encapsulating different services to work together in standard structure [17]. SOA is a paradigm in which manufacturing systems can work to achieve the desired results, however, it is often misinterpreted as a technology. SOA can be defined in many ways. For example, in [13] authors described it as a loosely coupled architectural style that is designed to fulfil the modern-day business needs of the organizations. Whereas, in [14] it is defined as an architectural style that is used for building autonomous and interoperable systems.

Interoperability is one of the major requirement for SOA to perform efficiently [13]. Moreover, it helps different systems to exchange information with ease, removing any interface obstacles when a service is exposed to its environment. SOA helps system to be autonomous yet interoperable at the same time [13]. The functionality of services provided by SOA based systems is at disposal of the boundary without any obstacles because of its loosely coupled characteristics. Moreover, service along with their schemas describing the functionality and standards for the service are independent of the platform. Service implementation can be altered without affecting the users for that service, as the implementation of the service is totally opaque. Furthermore, SOA communications are asynchronous meaning that when a system is asked for a service it respond to the requester without halting any other operations on the respondent [13].

The following Figure 3 shows the basic SOA pattern for any domain implementing it. In this SOA pattern, there are three main building blocks, service requester, service broker and service provider. The service requester, request for its desired service on the web. On the other hand, a service broker helps in finding the needed service for the service requester. Finally, a service provider is the one that owns the required service. If the number of service providers is more than one, then one is selected among them [15].

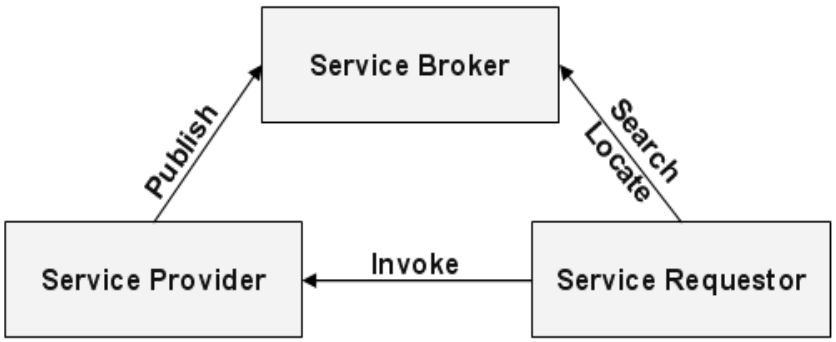

Figure 3: *Pattern describing building blocks of SOA [15]*

In [16], SOA is used for data acquisition and analytics from devices by deploying web services (WS) at device level. The publish/subscribe mechanism will be used to gather data from the devices and analyzed at respective application. The data acquired in this case will mostly be of KPIs, which are performance evaluators of the systems. The authors in [16], implemented the aforementioned approach on a testbed, which is presented in Figure 4. These KPIs are calculated by formulas specified for them depending on several variables such as quality values, production time, and availability of the system [16].

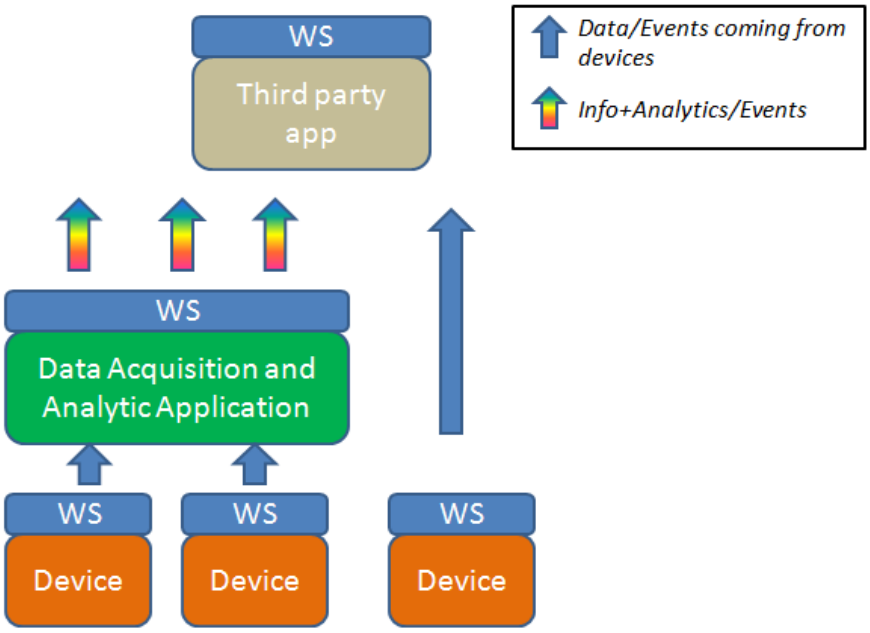

Figure 4: *Service-oriented production system used for data acquisition and analytics [16]*

WS is the realization of SOA framework, SOA implementation are carried out using WS technology. Previously, Distributed Component Object Model (DCOM), and Object Request Broker (ORB) based on Common Object Request Broker Architecture (CORBA) specification were used for implementing SOA [56].

WS are software systems that ensures interoperability in a machine to machine communication. Initially, WS were implemented with help of Simple Object Access Protocol (SOAP) and Web Service Definition Language (WSDL) [56]. SOAP is an XML based protocol that is used to connect application. Whereas, the interface for these SOAP web services is written in WSDL, which is a machine-readable format. SOAP messages can be transmitted over any protocol such as HyperText Transfer Protocol (HTTP), File Transfer Protocol (FTP), Transmission control Protocol (TCP) and Blocks Extensible Exchange Protocol (BEEP), but the most commonly used among them is the HTTP. However recently, another approach using REST has been used for implementation of WS [56]. REST is gaining popularity because of its faster learning curve, as every operation in REST is an HTTP request and the response can be a plain JavaScript Object Notation (JSON), XML or any other available format. A detail description of REST is presented in the next subsection.

SOA fulfills the requirements of automation systems in meeting their business and technical requirements. WS gave SOA the foundation for its acceptability across different domains and helped in overcoming the challenges that were associated with the development of SOA based applications [56]. The incorporation of SOA with in the architectural structure of industry is also providing a uniform technology for industries to collaborate [22].

2.3.2 Representational State Transfer (REST)

REST architecture style was first introduced by Roy Fielding, in his doctoral dissertation [19]. Roy presented some principles and guidelines named as constraints. These constraints or principles describe the architecture of systems and interactions that make up the Web. There are six constraints, which are discussed below.

The first constraint is about client-server architectural style. Separating the concerns related to client-server is the basic idea in this constraint. It will give client and server a sense of independence and will help in improving scalability. The next constraint impose condition on client-server interaction. Client must ensure that request must contain all the necessary information to interpret the request. During request, the communication between client and server must be stateless. Third constraints enhance the efficiency of the network. The cache constraint suggests that data in response to client request can be either cacheable or non-cacheable. The cacheable response data can be used for later equivalent request by client cache.

The layered system constraint permits the architecture to consist of hierarchal layers which can only interact with the layer next to it and not able to see beyond that, thus decreasing the overall complexity of the architecture and making each layer independent of others. The fifth constraint is code on demand, which is an optional constraint that allows the functionality to update for the client side independently of the server side. This helps the client in a way that many pre-implementation features are already executed.

The most important characteristic of REST style architecture is its uniform interface between different components. REST defines a set of guidelines and principles for transmitting data over a standardized interface. There are certain guiding principles for uniform interface, which are the following:

1. That Uniform Resource Identifiers (URIs) are used as resource identifiers and every resource or entity should be identifiable by a single URI. Instead of using XML to make a request, REST depend on a simple Uniform Resource Locator (URL) in many cases [21].
2. Manipulation of resources through representations is the other guiding principle for uniform interface [21]. The identified resource can be represented in XML, SVG

(Scalable Vector Graphics) or JSON. The user can ask for the output response in its preferred format in the request URI as per the need of application and ease of usability.

3. The request or messages send from client to server are mostly self-descriptive. The transfer of request is over HTTP only, the commonly used HTTP verbs for performing different task on server are GET, POST, PUT, and DELETE. GET is used for retrieving some data about the resource from the server or getting the state of the resource. POST is used to make a new sub resource on the server. PUT is used to edit or update an existing resource. And DELETE is used to remove an existing resource. Besides these four methods other commonly used HTTP methods are OPTIONs and HEAD. Moreover, if we look in deep to these methods GET, PUT and DELETE requests can be made as many times as one want without effecting the server thus they are called idempotent and POST method if get repeated will create a new entry every time on the server, is therefore non-idempotent.
4. The fourth principle is that of HATEOAS (Hypermedia as the engine of application state) which means that any exchange or state of interaction between client-server is through hyperlinks or hypermedia i.e., links, or URIs. This principle enforces web services to return the necessary links in the returned body (or headers) of the object itself or related objects [20][21].

REST web services are now majorly in use due to its small learning curve as well as ease of usability. The performance of REST web services is much better in comparison to SOAP, REST is lightweight and much faster in operation than SOAP. REST doesn't imply any strict rules or standards, which is the case in SOAP services, and that is one of the major reasons developer tends to use REST services more. REST satisfies the needs of e-commerce applications with its ease of usability but when it comes to sophisticated Business to business interactions that involve multi-step business interaction REST is not that effective as in the earlier case [23].

2.4 Key Performance Indicators

Several Performance measuring factors are used in industry to evaluate the progress and growth of all types of processes involved in an industry. Strictly speaking, in the context of production industry, these processes includes production scheduling and planning, inventory management and quality management. These performing measuring indicators are identified on the basis of their relevance and importance to the overall performance of the industry starting from a single work unit and worker to the entire production line.

The 'Performance indicators' term is now replaced by the new term KPIs in the industrial domain because of management's sole interest in the critical and key factors in the manufacturing system. KPIs drive the industry towards success and play a more vital role in improving a company's efficiency and profitability. In this era of competitive-

ness manufacturing industries are very keen to know the key factors in their manufacturing line to invest in them to excel from their competitors and have greater market share. To have a more better chance of self-assessment and improvement, identifying and evaluating the critical KPIs for a manufacturing system is the key. However, identifying a general set of KPIs that can fully depict and monitor the performance of any industrial environment and can be applicable to any general production line is a tiresome task. In [35], one such effort is made to propose a methodology to identify which can help in identifying a basic set of KPIs for the production line. An 8-step iterative close-loop model is suggested for introducing and monitoring different KPIs on a production line in relation to the set goals for the production. Figure 5 depicts that 8-step close-loop model for KPIs identification.

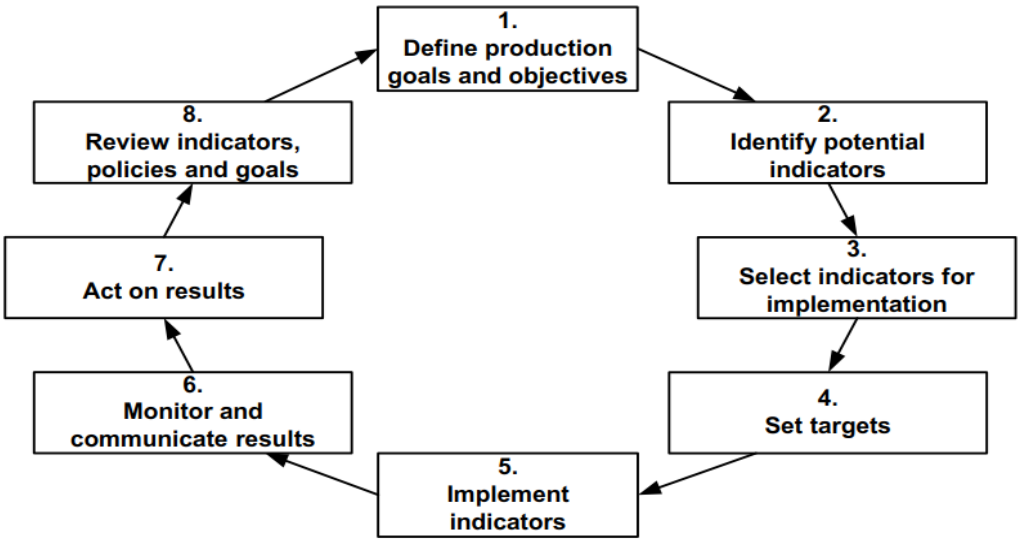

Figure 5: *8-step iterative close-loop model for KPIs identification [35]*

The close-loop model starts by setting the goals and objectives of the production line, then the identification of potential indicators that can contribute to the performance of the line are identified and selected for implementation. Setting targets for each indicator and implementing indicators on the production line are next steps in the close-loop process. Moreover, these implemented KPIs are monitored and the results achieved from the monitoring of these KPIs is continuously communicated to the responsible authorities within the industry so that steps for continuous improvement can be taken. The seventh step of acting on the achieved results is the most critical one, corrective measures are taken on the basis of the KPIs for improving the productivity of the line. Finally, in the last step a complete review of the whole process from identifying the indicators and to acting on the results is done that can lead to identification of new KPIs or elimination of some previous KPIs. In [35], the authors define a set of four key properties for each KPI that must be defined when implementing a KPI. These properties include Unit of Measurement, Type of measurement, Period of measurement and lastly the Boundaries.

Moreover, in [35], a three-level framework is designed that sort the performance indicators on the basis of their importance in the industrial environment. The first level includes KPIs related to safety and environment that can comply with international standards and rules for the manufacturing industry. The second level has all the KPIs that are related to production planning, scheduling, quality and inventory operations. The last level has KPIs related to the workers working in the industrial setup and the issues related to them. Figure 6 illustrates the above discussed three levels of the KPIs framework in a hierarchical way.

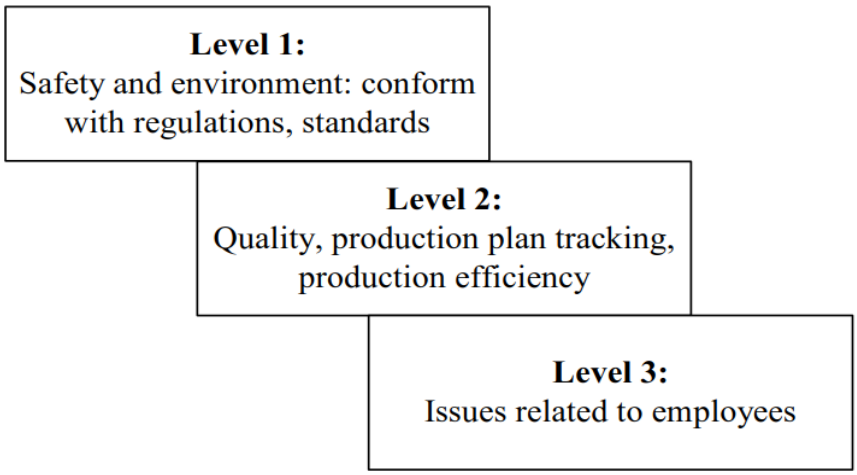

Figure 6: *KPIs Framework [35]*

The five KPIs that were identified by applying the proposed methodology in [35] are quality, safety and environment, issues related to employees, production plan and schedule tracking and production efficiency. These KPIs are given in Table 1 along with the indicators that comes in handy in the manipulation process of these KPIs.

However, despite the proposed methodology and framework in [35], it only defines five KPIs, which are alone not enough to completely monitor the performance of a production line. Moreover, there is a need of providing an architecture for implementation of these KPIs on a real production line along with the means of designing a database model as well as visualizing these KPIs.

Table 1: *KPIs along with indicators from the industrial environment [36]*

KPIs	Indicators
Safety and Environment	Number of accidents at work
	Number of hazardous alarms
	Fresh water consumption
	Waste generated before recycling
	Number of penalties due to releasing waste in environment
Production Efficiency	Efficiency of employees in production
	Infrastructure efficiency
	Material used (total and per product)
	Energy used (total and per product)
	Unit product time
	Quality of internal and external services
	Production shutdowns
Quality	Percent of final product, which do not meet quality criteria
	Percent of raw material, which do not meet quality criteria
	Size of production losses
	Quality of internal and external services
Production plan tracking	Percent of production orders finished late
	Number of penalties
	Percent of production orders finished ahead
Employees' issues	Complete job satisfaction of employees
	Lost work due to injury and illness
	Average length of service of employees
	Employees' proposal for improvements and innovations

In [44], Javon et al. implemented the approach of using production KPIs as a reference value for the closed loop production control system. The chosen KPIs were Productivity, Mean production cost and Mean production quality. However, none of these KPIs are directly measurable from the production line rather an indirect processing of some process variables is done in order to calculate. Figure 7 illustrates the close loop system that is designed in [44] with help of MATLAB, Simulink and other simulation tools.

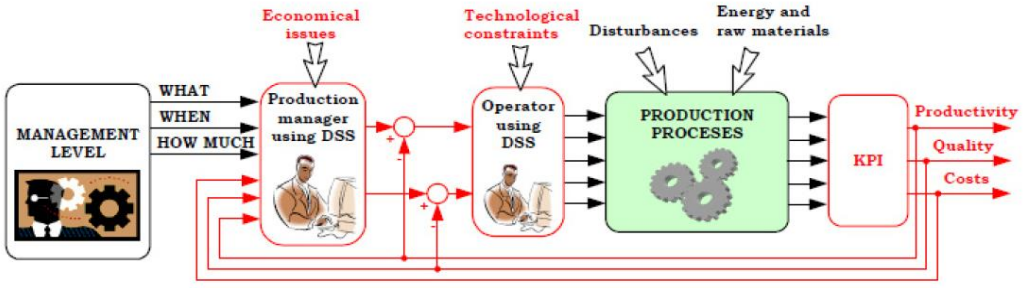

Figure 7: *Closed-loop control system of production process [44]*

The three identified KPIs serve as the output control variables in Figure 7. Thus, to achieve the desired set points input variables are tuned. The input variables used in [44] Figure 7 are low level indicators in the production line such as Production speed, Raw material quality and batch schedule. Delays and lack of storage capacities along with several other indicators in the production process are used as the mean of depicting disturbances in the close-loop system.

2.4.1 ISO 22400 Standard

Due to the utter importance of KPIs for manufacturing industry, the International Standard Organization (ISO) has worked towards identifying and designing a very basic set of performance measuring indicators at manufacturing operation management level of the industry that can be critical towards the success of any manufacturing industry. ISO has designed this standard under the name 'ISO 22400-Automation systems and integration — Key performance indicators (KPIs) for manufacturing operations management'. The ISO 22400 consists of 34 KPIs in which some may differ in terms of their implementation, depending upon the industrial environment which they are intended for. The KPIs identified and described in the ISO 22400 are intended for factory managers that are majorly responsible for the performance of the production site. The audience for these KPIs also include all the personnel working in the industrial environment that has a role in planning production activities and designing manufacturing systems. The ISO 22400 adopt the physical equipment model for hierarchy from the IEC 62264-3, that has some general terminologies such as Enterprise, site and area along with specific vocabulary for work units and centers. Figure 8 below illustrates that role-based hierarchy for equipment in industrial environment.

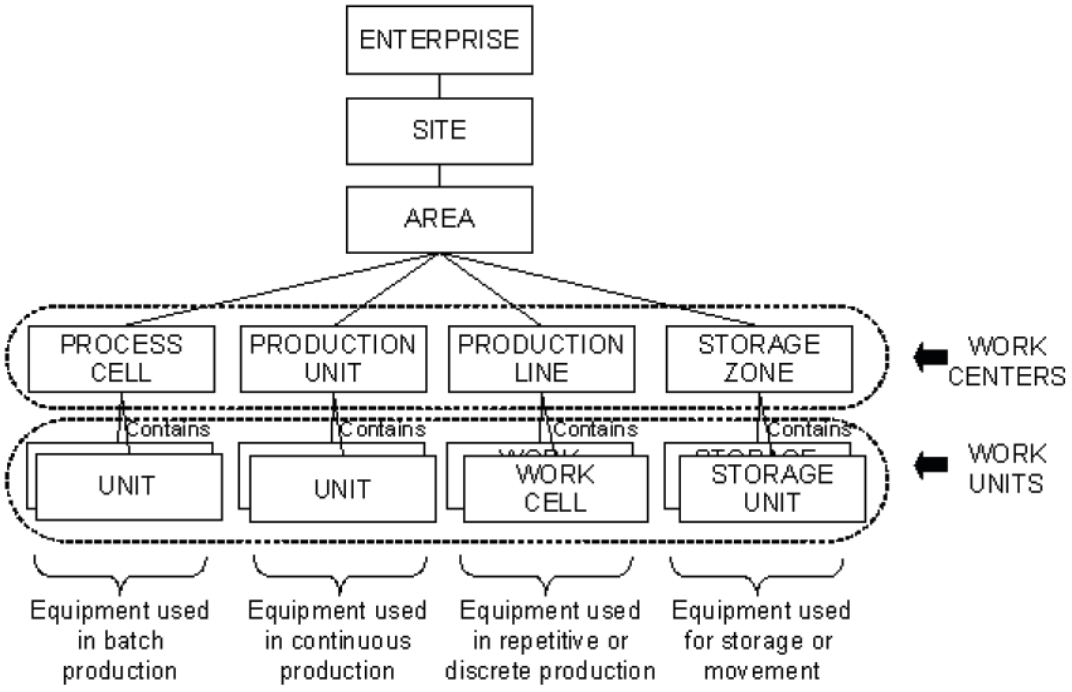

Figure 8: *Role based equipment hierarchy [25]*

ISO has released two documents, which give the definition, description and scope of these KPIs. Following are the two standards that are released.

1. ISO-22400-1: Overview, concepts and terminology
2. ISO-22400-2: Definitions and descriptions

The other two parts of the ISO-22400 standard are planned to be released in the future. These ones are:

3. ISO-22400-3: Exchange and use
4. ISO-22400-4: Relationships and dependencies

The first part of the standard ISO-22400-1 [24], gives the overview and concept as well define the terms that are used in constructing a KPI, the basic concepts and terms that form the KPI framework are described in this part of the International Standard. The second part ISO-22400-2 [25], introduces 34 KPIs that can be used at MOM level. Moreover, a complete description of each KPI is presented that includes their definitions, range, scope, formulas, timings and audiences. These KPIs are developed or identified by passing it through a complete lifecycle as shown in Figure 9.

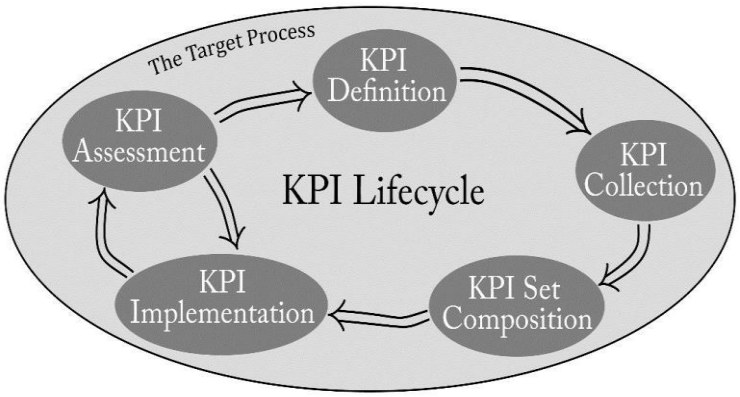

Figure 9: *KPI Lifecycle [25]*

The ISO-22400-2 [25] also defines a template and model for KPIs, which is shown in Table 2. This template shows how a KPI can be described and all the necessary terminology and fields related to a KPI.

Table 2: *Structure of a KPI in ISO-22400 [24]*

KPI Description	
Content	
Name	Name of the KPI
ID	A user defined unique identification of the KPI in the user environment
Description	A brief description of the KPI
Scope	Identification of the element that the KPI is relevant for, which can be a work unit, work center or production order, product or personnel
Formula	The mathematical formula of the KPI specified in terms of elements
Unit of measure	The basic unit or dimension in which the KPI is expressed
Range	Specifies the upper and lower logical limits of the KPI
Trend	Is the information about the improvement direction, higher is better or lower is better
Context	
Timing	A KPI can be calculated either in • real-time - after each new data acquisition event • on demand - after a specific data selection request • periodically - done at a certain interval, e.g. once per day
Audience	Audience is the user group typically using this KPI. The user groups used in this part of ISO 22400 are • Operators – personnel responsible for the direct operation of the equipment • Supervisors – personnel responsible for directing the activities of the operators • Management – personnel responsible for the overall execution of production
Production methodology	Specifies the production methodology that the KPI is generally applicable for • Discrete • Batch • Continuous
Effect model diagram	The effect model diagram is a graphical representation of the dependencies of the KPI elements that can be used to drill down and understand the source of the element values. NOTE This is a quick analysis which supports rapid efficiency improvement by corrective actions, and thus reduces errors
Notes	Can contain additional information related to the KPI. Typical examples are • Constraints • Usage • Other information

The 34 KPIs defined at MOM level in ISO-22400-2 are divided in to four types based on different processes in the manufacturing systems. These four types are production, maintenance, quality, and inventory operations management.

The production operations management KPIs deal with production line activities, such as monitoring the flow of production orders and batches, scheduling machines and workers, ensuring completion of orders in time. These KPIs are mostly related to product managers and workers that work close to the production line. For example, KPIs in this category are availability, allocation efficiency, utilization efficiency and technical efficiency.

The maintenance operations management KPIs are regarding the maintenance of all the manufacturing resources, such as machines, robots and other tools. It includes planning maintenance activities for the production line periodically. For instance, KPIs in this

category are mean time to failure, setup rate, mean time to restoration and corrective maintenance ratio.

The following Table 3 divides all the 34 KPIs in to the above-mentioned categories.

Table 3: *Types of KPIs based on ISO 22400*

KPIs	Production	Maintenance	Inventory	Quality
Work efficiency	×			
Allocation ratio	×			
Throughput rate	×			
Allocation efficiency	×			
Utilization efficiency	×			
Overall Equipment effectiveness index	×			
Net equipment effectiveness index	×			
Availability	×			
Effectiveness	×			
Quality Ratio				×
Set up ratio	×			
Technical efficiency	×			
Production process rate	×			
Actual to planned scrap ratio				×
First pass yield				×
Scrap ratio				×
Rework ratio				×
Fall off ratio				×
Machine capability index	×			
Critical Machine capability index	×			
Process capability index	×			
Critical Process capability index	×			
Comprehensive energy consumption	×			
Inventory turns			×	
Finished good ratio	×			
Integrated goods ratio	×			
Production loss ratio	×			
Storage and transportation loss ratio			×	
Other loss ratio			×	
Equipment load rate	×			
Mean operation time between failures		×		
Mean time to failure		×		
Mean time to restoration		×		
Corrective maintenance ratio		×		

The quality operations management KPIs are of great importance in any manufacturing system, they ensure that all products produced are of best quality. These KPIs indicate the performance of whole production line in terms of quality perspective. Top-level management is mostly interesting in the quality of the products produced rather than small details about the production line, thus these quality operation management KPIs can help them in getting the overview of the whole manufacturing plant. Example of vital quality KPIs are quality ratio, rework ratio and Actual to planned scrap ratio.

Inventory operations KPIs deal with activities such as transportation of raw material from warehouse to work centers, dispatching of finished products and keeping track of inventory in the storage. For example, KPIs in this category are inventory turn and Finished goods ratio.

Moreover, ISO-22400-2 also present time models for the manufacturing industry that help in defining and identifying a relationship between different times in use. There are three time models described that carter different domains in an industrial environment. These models include time model for work units, time model for processing production order and time model for personnel, which are represented in Figure 10, Figure 11 and Figure 12 respectively.

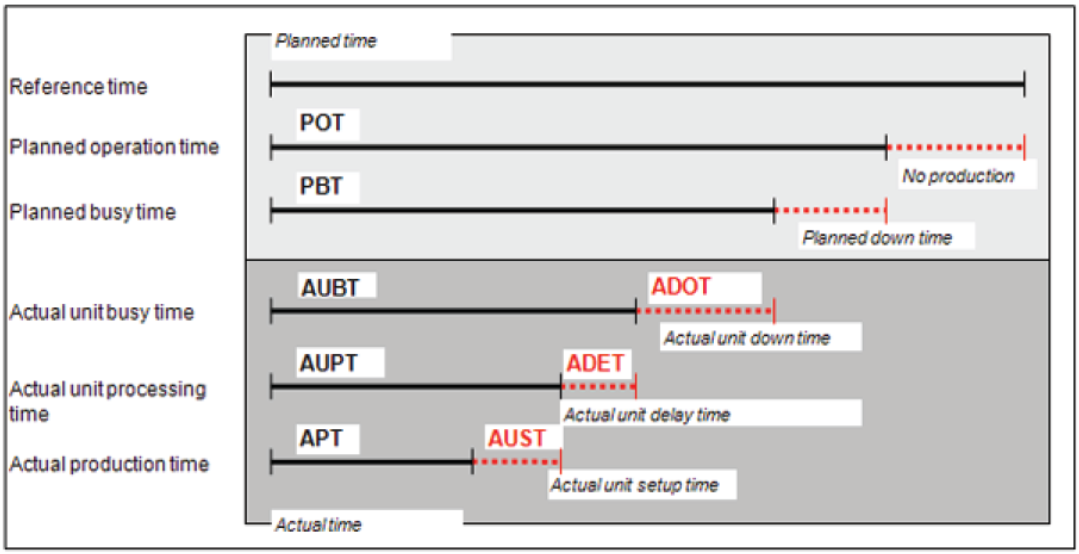

Figure 10: *Time lines for work unit [25]*

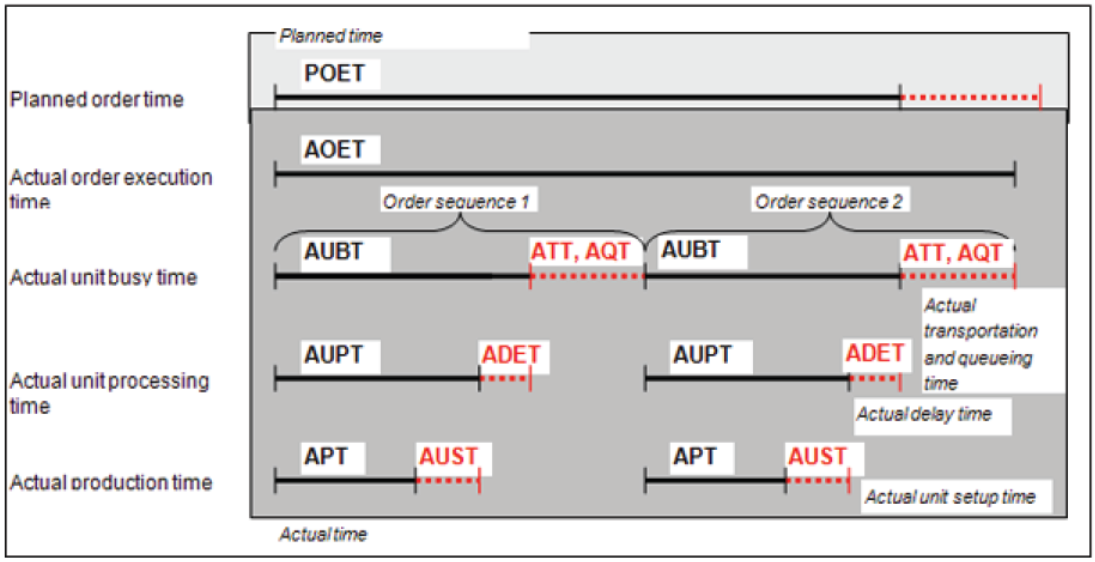

Figure 11: *Time lines for production order processing [25]*

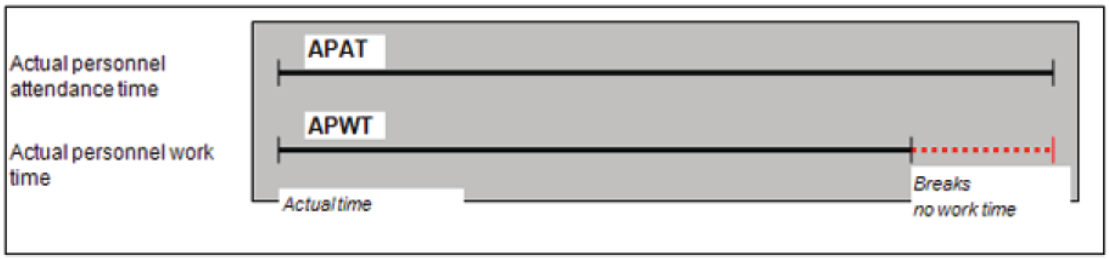

Figure 12: *Time lines for personnel [25]*

The aforementioned four different types of KPIs defined in Table 3 are further divided in to eight different subcategories to make it easy for the industry to interpret and classify these KPIs across their entities. These subcategories include resource management, detailed scheduling, definition management, dispatching, tracking, data collection, execution management, and analysis.

2.4.2 Key Performance Indicator Markup Language

The Key Performance Indicator Markup Language (KPI-ML) is the first step for implementing the defined KPIs in ISO-22400. The first version of a KPI-ML was introduced in May 2015 by the MESA. According to MESA, "*KPI-ML is an XML implementation of the ISO 22400 standard, Automation systems integration - Key performance indicators (KPIs) for manufacturing operations management. KPI-ML consists of a set of XML schemas written using the World Wide Web Consortium's XML Schema language (XSD) that implement the data models in the ISO 22400 standard" [26].*

One example of KPIML is presented for one KPI in the following Figure 13. The XML contains all the information related to a KPI that is provided in the ISO 22400-2 standard. It should be noted that XML for other KPIs are shown in Appendix A – KPIML for the Implemented KPI's.

```
<?xml version="1.0" encoding="UTF-8"?>

<KPIDefinition
xsi:schemaLocation="http://www.mesa.org/xml/KPI-ML-V01 KPI-ML-V01.xsd"
xmlns="http://www.mesa.org/xml/KPI-ML-V01"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <ID>AV100</ID>
 <Description> Availability is a ratio that shows the relation bet ween
the actual production time (APT) and the planned busy time (PBT) for a
work unit. </Description>
 <Name> Availability </Name>
 <Scope> Work unit </Scope>
 <Formula> Availability = Actual Production Time/Planned Busy Time
</Formula>
 <UnitOfMeasure> % </UnitOfMeasure>
 <Range>
        <ID> Natural </ID>
        <Description> Natural Range </Description>
        <LowerLimit> 0 </LowerLimit>
        <UpperLimit> 100 </UpperLimit>
 </Range>
 <Trend> Higher-is-better </Trend>
 <Timing> On-demand </Timing>
 <Timing> Real-time </Timing>
 <Audience> Operator </Audience>
 <Audience> Supervisor </Audience>
 <Audience> Management </Audience>
 <ProductionMethodology> Batch </ProductionMethodology>
 <ProductionMethodology> Continuous </ProductionMethodology>
 <Notes> “Availability indicates how strongly the capacity of a work unit
for the production is used in relation to the available capacity. The
term availability is also called degree of utilization or capacity
factor.” </Notes>
</KPIDefinition>
```

Figure 13: *Availability KPI XML based on KPIML*

2.5 Knowledge Based Systems

Due to rapid growth of manufacturing industry and technological progress, an efficient knowledge management is important for the organizations. KBS has gained a vital role in industry’s effort to share and manage knowledge. KBSs have an impact on every level of organizational knowledge: individual, group, organizational and knowledge links [54]. To extract the knowledge and solve problems through different reasoning techniques and processes, KBS are designed. In the context of this thesis, KBS can best be defined as a set of knowledge description statements, which are presented with the help of specific Knowledge Representation (KR) language and that can be queried or extended with a program [41]. For the representation of knowledge, several languages are used such as Ontology Web Language (OWL), Prolog, Answer set programing language, and constraint programming [52].

2.5.1 Ontology

Designing a Knowledge Base (KB) for the KBS is considered as the most important and major task in implementing a KBS. Ontologies are considered as the one of the candidate that can serve as the knowledge base for the KBS.

Ontology is defined as *"explicit specification of a conceptualization"* by Thomas Gruber in [39]. In general, ontologies are used to represent concepts, relationships and other properties that are necessary for modeling a specific system. An ontology describes the related vocabulary and relationships for a specific domain thus developing a common understanding of knowledge representation and information sharing [38]. The core motive of designing an ontology is to share common knowledge among researches and software agents of the structure of information as well to enable reusability of that specific domain knowledge. By developing mutual understanding of the structure and vocabulary of the information, make it easy for data analysts and software agents to extract explicit as well implicit knowledge from various sources [38].

Ontologies can be classified based on different factors and aspects. In [37], ontologies are classified based on the domain in which they can contribute. Ontologies ranges from a broader domain of KR (Knowledge Representation) ontology towards a narrow domain of an Application specific ontology. Figure 14 shows different types of ontologies that can be made ranging from a wider to a narrow domain. Moreover, it shows a trade-off between high usability and high reusability going from one domain towards another, as both are inversely proportional to each other.

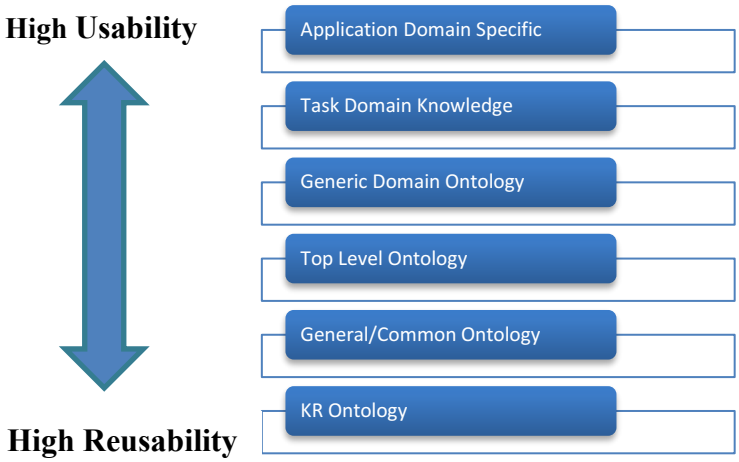

Figure 14: *Classification of Ontologies, based on [37]*

To design an ontology certain methodologies are used. For instance, the NeOn (Networked Ontologies) methodology is used, which define nine scenarios for development of ontology [53]. These scenarios depend on the types, availability and handling of resources. Another example of methodology for development of ontology is presented in

a guide known as *Ontology Development 101* [39]. In this ontology creation guide seven steps are described for designing ontology. In [39], the first step in ontology development is to define a scope of the ontology and to identify the domain in which areas this ontology will be used. The ontology developed in this thesis caters the manufacturing system domain. Once the domain of the ontology is defined, reusing other existing ontologies in that domain can be used or merged with it. The next step in ontology designing is to gather the terminologies and vocabulary used in that domain. The ontology designed in this thesis contains specific terminology of manufacturing system domain along with its relationship with different key performance indicators for monitoring.

Furthermore, designing an ontology include organizing the terms in a hierarchal structure known as objects/classes. These classes can further be divided in to subclasses. Different classes can be connected to each other through different properties and relationships. For each class, certain other properties such as domain and range can also be defined. These steps are followed by creating an instance/individual of a class. These instances or individuals are real case scenarios that relates to an object. Individuals with in the same class more or less exhibits some similarities in terms of their properties.

2.5.2 Protégé

Protégé is an open-source tool, free to use, and provides a set of features to design ontologies for knowledge-based applications [43]. Protégé is an ontology editor tool that has the full support for the design of Resource Description Framework (RDF) based ontologies, which also gives the feature of connectivity with description logic reasoners, such as HermiT[1] and Pellet[2].

In protégé, a user can create and edit more than one ontology in a single workspace. Protégé provides a user interface that can be modified according to the requirements of users. Moreover, it allows visualizing the relationships between different classes, subclasses and the properties attached to them in different hierarchical structures. It has the ability of tacking inconsistencies with the help of advanced explanation support. Additionally, it supports various formats such as RDF/XML, Turtle, OWL/XML, and OBO for ontology uploading and downloading. Moreover, Protégé also supports several refactor operations such as merging and moving axioms across different ontologies, and renaming multiple entities in ontologies.

2.5.3 SPARQL

The Simple Protocol and RDF Query Language (SPARQL) is commonly used for querying RDF-based ontologies. SPARQL is a protocol and query language that manipu-

[1] http://www.hermit-reasoner.com/
[2] https://www.w3.org/2001/sw/wiki/Pellet

lates and retrieve data from databases commonly stored in RDF format [42]. SPARQL queries are formatted in form of triples that permit the retrieval of results in form of RDF graphs. A basic SPARQL query consist of three elements: subject, predicate and object. Below, Figure 15 shows a basic example of a SPARQL query.

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

Figure 15: *A basic SPARQL query example*

The result of the above query will be achieved in form of columns for each of the three elements and rows containing the retrieved data for each one of them with their respective relation. Besides the SPARQL query language, SPARQL Update is used to add, edit and delete triples from RDF store.

2.6 Programming Technologies and Tools

This section provides a brief description of programming technologies and tools used in this research work. Hyper Text Markup Language (HTML), Cascading Style Sheet (CSS), Javscript (JS) and AngularJs are used for developing the frontend. Whereas, NodeJs is used as programming language for the backend development.

2.6.1 HTML/CSS

The is used for creating webpages and helps in describing the web documents. HTML along with CSS is very commonly used for formatting web pages. Most of the browsers, such as Internet Explorer, Google chrome and Firefox can interpret the web documents written in HTML. HTML documents have elements also known as tags, which serve as a building block for web pages. Some of the most common tags used for making web pages are body, heading, title, paragraph and table. HTML is an open technology, which is very user friendly and easy to update. The validation of HTML is another important aspect that ensure increase in web accessibility [28].

CSS works side by side with HTML document. The HTML only tells the browser what information and data is going to be displayed, whereas CSS instruct the browser about how to represent and format that data. CSS decides the font sizes, coloring, and spacing of each element of the HTML. Moreover, CSS takes care of the styling factor of the webpages to make it more attractive and readable for its users [29]. In this thesis,

HTML and CSS along with Google charts[3] will be used to visualize the concept of key performance indicators for the user by creating a webpage.

2.6.2 JavaScript

JS can enhance the interfaces HTML gives us. Fundamentally, JS is an object-oriented scripting programming language that adds interactivity and behavior to the web page. Moreover, JS is lightweight that feature makes it very easy and effective to run on the browser. JS can easily be integrated with HTML to make the static webpages dynamic and interactive. JavaScript is mostly famous for client side applications as it helps in validating the user inputs before sending it to server thus decreasing interaction and workload on the server. [30]

2.6.3 Node.js

Node.js is a runtime environment based on JavaScript that is designed on Google Chrome's JavaScript V8 Engine. Node.js is very effective and lightweight for real time web applications because of its event driven and non-blocking I/O model [31]. Node.js is mostly used for making server side applications. Node.js is open source, and can be used free of cost by anyone. Moreover, most of the data intensive I/O web applications are developed via node.js due to its light weight feature. These applications range from video streaming sites to chat applications, from *weather applications* to simple *To do application* and other single page web applications.

In this thesis, Node.js will be used to create the back-end. Some of the most common modules associated with node.js such as Express[4], Socket.io[5] will be used in order to provide the desired/implemented functionality. Express is used for setting up a routing table and middleware's to respond to HTTP Requests to execute different action according to the specified HTTP Method. Whereas, Socket.io is used for two-way communication based on user defined events in real time.

2.6.4 AngularJs

AngularJs is a JavaScript framework that is used to create dynamic webpages [45]. AngularJs can be included to an HTML page to extend its functionality and can be used as a toolset for building the framework that are specific to designed application. Furthermore, AngularJs has the feature of extensibility and has the ability to encompass other libraries. Besides this AngularJs has two-way data binding, which is, the view is updated whenever the model changes and similarly the model is updated whenever the view

[3] https://developers.google.com/chart/
[4] https://expressjs.com/
[5] https://socket.io/

is changed. AngularJs uses controllers that are the behavior behind the DOM elements. AngularJS helps in expressing the behavior in a user-friendly readable form without registering callbacks or listening to model changes every time, thus avoiding boilerplate of updating the DOM. Moreover, AngularJs has this additional feature of directives that let the user to create its own HTML syntax.

2.7 Virtual Engineering and Digitalization in the Industrial Domain

Manufacturing systems and productions lines resources have become very expensive and requires high standards of technical knowhow. Moreover, it gives very less options for design flexibility once installed. Due to high production volumes and busy schedules, the margin for error in such production lines is also very little. To solve such problems the concept of virtual engineering and digitalization emerged in manufacturing industry. Virtual engineering technologies enables an industry to model its real-world production line in the form of animations, simulations and Human Machine Interfaces (HMI) to control and test their systems, virtually [32].

Commissioning of automated systems is a cost and time-consuming process. Thus, virtual testing of the design alternatives and system prior to laying it down to the real production line has become very important part of the manufacturing industry, as it saves a lot of energy and minimize the risk of failure. Virtual simulation can help in identifying problems in the production line even before implementation in the real-world production line. Moreover, it helps in validation of the assembly process ad production capacity testing of production line before commissioning. Virtual systems that are integrated with the real production line are easy to monitor and control, thus assuring high quality and functionality of the line [33].

Moreover, virtual platforms have a great significance for researchers as it provides them a testbed for testing their concepts and ideas. Implementation of new concepts and ideas directly to the real-world production line can be very costly and risky. Testing new ideas and concepts is an iterative process that makes it almost impossible to apply them on real production lines. Virtual simulations are one of the best solution for such problems because provide a safe and cheap testing solution. In this thesis, a simulation of the real-world production line is used for implementation of key performance indicators.

2.7.1 FASTory Simulator

The production line used in this thesis as the testbed to implement the ISO-22400 KPIs is the FASTory line the one located at FAST laboratory, the Tampere University of Technology (TUT). This production line was originally used for assembling different parts of mobile phones, the line can assemble mobile phones by drawing mobile

phone's main parts (frame, screen and keyboard) with different colors and different shapes, now this line is used for research purposes [55]. In this thesis, the simulation of the real production line would be used which is also known as FASTory Simulator. The simulation is efficient to use instead of the real production line as it can help avoiding problems of electrical shutdown and other mechanical issues which are common in real production line.

FASTory Simulator was developed during the implementation of eScop (Embedded systems Service-based Control for Open manufacturing and Process automation) project solution on FASTory assembly line [58]. It was developed with the goal that it will provide a flexible Open-Knowledge Driven industrial system, which could be applied on manufacturing applications.

The production line has in total 12 work stations, each workstation has the following components.

- A robot, to perform the specified task at each workstation.
- A conveyor line, for movement of pallet between the workstations.
- Presence sensors, to detect the presence of the pallet.
- Radio Frequency Identification Device (RFID) readers, for pallet recognition (detecting the pallet ID).
- Stoppers, to stop the pallet in the zone.

A pallet is used for transportation of product from one workstation to the other over the conveyor line. Mechanically the conveyor line consists of three components, a straight conveyor, bypass conveyor and two junctions to link the conveyors. A motor is used to move all the belts of the module. There are four stoppers which are placed in different zones of the conveyor to facilitate the pallet to stop at specified location, one is at the end of bypass conveyor, another in the incoming junction and two in the straight conveyor. Figure 16 shows the testbed, FASTory line.

Figure 16: *Testbed - FASTory line [55]*

Figure 17 shows the complete interface of the real production line. The interface depicts all the workstations along with all the necessary sensors at each one of them. Moreover, it has two parts, one is the animated simulation of the real line and the other is the control panel for that simulation. Figure 17 also shows a legend in the upper right corner that has the symbols to represent all the sensors. Users can invoke any available services on the line by pressing the buttons in the control panel. Different services can be invoked on the FASTory line through RESTful client, as it works RESTful services. It also has the alert functionality that is the user can subscribe to get notifications about different events occurring on the line [55].

In the control panel, each workstation has its own set of buttons which give user the option to move the pallet from zone to the other. Moreover, the colored buttons give user an option to select among the three colors for the drawing. The other 9 buttons are used for selecting different recipes of frame, screen and keyboard respectively [55].

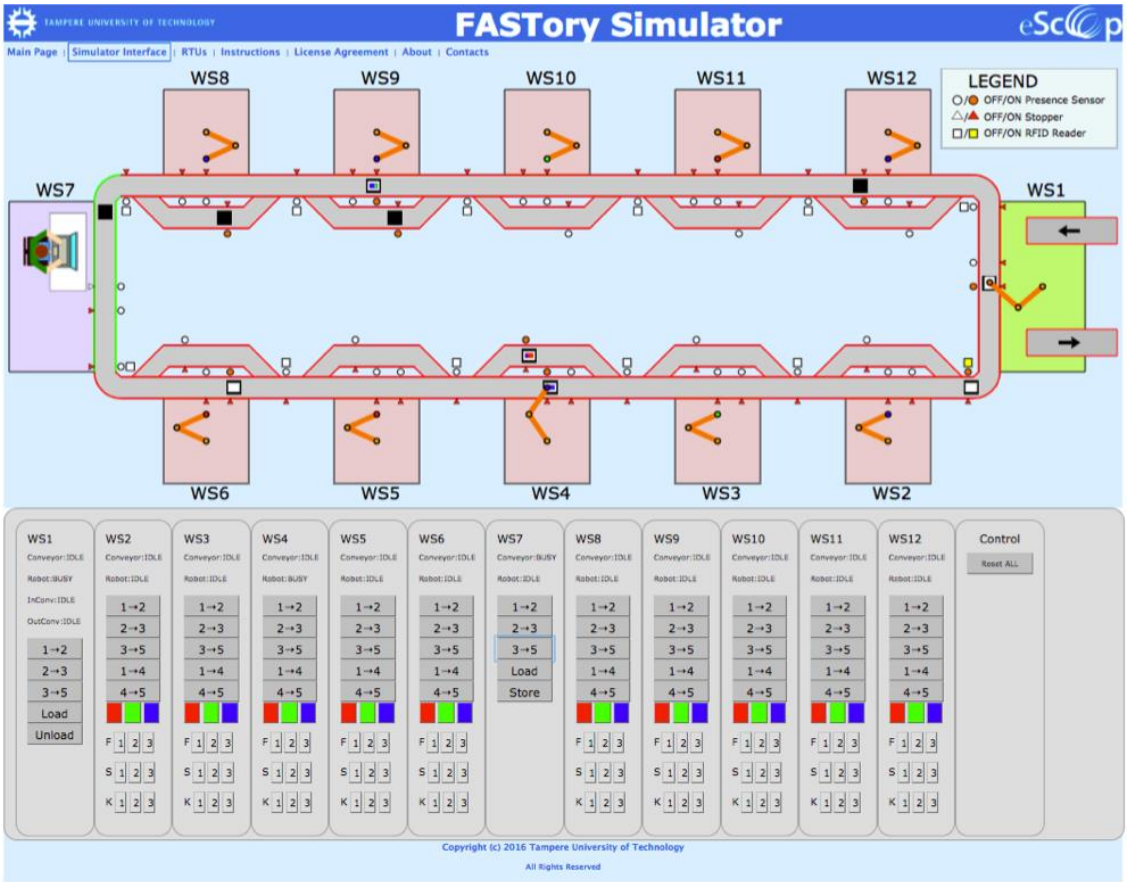

Figure 17: *FASTory Simulator interface*[6] *[51]*

The production line works in a manner that first pallet is loaded to the system at workstation 7, the pallet then reach workstation 1 for loading paper on the pallet. For loading pallet and paper, '*Load'* button in the control panel are used in the respective workstations. After loading the paper, the mobile phones are drawn on the paper between

[6] http://escop.rd.tut.fi:3000/

workstation 2 to 6, and 8 to 12. The drawing operation consist of three tasks, frame drawing, screen drawing, and keyboard drawing. The user at this point can also select among the three specified colors for their drawing. At last, the pallet goes to workstation 1 for unloading using the '*Unload*' button in the control panel. The user then had the option after completing one cycle of production to either unload the pallet at workstation 7 to stop production or continue to add new paper at workstation 1. The workstation 7 is also used as a storage or buffer for allowing temporary storing of products at any production stage for optimizing the overall production of the line.

3. APPROACH

This Chapter gives the details about the approach implemented to solve the research problems identified and accomplish the desired objectives. This chapter is divided in to two sections; the first section will describe in general the research methodology adopted and steps taken to move forward in this thesis. Moreover, the second section will explain the overall architectural view of the system and its components.

3.1 Research Methodology and Phases

The research objectives of this thesis revolve around the efficient implementation of the key performance indicators for manufacturing systems. The work done in this thesis has been through various stages of brainstorming, research, analysis and review of literature. The initial phases of identifying the problems and designing a work plan to solve these problems are the most critical. Following Figure 18 shows different phases of the research.

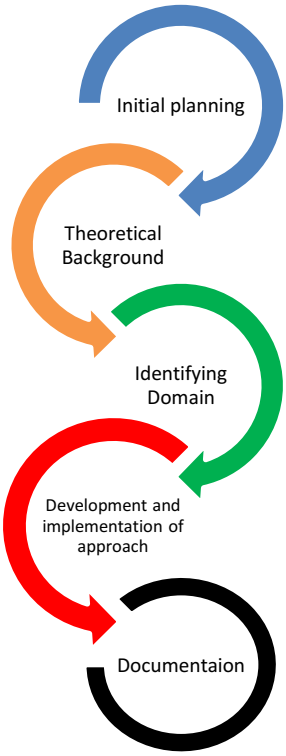

Figure 18: *Different phases of methodology*

- *Initial Planning*: In the initial planning phase, the most critical and important step is the identification of the research topic and areas. A vast amount of research and analysis is required to identify the research topic and areas. Additionally, a basic work plan is designed in order to move forward towards an efficient and state of the art solution of the problems identified for the thesis.
- *Theoretical Background*: Identifying an efficient solution is not possible without a valuable theoretical background and research. To lay a solid foundation for the thesis, a significant amount of research was done, which comprised of journal papers, articles, books and reports. The theoretical background phase, gives an idea of the previous work done in relation to the thesis topic and which approaches and methodologies were used in past to encounter related problems.
- *Identifying Domain*: After the theoretical background phase, the gaps and problems that need to be solved are identified. A thorough analysis and examination of literature in the previous phase, helps in narrowing down the domain of the thesis and thus laid the foundation for the thesis topic.
- *Development and implementation of approach:* The next phase in research methodology is the development of approach for solving the problems identified in the previous phases. Moreover, implementing that approach on the real-world manufacturing system and figuring out results and comparing them with previously implemented approaches gives a solid base and recognition to the suggested approach.
- *Documentation*: The final phase in the course of the thesis is to document all the findings and information in the form of dissertation. The background knowledge necessary to understand the thesis topic along with all the problems faced during the thesis along with their possible solutions and the research done to implement the approach is written in the form of well-structured document for future reuse and research.

3.2 Architectural Views

This section of the chapter explains the approach provided for the implementation of the key performance indicators. The overall architecture view of the system is illustrated in this section along with brief description of each component. The architecture of the system is divided in to five major components. Figure 19 show these components.

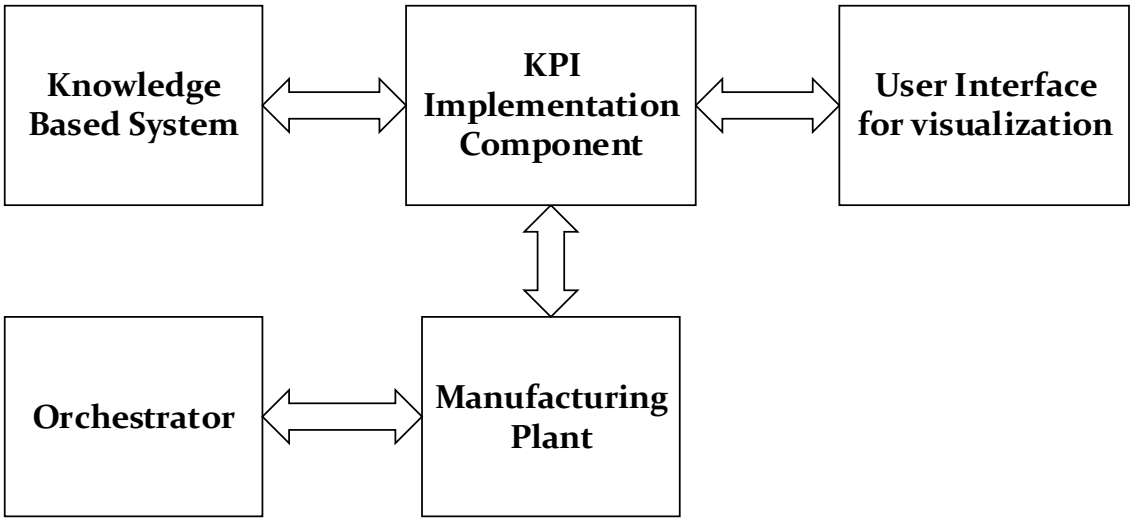

Figure 19: *Architectural view of the overall system*

3.2.1 Knowledge based System

As mentioned in the previous chapters, a KBS is used to model the use case manufacturing system. In this thesis, ontologies are used as knowledge base (KB) in the KBS, which will help in designing a reconfigurable and reusable model for future extendibility and research.

For this thesis, RDF data format has been selected, that is designed for Semantic Web used in building large-scale data sets. One of the reason for its popularity and extensive use is its flexibility and ease of use. Data can be stored in form of RDF graphs without the need of designing a schema for it beforehand. In addition to primary data, other data such as metadata, annotation and hierarchical information can easily be added. SPARQL, a query language for RDF, is used for extracting the data in the form RDF graphs as well updating the data collection. Moreover, SPARQL can unite data from different ontologies, as well as documents, inference engines, or anything in which knowledge is stored in form of labeled graph. In the RDF data format, querying supports agnosticism to a certain extent.

Figure 20 represents the basic architecture of a KBS and interaction between different building blocks. As shown in the Figure 20, a KBS comprises of three basic building blocks which are

- A KB that contains specific domain related knowledge necessary for computing and solving the problems.
- An engine, which is used to extract the knowledge stored, it tries to extract solution from the knowledge base.
- An interface, that helps user to communicate with knowledge base such as querying and updating the knowledge base. Fuseki server provides REST medium for interacting with the model. The model can be queried with HTTP GET re-

quest with the desire query, and can be updated with POST request containing the desire update.

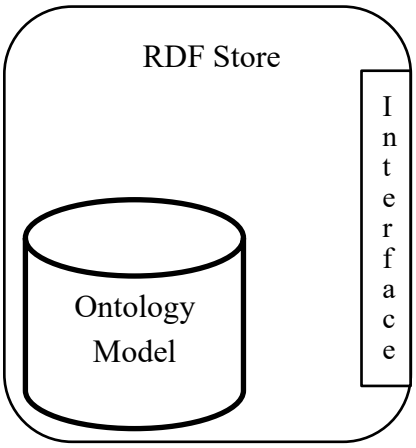

Figure 20: *Knowledge Based System*

For creating the ontology, an ontology editor that provides a platform for creating model based on the ontology language. As discussed in the Programming Technologies section, Protégé software is utilized for creating the model. Protégé not only provides a platform to create an ontology model, but also provides an inbuilt platform for querying the created model as well visualizing the structure [37].

In the ontology model, the whole system is divided in to three major classes, which are illustrated in Figure 21. The three classes are: *Equipment*, *KPI* and *KPI_Variable*.

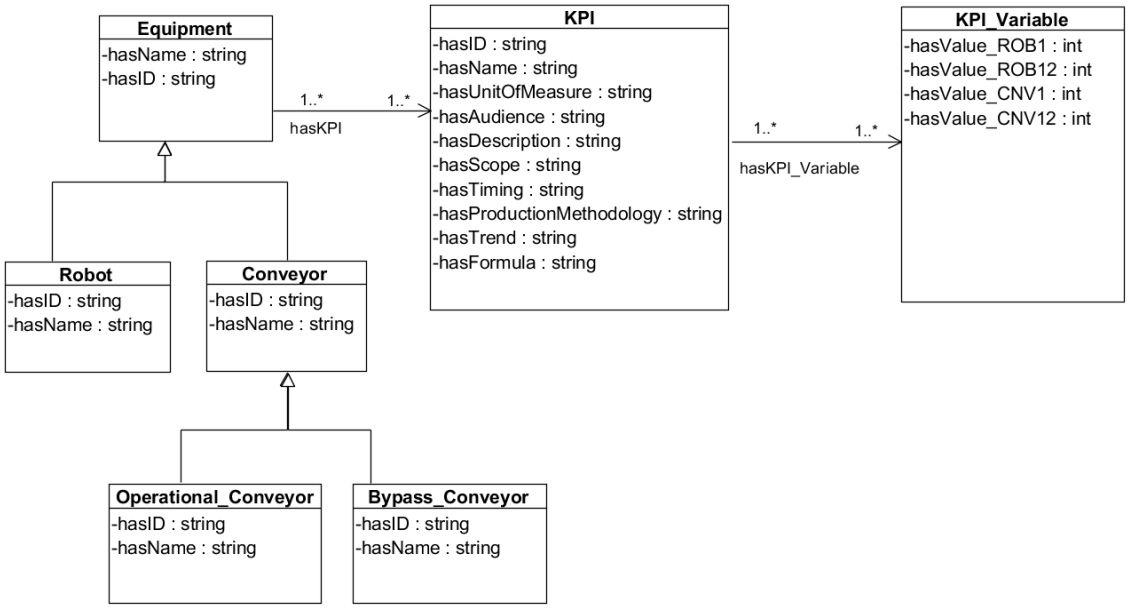

Figure 21: *Class diagram representation of Ontology model.*

The *Equipment* class contains all the equipment available at the factory floor. The *Equipment* class is further divided in to two subclasses, which are *Robot* and *Conveyor*. The *Conveyor* subclass is further divided to separate the *Operational-Conveyor* from

ByPass-Conveyor. Each individual included in this class have a *hasID* and *hasName* property. Moreover, each *Equipment* class individual is associated with one or many individuals of the KPIs class by *hasKPI* property.

The *KPI* class contains all the implemented key performance indicators defined in the ISO-22400 as well the KPIs which user create and define in accordance to their factory floor needs. Each individual of *KPI* class has all the data properties that are define in the data models by MESA international in KPIML. These properties include *hasID*, *has-Name*, *hasUnitOfMeasure*, *hasformula*, *hasDescription*, *hasAudience*, *hasTiming*, *hasTrend*, *hasScope* and *hasProductionMethodology*. Each individual in this class is associated to one or many individuals in the *KPI_Variable* class depending on the formula of each KPI.

The *KPI_Variable* class have all the variables that are enlisted in the formulas of each KPI. These KPI variables have *hasValue* property for all the equipment i-e Robots and Conveyors. These KPI variables stores the updated data generated from the simulator interface for all the robots and conveyors, which is used in creating the visualization such as pie charts and graphs.

3.2.2 Manufacturing Plant

Manufacturing plant serve as the major component, as all this implementation is objected to monitor and evaluate the performance of a manufacturing plant by visualizing its key performance indicators. As previously explained, FASTory line is utilized as a testbed for this research work.

- **FASTory Simulator as Discrete manufacturing system**

FASTory simulator is used instead of the real line because of the problems and difficulties associated with using real production line such as electrical and mechanical risk, running cost and most importantly the setup time that can slow down the process to large extent.

The FASTory simulator interface serve as the source of data for the KPI section. The KPI section of the simulator subscribe to all the events and changes on the multi-robot line simulator via REST. Any event that occurs on the multi-robot line simulator, the notification along with the related data is received in the KPI section for further manipulation and processing.

The simulator interface exposes the web services for getting the notifications. The user can subscribe to these notifications and can instantly receive it whenever there is some change on the interface, thus making it easy for monitoring the production line in real time. The data received with these notification messages have various attributes such as the pallet transfer status on conveyors, which contains the cell ID, the information of the

arrival and destination zone from which pallet has arrived, the zone to which pallet is moving and from which it is moving. Moreover, it also contains the pallet ID, time stamp and event ID for every invoked service. Based on the data received with in the notification message further processing is done.

3.2.3 KPI Component

For the purpose of this thesis, KPI component is used to implement the KPIs that are relevant to our manufacturing system. KPI Implementation component has vital significance in terms of data processing and interaction with other components of the system. The operations performed by KPI section are majorly divided in to three separate operation.

Firstly, it receives notifications from the manufacturing plant, based on these notifications and data received, it processes, manipulate and calculate the value for all the necessary variables in the KPI formulas for the respective cells. Secondly, the KPI section interacts with the KBS in order to update the data in the KB as well as to read the updated data through HTTP requests. Moreover, the KPI section also interacts with the front-end user interface and update it in the real time. For interaction with the front end, socket.io is selected as the medium of interaction because of its real-time bidirectional event-based communication thus achieving the desire real time functionality. The KPI component serve as a bridge between the front end and the knowledge based system. Additionally, all the changes on the front end are saved in the knowledge base via the KPI component.

Lastly, the KPI component serve as a control engine for all the operations. The control engine decides the sequence in which all the functions and operations will run in order to achieve orchestration of the system.

3.2.4 User Interface and Visualization

Visualization component is the integral part of the overall architecture system. Visualization component provides an interface to the user to interact, select the desire KPI and the equipment to monitor, and visualize with help of pie charts, line charts and bar graphs in run time. The Visualization component receives its data via a socket.io connection in real time whenever there is any update. The user section also gives user an option to create its own KPIs according to the needs of its specific manufacturing system.

For the development of visualization, AngularJS framework is selected along with HTML and CSS for representation and styling respectively. AngularJS extends the HTML vocabulary by adding new syntax. Moreover, AngularJS is selected because of its two-way data binding. Two-way binding keeps monitoring the expressions in view

template and whenever it changes, the corresponding part of the Document Object Model (DOM) is updated. Therefore, whenever any of the value changes on the view component (e.g., the value of input text) the respective bound model is updated, and whenever the model value changes, the view is automatically updated. Knowledge base discussed in the previous section is use to keep the inputs of the user until they are deleted by the user itself.

3.2.5 Orchestrator

The orchestrator is responsible for executing the production order provided by the user or production line manager. Then, the orchestrator takes the requirements and specification assigned by the user for each production order and execute the order according to user requirements. The user provides the total quantity of products to be produced in an order, the recipe of product and color for each component. The orchestrator then on the basis of these inputs start executing the order. By executing different production orders through orchestrator, the system will be able to monitor different KPIs for each order, such as quality of the products, workload on each workstation in case of each production order, and the time taken to execute the production order.

3.3 Sequence of Data Flow

The previous section describes the functionality of each of the component of the architecture used in this thesis. In this section, the flow of interaction between different components of the architecture is described. Figure 22 show the general sequence of how different modules interact with each other. Detail explanation of the below sequence diagram will be provided in the implementation chapter along with the mediums used to achieve each communication.

As shown in Figure 22, any event or change in the manufacturing plant is communicated to the node App. The node app after performing a series of operations extract useful information from the event and send request to the knowledge base for the update of that useful information. The knowledge base after updating the respective fields get another request from the node app that reads the updated data from the knowledge base. Once the node app gets the updated data, it calculates all the KPIs from its formula. After the KPIs are calculated in the KPI Component of the updated data it sends it to the user interface thus updating the visual graphics on the front end.

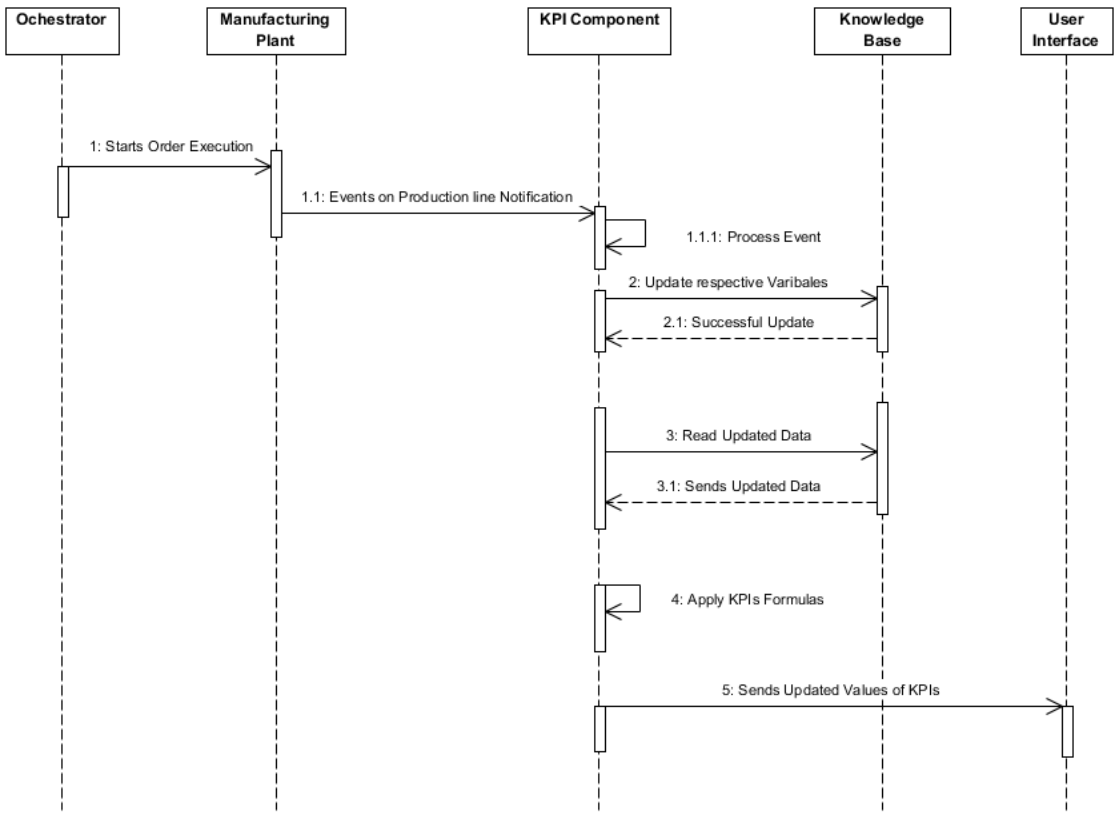

Figure 22: *Sequence Diagram showing general sequence of operations*

4. IMPLEMENTATION

This chapter presents a component diagram implemented in this thesis based on the architectural view of the system. Moreover, the chapter will explain the working of major components and their interaction protocol with other components involved in the architecture. In this chapter, a sequence of interaction will also be explained with the help of a detailed sequence diagram.

4.1 KPI Implementation

The KPI Implementation component is the major building block in the overall architecture of the system. It has major responsibilities of evaluating the KPIs formulas and interaction with other components in the architecture. The following Figure 23 represents the components involved in the architecture of this implementation.

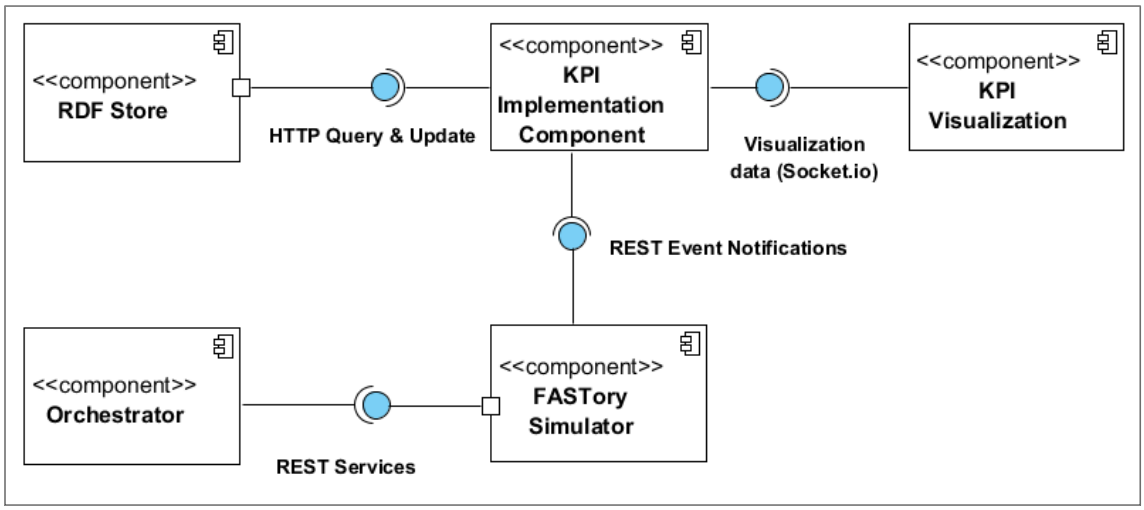

Figure 23: *Component diagram interacting in Architectural view*

The responsibilities assigned to KPI Implementation component can be divided in to three major responsibilities. Firstly, it retrieves the list of available KPIs and their respective formulas from the ontology that it needs to implement and visualize for the user. Secondly, it listens to the events received from the FASTory Simulator. Finally, it executes the formulas to find the values of each KPIs and send them to the front end for visualization. In this thesis work, five KPIs are identified that are applicable on to our production system and visualizing them will to monitor the performance of the production system. The following five KPIs are calculated with the help of the given formulas in the ISO 22400-2 standards [25].

- **Allocation Efficiency:** Allocation efficiency is defined as the ratio between Actual Unit Busy Time (AUBT) and Planned Busy Time (PBT). AUBT is the actual time a work unit is busy producing as well as the time it takes for transferring goods from one work station to the other. And PBT is usually estimated by the production and scheduling managers at the start of any production shift or order. The following equation (1) represents the formula for allocation efficiency KPI. It is calculated in percentage with range varying from a lower limit of 0 to upper limit of 100%.

$$Allocation\ Efficiency = \frac{Actual\ Unit\ Busy\ Time}{Planned\ Busy\ Time} \quad (1)$$

- **Utilization efficiency:** Utilization efficiency indicates how much of the work done is productive and can add value to the production order. It is calculated as the ratio between total time for which the work unit is productive and the total busy time of the work unit. It illustrates the productivity of each work unit during the execution of a production order. The following equation (2) represents the formula for utilization efficiency for each work unit. It is calculated in percentage with range varying from a lower limit of 0% to upper limit of 100%.

$$Utilization\ Efficiency = \frac{Actual\ Production\ Time}{Actual\ Unit\ Busy\ Time} \quad (2)$$

- **Availability:** Unlike allocation efficiency, which depicts the total busy time of each work unit with respect to the planned busy time, availability KPI shows the productive time a work unit spent in producing a product. It excludes the time spent in queuing or transferring products form one workstation to the other and only shows the time when the work unit is adding some value to the final production order. The following equation (3) shows the availability as the ratio between the Actual Production Time (APT) and the PBT for each work unit. It is also calculated in percentage with range varying from a lower limit of 0% to upper limit of 100%.

$$Availability = \frac{Actual\ Production\ Time}{Planned\ Busy\ Time} \quad (3)$$

- **Quality Ratio:** Quality ratio is defined as the ratio between good quantity and total produced quantity. Good quantity is considered as the products that fulfills the quality criteria or percentage set by the quality manager for every production order. The following equation (4) represents the above mention definition, it is also calculated in percentage and limits ranges from 0% to 100%.

$$Quality\ Ratio = \frac{Good\ Quantity}{Produced\ Quantity} \quad (4)$$

- **Scrap Ratio:** In contrast to the quality ratio, scrap ratio determines the number of scrapped products produced while executing a production order. Scrap ratio can be thought as the complete inverse of quality ratio,, it is the computed as the ratio between scrap quantity that didn't fulfil the quality criteria and total produced quantity, as shown in equation (5). Scrap quantity is all the quantity that does not fulfills the quality criteria set by the quality manager. It is also calculated in percentage with range varying from a lower limit of 0% to upper limit of 100%.

$$Scrap\ Ratio = \frac{Scrap\ Quantity}{Produced\ Quantity} \tag{5}$$

The abovementioned five KPIs are modelled in the ontology model and the list of these KPI is received form the ontology model. Afterward, these KPIs are formatted in proper JSON data structure as shown in Figure 25 and then each one of these formulas is executed one by one by initiating a 'new function' in the KPI Implementation component. The query in Figure 24 is sent initially to the ontology model via HTTP GET request to fetch the list of KPIs and their formulas.

```
var qs = require('qs');
var request = require('request');
var http = require("http");
var endpoint = 'http://localhost:3030/DS-1/sparql?'
var getFormulasQuery = "PREFIX
kpis:<http://www.semanticweb.org/KPI#>
Select ?kpi ?formula
Where
{
?z kpis:hasFormula ?formula.
Bind (strafter(str(?z),"http://www.semanticweb.org/KPI#>") as
?kpi)}";
var query = qs.stringify({ query: getFormulasQuery });
request.get(endpoint + query + '&format=json', function(error, re-
sponse, body) {
          if (!error && response.statusCode == 200){
               console.log(body);
          } else {
               console.log(error);
               }
});
```

Figure 24: *Query used to fetch the list of formulas*

The above JavaScript code is used to fetch the complete list of KPIs and their formulas. The above code is called the first time when the system starts and whenever the user adds a new KPI to the system by filling a 'Create New KPI Form' in the user interface of the system. The following result in Figure 25 is gathered by sending the above query to the ontology model.

```
{
 “Availability”: “(Actual_Production_Time/Planned_Busy_Time)*100”,
 “Allocation-Efficiency”: “(Actual_Unit_Busy _Time/Planned_Busy_Time)*100”,
 “Utilization-Efficiency”: “(Actual_Production_Time / Actual_Unit_Busy_Time)*100”,
 “Quality-Ratio”: “(Good_Quantity / Produced_Quantity)*100”,
 “Scrap-Ratio”: “(Scrap_Quantity / Produced_Quantity)*100”
}
```

Figure 25: *List of formulas recieved and formatted*

Furthermore, the KPI implementation component listen to the events notification from the FASTory simulator. The simulator sends HTTP POST notification whenever there is a change in the state of the simulator. These notifications are sent whenever any of the robot starts or stops drawing a particular recipe, or when any of the conveyor starts transferring pallet from on zone to another. The list of events available in the simulator are mentioned in Appendix B – FASTory Simulator Events.

The events received from the simulator are used to calculate different KPI variables from them. Some of the events that are used in this research work are *RobotStartDrawing* and *RobotStopDrawing, ConveyorStartTransferring* and *ConveyorStopTransferring,* and *PaperStartLoading* and *PaperStartUnloading.* Figure 26 shows an example of notification received when a Robot starts drawing a specific recipe. This event is used in calculating the busy time of the robots. Event notification body is received in JSON format that contains all the details of the change of the state. Notification contains the workstation number at which the operation is occurred, the type of operation that is occurred and the ID of the pallet on which the operation is performed.

```
{
  MSG: ‘RobotStartDrawing’,
  WS: ‘8’,
  PalletID: 1508488758200,
  Recipe: ‘1’,
  ServiceID: 1508488778004
}
```

Figure 26: *Example of Event notification received form Simulator*

Another example of event notification that is majorly used is for transferring the pallet from one zone to the other is given in Figure 27. This event in Figure 27 along with *ConveyorStopTransferrring* event is used to calculate the time a pallet takes from one zone to another. The *MSG* in the event notification shows the type of operation that is invoked on the FASTory simulator. *WS* indicates the workstation number that has performed the operation. The *PalletID* is the ID that is associated to each pallet at the start

whenever a new pallet is loaded. *ServiceID* is the ID of the service that is invoked. Each service has a unique id associated to it.

```
{
  MSG: 'ConveyorStartTransferring,
  WS: '2',
  From: '1',
  To: '2',
  PalletID: 1508488758200,
  ServiceID: 1508488778004
}
```

Figure 27: *Example of Event notification received form Simulator*

Besides the already described information, the above event notification in Figure 27 also give information about the zone from which pallet is transferring and to which zone it is transferred.

Once the KPI implementation component receive these event notifications about each operation, it extracts useful information from them. It calculates the respective KPI variables that are required for the evaluation of KPI formulas. In this thesis, almost five different variables are extracted from events received from the simulator in run time.

After the KPI variables are calculated from the events received from the FASTory simulator during the execution of the production order, the KPI implementation component updates the knowledge base with the values of these KPI variables along with the time stamp attached to them. The timestamp helps in retrieving the data from the RDF store on the basis of time for visualizing the KPIs. Each of these KPI variables is updated in the RDF store for each of the work unit in run time during the execution of the production order. The update is done via HTTP POST Update request to the RDF store from the KPI implementation component. The following JavaScript code in Figure 28 is used to send an HTTP POST request to update the data in RDF store.

```
var qs = require('qs');
var request = require('request');
var http = require("http");
var endpoint = 'http://localhost:3030/DS-1/sparql?'
var UpdateQuery = " PREFIX kpis:<http://www.semanticweb.org/KPI#>
PREFIX owl:<http://www.w3.org/2002/07/owl#>
insert
{[]
   kpis:Kpi_Variable ?VariableName;
   kpis:hasValue_ROB1 ?Production_Time;
   kpis:hasTimestamp ?time
}
where
{
   values (?VariableName ?Production_Time)
      {(kpis:Actual_Production_Time Working_Time)}
   bind (now() as ?time)
}";
var query = qs.stringify({ query: UpdateQuery });
request.get(endpoint + query + '&format=json', function(error, re-
sponse, body)
{
          if (!error && response.statusCode == 200)
                    {
               console.log(body);
          }
          else {
               console.log(error);
                         }
          });
```

Figure 28: *Update Query for KPI Variables*

Once the update of KPI variables is done in the RDF store, the next task for the KPI component is to extract all this data from the RDF store with a query. A query is sent in the HTTP GET request from the KPI implementation component to the RDF store that returns the data in the JSON format and is then formatted in proper data structure to execute the aforementioned five formulas of the KPIs and any other KPI created from the user. Besides JSON, data can be requested in other formats such as TSV and XML. The following code in Figure 29 is used to retrieve the data and then structured to be used to calculate the KPIs.

```
var qs = require('qs');
var request = require('request');
var http = require("http");
var endpoint = 'http://localhost:3030/DS-1/sparql?'
var GetDataQuery = " PREFIX kpis:<http://www.semanticweb.org/KPI#>
PREFIX owl:<http://www.w3.org/2002/07/owl#>
SELECT ?VariableName ?Production_Time ?time
{[]
    kpis:Kpi_Variable ?VariableName;
    kpis:hasValue_ROB1 ?Production_Time;
    kpis:hasTimestamp ?time.
FILTER(?VariableName=kpis:Actual_Production_Time)
}
order by ?time";

var query = qs.stringify({ query: GetDataQuery });

request.get(endpoint + query + '&format=json', function(error, re-
sponse, body)
          {
        if (!error && response.statusCode == 200)
                    {
              console.log(body);
          }
          else {
              console.log(error);
                        }
          });
```

Figure 29: *Query to retrieve the data of each KPI variables*

After the data for all the KPI variables is received from the RDF store, it is used in the calculation of KPIs with help of their formulas. The above-mentioned query in Figure 29, which is stored as variable *GetDataQuery* is executed every time there is a new update in the data due to events received from the FASTory simulator. Once the formulas are executed for each of the KPI, the result is sent to the user interface via socket.io for visualization in form of different visual graphs and charts. The value of these KPIs keep on updating in run time with new updates in the data coming from the FASTory simulator and so are the visual graphs on the front end.

Figure 30 explains the complete interaction of each component involve in the architecture as well as the sequence of these interactions. The process starts when the production manager places the order and the orchestrator starts executing the order on the FASTory Simulator till the users start visualizing the KPI. The user visualizes the graphs in run time as the production order is processed.

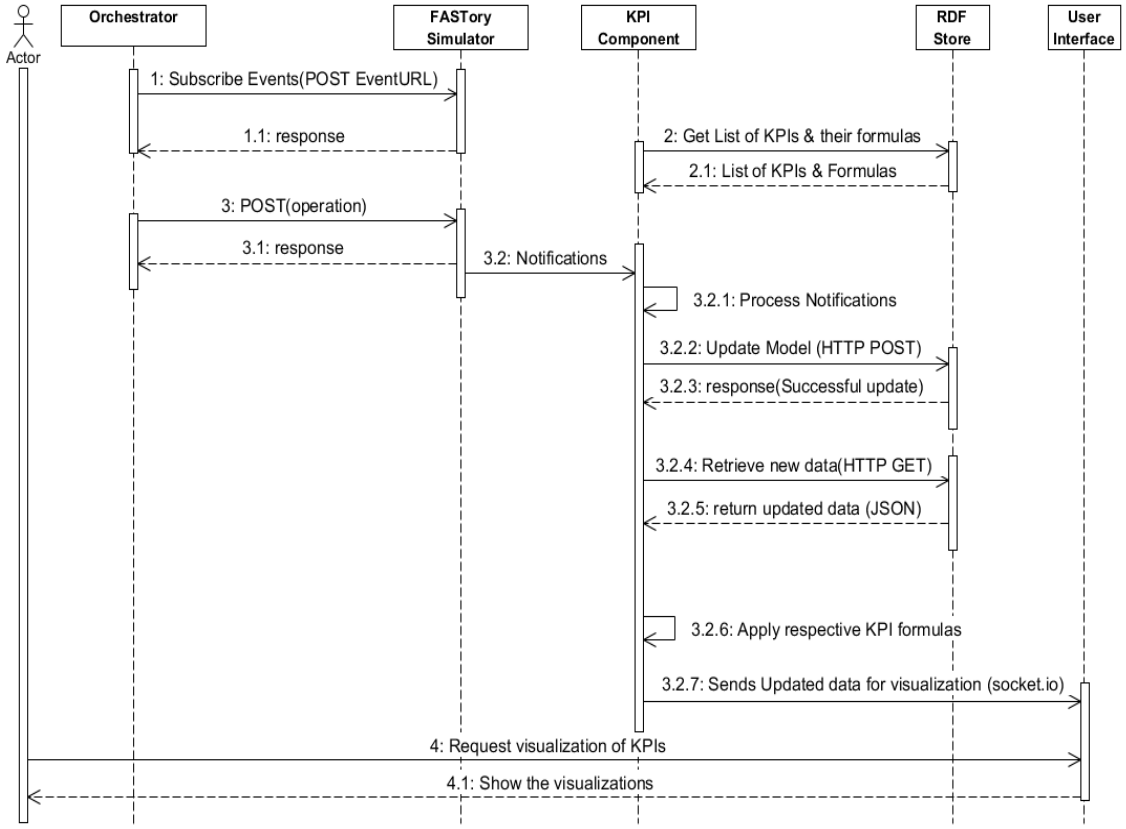

Figure 30: *Detailed Sequence Diagram showing different component interaction*

4.2 Create User defined KPIs

The next major objective of this thesis is to give user the option to create user defined KPI according to its need and importance. Manufacturing systems vary from each other in their structure and operations it performs, so the user may need some specific performance indicator that would be of great importance and will only be applicable to specific use case. Keeping in view this need of custom-made performance indicators that to monitor the performance manufacturing plant, this thesis work provides an opportunity to the user to create own KPI for the manufacturing plant.

In this thesis work, a form is made available on the user interface for the users that can be filled with all the specification a KPI should possess. Fields to fill in this form are obtained from the generic KPI description given in ISO 22400-2, also presented Table 2. Then, Figure 31 shows the form that the user will fill to make its own KPI.

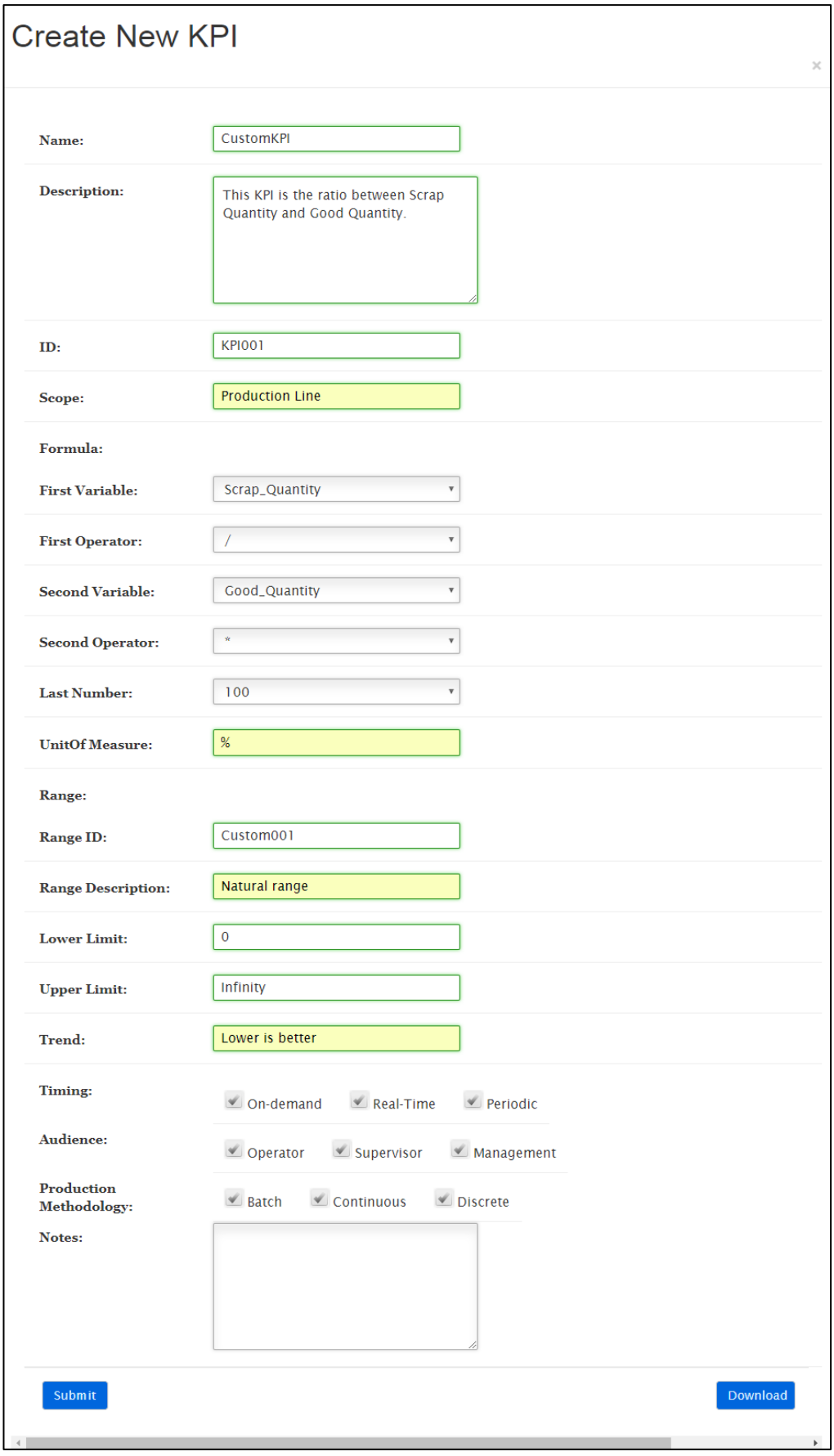

Figure 31: *Create user defined KPI Form*

Once the user fill and submit this form, an instance of this KPI is created under the KPI class in the RDF store. This new KPI instance contains all the data properties and object properties as specified in the form and the KPI implementation component start computing the values for this KPI as soon as it is added to the RDF store. Besides the addition to the RDF, an XML for this KPI is created based on KPIML schemas, which the user can download and save in XML file repository on its system. The KPI created in this thesis work for testing purpose has the following formula in equation (6). The customized KPI is the ratio between scrap quantity and good quantity. The unit of measure for this customized KPI is percentage and will be measured and visualized in run time with help of different visual graphs.

$$Customized\ KPI = \frac{Scrap\ Quantity}{Good\ Quantity} \tag{6}$$

However, this creates new KPI functionality comes with some limitations, which can be overcome in future research work in this field. The first limitation is the number of KPI variables available that the user can use. User can only use the KPI variables that are already available in the system, which restricts the range of creating KPIs out of the scope of the system. Moreover, the KPI created should only consist of two KPI variables that can be extended to three or more in future.

4.3 Use Case Scenario

The implemented approach is tested with the help of use case scenario. In this thesis work, two different production orders are executed and the results for them are monitored in run time. These two production orders have different specification such as different recipe, different color, and different products quantity. For this production order the data for five different KPI variables is gathered in run time. These five KPI variables are APT, AUBT, Scrap quantity, Good quantity, and Produced quantity. The APT and AUBT variables will be measured for each workstation separately whereas the others are measured for the whole production line. The APT is measured as the time difference between when robot start drawing and when robot stops drawing events. The AUBT is calculated as sum of the time when the robot is drawing and transferring pallet toward the drawing zone. The scrap quantity is calculated as the number of products that do not meet the quality criteria. The good quantity calculated as the number of products that are above minimum quality limit set. The produced quantity is measured as the total number of products that are produced. Afterwards, the data gathered from these use case scenarios is used in the formulas of the KPIs and visualized in the form of different visual graphics. Table 4 presents how each KPI variable is calculated during the process of production.

Table 4: *KPI variables and their calculation*

KPI Variables	**How they are calculated**
Actual production time (APT)	APT is the actual time in which the workstation is adding some value to the final production order. It excludes the time used for transportation of pallets across the workstation and queuing time. It is calculated as the time difference between 'Robot start drawing' and 'Robot stop drawing events.
Actual unit busy time (AUBT)	AUBT is the time when a workstation is busy. Besides the actual production time, it includes the time used for transportation of pallets across the workstation and queuing time. It includes the time when the robot is drawing as well as the time when the conveyor is transferring the pallet towards the drawing zone.
Produced Quantity	The produced quantity is the total amount of quantity produced until that moment in time. It is the sum of both good quantity and scrap quantity.
Good Quantity	The good quantity is considered as the quantity that meets the quality criteria. The quality criteria in this implementation is kept 80 percent. Any product above this quantity will be considered as good quantity.
Scrap Quantity	During execution of production order, material or quantity of product that cannot be reworked or used and has no value except for its material content. The scrap quantity is considered as the quantity that falls below the quality criteria. Any product below the 80 percent quality criteria is considered as scrap quantity.

5. RESULTS

This chapter presents the results achieved after implementing the methodology, tools and techniques proposed in the previous chapter and then running a sample order with help of an orchestrator. The results are presented in form of different graphs and charts obtained from the user interface designed. Moreover, the chapter discusses these results that how these can be interpreted.

5.1 KPI's Visualization

In order to test the implemented approach an orchestrator was designed that could run different production orders on the FASTory simulator. The orchestrator helps in orchestrating different services needs to be carried out on the product, such as loading pallets, loading paper, transferring pallets from one workstation to another for drawing the recipe set by the production manager and finally unloading the products. Two different production orders were executed on the FASTory simulator that have different specification such as different products quantity, different recipes and different colors. Following are the two production orders, which were used as an example to obtain the results.

Table 5: *Production Orders executed for obtaining results*

1st Order specifications	2nd Order specifications
{ "Order Quantity": 200, "Frame Recipe": 1, "Screen Recipe": 5, "Keyboard Recipe": 7, "Frame Color": "BLUE", "Screen Color": "RED", "Keyboard Color": "GREEN" }	{ "Order Quantity": 150, "Frame Recipe": 2, "Screen Recipe": 4, "Keyboard Recipe": 8, "Frame Color": "RED", "Screen Color": "GREEN", "Keyboard Color": "BLUE" }

The 'Planned Busy time' for execution of each one of these production order was set to be 60 minutes for each work unit. The execution of abovementioned production order yields the following results, which can be visualized in form of different visual graphs from Figure 32 to Figure 40. All these results were obtained towards the end of execution of production order, to compare both these results.

Table 6 presents the calculation obtained for the *Allocation efficiency* of all the used equipment. The AUBT for each equipment is shown in minutes. Robot 1 was busy for the 50.48 minutes of the total planned busy time of 60 minutes, that yields to 84.3% of

Allocation efficiency. Robot 2 and Robot 3 were busy for 45 and 1.74 minutes respectively during the execution of 1st production order. In the 2nd production order, Robot 1 remain busy for X amount of time which is less amount of time as compared to 1st order because of less number of products to make. Similarly, less time is consumed for Robot 2 and Robot 3 in 1st order as compared to 2nd order.

Table 6: *Allocation efficiency calculations for both the production orders*

Equipment	**1st order calculations for KPI Allocation efficiency**		**2nd order calculations for KPI Allocation efficiency**	
	AUBT (in minutes)	*Allocation efficiency*	*AUBT (in minutes)*	*Allocation efficiency*
Robot 1	50.48	84.3 %	37,02	61.7%
Robot 2	45	75 %	33.96	56.6%
Robot 3	1.74	2.9 %	1.14	1.9%

The values presented in table 4 can be visualized in form of pie chart in Figure 32 and Figure 33 for total *Allocation efficiency* of Robot 1 at the end of first order and second order respectively.

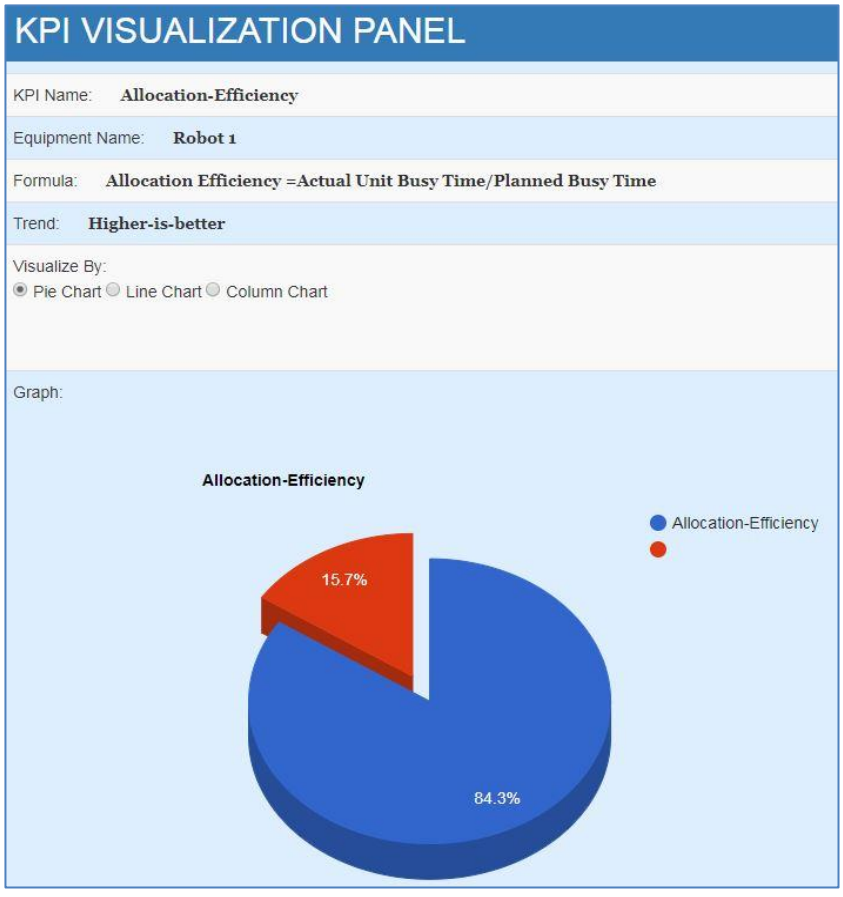

Figure 32: *Allocation Efficiency of Robot 1 for 1st production order*

Allocation efficiency indicates how well the tasks and workloads are divided among different resources to ensure smooth progress of the production orders and takes full advantage of economies of scale. It gives a measure of the planning done by the production and scheduling manager by indicating the amount of the total resources that is utilized and the remaining capacity of each resource that can still be used. Monitoring this KPI helps the production managers, to manage the production scheduling and allocating its resources a balance workload in order to increase the machines lifetime as well preventing excessive breakdowns and failures of robots and other machinery due to heavy workloads.

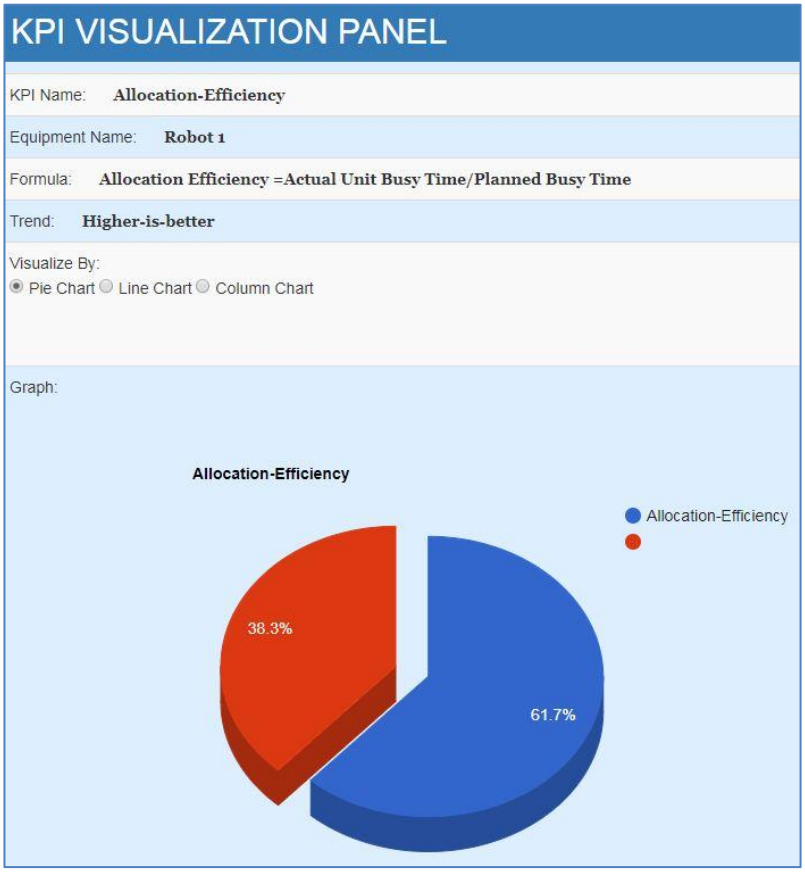

Figure 33: *Allocation Efficiency of Robot 1 for 2st production order*

Moreover, the general trend for the *Allocation efficiency* is, the higher its values are the better it is for production. It is calculated on-demand from the management, supervisor or operator. However, in this thesis it is calculated in runtime and can be visualized at any point of time during the execution of production order.

Table 7 presents the values gathered at the end of both production orders of APT and *Availability* KPI. The APT for Robot 1 is 40.56 minutes of the total planned busy time of 60 minutes that yields to 67.6% of *Availability*. Furthermore, the APT for Robot 2 and Robot 3 was 42.48 and 1.74 minutes respectively at the end of 1st production order. Due to less number of products in the 2nd order APT for all the robots used is less as compared to 1st production order.

Table 7: *Availability calculations for both the production orders*

Equipment	1st order calculations for KPI Availability		2nd order calculations for KPI Availability	
	APT (in minutes)	*Availability*	*APT (in minutes)*	*Availability*
Robot 1	40.56	67.6 %	30.72	51.2%
Robot 2	42.48	70.8 %	28.86	48.1%
Robot 3	1.5	2.5 %	1.02	1.7%

Figure 34 and Figure 38 illustrates the *Availability* KPI for Robot 2 after the execution of first and second production order in form of pie charts, respectively. The total production time used in the *Availability* formula excludes all the times a work unit is busy in transferring pallets and from one workstation to another workstation.

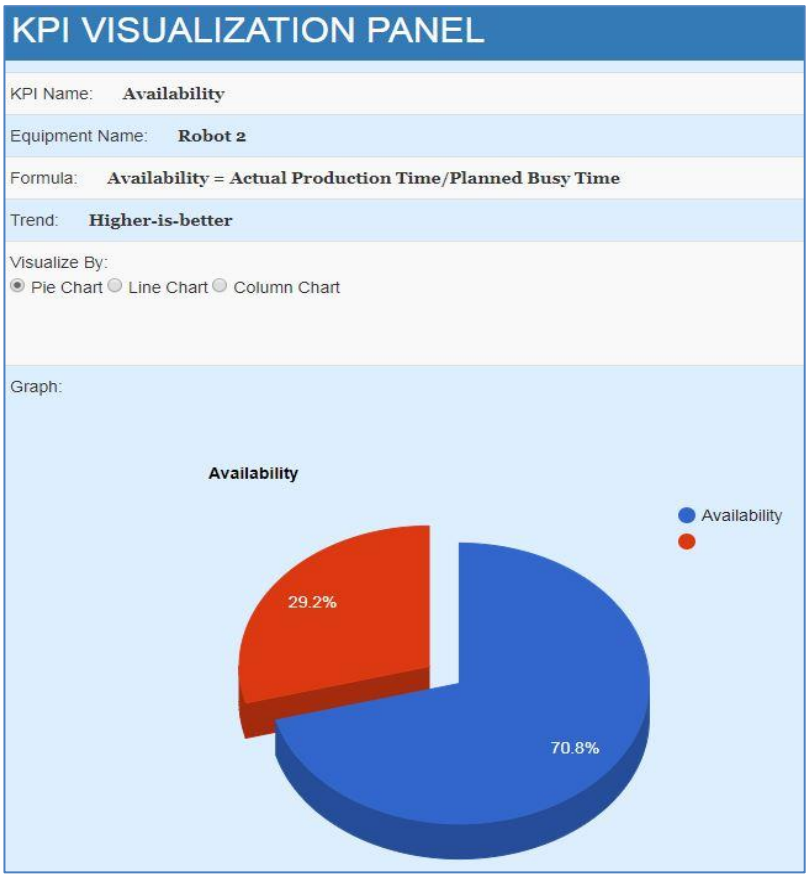

Figure 34: *Availability KPI of Robot 2 for 1st production order*

It can also be noted that the percentage of *Availability* KPI for Robot 1, 2 and 3 in both the order is less than the *Allocation Efficiency* in their respective counter parts. This is because of the facts that *Allocation efficiency* includes the time taken by a work unit in transferring and queuing in addition to the actual production time.

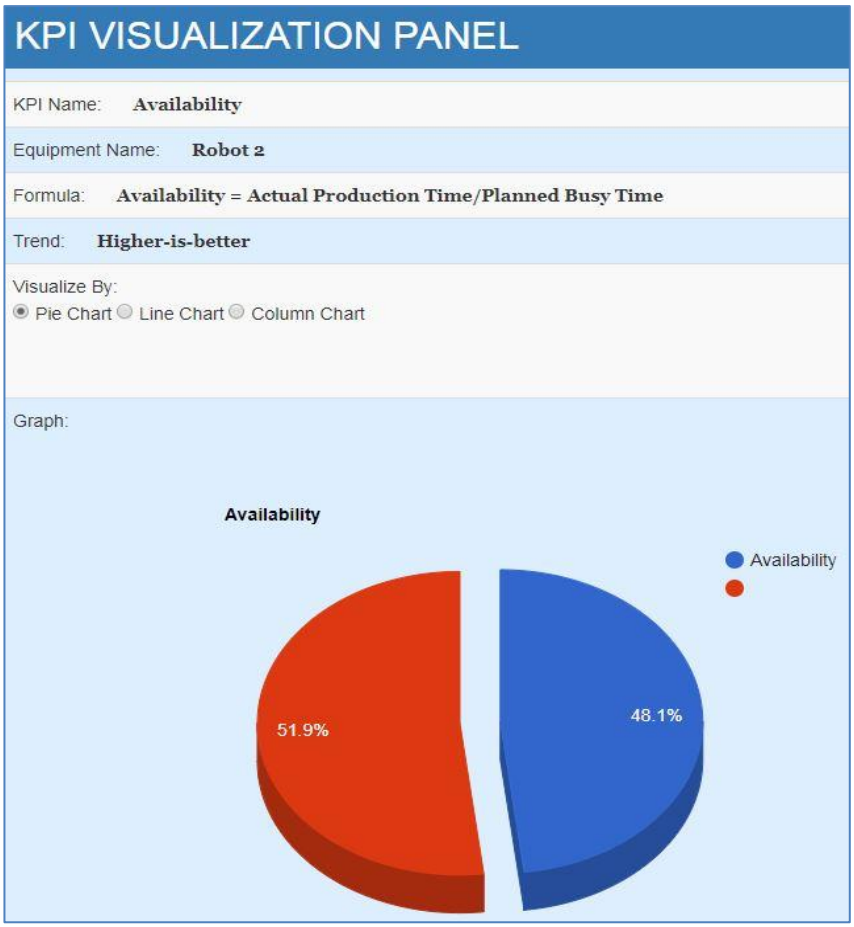

Figure 35: *Availability KPI of Robot 2 for 2nd production order*

Availability along with *Allocation efficiency* gives a very good view to the management and supervisors about the amount of productive time in the total busy time. It helps in better planning of scheduling and assigning tasks as well as orchestrating the production line in more optimized way. Just like Allocation efficiency, higher percentage of Availability is appreciated across the industry due to its direct proportionality with productivity of the work unit.

Table 8 shows the obtained for APT and AUBT along with the *Utilization efficiency* KPI for both the orders. The calculations are obtained for all the robots involved in execution of production orders. *Utilization efficiency* is a measure of total productivity of the work units and it is important in the industry because in market only the productive hours are paid thus the objective of the management or supervisor is to get higher values of utilization efficiency.

Table 8: *Utilization efficiency calculations for both the production orders*

Equipment	**1st order calculations for KPI Utilization Efficiency**			**2nd order calculations for KPI Utilization Efficiency**		
	APT (in minutes)	*AUBT (in minutes)*	*Utilization Efficiency*	*APT (in minutes)*	*AUBT (in minutes)*	*Utilization Efficiency*

Robot 1	40.56	50.48	80.2 %	30.72	37,02	82.9%
Robot 2	42.48	45	94.4 %	28.86	33.96	85%
Robot 3	1.5	1.74	83.6 %	1.02	1.14	85.1%

Figure 36 and Figure 37 represents the *Utilization efficiency* of both the production orders for Robot 1. The *Utilization efficiency* for Robot 1 in 1st order is 80.2% and 82.9% for the 2nd production order respectively. *Utilization efficiency* for both the orders is almost equivalent it is because of the fact that both theses production orders use the same orchestration plan, so the ratio between APT and AUBT almost remains the same. These percentages show that around 20% of the Actual unit busy time of workstation 1 is spent in transferring and queuing and the other 80% is spent in actual production of products in both the production orders.

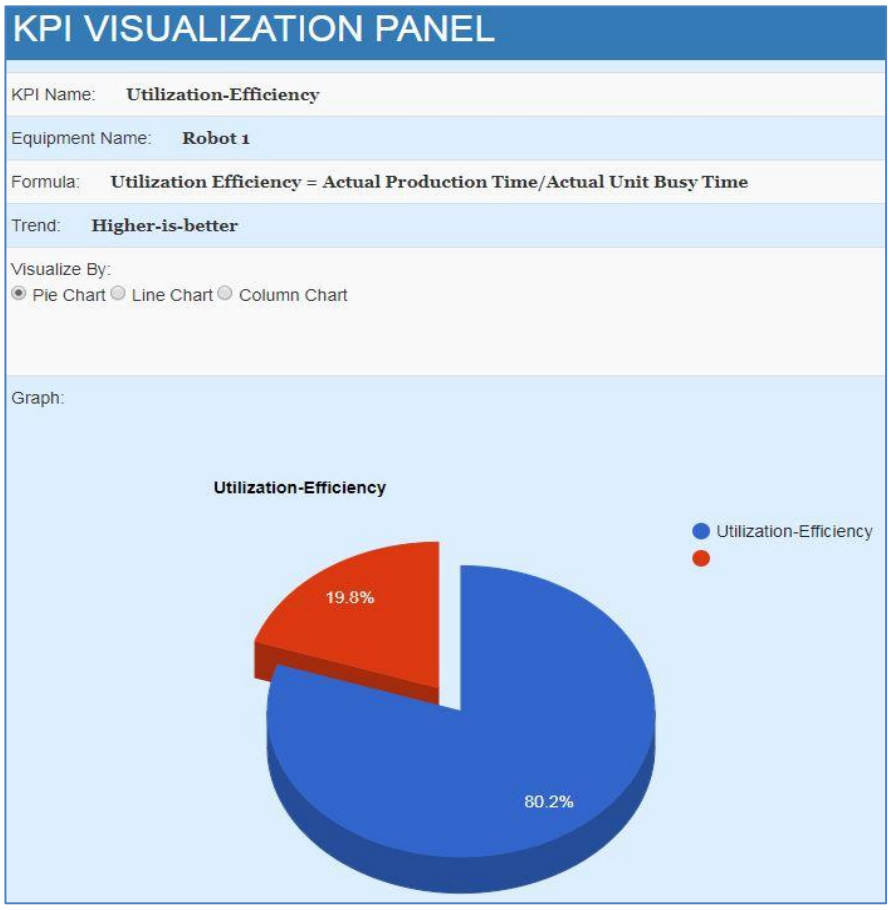

Figure 36: *Utilization Efficiency KPI of Robot 1 for 1st production order*

Utilization efficiency is a measure of total productivity of the work units and it is important in the industry because in market only the productive hours are paid thus the objective of the management or supervisor is to get higher values of *Utilization efficiency*.

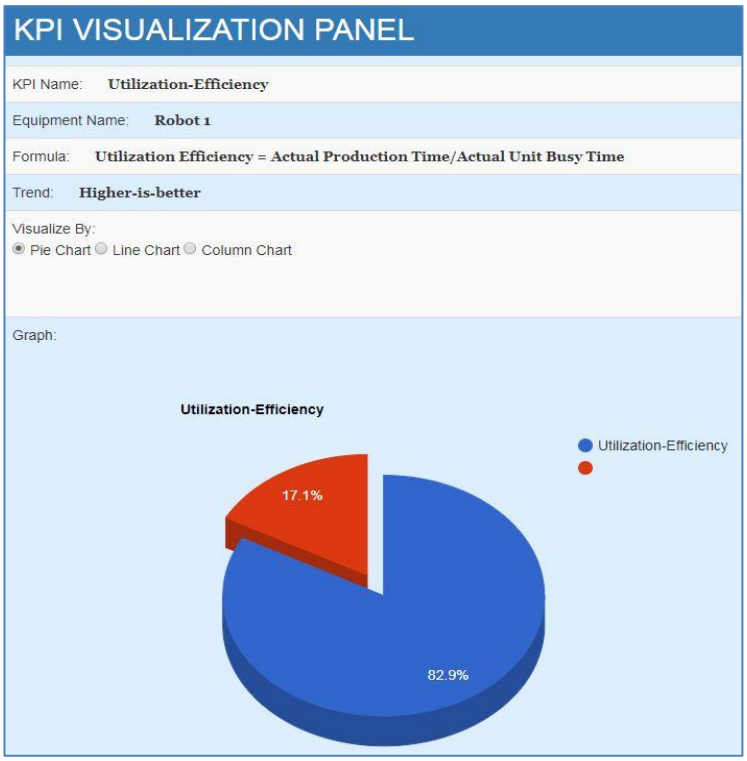

Figure 37: *Utilization Efficiency KPI of Robot 1 for 2nd production order*

Table 9 presents the number of good quantity and scrap quantity for both production orders. These quantities are further used in determining the quality ratio, scrap ratio as well as the values for the customized KPI. The 1st order consist of 200 products out of which 66 are scrapped while the other 134 meets the quality criteria of 80%. This results in to 67% of *Quality ratio* and 33% of *Scrap ratio*. The Customized KPI designed in this thesis is the ratio between scrap quantity and good quantity, as mentioned in the Implementation section. The value for Customized KPI in the 1st order is 49.2%, which shows that the number of scrap quantity is half of the good quantity.

Table 9: *Quality ratio, scrap ratio and customized KPI calculations for both the production orders*

KPI Variables and KPIs	**1st Production Order**	**2nd Production Order**
Good Quantity	134 products	93 products
Scrap Quantity	66 products	57 products
Produced Quantity	200 products	150 products
Quality Ratio	67 %	62%
Scrap Ratio	33 %	38%
Customized KPI	49.2 %	61.2%

In Figure 38 and Figure 39, the *Quality ratio* of both the production orders is shown at the end of execution. *Quality ratio* is very important in manufacturing and production industry as it directly affects the customers and their viewpoint about the products that is why it is one of very critical performance indicator for the management.

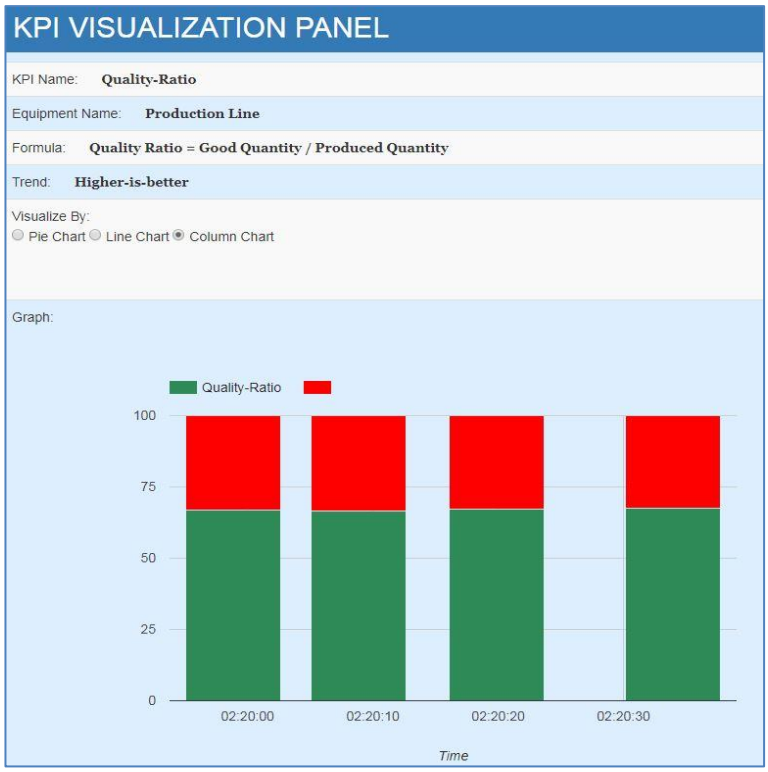

Figure 38: *Quality Ratio KPI for 1st production order*

Each bar in the Figure 38 and Figure 39 indicates the *Quality ratio* with respect to time and it is generated whenever a new product is produced and *Quality ratio* changes with it. Whenever a new product is produced, its quality criteria is checked and a new bar is generated showing the updated quality ratio for the products in the overall production order with respect to time.

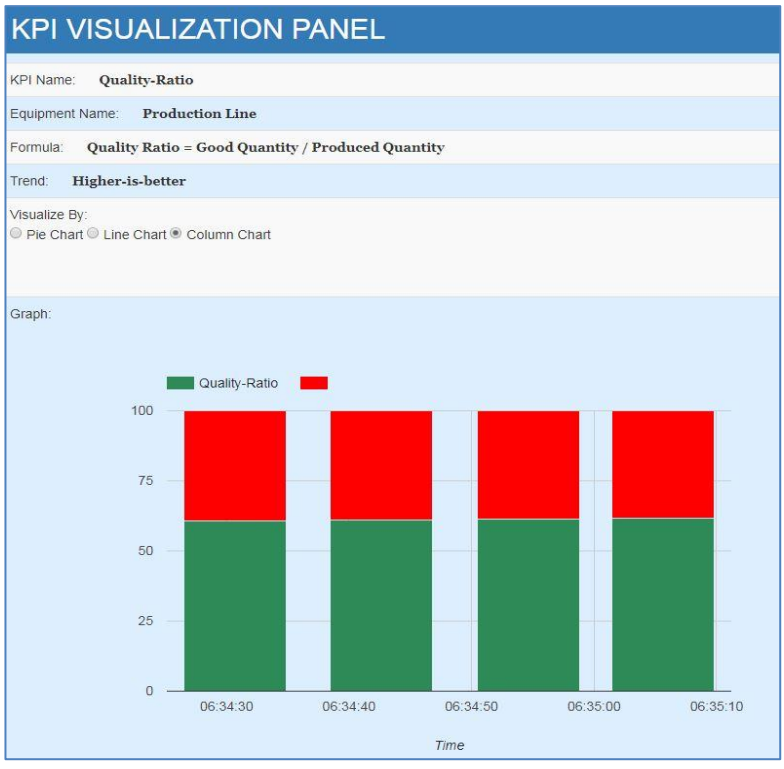

Figure 39: *Quality Ratio KPI for 2nd production order*

Similar to *Quality ratio* in manufacturing industry, special attention is given to monitor *Scrap ratio* as well. Figure 40 and Figure 41 provides a visualization of the scrap ratio in both the orders over a period of time. The *Scrap ratio* for the first production order is about 21% at the end of the order and 18% for the second order. These percentages indicate that this number of products failed to meet the quality criteria defined at the start of production shift. Moreover, the *Scrap ratio* also helps the management in identifying, if there is any faulty equipment or work unit in the production line if the graphs show very unusual behavior over a period of time. Besides it, also help in identifying the quantity of products which may needs a rework. Management appreciates lower value of *Scrap ratio* as it inversely relates to the quality of products.

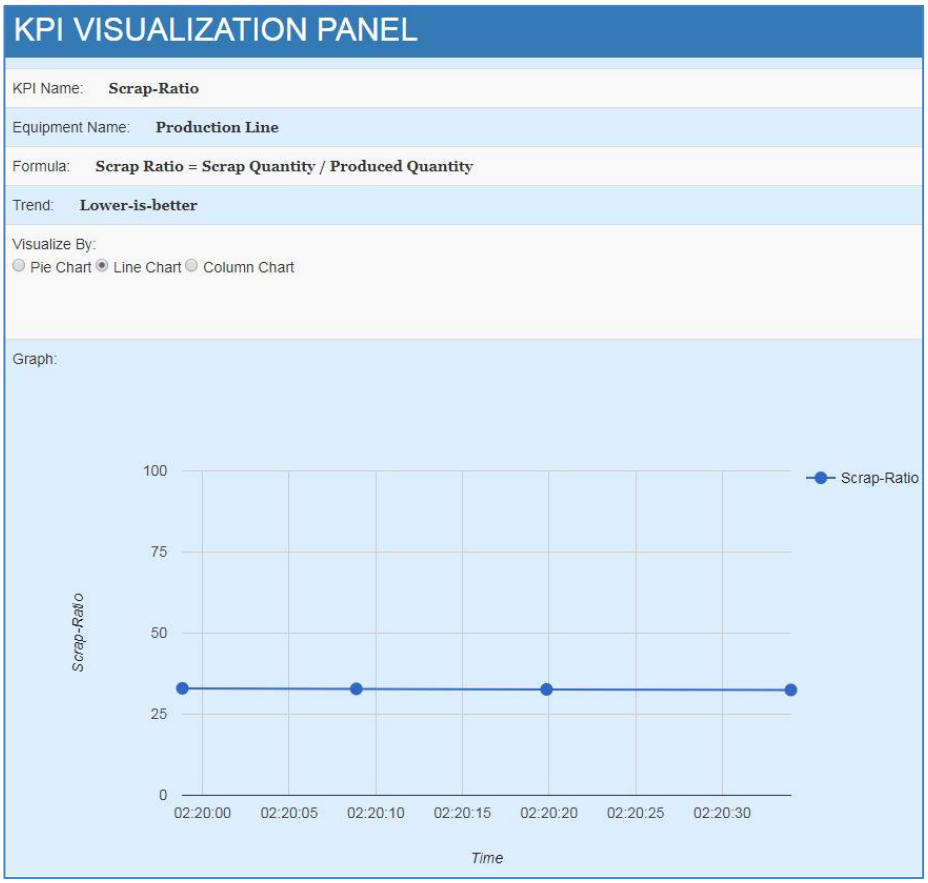

Figure 40: *Scrap Ratio KPI for 1st production order*

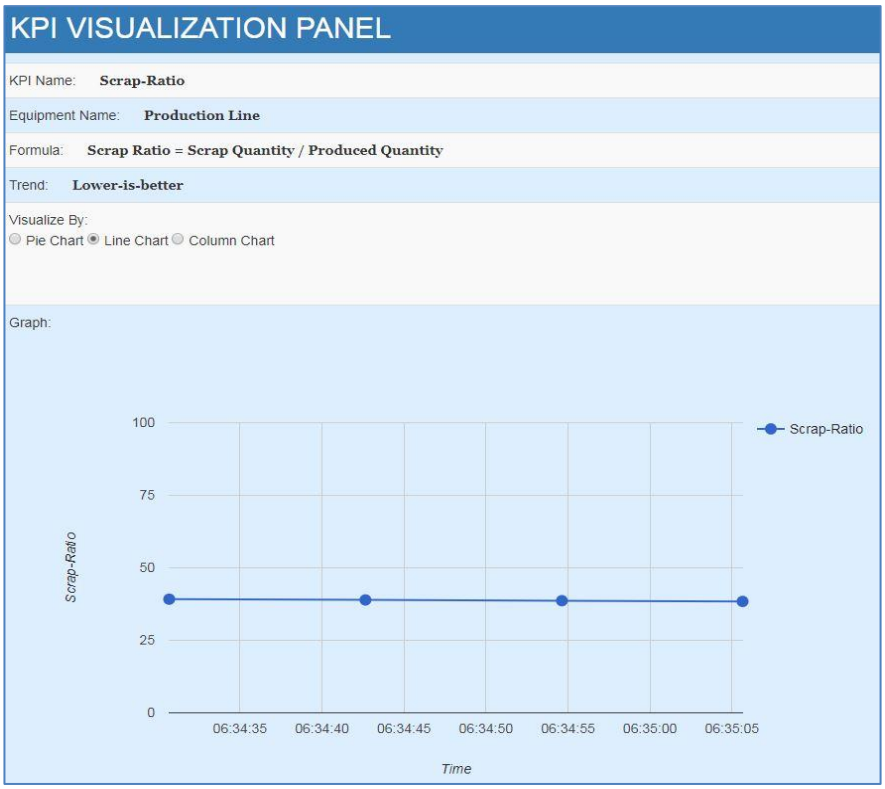

Figure 41: *Scrap Ratio KPI for 2nd production order*

Figure 42 shows the customized KPI for the 1st order that was created for testing the implemented functionality of user defined KPIs. As mentioned in the Implementation section that the customized KPI is the ratio between scrap quantity and good quantity.

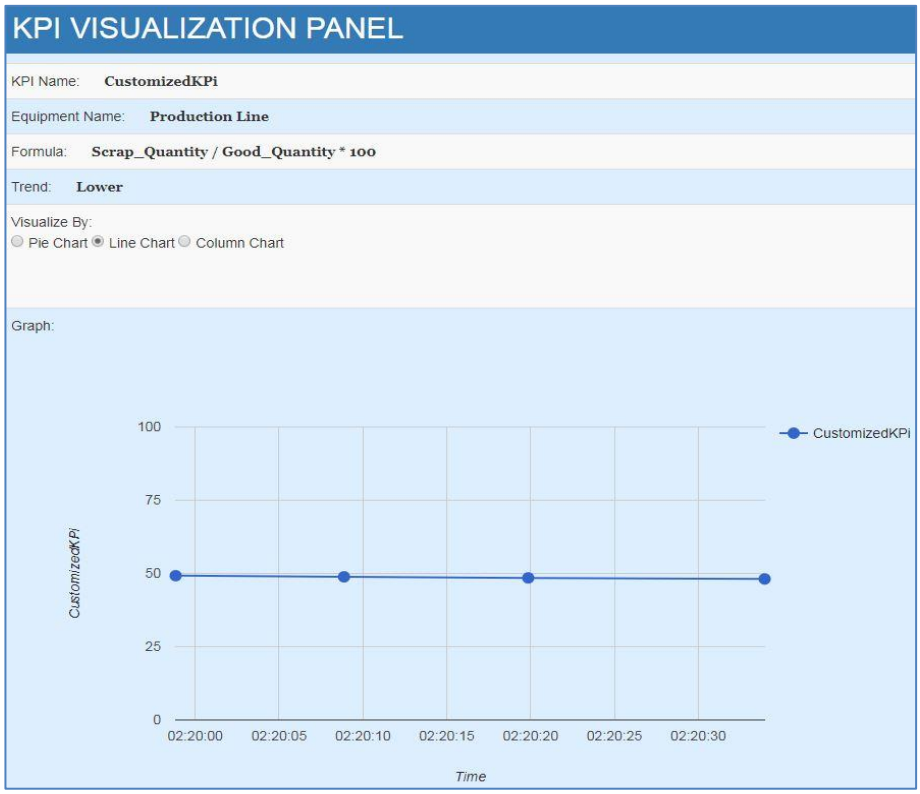

Figure 42: *Customized KPI for 1st production order*

6. CONCLUSIONS

This chapter concludes the work done during the course of this thesis and briefly discuss the objectives and results achieved by implementing the proposed methodology. Furthermore, it describes the future prospect in this research field along with challenges that still exist and needs to be solved related to the implementation of key performance indicators in manufacturing and production industry.

6.1.1 Summary of discussion

This thesis work implements the KPIs defined in ISO 22400 standard [24][25] taking help of the data models presented in KPIML on a multi-robot production line simulator. The author implemented the five major KPIs Availability, Allocation efficiency, utilization efficiency, Quality ratio and Scrap ratio on the test bed of FASTory simulator. The development phase includes the designing of a knowledge based by taking use of the ontology. The data models defined in KPIML for the KPIs were brought into use to design the ontology model within Protégé. The ontology model also serves as the knowledge base for storing data of different KPI variables in run time. In addition, an orchestration engine is designed that orchestrates the operations that take place on test bed. Different components interact with each other taking use of RESTful web services such as querying and updating data to the RDF store and receiving notifications from the test bed. Besides the implementation of KPIs, these KPIs are also visualized. Their values are rendered as charts on the user interface with the help of HTML, AngularJs, Bootstrap and different google charts. These charts include pie, line and column charts that are generated in runtime thus helping in monitoring the performance of the testbed.

Furthermore, the implementation also includes the option of creating customized KPI by just filling a form, which is available on the user interface in run time and will be able to visualize that newly made KPI instantly with the help of line chart. However, the presented solution is limited to only those KPIs that can be created with the help of available KPI variables. Moreover, another limitation is that the user created KPIs can only be visualized with the help of line chart for now.

In conclusion, this thesis provides one solution towards the implementation of ISO 22400 standards and KPIML in a factory environment. Moreover, it provides a generic framework for the implementation of standard KPIs in contrast to the previous research work done in this field, which rather just identify performance indicators by applying their own techniques and methodologies. In addition, this thesis not only proposed an

approach but also implements it on a test bed to validate the proposed methodology and generate results that can be visualized and analyzed.

6.1.2 Future Work

This thesis focuses more on implementing the KPIs and visualizing them in runtime. Only five of the 34 KPIs defined in ISO 22400 were implemented due to lack of data for several KPI variables in the FASTory simulator. Then, in future it can extended in a way that data for different KPI variables from varied data sources can be included in the knowledge base. Usually, companies have data in their legacy systems in form of excels sheets or other tables and JSON sources, an approach should be designed in order to add that data in form adds-on to the system and thus visualized in run time. Moreover, this thesis only focusses on production operation management and quality operation management related KPIs, other two important areas are inventory and maintenance related KPIs, which should be considered in future.

Furthermore, the future research should be more focused on designing a standalone tool that have ingredients for all the 34 KPIs defined in ISO 22400 standard that can be adaptable and extendable to any generic use case and should have the ability to encapsulate in to any production environment. In addition, a major focus should be given to enable users to create their own KPIs due to the fact that manufacturing and production industry vary drastically across the globe. Thus, it will be challenging to design one common solution; however, giving user the option to create its own KPI in accordance to its environment can solve this problem of largely different production environments.

REFERENCES

[1] S. Gershwin, “Manufacturing Systems Overview, HP Printer Case,” *Massachusetts Inst. Technol.*, 2016.

[2] R. Salzman, “Manufacturing System Design : Flexible Manufacturing Systems and Value Stream Mapping,” *Dr. Diss. Massachusetts Inst. Technol.*, 2002.

[3] G. Chryssolouris, Manufacturing Systems: Theory and Practice, 2nd ed. Springer-Verlag New York, 2006, ISBN – 978-0-387-28431-6.

[4] M. P. Groover, Automation, Production Systems, and Computer-Integrated Manufacturing, 3 edition. Upper Saddle River, N.J: Prentice Hall, 2007, ISBN – 978-0-13-239321-8.

[5] Professur für Prozessleittechnik, “Vorlesung Prozessrechen- und -leittechnik - ISA 95 & IEC/ISO 62264.” p. 32.

[6] D. Brandl, “Practical Applications of the ISA 95 standard,” 2012.

[7] D. Brandl, “What is ISA-95 ? Industrial Best Practices of Manufacturing Information Technologies with ISA-95 Models,” pp. 1–32, 2008.

[8] M. Lucke, “KPIs for Asset Management : A Pump Case Study,” 2016.

[9] B. Mehta and R. Y. Jaganmohan, *Industrial Process Automation Systems*, 1st Edition. Butterworth-Heinemann, 2014, ISBN – 9780128010983.

[10] A. I. Omer and M. . Taleb, “Architecture of Industrial Automation Systems,” *Eur. Sci. Journal, ESJ*, vol. 10, no. 3, pp. 273–283, 2014.

[11] S. Engell and I. Harjunkoski, “Optimal operation: Scheduling, advanced control and their integration,” *Comput. Chem. Eng.*, vol. 47, pp. 121–133, Dec. 2012.

[12] D. Brandl, “The IT Implications of ISA 95 and ISA 99,” pp. 1–12, 2005.

[13] “Enabling “Real World” SOA through the Microsoft Platform”, A Microsoft White Paper, Dec. 2006. Available at http://www.microsoft.com/biztalk/solutions/soa/whitepaper.mspx

[14] F. Jammes and H. Smit, “Service-oriented paradigms in industrial automation,” *IEEE Trans. Ind. Informatics*, vol. 1, no. 1, pp. 62–70, 2005.

[15] C. Popescu and J. L. M. Lastra, “Modeling breakdown handling for soa-based factory automation systems,” *Proc. IEEE Int. Conf. Ind. Technol.*, no. August, 2009.

[16] 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.

[17] “Service-Oriented Integration.” [Online]. Available: https://msdn.microsoft.com/en-us/library/ms978594.aspx. [Accessed: 10-Feb-2017].

[18] “Web Services Architecture.” [Online]. Available: https://www.w3.org/TR/ws-arch/. [Accessed: 06-Feb-2017].

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

[20] T. Fredrich, “RESTful Service Best Practices: recommendations for creating web services,” p. 40, 2013.

[21] P. Adamczyk, P. H. Smith, R. E. Johnson, and M. Hafiz, “REST: From Research to Practice,” 2011.

[22] A. W. Colombo, F. Jammes, H. Smit, R. Harrison, J. L. M. Lastra, and I. M. Delamer, “Service-oriented architectures for collaborative automation,” *IECON Proc. (Industrial Electron. Conf.*, pp. 2649–2654, 2005.

[23] Cesare Pautasso, “REST vs. SOAP: Making the Right Architectural Decision,” 2008.

[24] International Standard ISO 22400–1. Automation Systems and Integration – Key Performance Indicators (KPIs) for Manufacturing Operations Management - Part 1: Overview, Concepts and Terminology. Geneva: International Standard Organization (ISO), 2014.

[25] “International Standard ISO 22400–2. Automation Systems and Integration – Key Performance Indicators (KPIs) for Manufacturing Operations Management - Part 2: Definitions and descriptions.” Geneva: International Standard Organization (ISO), 2014.

[26] “Key Performance Indicator Markup Language, Version 01.” MESA International, May-2015.

[27] J. Horst, “A KPI Standard to Improve Process Performance,” MESA International , 2015.

[28] "The Basics Of HTML - W3C Wiki". W3.org. N.p., 2014. Web. 15 Feb. 2017.

[29] "CSS Basics - Web Education Community Group". W3.org. N.p., 2012. Web. 15 Feb. 2017.

[30] “What can you do with JavaScript - Web Education Community Group.” [Online]. Available: https://www.w3.org/community/webed/wiki/What_can_you_do_with_JavaScript. [Accessed: 18-Feb-2017].

[31] Foundation, “Node.js,” Node.js. [Online]. Available: https://nodejs.org/en/. [Accessed: 18-Feb-2017].

[32] H. G. Lemu, “Virtual engineering in design and manufacturing,” *Adv. Manuf.*, vol. 2, no. 4, pp. 289–294, 2014.

[33] Z. Liu, N. Suchold, and C. Diedrich, “Virtual Commissioning of Automated Systems,” *Automation*, pp. 131–148, 2012.

[34] "Fastory Simulator". [Online]. Available: Escop.rd.tut.fi. [Accessed: 18-Feb-2017].

[35] A. Rakar S. Zorzut and V. Jovan "Assesment of Production Performance by Means of KPI " Control 2004 pp. 6-9 2004.

[36] B. Zhang, C. Postelnicu, and J. L. M. Lastra, “Key Performance Indicators for energy efficient asset management in a factory automation testbed,” *Ind. Informatics (INDIN), 2012 10th IEEE Int. Conf.*, pp. 391–396, 2012..

[37] B. Ramis Ferrer, “An ontological approach for modelling configuration of factory-wide data integration systems based on IEC-61499,” 2013.

[38] F. Shuli, L. Wenling, Z. Xiankun, and L. Xin, “The Knowledge Description of Teaching Resource and Its Application,” Jun. 2010.

[39] N. F. Noy and D. L. McGuinness, *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.

[40] Gruber, T.; “Ontology, in the Encyclopedia of Database Systems”, Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag, 2009. Paper available online: http://tomgruber.org/writing/ontology-definition-2007.htm

[41] “FOLDOC (the Free On-Line Dictionary of Computing) - Computing Dictionary.” [Online]. Available: https://foldoc.org/. [Accessed: 20-Feb-2017].

[42] “SPARQL Query Language for RDF.” [Online]. Available: https://www.w3.org/TR/rdf-sparql-query/. [Accessed: 20-Feb-2017].

[43] “protégé.” [Online]. Available: https://protege.stanford.edu/. [Accessed: 20-Feb-2017].

[44] V. Jovan and S. Zorzut, “Use of Key Performance Indicators in Production Management,” in *2006 IEEE Conference on Cybernetics and Intelligent Systems*, 2006, pp. 1–6.

[45] "AngularJS — Superheroic JavaScript MVW Framework", *Angularjs.org*, 2017. [Online]. Available: https://angularjs.org/. [Accessed: 05- Feb- 2017].

[46] T. Gama and V. Cavenaghi, “Measuring performance and Lean Production: a review of literature and a proposal for a performance measurement system,” in *Proceedings of the Production and Operation Management Society (POMS) 20th Annual Conference*, 2009

[47] M. Georgoudakis, C. Alexakos, A. Kalogeras, J. Gialelis, and S. Koubias, “Decentralized Production control through ANSI / ISA-95 based ontology and agents,” in 2006 IEEE International Workshop on Factory Communication Systems, 2006, pp. 374–379.

[48] S. M. Kannan et al., “Towards Industry 4.0: Gap Analysis between Current Automotive MES and Industry Standards Using Model-Based Requirement Engineering,” in 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), 2017, pp. 29–35.

[49] J. Cottyn, H. V. Landeghem, K. Stockman, and S. Derammelaere, “The combined adoption of production it and strategic initiatives - Initial considerations for a Lean MES analysis,” in 2009 International Conference on Computers Industrial Engineering, 2009, pp. 1629–1634.

[50] M. I. Mahmoud, H. H. Ammar, M. M. Hamdy, and M. H. Eissa, “Production operation management using Manufacturing Execution Systems (MES),” in 2015 11th International Computer Engineering Conference (ICENCO), 2015, pp. 111–116.

[51] B. Ramis Ferrer, W. M. Mohammed, A. Lobov, A. Moreno Galera, and J. L. M. Lastra, “Including Hu-man Tasks as Semantic Resources in Manufacturing Ontology Models,” in IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society, 2017.

[52] M. Dastani, K. V. Hindriks, P. Novák, and N. A. Tinnemeier, “Combining Multiple Knowledge Representation Technologies into Agent Programming Languages.,” in DALT, 2008, pp. 60–74.

[53] D. Schachinger, W. Kastner, and S. Gaida, “Ontology-based abstraction layer for smart grid interaction in building energy management systems,” in 2016 IEEE International Energy Conference (ENERGYCON), 2016, pp. 1–6.

[54] S. Dutta, “Strategies for implementing knowledge-based systems,” IEEE Transactions on Engineering Management, vol. 44, no. 1, pp. 79–90, Feb. 1997.

[55] W. M. Mohammed, A. Lobov, B. R. Ferrer, S. Iarovyi, and J. L. M. Lastra, “A web-based simulator for a discrete manufacturing system,” in IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, 2016, pp. 6583–6589.

[56] R. Ramanathan and T. Korte, “Software service architecture to access weather data using RESTful web services,” in Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT), 2014, pp. 1–8.

[57] T. Cucinotta et al., “A Real-Time Service-Oriented Architecture for Industrial Automation,” IEEE Transactions on Industrial Informatics, vol. 5, no. 3, pp. 267–277, Aug. 2009.

[58] 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,” Proceedings of the IEEE, vol. 104, no. 5, pp. 1142–1154, May 2016.

APPENDIX A – KPIML FOR THE IMPLEMENTED KPI'S

This section present the XML generated for the KPIs implemented in this thesis on the basis of ISO 22400-2 standard and KPIML.

Quality Ratio

```
<?xml version="1.0" encoding="UTF-8"?>

<KPIDefinition
xsi:schemaLocation="http://www.mesa.org/xml/KPI-ML-V01 KPI-ML-V01.xsd"
xmlns="http://www.mesa.org/xml/KPI-ML-V01"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <ID> QR100 </ID>
 <Description> The quality ratio is the relationship between the good
quantity (GQ) and the produced quantity (PQ).</Description>
 <Name> Quality Ratio </Name>
 <Scope> Work unit, work centre, area, product, time period, product,
defect types </Scope>
 <Formula>
          Quality Ratio = Good Quantity / Produced Quantity
 </Formula>
 <UnitOfMeasure> % </UnitOfMeasure>
 <Range>
       <ID> Natural </ID>
       <Description> Natural Range </Description>
       <LowerLimit> 0 </LowerLimit>
       <UpperLimit>100 </UpperLimit>
 </Range>
 <Trend> Higher-is-better </Trend>
 <Timing> On-demand </Timing>
 <Timing> periodically </Timing>
 <Timing> Real-time </Timing>
 <Audience> Operator </Audience>
 <Audience> Supervisor </Audience>
 <Audience> Management </Audience>
 <ProductionMethodology> Batch </ProductionMethodology>
 <ProductionMethodology> Continuous </ProductionMethodology>
 <ProductionMethodology> Discrete </ProductionMethodology>
 <Notes> "This indicator is usable as real-time indicator for the operator
level." </Notes>
</KPIDefinition>
```

Allocation Efficiency

```
<?xml version="1.0" encoding="UTF-8"?>

<KPIDefinition
xsi:schemaLocation="http://www.mesa.org/xml/KPI-ML-V01 KPI-ML-V01.xsd"
xmlns="http://www.mesa.org/xml/KPI-ML-V01"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <ID>AE100</ID>
 <Description>  The allocation efficiency is the ratio bet ween the
actual allocation time of a work unit expressed as the actual unit busy
time (AUBT) and the planned time for allocating the work unit expressed
as the planned unit busy time (PBT). </Description>
 <Name> Allocation Efficiency </Name>
 <Scope> Product, production order, and work unit </Scope>
 <Formula> Allocation Efficiency= Actual Unit Busy Time/Planned Busy Time
</Formula>
 <UnitOfMeasure> % </UnitOfMeasure>
 <Range>
        <ID> Natural </ID>
        <Description> Natural Range </Description>
        <LowerLimit> 0 </LowerLimit>
        <UpperLimit> 100 </UpperLimit>
 </Range>
 <Trend> Higher-is-better </Trend>
 <Timing> On-demand </Timing>
 <Audience> Operator </Audience>
 <Audience> Supervisor </Audience>
 <Audience> Management </Audience>
 <ProductionMethodology> Discrete </ProductionMethodology>
 <ProductionMethodology> Batch </ProductionMethodology>
 <ProductionMethodology> Continuous </ProductionMethodology>
 <Notes> “The allocation efficiency indicates how strongly the planned
capacity of the work unit is already used and how much planned capacity
is still available.
The allocation efficiency is only affected by the actual unit idle time
while the availability KPI takes the actual unit delay time into
account.” </Notes>
</KPIDefinition>
```

Scrap Ratio

```
<?xml version="1.0" encoding="UTF-8"?>

<KPIDefinition
xsi:schemaLocation="http://www.mesa.org/xml/KPI-ML-V01 KPI-ML-V01.xsd"
xmlns="http://www.mesa.org/xml/KPI-ML-V01"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <ID>SR100</ID>
 <Description> The Scrap ratio is the relationship between the Scrap
quantity (SQ) and the produced quantity (PQ). </Description>
 <Name> Scrap Ratio </Name>
 <Scope> Work unit, work centre, area, product, time period, product,
defect types </Scope>
 <Formula> Scrap Ratio = Scrap Quantity / Produced Quantity </Formula>
 <UnitOfMeasure> % </UnitOfMeasure>
 <Range>
        <ID> Natural </ID>
        <Description> Natural Range </Description>
        <LowerLimit> 0 </LowerLimit>
        <UpperLimit> 100 </UpperLimit>
 </Range>
 <Trend> Lower-is-better </Trend>
 <Timing> On-demand </Timing>
 <Timing> periodically </Timing>
 <Timing> Real-time </Timing>
 <Audience> Operator </Audience>
 <Audience> Supervisor </Audience>
 <Audience> Management </Audience>
 <ProductionMethodology> Batch </ProductionMethodology>
 <ProductionMethodology> Continuous </ProductionMethodology>
 <ProductionMethodology> Discrete </ProductionMethodology>
 <Notes> “This indicator is usable as real-time indicator for the
operator level.” </Notes>
</KPIDefinition>
```

Utilization Efficiency

```
<?xml version="1.0" encoding="UTF-8"?>

<KPIDefinition
xsi:schemaLocation="http://www.mesa.org/xml/KPI-ML-V01 KPI-ML-V01.xsd"
xmlns="http://www.mesa.org/xml/KPI-ML-V01"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <ID>UE100</ID>
 <Description> The utilization efficiency is the ratio bet ween the
actual production time (APT) and the actual unit busy time (AUBT).
</Description>
 <Name> Utilization Efficiency </Name>
 <Scope> Work unit </Scope>
 <Formula> Utilization Efficiency = Actual Production Time/Actual Unit
Busy Time </Formula>
 <UnitOfMeasure> % </UnitOfMeasure>
 <Range>
        <ID> Natural </ID>
        <Description> Natural Range </Description>
        <LowerLimit> 0 </LowerLimit>
        <UpperLimit> 100 </UpperLimit>
 </Range>
 <Trend> Higher-is-better</Trend>
 <Timing> On-demand </Timing>
 <Timing> Real-time </Timing>
 <Audience> Operator </Audience>
 <Audience> Supervisor </Audience>
 <Audience> Management </Audience>
 <ProductionMethodology> Batch </ProductionMethodology>
 <ProductionMethodology> Continuous </ProductionMethodology>
 <ProductionMethodology> Discrete </ProductionMethodology>
 <Notes> “This indicator identifies the productivity of work units.
Because only the production time affects an added value which will be
paid by the market, the goal should be to get a high indicator value.
</Notes>
</KPIDefinition>
```

APPENDIX B – FASTORY SIMULATOR EVENTS

Table 10: *Events available in FASTory simulator*

Events	**Description**
Pallet Loaded/Unloaded	This notification is received whenever a new Pallet is loaded or any pallet is unloaded from the line.
Paper Loaded/Unloaded	This notification is received whenever a new Pallet is loaded or any pallet is unloaded from the line.
Robot Start/Stop Drawing	This notification is received whenever any of the Robot starts or stops drawing.
Conveyor Start/Stop transferring	This notification is received whenever any of the conveyor starts or stops transferring a pallet. It contains the zone numbers of the workstations from which it is transferring and to which it is transferring.
Pen changed	This notification is received whenever any of the Robot changes the pen it is using (Red, Blue & Green).
LowInkLevel	When the level of ink reaches a low level, the *LowInkLevel* event occurs..
OutOfInk	Whenever the ink in the robot pen finishes, the *OutOfInk* event occurs.
Z1_Changed	Whenever there is a change in pallet Id on zone 1 changes this event occur.
Z2_Changed	Whenever there is a change in pallet Id on zone 1 changes this event occur.
Z3_Changed	Whenever there is a change in pallet Id on zone 1 changes this event occur.
Z4_Changed	Whenever there is a change in pallet Id on zone 1 changes this event occur.