TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Ahmed Rabee Sadik

Design and Implementation of an Expert System for Monitoring and Management of Web-Based Industrial Applications Master of Science Thesis

Examiner: Professor José L. Martinez Lastra
Examiner and topic approved in the Automation, Mechanical and Materials Engineering Faculty Council meeting on
5 December 2013

"Life Is So Mysterious, We Do Not Know When,
Why Or Where We Begin Or End It. But We
Do Know We Need To Struggle To Keep
It Goes Forward"

Abstract

Tampere University Of Technology
Master Degree Programme in Machine Automation
Sadik, Ahmed: Design and Implementation of an Expert System for Monitoring and Management of Web-Based Industrial Applications
Master of Science Thesis: 91 pages, 21 appendices pages
Tampere – Finland – April 2013
Major: Factory Automation
Examiner: Prof. Jose L.Martinez Lastra

Keywords: Industrial Information Technology – Artificial Intelligence – Expert System – Web-Services – WSDL – SOAP – Service Oriented Architecture – Drools Rule Engine – Web Based SCADA – Monitoring System– Complex Event Processing – ANSI/ISA95 – Industrial System Integration – CAMX IPC-2541 – Message Oriented Architecture – E-Business – Industrial Management

Human is an intelligent creature – intelligent in design and behaviour. From the human first second on the earth, he is trying to collect the knowledge and use it for surviving and extending his own kind. Human knowledge collection is all based on his observations and discovering for his own environment, the information with time turns to be the human experience which is the main sort of his intelligence of dealing with different situations.

Expert system is one branch of artificial intelligence science which the human inspired from his own being. Human always tries to inherit his own experiences to the next generations. But with the vast wide spreading of the information in the present century, a new need imposed itself to emulate the human experience and behaviour in a similar way; from this point expert computer systems have been invented.

Expert system is mainly transforming the human experiences into software forms. To act in a similar manner the human behaves. The expert system is always collecting a huge amount of information from its domain, and transform them to knowledge, using those rules the human assigned based on his own experiences [1].

In industry we try to apply the same concept to have intelligent automated system, but for this purpose; all the information should be in an easy form of industrial language and follow a reliable industrial protocol to communicate in an efficient way.

As the internet is the main source of the data on our planet currently, it was so convenient to structure all the industrial data in same language the internet use and follow similar communication protocols. From industrial point of view a web based monitoring systems [2] – should be the base of information for the mentioned expert system.

During this master thesis we achieve this goal, by dividing the problem into two main sub problems.

The first part is to implement a web based monitoring system on PLC controlled production line made by FESTO and used for teaching purpose in TUT - Tampere University of Technology – FASTory lab facilities.

The second part is to design and implement a convenient industrial expert system to process this web based monitored information for managing from the business point of view.

Preface

The actual working in this thesis started in June 2012, the thesis work has been done at Tampere University Technology – production department – FASTory lab and it was funded as one of the packages of PLANT Cockpit project, under the direction of Prof. Jose L.Martinez Lastra.

I have been always interested in Artificial Intelligence, and I am very happy that my thesis work was related to it. Actually, the thesis work was the inspiring force to pushing to start my own business for developing a social network based on expert system core. The project aims are individuals' self-improvement, open source innovation and strategic management for the big enterprises, A Beta version of the platform has been already released under the domain Novogenie.com, however still the final product under developing.

I would love to thank all the members in FASTory lab, who supported me with help and effort for finishing my thesis, I would love specially to thank my teamwork Johannes Minor, Jorge Gracia, Luis Gonzalez and Borja Ramis, also I would love also to thank Prof. Jose L.Martinez Lastra for his support all the time I studied in Machine Automation master program.

I would love to mention that I really enjoyed the experience of studying and working in multinational environment. I got a lot of new and different talents from living and understanding the different cultures, either they are similar or different from my original culture.

Finally I would love to send my greeting and respect for my own family and my home land Egypt.

In Tampere, April 3rd, 2013
Ahmed Sadik

Table of Contents

List of Figures

List of Abbreviations

A.I.	Artificial intelligence
A2A	Application to application
ANSI	American National Standards Institute
API	application programming interface
AS/RS	Automatic Storage and Retrieval System
BRMS	Business Rule Management System
CAMX	Computer Aided Manufacturing using XML
ES	Expert System
ESB	Enterprise Service Bus
FIFO	First In First Out
H2A	Human to application
HTML	HyperText Markup Language
HTTP	Hyper Text Transfer Protocol
I/O	Input/Output
ICD	Industrial Control Domain
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISA	Industry Standard Architecture
JBI	Java Business Integration
JTEC	Japanese Technology Evaluation Centre
KPI	Key Product Indicator
LAN	Local Area Network
MathML	Mathematical Markup Language
MES	Manufacturing Execution Systems
PLC	Programmable Logic Controller
RPC	Remote Procedure Call
RSS	Really Simple Syndication
SCADA	Supervisory Control and Data Acquisition
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
URL	Universal Resource Locator
W3C	World Wide Web Consortium
WSDL	Web-Service Description Language
WWW	World Wide Web
XML	Extensible Markup Language

CH1 – INTRODUCTION

Background

Automation nowadays is one key of success of every industry or firm. However automation layer is linked to other layers within the same firm such as manufacturing operations management and business planning and logistics. This means all the layers should be well bind together from the information point of view, to enable everyone in each layer to obtain the amount of information needed in the right format. Therefore, a flexible solution and model has been developed to define a standard of integration the manufacturing and business layer within the same project. This Standard has been known as ISA-95 [3], the standard schematic levels and tasks can be summarized in Figure 1

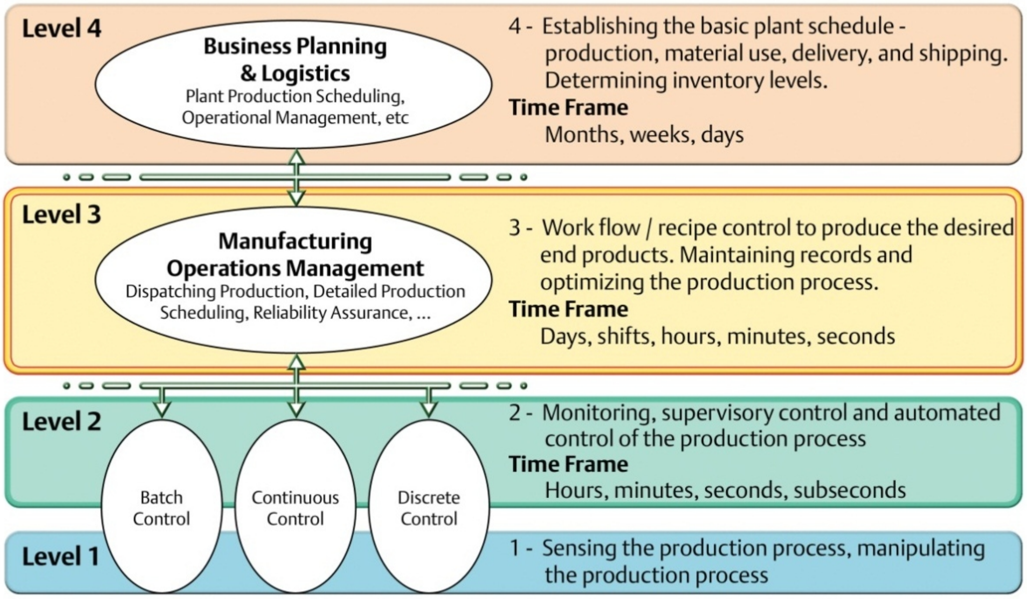

Level 4
Business Planning & Logistics
Plant Production Scheduling, Operational Management, etc
4 - Establishing the basic plant schedule - production, material use, delivery, and shipping. Determining inventory levels.
Time Frame
Months, weeks, days
Level 3
Manufacturing Operations Management
Dispatching Production, Detailed Production Scheduling, Reliability Assurance, ...
3 - Work flow / recipe control to produce the desired end products. Maintaining records and optimizing the production process.
Time Frame
Days, shifts, hours, minutes, seconds
Level 2
Batch Control
Continuous Control
Discrete Control
2 - Monitoring, supervisory control and automated control of the production process
Time Frame
Hours, minutes, seconds, subseconds
Level 1
1 - Sensing the production process, manipulating the production process

Figure 1 – ISA-95 different levels Hierarchy

In this model it is obvious that every layer has different needs from the kind of required information from the shop floor i.e., level 1, also the time frame is different due to every layer responsibilities – ISA-95 proposed XML language as a standard language to exchange the information from sensory and automation level to the business planning and logistics [4].

XML is different from other process automation industrial languages such like Modbus, Fieldbus, Profibus, Hart and others [5] as in those protocols the messages are more or less a serious of bits, which every bit has a certain meaning due to the protocol definition, however XML is a textual structured presentation of a certain domain as it will be clarified later in more details in chapter 3 of the thesis. XML not only gives the advantage of integration of different enterprises of the project but also to get access to the project from the WWW – World Wide Web – or in other words the internet over the regular Web browsers as XML format can be easily displayed as HTML format.

From this prospective, a new need to evolve the old automation systems to a web based automated system, process all the information obtained with intelligent approach to monitor /control them.

During this thesis work, we try to apply the mentioned concept on a fully automated production line which will be described in details in next chapter of this thesis

Industrial implementation of the thesis work

The thesis work has been done as a validation case study for a PLANT Cockpit project [6] which launched by the European commission as a part of (Factories of the Future) partnership,

The partners of the project are belonging to well-known five industrial firms which are (Acciona, BMW Group, Comau, Doehler Group, Intel) working with highly qualified research team from many four different European universities which are Ècole Polytechnique Fédérale de Lausanne, Politecnico di Milano, Tampere University of Technology, Technische Universität Dresden) plus one research centre (Tecnalia).

The following points summarize the overall objectives of the project

- Real-time visibility extending from the shop-floor level to the business level
- Continuous monitoring and control of material flow and resource utilization
- Continuous increase of operational excellence
- The avoidance of shortage situations (e.g. material, staff)
- The identification of abnormal situations (e.g. rising energy use indicating a problem)
- Identification and quantification of bottlenecks and optimization potentials (e.g. energy use)
- Multi-objective decision making,
- Fast re-configurability of production processes due to unplanned events or market demands
- Seamless communication between all stakeholders in the process (e.g. production manager, foremen...)

Problem statement

The most already existing infra-structure automation systems are controlled by the mean of Programmable logic controllers or any other similar controllers, which are the main elements of automation level in every factory or firm.

However the new technological requirements push us for searching for a new methods, to integrate this existing first layer of the shop floor to second layer of ISA-95 model, using the latest technology for factory information system, which in our case is a Web-Service technology. Designing a monitoring system based on a XML related standard is the first problem part we are concerning about in this thesis work.

Furthermore, the second problem part is located in the third layer of ISA-95.As we are trying to process the XML messages we already generated in the second layer as a Web-Service messages, to manage and analysis them with a rule engine which is containing all our experience and knowledge about our case study and able of executing a certain reasoning.

Proposed solution and thesis work

Due to ISA-95 description for the second and the third layer in its model

For the first part of the problem, the proposed solution was to select a PLC automated system, then add a new Web-Service device for every PLC or controller, to acquire all the sensors events and the controller actions and reformat them into XML messages using Web-Service technology.

A FESTO production line has been selected to be our case study for the problem we try to solve during the thesis work. Every station of the line is controlled by SIMENS step 7 PLC which is one of the most common industrial controllers in all the factories.

An INICO S1000 controller has been selected to be the Web-Service standalone controller, to use it as our hardware shell, enables us to design and implement our Web-Service solution within every FESTO station.

Following a Web-Service standards to design required appropriate services for our case study. Design our events types based on an analogy with CAMX IPC-2541 standard which is defining a generic XML schemas for electronic manufacturing, thus we can monitor all the events we can get from our Web-Service XML message over the internet or any network.

The other part of the proposed solution, is to create an expert system using Drools software as a rule engine to process all the XML events acquired from the low automation level. Imply new knowledge and store them based on the knowledge basement of our system. Convert them to key product indicators – KPIs – to enable the people who work in management and business level to give them useful information from their prospective about what is going on in the shop floor.

Preliminary design concepts

Some main concepts design have been into consideration, the whole implementation was following those concepts as pillars of design.

- The first design consideration is to follow industrial automation system architecture standard, to form the overall frame of work during the thesis.

- The second concept, is to find a convenient way to upgrade the old PLC automation technology – which already exists in most of the industrial premises – the way should be non-destructive.

- The third concept, all the work cannot stop during installation and programming the new technology hardware/software. In another words, no time wasting from the production should be into consideration.

- The forth concept, is that the new technology should be as generic as possible to fit the case study and all the similar cases. In same time, the design itself should follow a certain known reliable standard.

- The fifth concept, is to use smart system for monitoring and managing the application, separate the level of the technical details from business and production information during the monitoring.

- The last concept, is to reduce the cost as much as possible for applying the new technology.

The state of art - Literature Survey

Rathwell, Gary – IEEE 2006 "**ISA-95- Setting the Stage for Integration of MES and ICD Systems".** This paper discusses the developing of other standard like ISA88 to ISA95. In addition to, the reasons for its existence. Clarify the different layers of the model, how to use it, and the advantages of using it. Finally, gives a good real case study of using the recommendation of ANSI/ISA-95 for integrating the industrial enterprise, to provide an overall master planning approach for ICD (industrial control domain) and MES (manufacturing execution systems) [7].

Doug Tidwell - James Snell - Pavel Kulchenko "**Programming Web-Services with SOAP**" First edition - December 2001. This book discusses in details, providing practical examples about the technical side for the Web-Services, how to create a SOAP Web-Service and exchange the SOAP messages. Implementation of UDDI to write a dynamic Web-Service, how to secure your web page. Finally the future of the Web-Service applications from the point of view of the authors.

Ahmad Yasseen Al-Obaidy Master thesis – 2006 – computer science "**Design and Implementation of Web based SCADA System**". This master thesis gives a detailed description for the SCADA system components and implementation. The new concept of merging the old SCADA technology to communicate over the Web-Service technology.

Edward Feigenbaum Chair - Peter E. Friedland - Bruce B. Johnson - H. Penny Nii -Herbert Schorr - Howard Shrobe – First edition – May 1993 "**KNOWLEDGE-BASED SYSTEMS IN JAPAN**" This book produced by JTEC (Japanese Technology Evaluation Centre). Starting by the history of the artificial intelligence and the expert system as a branch of it. Then the book is expanding to show how to build an expert system in details. After going through the theoretical part of the expert system, it shows a really successful studying case in Japan of applying an expert system to manage the business process.

Goran Simica, Vladan Devedz – Expert Systems with application, Elsevier Science – 2003 "**Building an intelligent system using modern Internet technologies".** [8] The paper gives detailed case study for implementing a Java based expert system shell with Apache JServ Java package. To support the dynamic interoperation the Web-Service technologies over the internet. The case study was concerning the radio frequency field. The final expert system form is working as a code tutor to give the user recommendations to the learners who are interested in this field.

RJ St Jacques - 2008 – white papers **"XESS the XML Expert Shell".** [8] This white paper explains and compares in details the difference between the different ES (Expert system) Shows the advantages and the disadvantages of them. Moreover, the language interpreter used to program the knowledge base and the reasoning engine. The main subject of the white paper is XESS Expert System shell**,** the way the shell use to deal with XML messages as an input for the shell reasoning engine. Furthermore, processing those messages. How to use Java language to write the rules and parse the XML messages. Supporting the explanation by guiding examples.

Work Objectives

- Integrating the old automation technologies which exist in shop floor to the current factory information technology, specifically Web-Service technology without destroying the shop floor infra-structure or interrupt the production process.

- Applying the model of ISA-95 on our case study at layer two and three, using the current standards and protocols to implement them on those two layers.

- Designing Web-Service messages based on the World Wide Web Consortium – W3C – standards.

- Designing generic XML events based on analogy with CAMX standard.

- Monitoring the Web-Service messages and XML events as indicators for the technical event they need to be shown in layer two of ISA 95.

- Defining the convenient key product indicators – KPIs – in a generic way to fit most of similar production lines to our case study.

- Translating those KPIs as Drools software rules, for management of the system in layer three of ISA-95 model.

Thesis Layout

- Chapter 1 is introduction for the general and brief concepts of design, the main problems we try to solve, the proposed solution, the thesis literature survey, and the main objectives we try to obtain during this work.

- Chapter 2 is description for the study case system which in this thesis has been used as a practical automated system to apply the concept of web based monitoring system to process the data captured from by the implemented Expert system. Also it goes briefly through the structure of ANSI/ISA95 standard as it is the framework of the thesis work.

- Chapter 3 is discussing the concepts of Web-Services and Web-Service architecture, go in details what is the XML, WSDL, SOAP and UDDI.

- Chapter 4 is to show the implementation of the Web-Services for our case study and showing the results we got from it.

- Chapter 5 is introducing the concept of the artificial intelligence and relate it to expert system, utilization of droll software as a rule engine with Java as a knowledge base, designing and implementing the system and illustrating the results.

- Chapter 6 is focusing of integration of all the thesis results to imply a rigid conclusion – purposing the future work.

CH2 – THE CASE STUDY – FESTO MPS 500

Line components and Operation

Our practical study case is FESTO MPS 500 fully automated assembly line used in Tampere University of Technology – production department – FASTory lab for educational purpose.

MPS 500 system is a successively expandable system consisting of individual stations [7]. Each one has its own PLC controller and its own description and use manual. However, the central unit is always the transport system. As shown in Figure 2 below

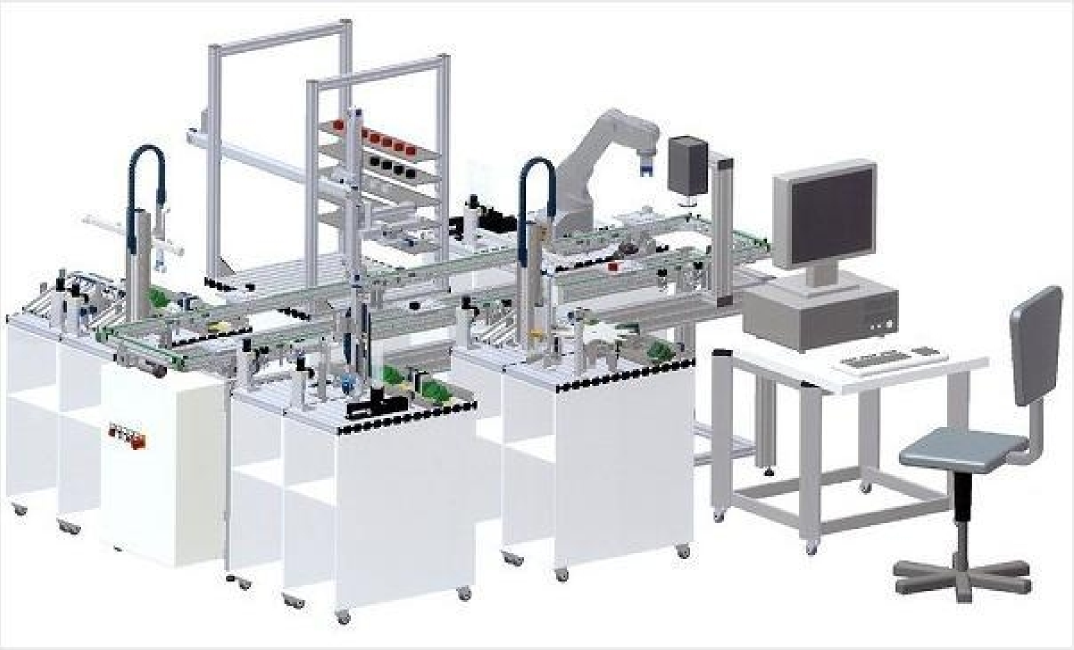


Figure 2 – FESTO MPS 500

The current layout of the line as shown in Figure 3 contains the following units

1. Distributing unit
2. Testing unit
3. Handling unit
4. Processing unit
5. Assembling and robot unit
6. Automatic Storage and Retrieval System (AS/RS) warehouse

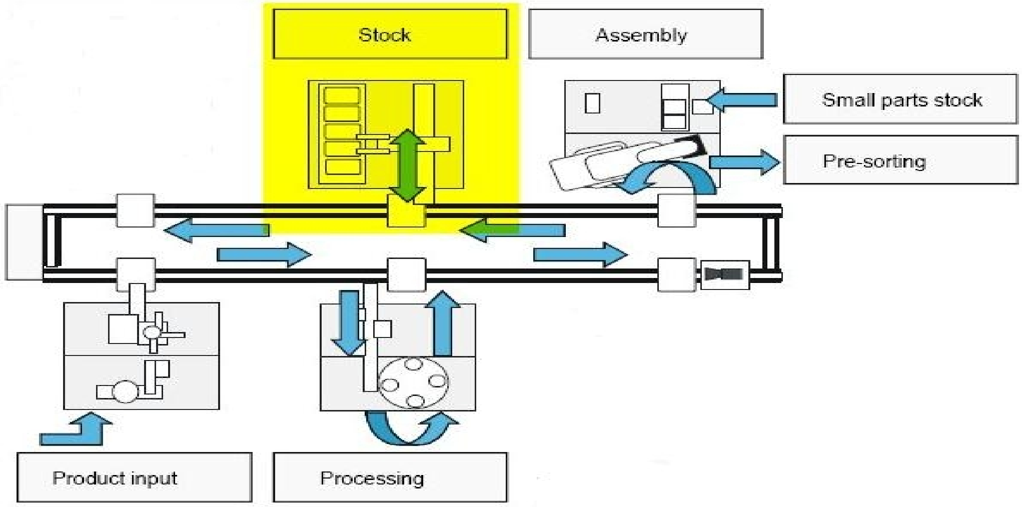


Figure 3 – The layout of FESTO MPS 500 line

The final product of FESTO MPS 500 line is a plastic cylinder which is composed from cylinder body, cover, spring and piston. The assembled cylinder and its components can be seen in Figure 4

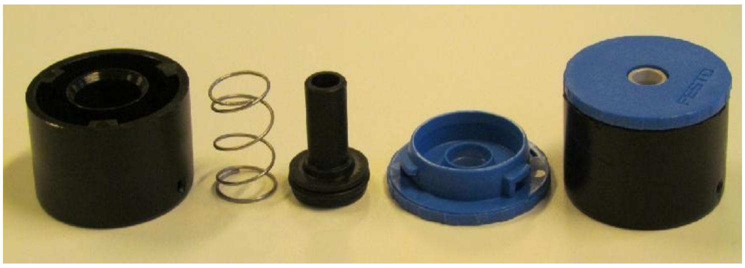


Figure 4 – The assembled cylinder and its components

In complete set up of a MPS 500 system standard sequence, the cylinder bodies are separated from the distribution station to be transferred to the testing station. The testing station checks the condition of the cylinder bodies, ejects the junk in case of occurrence and transfers the faultless pieces to the transport system. Consequently, the product input into the system takes place from the distribution station.

The cylinder bodies are transported to the processing station thanks to a PIC-alfa station. Then, the cylinder bodies should be processed, followed by testing. The PIC-alfa station returns the cylinder bodies to the transport system.

The robot station assembles a model cylinder from a basic body. Finally, the AS/RS warehouse is able to store assembled cylinder

Distributing unit

Figure 5 shows the distribution unit, the main function of the Distributing station is to separate the cylinder bodies from the Stack magazine. 8 different cylinder bodies can stored in the magazine stack, at least one cylinder body shall be exist inside the magazine tube in order the unit can start.

A single acting cylinder pushing the cylinder bodies sequentially to be picked up by the swinging arm via the vacuum gripper fixed on the end of the arm to deliver it to the next station which is testing.

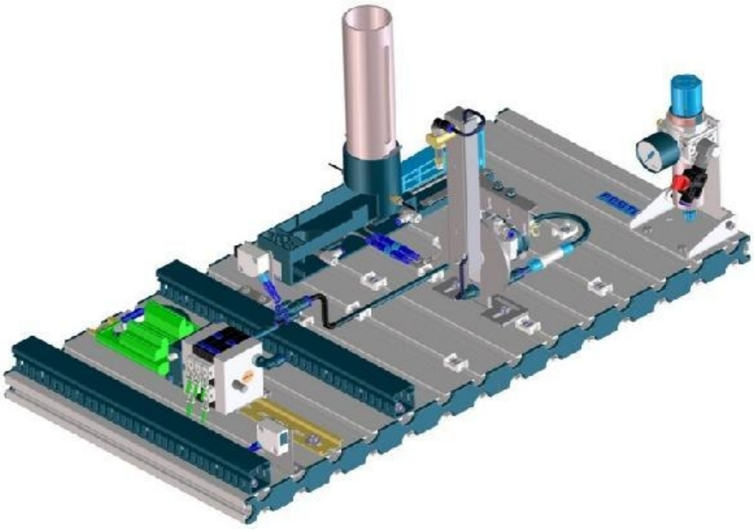


Figure 5 – Distribution unit

Testing unit

Figure 6 shows the testing unit, the main function of the Testing unit is to determine the different characters of the cylinder bodies such as colour and height. The unit can different between coloured red or silver cylinder bodies and black one, In same time analogue height sensor will measure every cylinder base transport by the testing unit from the distribution to the main conveyor system.

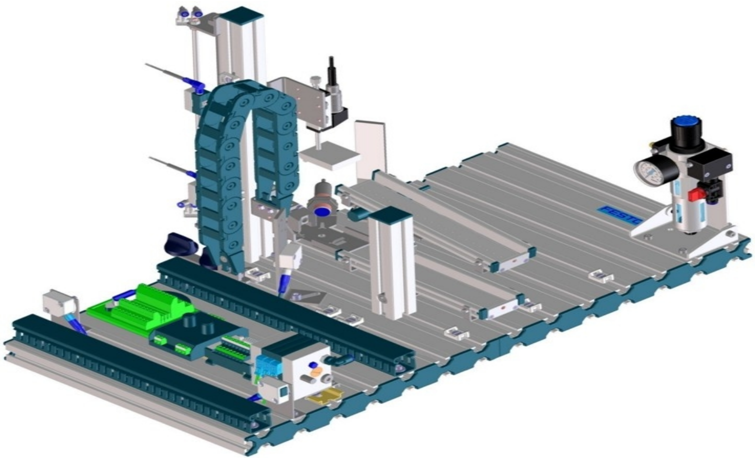


Figure 6 – Testing unit

Handling unit

The main function of the handling unit to manipulate the cylinder bodies from the conveyor system to the processing unit to be processed then back again to the conveyor system, Figure 7 shows the different components of the handling unit, it is equipped with a flexible twin-axis handling device, which retrieves the cylinder bodies by means of a pneumatic gripper. A colour detection sensor is attached to that gripper to distinguish between 'black' and 'non black' cylinders. The cylinders can be transfer again to the conveyor system or deposited based on customized sorting criteria defined by the PLC program.

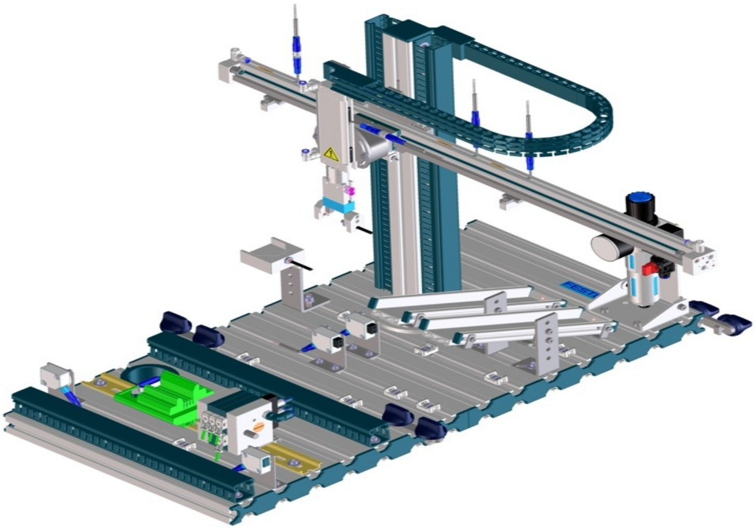


Figure 7 – Handling unit

Processing unit

The function of the processing unit is to emulate the process of rubbing the cylinder inside holes. The unit contains a rotating table which takes the cylinder first to check if it is flipped or in right position, if the cylinder is not flipped it will move to the next position, thus the rubbing tip will go down to rub the cylinder hole. Figure 8 shows the different components of the Processing unit.

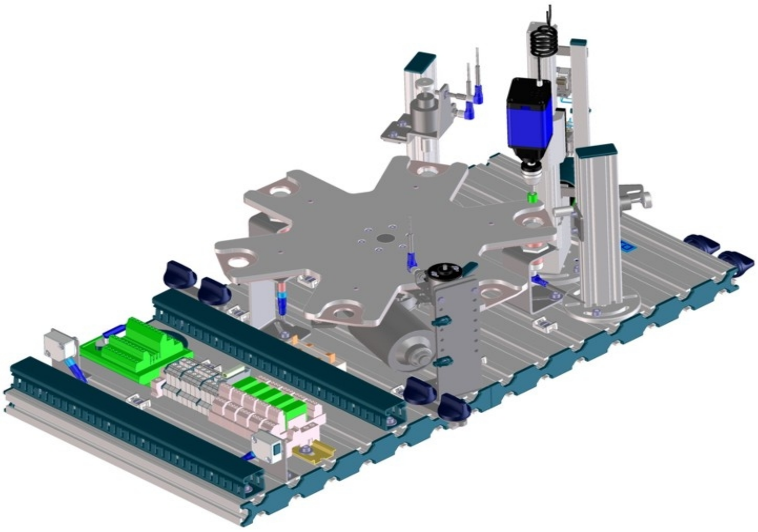


Figure 8 – Processing unit

Assembling and robot unit

The main function of the assembly station is to provide the small parts of the cylinder to the robot to assemble it to the cylinder body.

Figure 9 shows the arrangement of the assembly station which contains a stack tube for springs and another for covers, dedicated pallet for two different colours pistons.

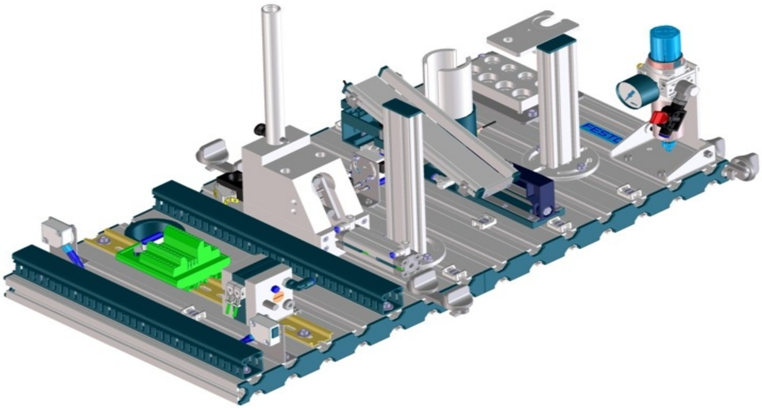


Figure 9 – Assembly station

AS/RS20 warehouse

The main function of the warehouse is to store/retrive the final assembled cylinders,
In our case study the warehouse is the final destination of the assembled cylinder, no retrieve process is executed during this production cycle.

The capacity of the warehouse is for 35 assembled cylinder. The warehouse has three axes Cartesian robot to pick up the cylinders from the conveyor system to the warehouse shelves. The principle of storing the cylinder is FIFO – first in, first out – principle as shown in Figure 10.

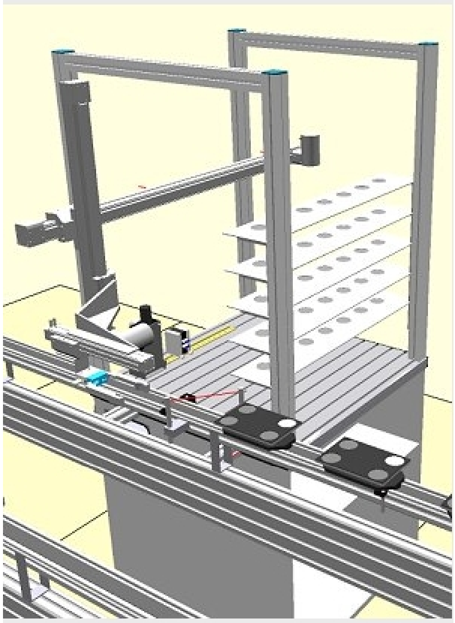
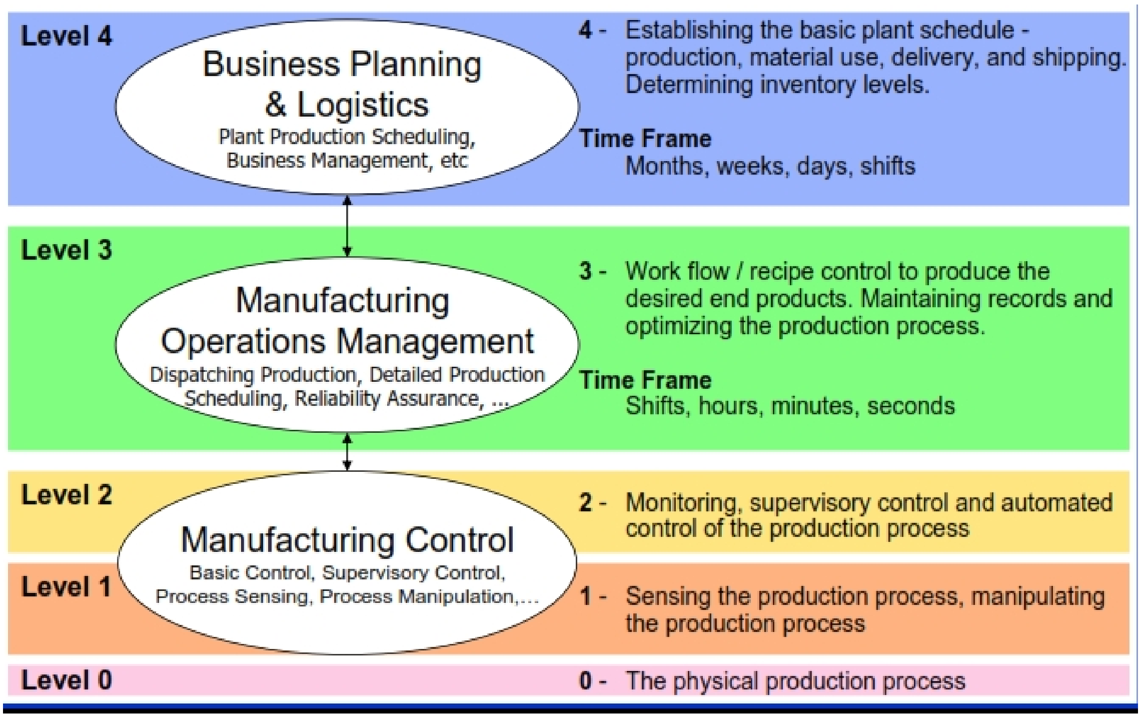


Figure 10 – Warehouse unit

ANSI/ISA95 standard

The previous description of the project case study represents the physical automation layer of ANSI/ISA95 standard. Before getting to the next chapter – which will discuss the theoretical part of the Web-Service technology – That will be used in designing the third layer of the model. As ANSI/ISA95 standard is the main skeleton of the project, it would be good to explain more, the details of the ANSI/ISA95 standard shown in Figure 11



Level 4
Business Planning & Logistics
Plant Production Scheduling, Business Management, etc
4 - Establishing the basic plant schedule - production, material use, delivery, and shipping. Determining inventory levels.
Time Frame
Months, weeks, days, shifts
Level 3
Manufacturing Operations Management
Dispatching Production, Detailed Production Scheduling, Reliability Assurance, ...
3 - Work flow / recipe control to produce the desired end products. Maintaining records and optimizing the production process.
Time Frame
Shifts, hours, minutes, seconds
Level 2
Manufacturing Control
Basic Control, Supervisory Control, Process Sensing, Process Manipulation,...
2 - Monitoring, supervisory control and automated control of the production process
Level 1
1 - Sensing the production process, manipulating the production process
Level 0
0 - The physical production process

Figure 11 – Layers tasks in ANSI/ISA95 standard

ANSI/ISA95 is an American ANSI standard created by an ISA committee called SP95. ISA-95 is the international standard for the integration of enterprise and control systems A group of volunteer industrial experts worked for developing the standard, this why it used later as an open source standard for structuring an industrial firm due to common agreed standard. The standard is divided into the following sections

- ISA-95.01 Models & Terminology
- ISA-95.02 Object Model Attributes
- ISA-95.03 Activity Models
- ISA-95.04 Object Models & Attributes
- ISA-95.05 B2M Transactions

Proceeding from the concept of modularity, ANSI/ISA95 is dividing the industrial enterprise into many different modules, every module called layer
The layers of ANSI/ISA95 are

- Layer 0: this layer describes the actual physical process
- Layer 1: the activity of sensing the physical process and pass the sensing data to the controller devices
- Layer 2: the activity of monitoring and supervision of the controlled process
- Layer 3: the activity of monitor the work flow from the manufacturing point of view, such as the recipe control, production and defection rate
- Layer 4: the activity of management the manufacturing level from the business point of view

The different sections of the standard contains

- Definitions for the different domain tasks in the industrial firms
- Definitions for the time frames of every layer to be proportional with those tasks assigned for every layer.
- Definition for the information flow and exchange models between the different layers in the hierarchy.

The standard defined the XML message format as the recommended format for exchanging the information between the different level in the model
The main target of ANSI/ISA95 standard is to

- Ease and decrease the time and the cost of the integration process between the control system, manufacturing system and business during the whole life cycle of the enterprise
- Upgrade the existing industrial premises to fit the existing model and improve the organization overall performance by giving more manufacturing and management capabilities
- Ability of expansion of the integrated project, from both the hardware and the software prospective, without destroying the old infrastructure

One of the main concepts in ISA95 section 1 and 2 is to analysis the project into resource objects and process measuring criteria
The resource object can be

1. Personal resource
2. Equipment resource
3. Material or energy resource
4. Process segments

The process measuring criteria are

1. Capabilities and capacities of the work force
2. Product and recipe definition
3. Production schedule
4. Production performance

In the next chapters, the previous mentioned point will be always be into consideration either during designing the Web-Services for layer 2 of the model or the expert system for layer 3.

CH3 – WEB-SERVICES

Introduction

Web-Service is one of the most grown technologies in our current area, because of the strong trend of using the current IT technology for the sake of business integration as a new market demand to share the information as open sources over the internet.

The Web-Service technology started to flourish, whenever the internet became available for normal users in the nineties; the public users started utilizing it in everyday activities, as it became later the main aid for everyone to collect whatever information wanted from any place in the world, that motivate the different related computer and information technology world societies to produce a standard internet protocols, to enable the normal user to access the internet whatever the platforms they are using.

In middle of the ninetieth century, it was clear that the WWW standards can simply and successfully provide any internet user by any information, presented by any browser over the network; this was called H2A – Human to Application – scenario. However people thought as the internet infrastructure already exists they can use it for exchanging the information between the different applications.

The idea found a lot of support and encouragement and it has been named later as A2A – Application to Application – scenario. But the idea was only studied from the hardware infrastructure not from the software wise, till this point HTTP was all the way successful to retrieve information between human and application, even it was able to exchanging information between two different simple application following a simply way of linking two document paths using hypertexts. This simple way did not provide any applicable solution for complex A2A process.

In the end of the ninetieth century SOAP – Simple Object Access Protocol - has been introduced by Mircrosoft and it was supported by IBM – the protocol depended on exchange the information in a form of structured text or in other words XML – Extensible Markup Language. The SOAP became the most common protocol to deal with Web-Services applications, not long time after IBM, Microsoft and Ariba successfully implemented the WSDL – Web-Services Description Language, then the UDDI - Universal Description, Discovery and Integration.

XML, WSDL, SOAP and UDDI has been improved and still improving to fit the new applications requirement [8].

In this chapter we will discuss briefly the definition of Web-Service, the components of Web-Service and architecture, the benefits and the challenges to use.

Service Oriented Architecture (SOA)

A service-oriented architecture (SOA) is a set of business-aligned services that collectively fulfil an organization's business process goals and objectives. These services can be choreographed into composite applications and can be invoked through Internet-based open standards and protocols. [9]

As it is clear from the definition of the SOA, it is a collection of many protocol and technologies together, encapsulated in one stack to let the business staff in an enterprise able to connect to their customers and explore them and being easy explored by them, deal with the partners and competitors in same business field of in related fields and also to monitor and control the enterprise of a portion of it.

In same time the SOA is a standard model for those who work in technical customs such as programmers or web developers who design the applications to link the enterprises using the web technology.

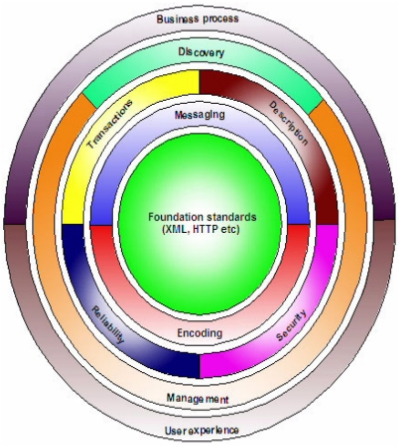


Figure 12 – Service Oriented Architecture Onion

Figure 12 shows the different layers of the SOA as every layer contains specific protocols for executing certain tasks.

Service Oriented Architecture components

The main components in the SOA are shown in Figure 13. In this figure three main entities are exist which are [10]

1. Service provider
2. Service registry/ broker
3. Service customer / requestor

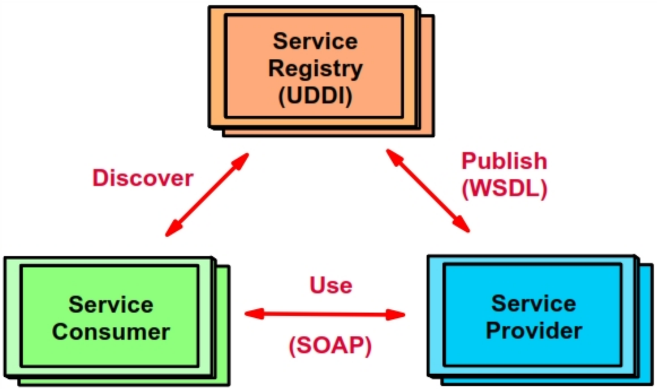


Figure 13 – Service Oriented Architecture main components

The service provider is this one who owns the services and can publish or offer them on the web, deciding which services list can be exposes, or published by the web broker. Moreover the agreement type with the service requestor, the service provider uses the WSDL protocol to enable him to publish his services.

The service broker is this entity which announces the different service lists allowed to publish by the providers for the whole service requestors in the domain. The broker can be follow different trend for marketing his services depending on the business model it has been used to build this broker. But most of the brokers will use UDDI protocol to be discovered and discover the service consumer. Moreover it still use the WSDL protocol to process the published message from the service provider and publish the service offers for it.

The service consumer is the end user who is searching for a certain service over the network from those services it can discover dynamically from the service broker. Whenever the service consumer will find his service, the agreement shall be obtained between it and the broker, which in turn will inform the service provider about certain consumer asking for a certain service. At this moment the service broker role has been ended and the service consumer and provider will start to exchange the agreed services in a direct way using SOAP protocol.

The service oriented architecture mechanism is superior than message oriented architecture mechanism which has been used for CAMX standard. The difference is that the broker in message oriented architecture still involved after delivering the messages for the consumer even after agreement, which in turn makes the broker is the narrow bottle neck of the architecture. However this problem does not exist at all in the service oriented architecture.

One important information about the SOA components, that every component is not bounded by a certain role. In other words a service provider can act as service consumer as well as a broker; it depends on the need of the component to offer, demand or manage the services.

What is the Web-Service?

Web-Service architecture has been discussed in the previous, but what about the Web-Service definition.

A Web-Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web-Service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with a XML serialization in conjunction with other Web-related standards [11].

If we try to put the Web-Service in few words the Web-Services are self-contained, modular applications that can be described, published, located, and invoked over a network. [12].

The Web-Service technology is the most suitable choice fits the Web-Service architecture, as the properties of it works as advantages that aimed to be obtained using the service oriented architecture.

The properties or advantages can be summarized in the following points [13]

Self-contained application and independency of the language of the operating system

The Web-Service applications are only needed XML client and HTTP client to be processed on the client side. Or SOAP server and HTTP server on the server side.
Those protocols are commonly essential within any existing operating system or platform which made Web-Service technology is independent and self sufficient.

Self-describing application
As the core of the Web-Service application is XML which marked as a structure language to organize data in a text like format as will be discussed later in more details.
This property makes Web-Service is so easy to be readable even by any non-professional programmers.

Encapsulated, modularity, integrity and expandability
Designing a Web-Service application is like designing a piece in a puzzle which can be integrated with more pieces to give more complicated services, thanks that all the Web-Services already designed based on a well-known architecture.

The Web-Service applications can extend so easily, moreover the new services can inherit a lot characteristics from the parent services.

Interoperability
As the Web-Service does not need a certain platform, this eliminate the problem of exchanging of services among the different operating systems.

Discoverability
Other important core protocol for Web-Service application is UDDI which will be discussed more in this chapter; this protocol aids publishing and discovering the service within the network domain.

Open source and standard language
All the standards involved in designing a Web-Service application are open and free standards, which made it easy for anyone to create his own application. On other hand, sharing the information.

Dynamic language
Web-Service applications are meant to be published, invoked, deployed and discovered because the characteristics it inherited from WSDL and UDDI protocols.
That eases establishing the e business process over the internet.

Loosely coupled application
No configuration is needed for Web-Service applications data exchange, which allow the coordination and accordingly flexibility of integration.

Security
Due to the fact that Web-Service applications are meant to be invoked on a certain network which mostly the internet, security is a crucial issue for any Web-Service application. XML-Encryption and XML-Signature has been specified clearly by different W3C

and IETF recommendations. WS-Security policy language and WS-Trust defines a complete model for all the required syntax for a secure data exchange and integrity over the internet.

Those advantages/ properties above are all inherited from other protocols the technologies have been involved for designing a Web-Service application, as it is clear in Figure 14.

The figure shows the protocol stack of the Web-Service protocol stack. As it can be seen easily every layer of the protocol concern certain task and provide the Web-Service application with a new point of strength.

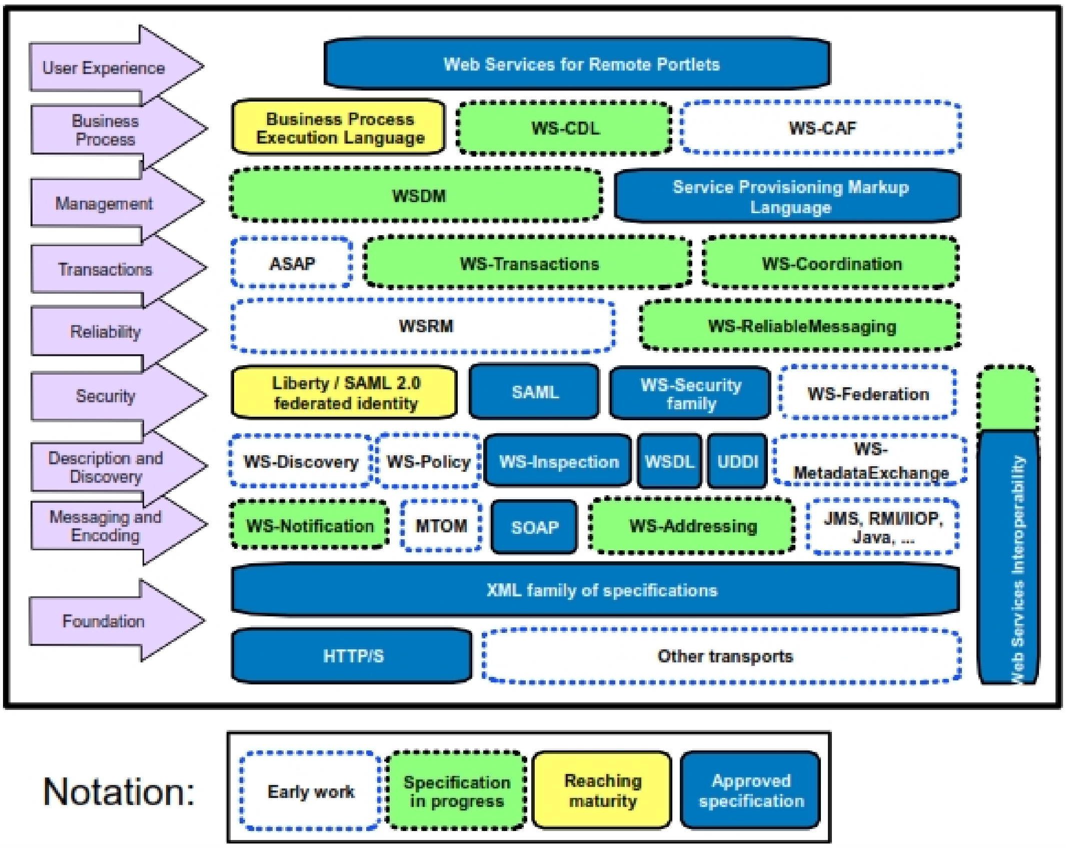


Figure 14 – The protocol stack of Web-Service application

Web-Service core protocols

EXtensible Markup Language – XML

XML is an extensible markup language, originally invited to store the data and the information in structured text way to describe electronic text – in simple words XML is meant to be the metalanguage for the electronic text.

XML is an extansible language that means you can always extend the old files, as the language is simply composed of customized tags and elements, which named flexibly due to the creator of the file choices. This feature of the XML to describe annotation or other marks within a text in form elements – which on contrary with HTML can be defined – made XML a flexible markup language. [14]

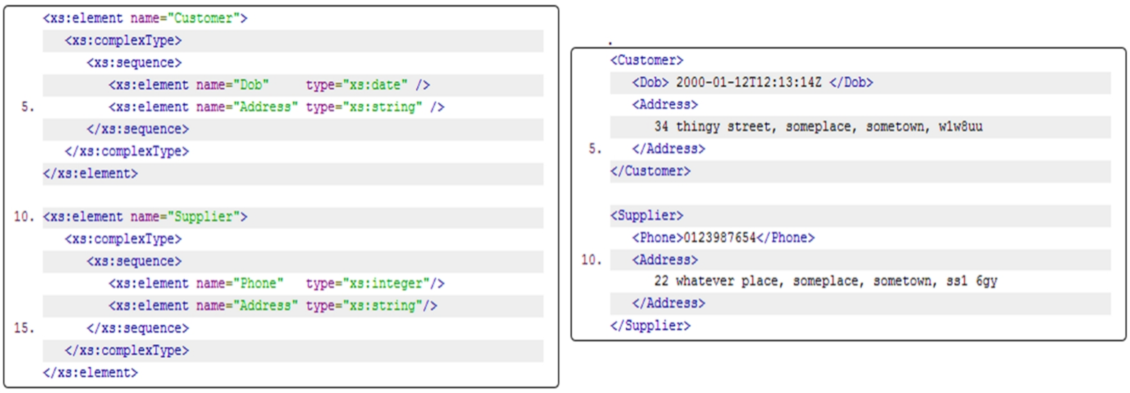
However XML is not just a language but it is metalangue, which means you can create and define new languages using it. On other words, it is a core language for many other languages such as RSS or MathMl. [15]

XML facilitates the data sharing and transfer as it is based on simple text format, that gave it much stronger privilege which is platform independency, as with XML you can easily change, store or transfer files among different operating platforms, that solved a curtail concern for internet application developers, as in internet world many different operating systems are connected in same time.

From the same prospective of object oriented language. XML object or element can have many instances all of them follow the definitions of the element, having the same attributes but with different values, all those definitions should be written in a separate file which used to know as XML schema or shortly XSD file.

A XML Schema [16]:

- defines elements that can appear in a document
- defines attributes that can appear in a document
- defines which elements are child elements
- defines the order of child elements
- defines the number of child elements
- defines whether an element is empty or can include text
- defines data types for elements and attributes
- defines default and fixed values for elements and attributes



```
<xs:element name="Customer">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Dob"     type="xs:date" />
      <xs:element name="Address" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Supplier">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Phone"   type="xs:integer"/>
      <xs:element name="Address" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<Customer>
  <Dob> 2000-01-12T12:13:14Z </Dob>
  <Address>
    34 thingy street, someplace, sometown, w1w8uu
  </Address>
</Customer>

<Supplier>
  <Phone>0123987654</Phone>
  <Address>
    22 whatever place, someplace, sometown, ss1 6gy
  </Address>
</Supplier>
```

Figure 15 – An example of one instant of xml in accordance with its schema

Figure 15 is a sample of a textual presentation of one XML instant of predefined complex type element in the adjacent schema.

In the schema there are two different elements, the first one is a description of a customer and the other for a supplier. Every element of them have other two sub elements or children elements. Every individual child has its own attributes which appear have some values in the XML files.

Web-Service Description Language – WSDL

WSDL is a XML originated language that describes and defines a plenty of services structure that can be offered as a service provider or requested as a service consumer. To be more precious, WSDL is describing the communication in the network in a structured way using XML.

Moreover exchanging the messages between the end points, those messages can be either document oriented or procedure oriented messages. Inside the WSDL file, the end points shall be explicitly well defined. Furthermore, the binding between the messages and used protocol sending or receiving those messages, mainly SOAP and HTTP GET/POST protocols will used for sending/receiving XML messages.

WSDL has similar properties to XML, as it is a network topology independent, and easy to expand it by adding new end points or services within the same file.

The main elements to construct a WSDL document are [17]

- **Types** a container for all the data type using standard way such as XML schema
- **Message** an abstract, typed definition of the data being communicated
- **Operation** an abstract description of an action supported by the service
- **Port Type** an abstract set of operations supported by one or more endpoints

- **Binding** a concrete protocol and data format specification for a particular port type
- **Port** a single endpoint defined as a combination of a binding and a network address
- **Service** a collection of related endpoints

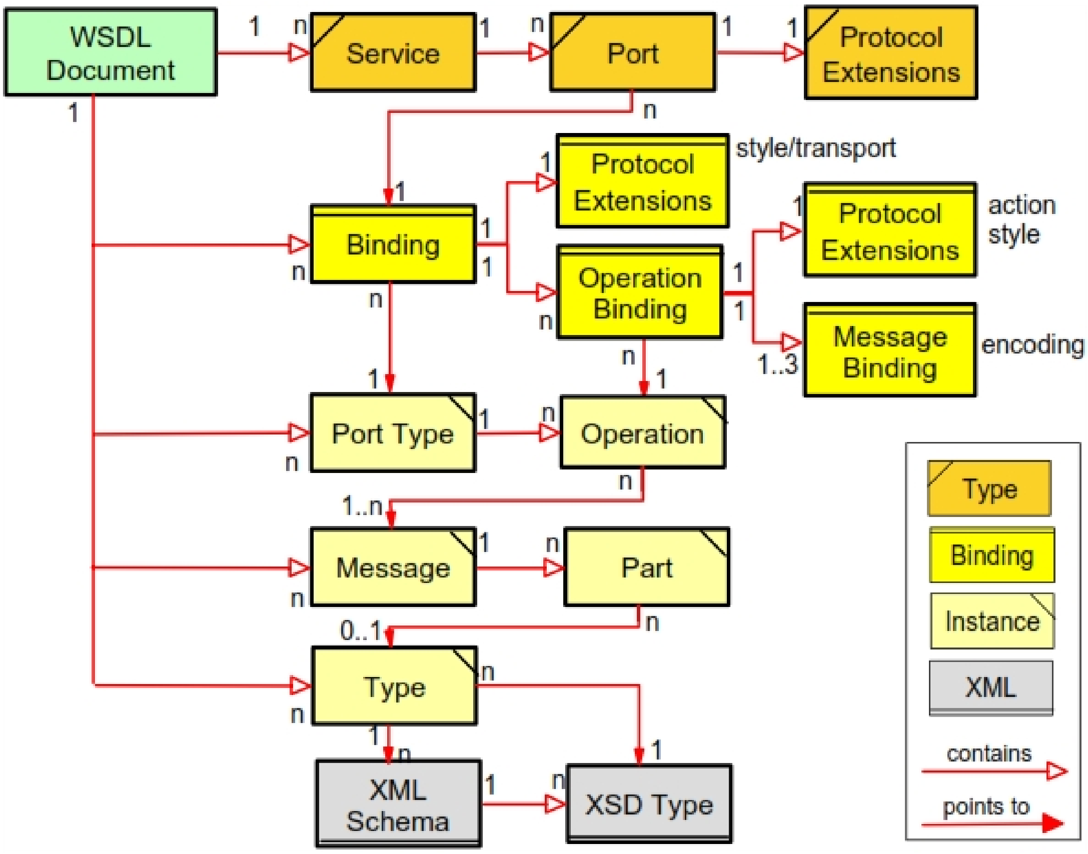
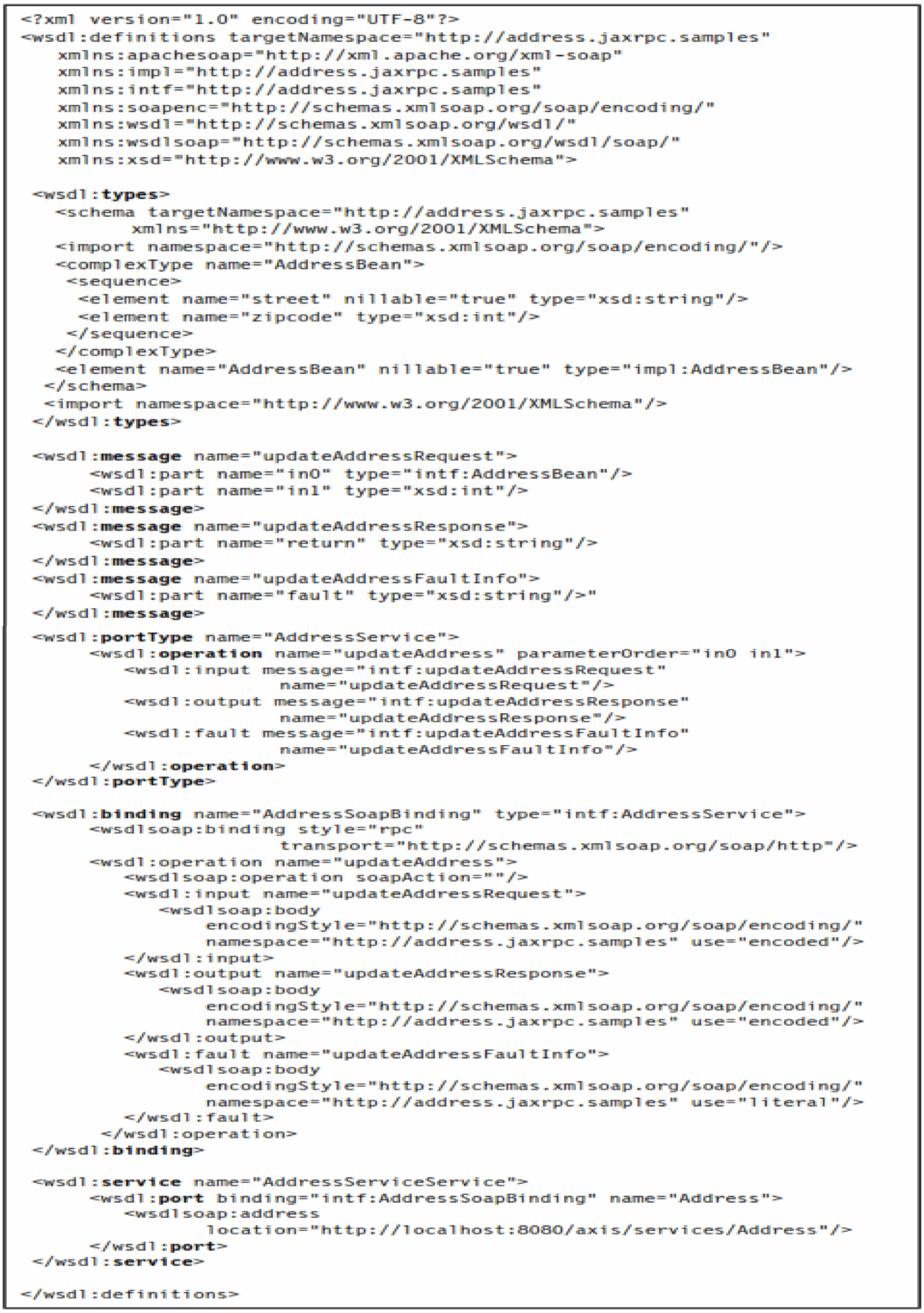


Figure 16 – The main WSDL document elements

Figure 16 illustrates an example for a WSDL document. The diagram is telling that every WSDL file can contain zero or more services. Every service can target a zero or many end points, in other words ports.

Every port can use certain protocols for communication which already defined into the bind. The port type will contain zero or many operation that can be processed by a certain port with a certain communication protocol or bind.

The operation define some action that will be token by the service, this operation or action is more or less a set of XML messages that already defined by the XSD ,An example of a complete WSDL documentation can be shown in
Figure 17



```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://address.jaxrpc.samples"
   xmlns:apachesoap="http://xml.apache.org/xml-soap"
   xmlns:impl="http://address.jaxrpc.samples"
   xmlns:intf="http://address.jaxrpc.samples"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

 <wsdl:types>
   <schema targetNamespace="http://address.jaxrpc.samples"
          xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="AddressBean">
    <sequence>
     <element name="street" nillable="true" type="xsd:string"/>
     <element name="zipcode" type="xsd:int"/>
    </sequence>
   </complexType>
   <element name="AddressBean" nillable="true" type="impl:AddressBean"/>
  </schema>
  <import namespace="http://www.w3.org/2001/XMLSchema"/>
 </wsdl:types>

 <wsdl:message name="updateAddressRequest">
      <wsdl:part name="in0" type="intf:AddressBean"/>
      <wsdl:part name="in1" type="xsd:int"/>
 </wsdl:message>
 <wsdl:message name="updateAddressResponse">
      <wsdl:part name="return" type="xsd:string"/>
 </wsdl:message>
 <wsdl:message name="updateAddressFaultInfo">
      <wsdl:part name="fault" type="xsd:string"/>"
 </wsdl:message>
 <wsdl:portType name="AddressService">
      <wsdl:operation name="updateAddress" parameterOrder="in0 in1">
         <wsdl:input message="intf:updateAddressRequest"
                      name="updateAddressRequest"/>
         <wsdl:output message="intf:updateAddressResponse"
                      name="updateAddressResponse"/>
         <wsdl:fault message="intf:updateAddressFaultInfo"
                      name="updateAddressFaultInfo"/>
      </wsdl:operation>
 </wsdl:portType>

 <wsdl:binding name="AddressSoapBinding" type="intf:AddressService">
      <wsdlsoap:binding style="rpc"
                      transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="updateAddress">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="updateAddressRequest">
            <wsdlsoap:body
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                namespace="http://address.jaxrpc.samples" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="updateAddressResponse">
            <wsdlsoap:body
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                namespace="http://address.jaxrpc.samples" use="encoded"/>
         </wsdl:output>
         <wsdl:fault name="updateAddressFaultInfo">
            <wsdlsoap:body
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                namespace="http://address.jaxrpc.samples" use="literal"/>
         </wsdl:fault>
       </wsdl:operation>
 </wsdl:binding>

 <wsdl:service name="AddressServiceService">
      <wsdl:port binding="intf:AddressSoapBinding" name="Address">
         <wsdlsoap:address
                location="http://localhost:8080/axis/services/Address"/>
      </wsdl:port>
 </wsdl:service>

</wsdl:definitions>
```

Figure 17 – An example of WSDL document

Simple Object Access Protocol – SOAP

SOAP is a simple object access protocol is standard communication protocol for exchanging structured information between applications in a distributed decentralized network over HTTP protocol as a mean of transport. SOAP is XML based protocol this why it is platform independent and can be simply extended.

SOAP became the standard protocol of message exchange concerning Web-Services. As it is a light weight protocol, which mainly do two different tasks. [18]

1- Sending requests or receiving responses over HTTP transport layer
2- Processing, encoding and decoding of XML messages

Figure 18 shows the message exchange between Web-Service applications

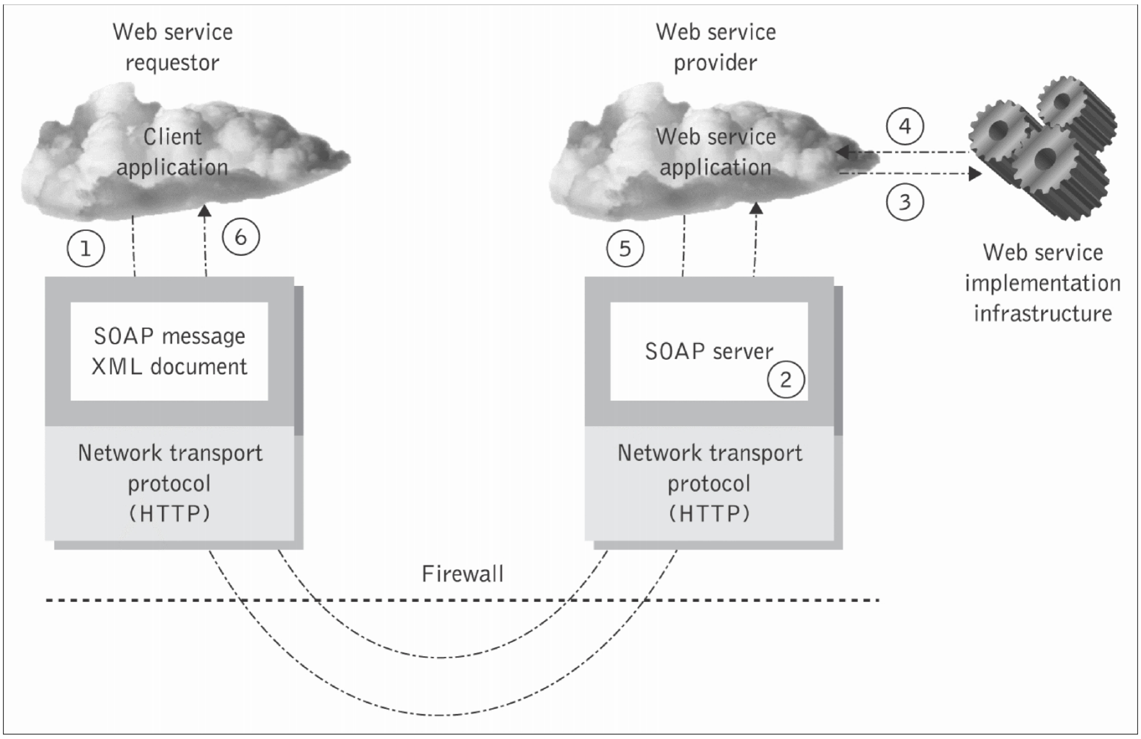
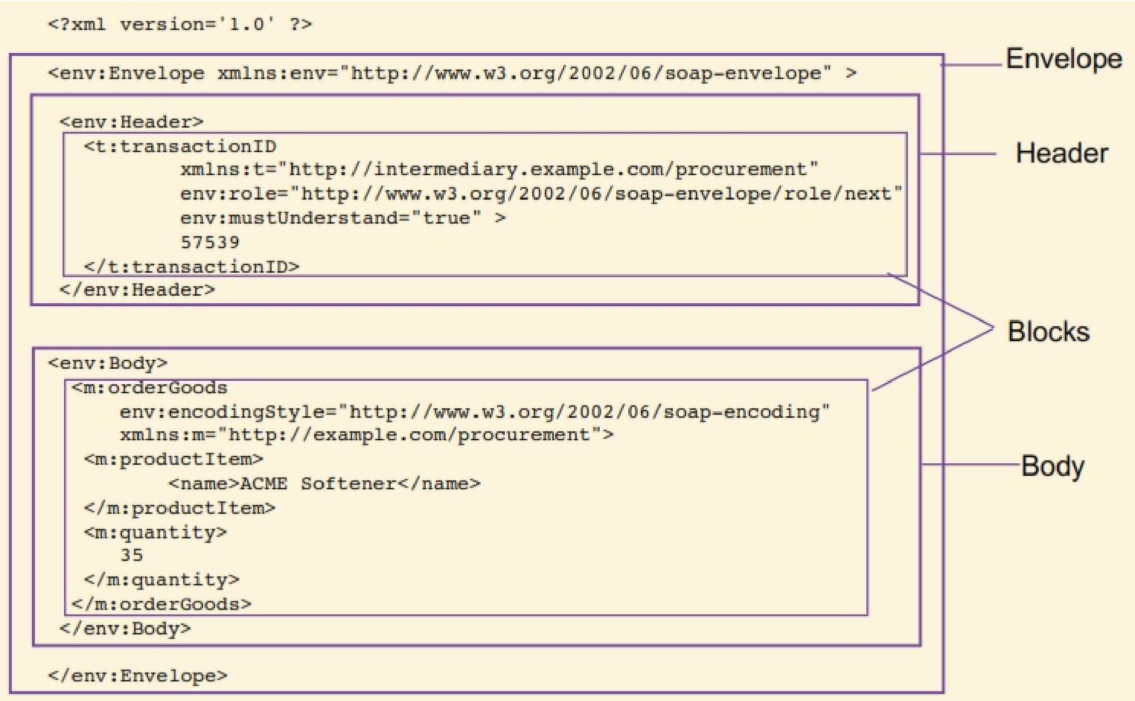

Web service requestor
Client application
1
6
SOAP message XML document
Network transport protocol (HTTP)
Web service provider
Web service application
5
SOAP server
2
Network transport protocol (HTTP)
4
3
Web service implementation infrastructure
Firewall

Figure 18 – Communication of XML Web-Service messages using SOAP

A SOAP message is an ordinary XML document which contains four main parts:

- An Envelope part to encode the XML message in SOAP format
- A Header part which contains header information
- A Body part which contains call and response information
- A Fault part in case of error it should contain error information

A simple SOAP message is shown in Figure 19



```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2002/06/soap-envelope" >
 <env:Header>
  <t:transactionID
          xmlns:t="http://intermediary.example.com/procurement"
          env:role="http://www.w3.org/2002/06/soap-envelope/role/next"
          env:mustUnderstand="true" >
          57539
  </t:transactionID>
 </env:Header>

<env:Body>
 <m:orderGoods
     env:encodingStyle="http://www.w3.org/2002/06/soap-encoding"
     xmlns:m="http://example.com/procurement">
  <m:productItem>
         <name>ACME Softener</name>
  </m:productItem>
  <m:quantity>
     35
  </m:quantity>
 </m:orderGoods>
 </env:Body>

</env:Envelope>
```

Figure 19 – SOAP message example

The services message exchange using SOAP can be done using many different patterns

1. One-way
2. Asynchronous two-ways
3. Request-response
4. Workflow-oriented
5. Publish-subscribe
6. Composite

In one way message pattern, SOAP messages are transferring in only one direction as it is shown in Figure 20. This pattern is suitable for certain applications such as resource monitoring applications.

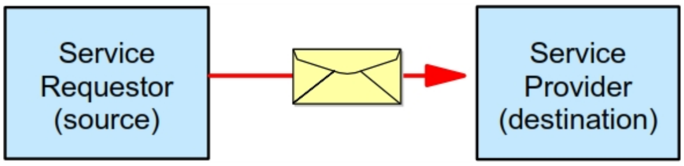


Figure 20 – One way SOAP message exchange pattern

In asynchronous two ways message pattern shown in Figure 21, the service requestor always expects a response from the provider. However there is not any certain time frame or limits as it is asynchronous.

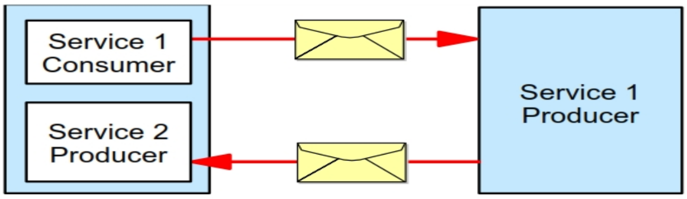


Figure 21 – Two way asynchronous SOAP message exchange pattern

The most common used mode in SOAP communication protocol is request – response pattern or sometimes called a remote procedure call – RPC as it is shown in Figure 22. A synchronous response is always expected every time there is a communication between the requestor and the provider.

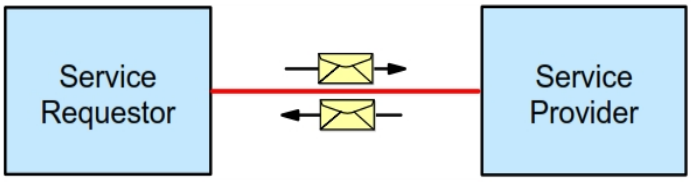


Figure 22 – Request/response SOAP message exchange pattern

In workflow oriented SOAP message exchange mode, is always implemented in case many service providers are cooperating to integrate a services package, as it is clear in Figure 23 the message cross from one service provider to another to build finally the whole work package.

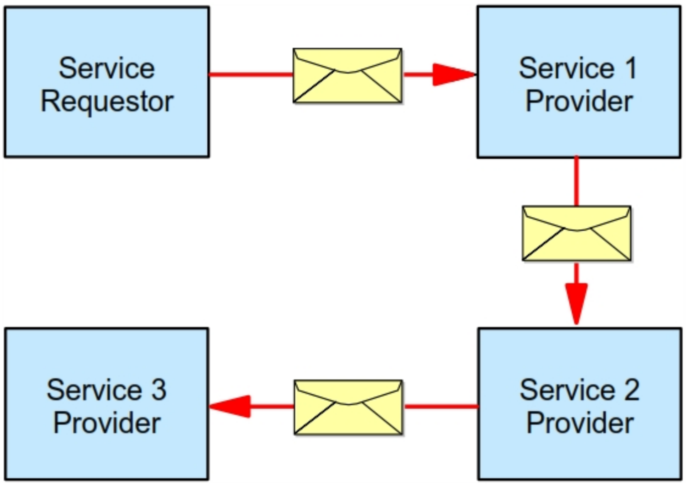


Figure 23 – WorkFlow oriented SOAP message exchange pattern

The publish subscribe SOAP message exchange mode is even based mode, and useful in case of informing many providers, regardless which provider will get the request first as it is shown in Figure 24.

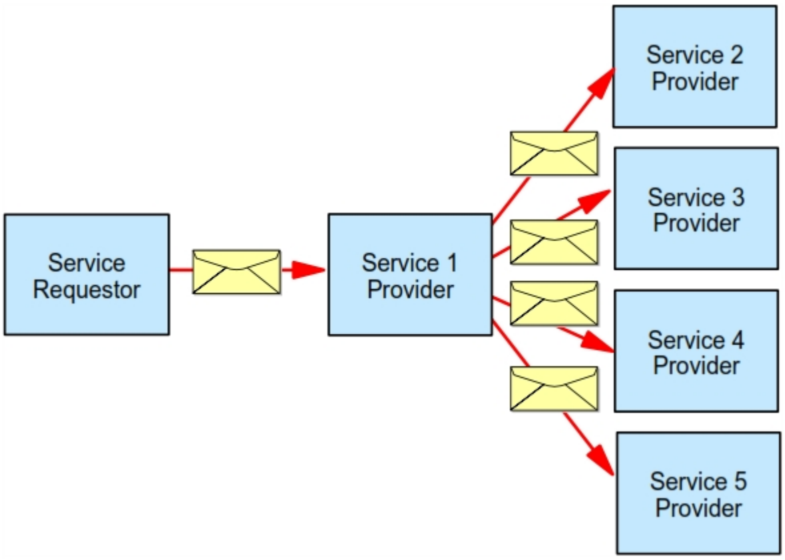


Figure 24 – Publish/subscribe SOAP message exchange pattern

The last SOAP message exchange pattern is a composite mode which is a bit similar to publish subscribe pattern. The difference is the existence of a new component call a composite service provider which is controlling the requests for different providers based on the business logic. This pattern is shown in Figure 25.

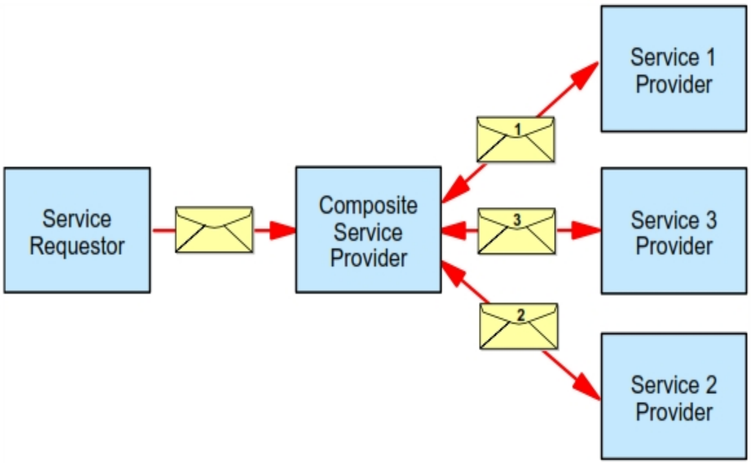


Figure 25 – Composite SOAP message exchange pattern

Universal Description, Discovery, and Integration UDDI

Universal Description, Discovery, and Integration (UDDI) is a standard protocol use the SOAP communication protocol to link the service client side application programming interface (API) side to the SOAP server, UDDI main function is to store or retrieve back the information between the different entities in the Web-Service paradigm. WSDL file is contains the UDDI to describe the appropriate Web-Service interfaces.

In the Web-Service world many different Web-Services nodes are running in different places by different providers. UDDI protocol is gathering all those Web-Services nodes into so called a UDDI registry.

The main aim of using UDDI was to simplify the process of electronic commerce, as it eases for the different Web-Service clients, with different set of information presentation protocol to discover provider. In other words, it concerns solving the problems that usually occurred during business to business (B2B) interaction, because it defines the structure of the Web-Service registries and those APIs, who are willing to access those registries. It can be seen that UDDI is a search engine protocol for those clients who are seeking for certain applications.

The implemented UDDI protocol will define if the Web-Service application either dynamic or static. In static Web-Service usually both the service provider and the service customer will get to know each others in advance during the design stage. The addresses of the service provider and requestor will be will included within the WSDL file, which in turn will be the access points for every partner during the runtime.

However the dynamic Web-Service does not really require that the service provider and customer should know each others in advance in the design stage. More complicated exploring mechanisms at the runtime have to be applied to enable every client to call a certain interface and finding one or more provider for it.

In this thesis, a static Web-Service will be implemented for designing the Web-Service based monitoring system for the mentioned case study; this will be described in details in the next chapter.

The difference between message oriented architecture and service oriented architecture

During the next chapter, the design of XML schema for the project will be based on the recommendation of CAMX IPC. Therefore, it would be useful to briefly understand the concept of a middleware message oriented architecture and how it can differ from service oriented architecture.

As it can be seen in Figure 26, the middleware message oriented architecture can have two kinds of messaging model [19], either point to point communication mode or publish/subscribe communication mode. In point to point communication mode a direct asynchronous message exchange mechanism will be involved between the message producer and consumer.

A middleware broker should be managing the message exchange between the different software entities. The most common managing procedure that the broker will follow in is FIFO – First In First Out – queue, to deliver a certain message to only one consumer.

The other utilized message communication mode is publish/subscribe. In this mode, all the message consumer will subscribe from the broker the message list they are willing to receive, in same time the all the publisher will publish their messages to the middleware broker.
The broker will follow the same mechanism of FIFO queuing to distribute the messages for the subscriber. One message can be send to many consumers

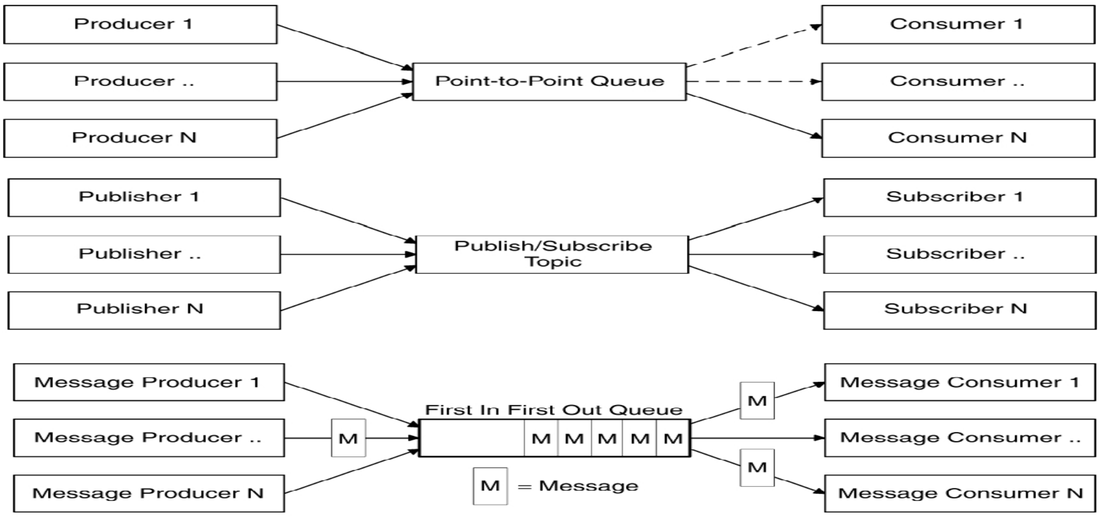


Figure 26 – Middleware message oriented architecture

The difference between the middleware message oriented paradigm and service oriented paradigm, is that the first one always involving the broker in the message exchanging either in point to point mode or in publish/subscribe mode, which makes a lot of communication traffic and waste a plenty of time during the communication process.

This cons does not exist in service oriented architecture, as the service registry which plays the role broker, only guide the service consumer to the service provider or many providers who can provide with a needed service. This way a direct peer to peer communication will be established between the service provider and consumer without wasting the registry time for managing a lot of messages in the network.

For designing our Web-Services for this project, we will use the advantage that CAMX IPC standard already defined standard XML schemas for electronic shop floor equipment communication messages, which used to be used under the concept of middleware message oriented architecture. But we are going to modify them and use them under the concept of service oriented architecture. This way we still follow a reliable standard in same way avoiding the cons of the middleware architecture.

CH4 – DESIGN AND IMPLEMENTATION OF WEB-BASED MONITORING SYSTEM FOR THE CASE STUDY

Introduction

The aim of this chapter is to show the designing and implementing the web-based monitoring system for our case study, which has been explained how it works in chapter two of this master thesis. The design of the web-services and message exchanging patterns will be based on the theoretical discussion for the Web-Service technology in the previous chapter. Finally showing the results of the monitoring system.

It is important to emphasize that the main concept of design of the web-based monitoring system we are implementing here is built on the recommendation of ISA-95 standard, which means two crucial points had been into consideration from the very beginning.

The first point is that the implemented monitoring system will be used in layer two of ISA-95 model, which is more concerning with technical data about the process in a relatively short time period. This time period can be seconds, minutes or hours.

The second point is that this web-based system will be a base for the expert system, which located due to ISA-95 model in layer three.

In this layer the technical information captured from layer two should be interpreted into more business like information, to fit the project planners and managers who are meant in this layer. The rate of information monitoring is a bit longer than the previous layer, it can be one hour, one working shift, day or week.

Web-Service hardware

As it was mention in the previous chapter, the Web-Service is software designed to support interoperable machine-to-machine interaction over a network. That means that there should be certain hardware in accordance with this software to contain it, and execute all the services from the physical point of view. For this purpose in our project we used Web-Service hardware used to be known as INICO S1000.

The S1000 is a programmable Remote Terminal Unit (Smart RTU) device which offers process control capabilities, as well as a Web-based Human-Machine Interface (HMI), support for Web-Services. The S1000 is designed to operate in typical industrial settings and is compatible with most industrial signal types and levels [20].

Figure 27 clarify the idea of distributing the different Web-Service devices over the Ethernet LAN. The same idea already applied over our case study, as there is a dedicated INICO S1000 Web-Service controller attached to every PLC controller for every machine in our case study, described in chapter 2 of this thesis.

Even the S1000 has the same ability of the regular PLC to control the process itself, it was sufficient only to hardwire it to the exciting PLC for many reasons as mentioned in the problem station. That approach will save money and time as we do not need to stop the process, or reprogram the new controller again. This was from the very beginning one target of the this thesis work.

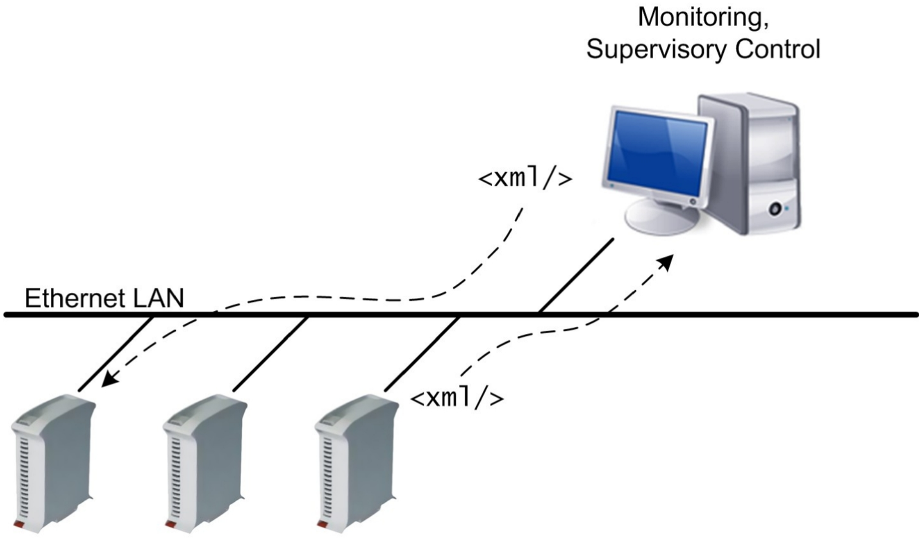


Figure 27 – The distribution of the INICO S1000 devices over the Ethernet network

Design of the XML schema

The XML schema is the first milestone in designing a Web-Service. From the same prospective of the object oriented programming, the XML schema can be seen as a container for all the definition of the objects in the project has been written as XML script. While all the XML messages will be some instances of this object contained inside this XML schema.

Every object can be created as a complex element which contains many other simple elements inside and organise them in a convenient way. An object oriented model shown in Figure 28, expresses the three objects which our case study or any other production line. The model contains three objects, the workstation, the workpiece and the labourer.

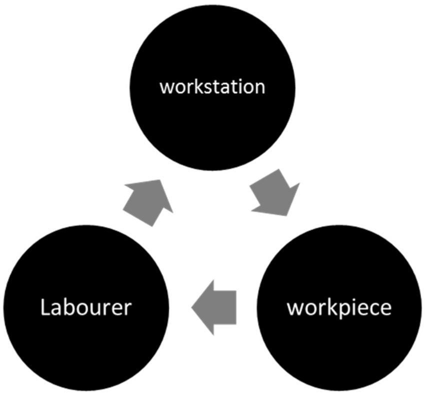


Figure 28 – The case study object oriented model

The first object we are going to discuss in details in the schema is the workstation object as show in Figure 29

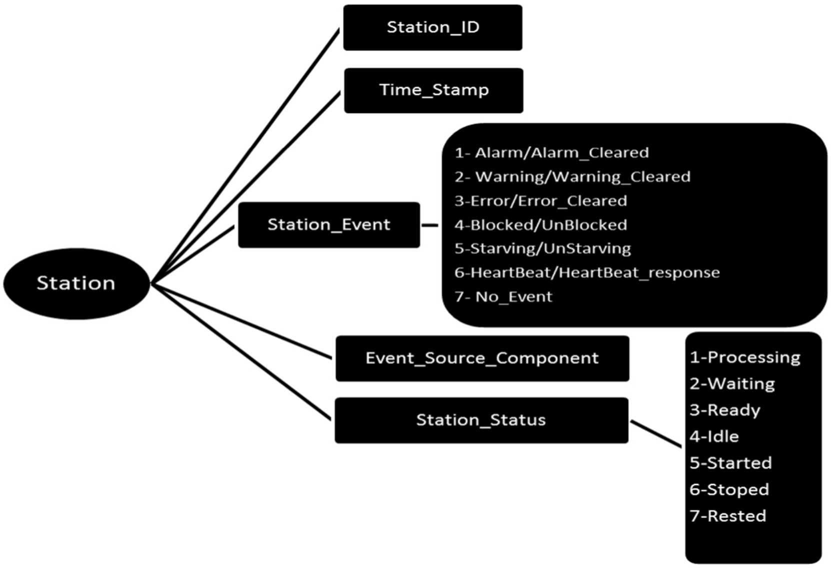


Figure 29 –Workstation object elements

As we have many stations in the case study, a child element Station_ID should contain every station name. The second child element for workstation schema is the time stamp which is quite important element for understanding the time of every event sent by different stations. The station event element can be one of those thirteen events, which listed in the previous figure. Those event statuses are following the recommendation of CAMX IPC- 2541 standard [20] which defines many XML schemas for generic requirement for electronic shop floor equipment communication messages.

Figure 30 shows the original CAMX states diagram which has been used to drive the different statues included in the Workpiece object.

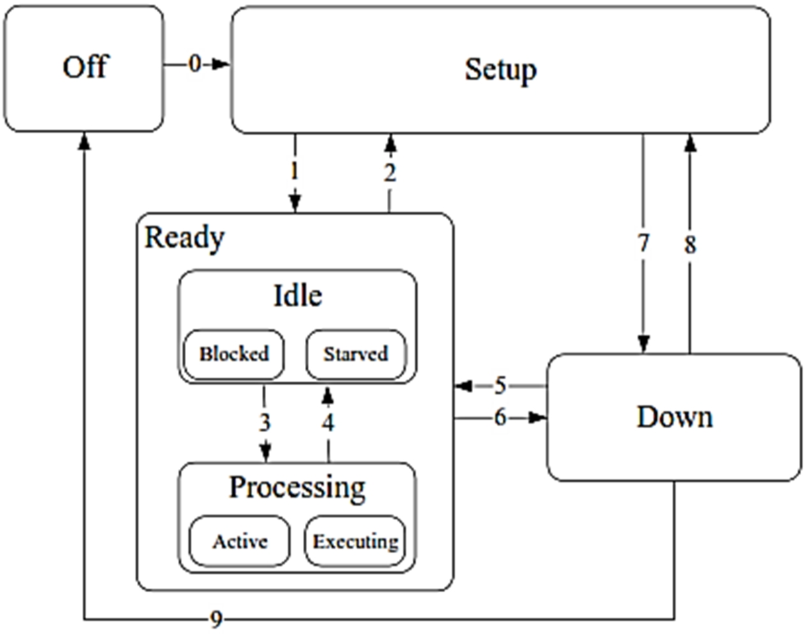


Figure 30 – CAMX Equipment State Diagram

An analogy has been done from those XML schemas which they are already defined into this standard to make them fitting our case. In other words we used XML schemas which meant to be used in message oriented architecture to be used for building the schema for the service oriented architecture. The same procedure has been done for selecting the station status. The last child element in the workstation object is the source component of event, this element should be associated with the station event as it should be the component inside the workstation that cause the event.

The second object in the schema is the workpiece. Similar to the workstation object, the workpiece has an identification number and time stamp child elements for every event occurred by or to any work piece. To understand the location of the workpiece within the production line, every message from the workpiece should have the workstation ID that processing it. The workpiece schema also has two child elements to contain some properties of the work piece, the first element is the workpiece color which either black or non-black. The second element based on the first element as if the workpiece color is black that means it is defected workpiece, otherwise it is undetected.

The final child element is the workpiece statuses which are also following the recommendation of CAMX IPC- 2541 standard. Shows a schematic for XML schema of the workpiece object.

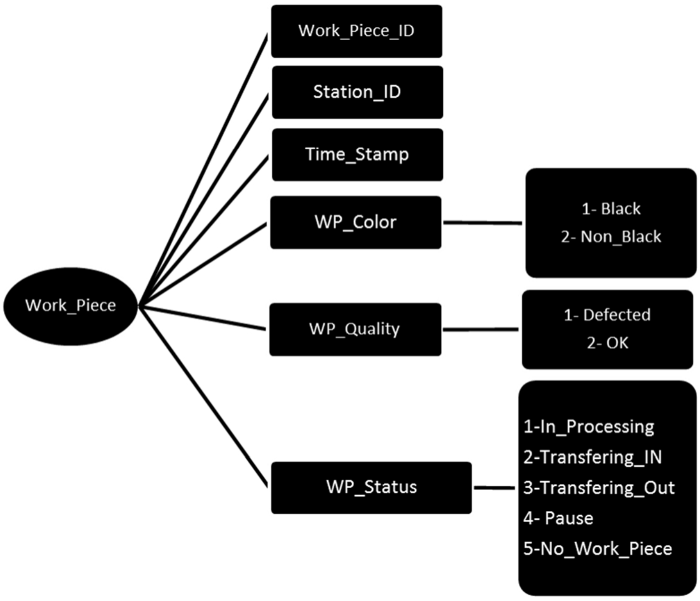


Figure 31 – Workpiece object elements

The final object in the project schema is the labourer object, as shown in Figure 32. Similar to workstation and workpiece objects every labourer has an identification number child element, time stamp child element for expressing the timing, station ID child element to express the location of the labourer over the production line. And finally labourer action child element which followed the CAMX standardization.

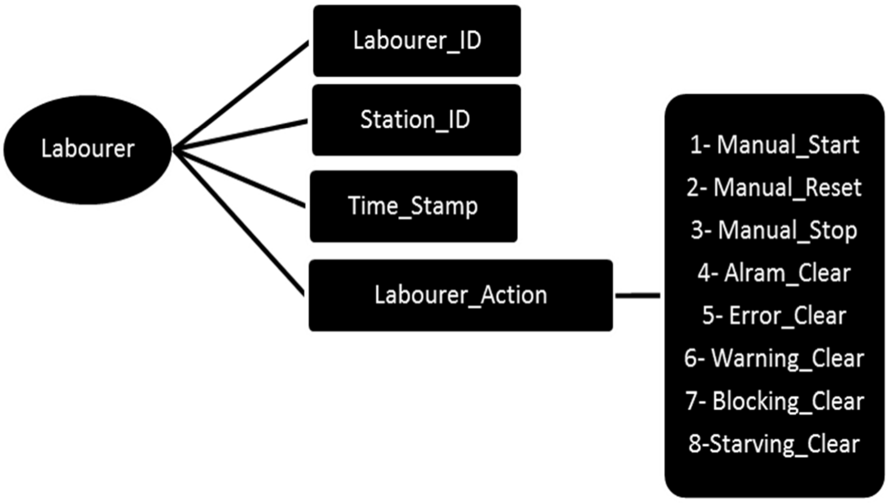


Figure 32 – A presentation for the labourer object elements

Figure 33 is collecting all the elements for the project XML schema; we have discussed earlier in details in one chart.

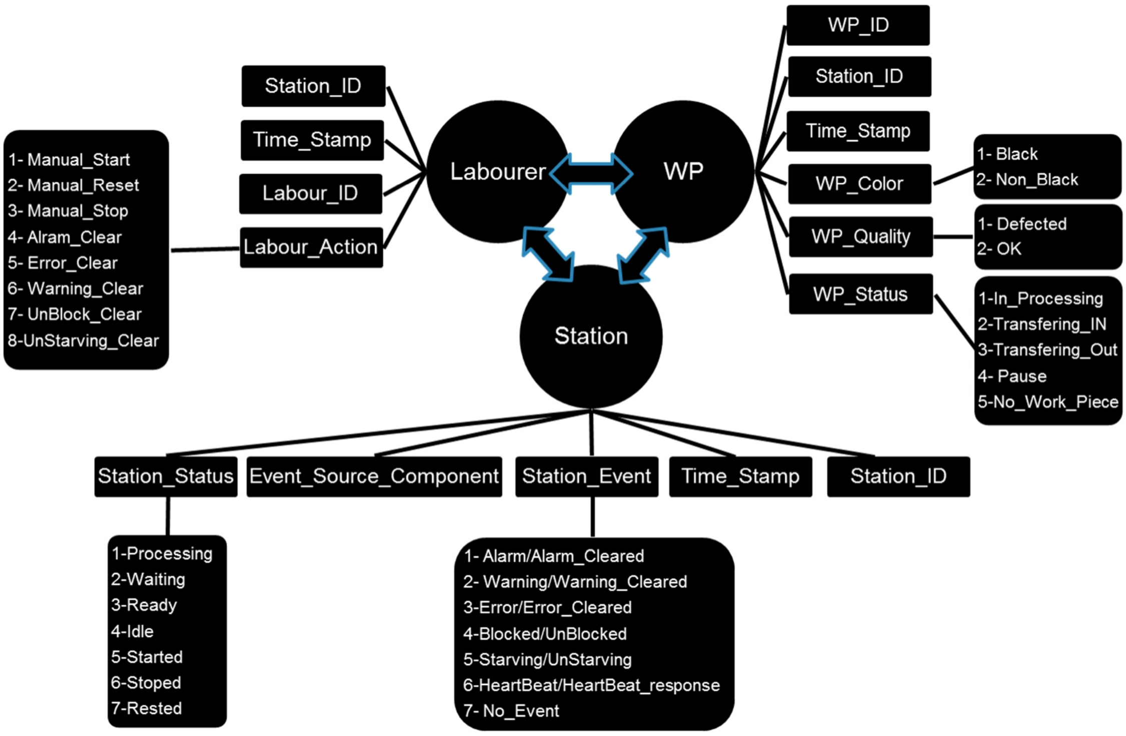


Figure 33 – A complete view of the project XML schema elements and types

It is worth to emphasize that one of main design concepts during this thesis work was to design the work as generic as possible, to fit as a wide range of cases not only our case studies .In same time, to save time and effort inside the same project. As the description of the case study in the second chapter of this thesis work, every machine will deal with the same objects we created, which means this XML schema will fit the whole assembly line.

All the original schemas that has been used from the CAMX IPC- 2541 standard can be found in Appendix 1

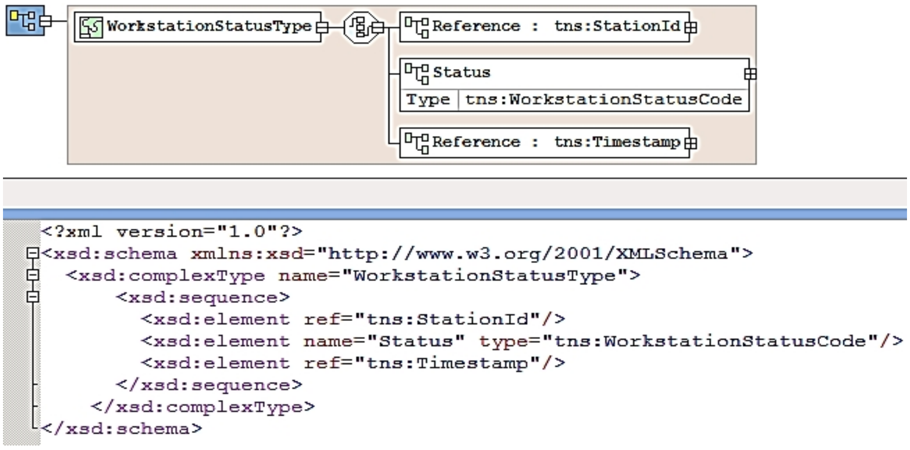
Design of the WSDL document

Web-Services as mentioned in its definition, should be described by a WSDL document, The WSDL document is built on the roots of the XML schema. As the XML schema define the types included inside the WSDL file
The same concept of designing a generic XML schema for the case study was followed to design the WSDL document.

In the project WSDL file we have five different message types

1. Work Station Status Type

The first type in the project WSDL document is shown in Figure 34, it contains three main elements as it is clear in the graphical presentation. The station ID which is a string expressing the name of the station. Every station should have a unique name. The timestamp is to indicate the time whenever the service will use this type included in any message. The station status could be ready – starting – idle – rested –processing – waiting or stopping. Those statuses originate from the definition of the work station object which discussed earlier.

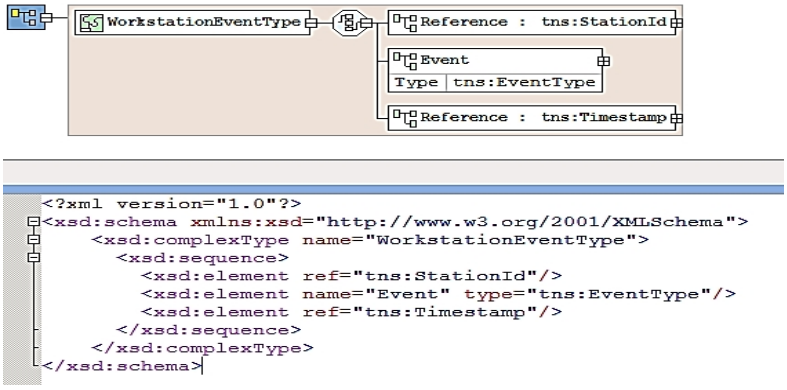

WorkstationStatusType
Reference : tns:StationId
Status
Type tns:WorkstationStatusCode
Reference : tns:Timestamp

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="WorkstationStatusType">
      <xsd:sequence>
        <xsd:element ref="tns:StationId"/>
        <xsd:element name="Status" type="tns:WorkstationStatusCode"/>
        <xsd:element ref="tns:Timestamp"/>
      </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Figure 34 – Work Station Status Type included in the generic WSDL document

2. Work Station Event Type

The second type in the project WSDL document is shown in Figure 35, it contains three main elements as it is clear in the graphical presentation. The work station event could be alarm – alarm cleared – warning – warning cleared – error – error cleared – unblocked – starving – unstirred – heartbeat – heartbeat response or no event.

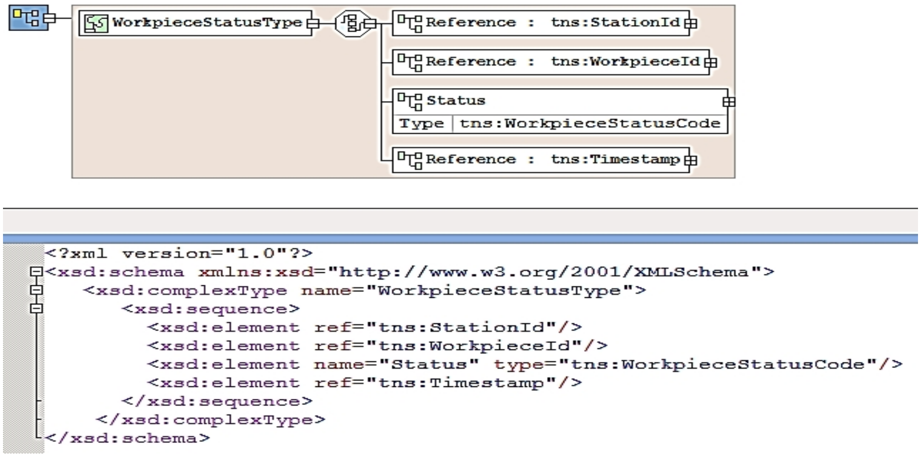


```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:complexType name="WorkstationEventType">
      <xsd:sequence>
        <xsd:element ref="tns:StationId"/>
        <xsd:element name="Event" type="tns:EventType"/>
        <xsd:element ref="tns:Timestamp"/>
      </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Figure 35 – Work Station Event Type included in the generic WSDL document

3. Workpiece Status Type

Similar to work station status type, the third type is workpiece status type show in Figure 36, every workpiece has a unique ID and should be identified by the station ID as well, as the workpiece is moving from work station to another. With time stamp for every change in the workpiece status. The workpiece status should be in processing – transferred in – transferred out – paused or no workpiece.

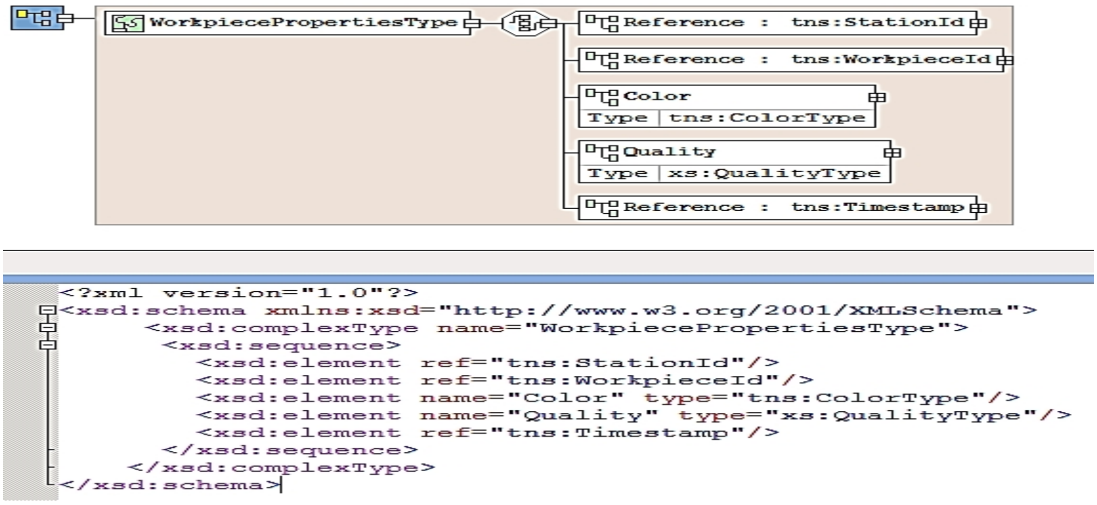

WorkpieceStatusType
Reference : tns:StationId
Reference : tns:WorkpieceId
Status
Type tns:WorkpieceStatusCode
Reference : tns:Timestamp

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:complexType name="WorkpieceStatusType">
      <xsd:sequence>
        <xsd:element ref="tns:StationId"/>
        <xsd:element ref="tns:WorkpieceId"/>
        <xsd:element name="Status" type="tns:WorkpieceStatusCode"/>
        <xsd:element ref="tns:Timestamp"/>
      </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Figure 36 – Workpiece Status Type included in the generic WSDL document

4. Workpiece Properties Type

The workpiece properties type is similar to the workpiece status type in the workpiece ID, the station ID and the time stamp. The different two elements are the work piece color which is either black or nonblack and the workpiece quality which either ok or defected. Figure 37 shows the workpiece properties type.

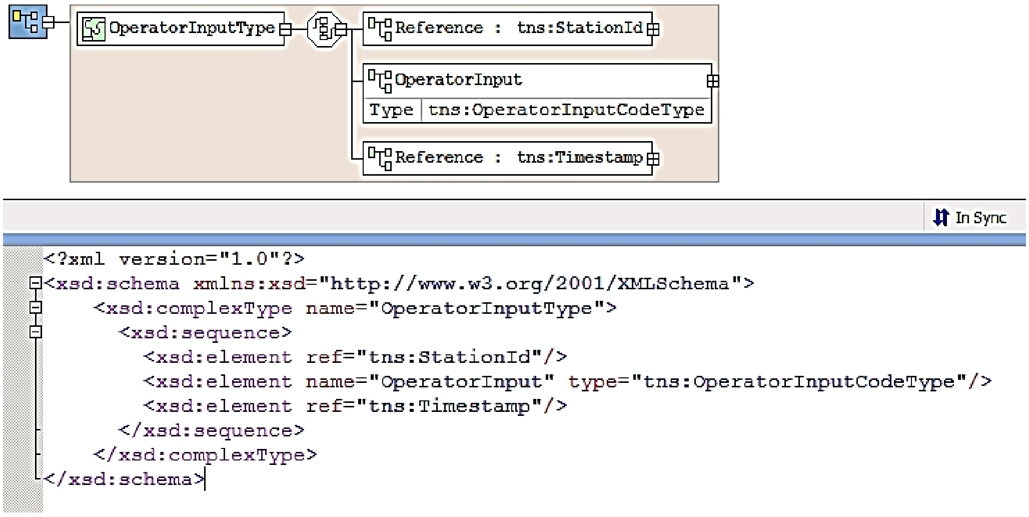

WorkpiecePropertiesType
Reference : tns:StationId
Reference : tns:WorkpieceId
Color
Type tns:ColorType
Quality
Type xs:QualityType
Reference : tns:Timestamp

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:complexType name="WorkpiecePropertiesType">
       <xsd:sequence>
         <xsd:element ref="tns:StationId"/>
         <xsd:element ref="tns:WorkpieceId"/>
         <xsd:element name="Color" type="tns:ColorType"/>
         <xsd:element name="Quality" type="xs:QualityType"/>
         <xsd:element ref="tns:Timestamp"/>
       </xsd:sequence>
     </xsd:complexType>
</xsd:schema>
```

Figure 37 – Workpiece Properties Type included in the generic WSDL document

5. Operator Input Type

The last type has been used in the WSDL document is the operator input type which shown in Figure 38 . The Operator input type can be manual start – manual stop – manual reset – alarm clearing – error clearing – warning clearing – blocking cleaning – starving cleaning. Also every operator should has a unique ID.


OperatorInputType
Reference : tns:StationId
OperatorInput
Type
tns:OperatorInputCodeType
Reference : tns:Timestamp
In Sync

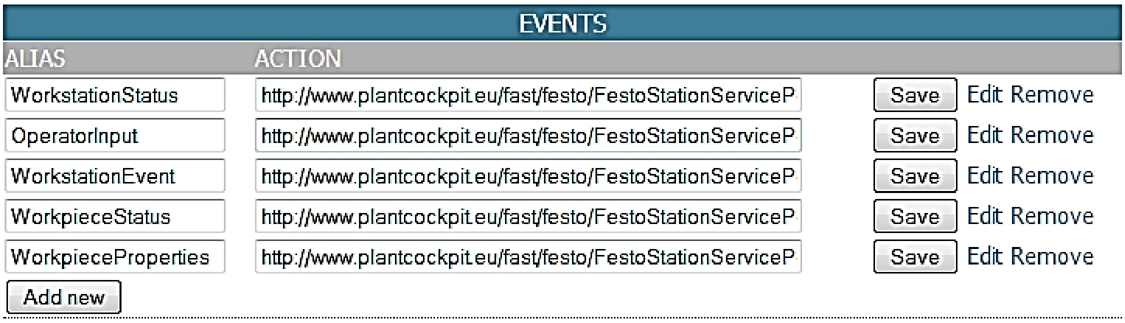
```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:complexType name="OperatorInputType">
      <xsd:sequence>
        <xsd:element ref="tns:StationId"/>
        <xsd:element name="OperatorInput" type="tns:OperatorInputCodeType"/>
        <xsd:element ref="tns:Timestamp"/>
      </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Figure 38 – Operator Input Type included in the generic WSDL document

After defining the types from the schema, the different messages can be defined based on those types. Three different kinds of messages can be defined for the Web-Services using INICO S1000

• **Event messages**: these are messages that are sent from the S1000 to one or more PC applications. Typically, these messages will report events such as the completion of a process, a significant change in a process parameter, or a fault/error/warning condition. In order to receive a copy of these messages, server application must subscribe to the S1000, so that an internal list of interested applications can be maintained. The number of subscribed applications for an event message can be zero, one, or many. [20]

The five events messages have been defined inside the INICO S1000 as shown in Figure 39



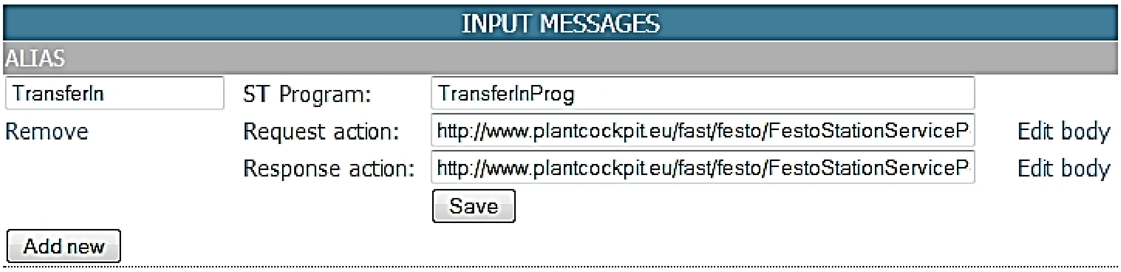
EVENTS

ALIAS	ACTION
WorkstationStatus	http://www.plantcockpit.eu/fast/festo/FestoStationServiceP
OperatorInput	http://www.plantcockpit.eu/fast/festo/FestoStationServiceP
WorkstationEvent	http://www.plantcockpit.eu/fast/festo/FestoStationServiceP
WorkpieceStatus	http://www.plantcockpit.eu/fast/festo/FestoStationServiceP
WorkpieceProperties	http://www.plantcockpit.eu/fast/festo/FestoStationServiceP

Figure 39 – The five event messaged definition inside INICO S1000

• **Input messages**: these are messages that are sent from the server application to the S1000. Typically, these messages will convey commands to execute a particular action, or can also be used to transfer configuration data. They can also be used to request data, such as the state of a process that is being executed or the value of a measured parameter. A response message (S1000 ➔ server application) can be configured, or not depends on the need of the response. [20]

There is one input request message and one input response message have been defined in input message section inside the INICO S1000 as shown in Figure 40



INPUT MESSAGES

ALIAS

TransferIn

ST Program: TransferInProg

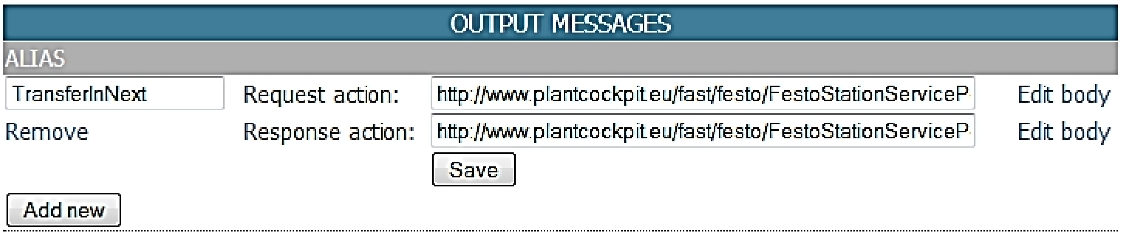
Request action: http://www.plantcockpit.eu/fast/festo/FestoStationServiceP

Response action: http://www.plantcockpit.eu/fast/festo/FestoStationServiceP

Figure 40 – The input message and its response definition inside the INICO S1000

The purpose of this message is to receive the workpiece ID coming from the previous station to keep tracking the workpiece and add new properties to it.

• **Output messages**: these are messages that are sent from the S1000 to a server application. As with input messages, they may or may not use a response message. [20]
There is one output request message and one output response message have been defined in output message section inside the INICO S1000 as shown in Figure 41



OUTPUT MESSAGES

ALIAS

TransferInNext | Request action: http://www.plantcockpit.eu/fast/festo/FestoStationServiceP | Edit body

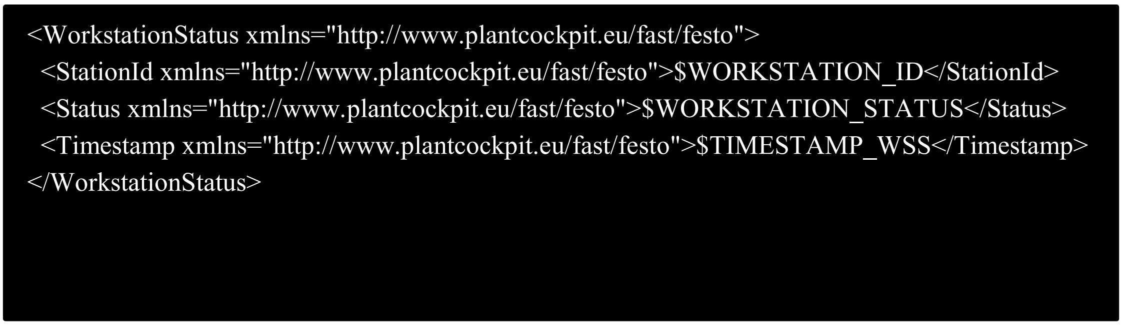
Remove | Response action: http://www.plantcockpit.eu/fast/festo/FestoStationServiceP | Edit body

Save

Add new

Figure 41 – The output message and its response definition inside the INICO S1000

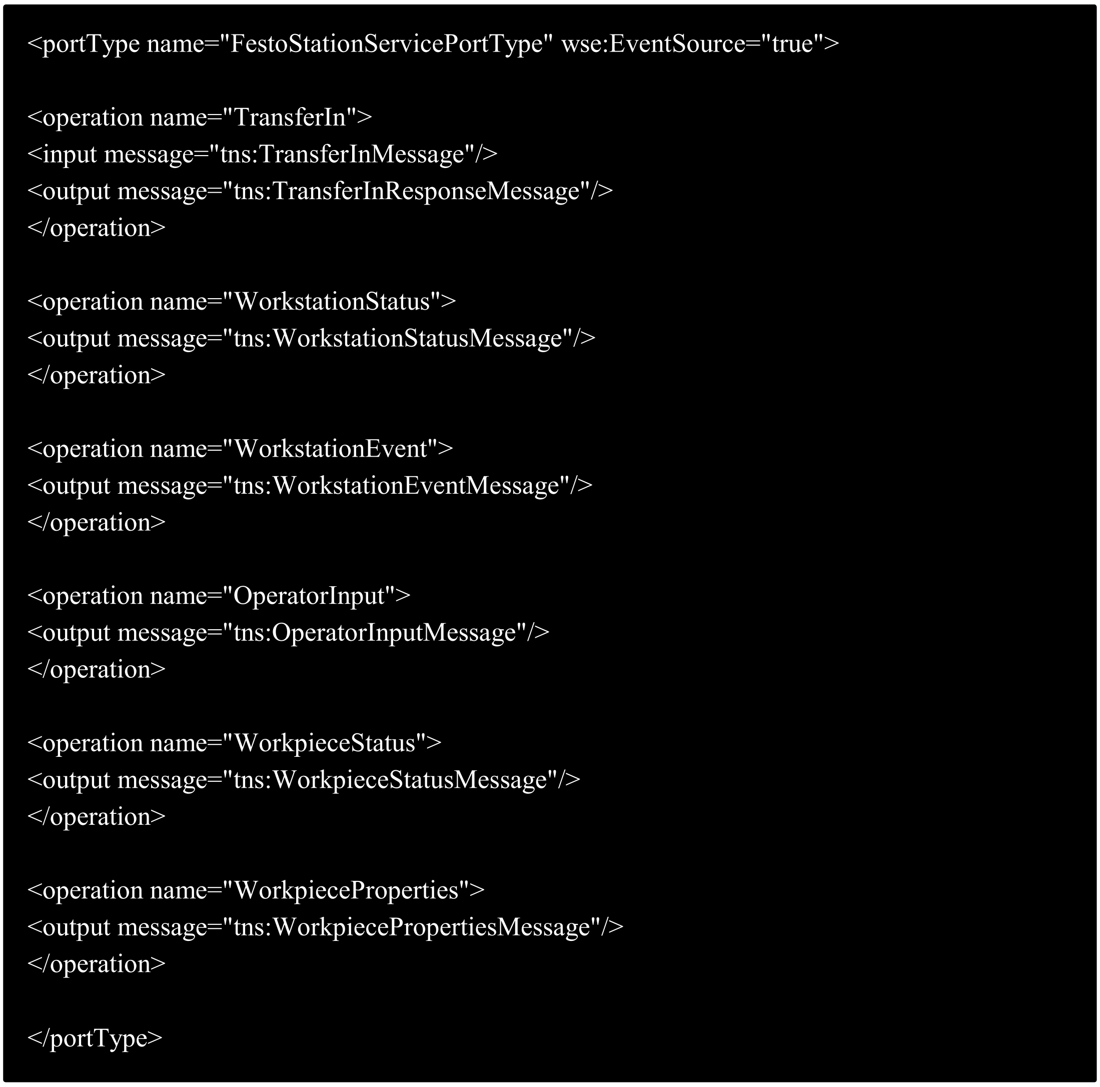
The purpose of this message is to send the workpiece ID to the next station to continue processing it
Regarding that every individual message of the previous either event, input or output messages should contains a XML instance of the previous types we discussed before
For example the WorkStationStatus message should look like the following XML format showed in Figure 42 inside the WSDL file



```
<WorkstationStatus xmlns="http://www.plantcockpit.eu/fast/festo">
  <StationId xmlns="http://www.plantcockpit.eu/fast/festo">$WORKSTATION_ID</StationId>
  <Status xmlns="http://www.plantcockpit.eu/fast/festo">$WORKSTATION_STATUS</Status>
  <Timestamp xmlns="http://www.plantcockpit.eu/fast/festo">$TIMESTAMP_WSS</Timestamp>
</WorkstationStatus>
```

Figure 42 – An example of one of the five event messages

As it is clear every element inside the XML message should have some parameter which defined inside the ST logic and the global variables of INICO S1000.Many instances can be generated from this message depends on the values of the inside variables
After all the messages are already defined, those messages will be combined into operations and can be gathered into so called port type as shown in Figure 43



```
<portType name="FestoStationServicePortType" wse:EventSource="true">

<operation name="TransferIn">
<input message="tns:TransferInMessage"/>
<output message="tns:TransferInResponseMessage"/>
</operation>

<operation name="WorkstationStatus">
<output message="tns:WorkstationStatusMessage"/>
</operation>

<operation name="WorkstationEvent">
<output message="tns:WorkstationEventMessage"/>
</operation>

<operation name="OperatorInput">
<output message="tns:OperatorInputMessage"/>
</operation>

<operation name="WorkpieceStatus">
<output message="tns:WorkpieceStatusMessage"/>
</operation>

<operation name="WorkpieceProperties">
<output message="tns:WorkpiecePropertiesMessage"/>
</operation>

</portType>
```

Figure 43 – The port type definition included inside the WSDL file

In this port definition we have only one operation called Transferin, however we have two messages are using the same operation. The first one is input message transfer to the station itself and the second one is the output message transfer from the station to the next one.

After defining the port types, the only thing left is to define the binding which is the protocol used to exchange the messages, which in our case is HTTP

The service can contain many ports. However in our project it contains only one port Publish subscribe SOAP message exchange pattern mode has been used during the implementation of the project. This because in this project many providers for similar services. Thus, it would be easier if there are many clients for those services to subscribe

whatever they want, depending on their need. In our case the client is any computer connected to the same network which all the INICO S1000 connected
Downloading the WSDL file on INCO S1000, will enable every INICO of publishing those services over the network. The server application should subscribe every needed service from every station over the case study. In designing of the our project, the Web-Service can be discovered dynamically, it means that any client on the network, has dynamic UDDI can subscribe those services from any machine, without any need for knowing in advance all the service providers – in our case every machine – to subscribe them. This concept has been explained in details at the end of the previous chapter.

Sending and receiving SOAP messages between the stations - Web-Service programing

The next step after downloading the generic WSDL document on every S1000 to program the controller itself to use the services included inside document. As it was shown in designing the WSDL, the core of the services is XML messages which include some parameters to be shown or subscribed by the service requestor.

As the INICO S1000 device is hardwired to SIMENS PLC for every station, all the sensory information and the PLC control actions can be acquired to the INICO S1000 device. By defining global variable inside the S1000, those values which already included inside the XML messages as it was already shown in Figure 42.

This message holds three global variables; $WORKSTATION-ID, $WORKSTATION-STATUS, and $TIMESTAMP-WSS.Those variables also can be changed from the ST logic inside the PLC.

For instance in this case, the time stamp should be changed every time by the ST logic every time for initializing this service. The most important part in the ST logic was to pass the workpiece ID, workpiece quality and color from station to another.

Every station INICO S1000 already has its own IP. By this mean definition of neighbours(s) of every station can be done inside the global variables of every S1000.

The following commands has been used to send, receive, respond or publish the messages. [20]

- WS_SEND(msg_alias, destination_address): an output message is sent to destination_address. Any variable data should be set/copied to the corresponding string variables before this command is invoked.

- WS_RESPOND(msg_alias): a response to an input/command message is generated and sent back. Any variable data should be set/copied to the corresponding string variables before this command is invoked.
- WS_GET_RESPONSE(msg_alias): Returns a response code according to the status of the output message transmission.

- WS_PUBLISH(msg_alias): an event message is generated and sent to every subscribed a server application. Any variable data should be set/copied to the corresponding string variables before this command is invoked.

For tracking the workpiece ID from station to another, a certain message exchange logic has been implemented. As in the very beginning of the process, whenever there is a new workpiece been fed into the distribution station, a random ID will be assigned to it. In same time, this ID will be broadcasted to all the stations. Every station has ID buffer or array composed from eight IDs, as the capacity of the case study is eight pallets.

The ID will be stored in the appropriate place of this software buffer, till the previous station send to the next station a workpiece output message and confirmed by workpiece input message. At this time the workpiece ID will move from the current station software buffer, take the place of the previous ID. The soap message exchange is shown in Figure 44

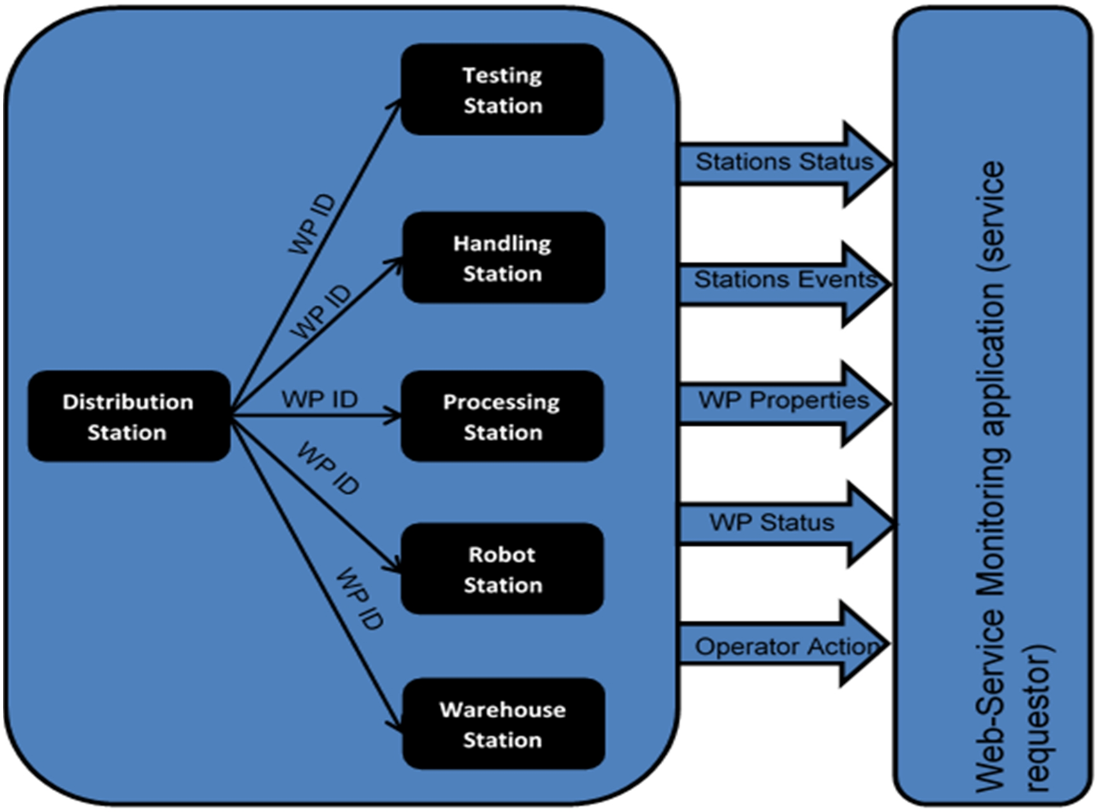


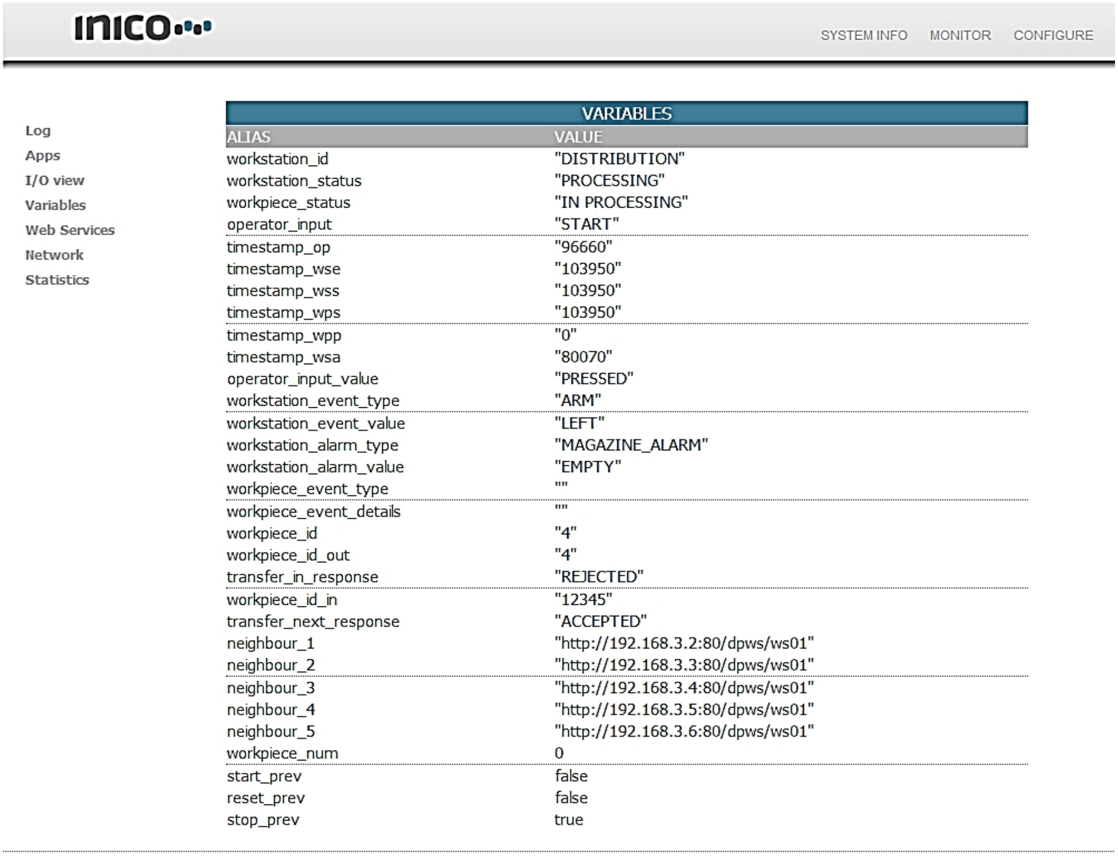
Figure 44 – SOAP messages exchange between the station and the monitoring server

Monitoring the Web-Services application

There are two different ways can be followed to monitor the Web-Services. The first way is to access any of the stations INICO S1000 by any PC connected to the same network .By any known browser.

By accessing the device IP, every INICO S1000 has a user friendly HMI, which enable the user to see online the changes in every variable included within every service.

Figure 45 shows the different values of the variables inside the INICO S1000 for the distribution station



VARIABLES	
ALIAS	VALUE
workstation_id	"DISTRIBUTION"
workstation_status	"PROCESSING"
workpiece_status	"IN PROCESSING"
operator_input	"START"
timestamp_op	"96660"
timestamp_wse	"103950"
timestamp_wss	"103950"
timestamp_wps	"103950"
timestamp_wpp	"0"
timestamp_wsa	"80070"
operator_input_value	"PRESSED"
workstation_event_type	"ARM"
workstation_event_value	"LEFT"
workstation_alarm_type	"MAGAZINE_ALARM"
workstation_alarm_value	"EMPTY"
workpiece_event_type	""
workpiece_event_details	""
workpiece_id	"4"
workpiece_id_out	"4"
transfer_in_response	"REJECTED"
workpiece_id_in	"12345"
transfer_next_response	"ACCEPTED"
neighbour_1	"http://192.168.3.2:80/dpws/ws01"
neighbour_2	"http://192.168.3.3:80/dpws/ws01"
neighbour_3	"http://192.168.3.4:80/dpws/ws01"
neighbour_4	"http://192.168.3.5:80/dpws/ws01"
neighbour_5	"http://192.168.3.6:80/dpws/ws01"
workpiece_num	0
start_prev	false
reset_prev	false
stop_prev	true

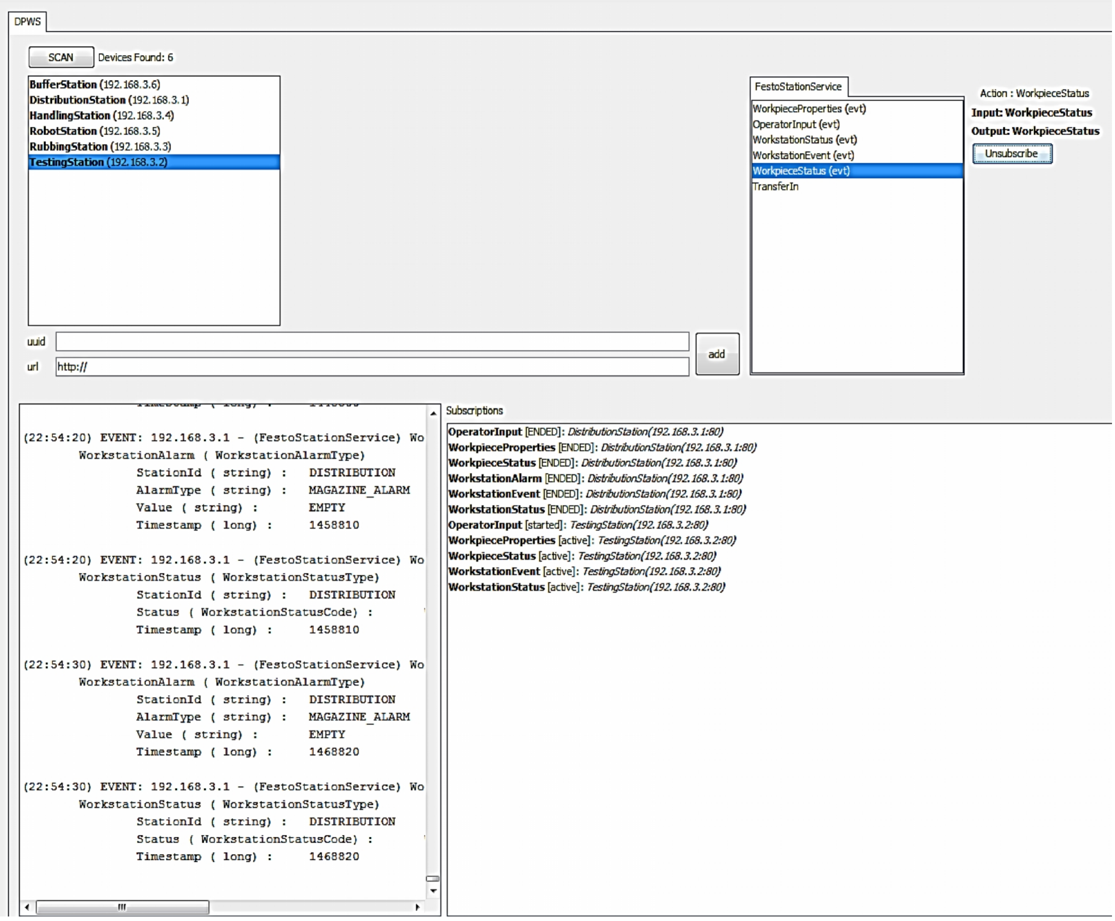
Figure 45 – INICO variables HMI for the distribution station

The Web-Services variables values for every station can be seen in Appendix 2

However the previous way to monitor the services need to watch different stations on different windows of the internet browser. Which is not so practical solution for monitoring.

Another tool was used to monitor all the services from all the stations in the same time, DPWS explorer, can discover any WSDL document running through any device in the network. When you run the tool over the server, all the service clients will be discovered automatically by the tool the same as it is clear in
Figure 46

In this figure, it is clear that every station from the six mentioned stations in our case study are discovered. Every station has a unique IP, and all of them offer the same services with different parameters values, as the WSDL document for all the station was generic from the very beginning. The user should subscribe the services he is interesting to monitor; different services from different stations can be monitored in the same time.
However, whenever the user is not interested in some services to monitor he can still unsubscribe them from the monitoring window



Devices Found: 6

BufferStation (192.168.3.6)
DistributionStation (192.168.3.1)
HandlingStation (192.168.3.4)
RobotStation (192.168.3.5)
RubbingStation (192.168.3.3)
TestingStation (192.168.3.2)

url: http://

WorkpieceProperties (evt)
OperatorInput (evt)
WorkstationStatus (evt)
WorkstationEvent (evt)
WorkpieceStatus (evt)
TransferIn

Action : WorkpieceStatus
Input: WorkpieceStatus
Output: WorkpieceStatus

```
(22:54:20) EVENT: 192.168.3.1 - (FestoStationService) Wo
        WorkstationAlarm ( WorkstationAlarmType)
                StationId ( string) :   DISTRIBUTION
                AlarmType ( string) :   MAGAZINE_ALARM
                Value ( string) :       EMPTY
                Timestamp ( long) :     1458810

(22:54:20) EVENT: 192.168.3.1 - (FestoStationService) Wo
        WorkstationStatus ( WorkstationStatusType)
                StationId ( string) :   DISTRIBUTION
                Status ( WorkstationStatusCode) :
                Timestamp ( long) :     1458810

(22:54:30) EVENT: 192.168.3.1 - (FestoStationService) Wo
        WorkstationAlarm ( WorkstationAlarmType)
                StationId ( string) :   DISTRIBUTION
                AlarmType ( string) :   MAGAZINE_ALARM
                Value ( string) :       EMPTY
                Timestamp ( long) :     1468820

(22:54:30) EVENT: 192.168.3.1 - (FestoStationService) Wo
        WorkstationStatus ( WorkstationStatusType)
                StationId ( string) :   DISTRIBUTION
                Status ( WorkstationStatusCode) :
                Timestamp ( long) :     1468820
```

OperatorInput [ENDED]: *DistributionStation(192.168.3.1:80)*
WorkpieceProperties [ENDED]: *DistributionStation(192.168.3.1:80)*
WorkpieceStatus [ENDED]: *DistributionStation(192.168.3.1:80)*
WorkstationAlarm [ENDED]: *DistributionStation(192.168.3.1:80)*
WorkstationEvent [ENDED]: *DistributionStation(192.168.3.1:80)*
WorkstationStatus [ENDED]: *DistributionStation(192.168.3.1:80)*
OperatorInput [started]: *TestingStation(192.168.3.2:80)*
WorkpieceProperties [active]: *TestingStation(192.168.3.2:80)*
WorkpieceStatus [active]: *TestingStation(192.168.3.2:80)*
WorkstationEvent [active]: *TestingStation(192.168.3.2:80)*
WorkstationStatus [active]: *TestingStation(192.168.3.2:80)*

Figure 46 – DPWS explorer Web-Services monitoring

CH5 – DESIGN AND IMPLEMENTATION OF THE EXPERT SYSTEM USING DROOLS SHELL

Artificial intelligence definition, branches and applications

Artificial Intelligence (A.I) is a growing branch of computer science that aims to extract the human way of thinking and trends of behaviour to comprehend it into a software program [22].A detailed definition for the artificial intelligence can be seen as following A.I. is the study of the computations that make the machine able to reason, perceive and act [23] to perform functions that require intelligence and experience of the human being [24]. These computation intents to express the people intelligence and senses into mathematical forms and logical models [25] .

A lot of branches are originally descended from Artificial Intelligence. Knowledge presentation and interference, common sense knowledge and reasoning, pattern recognition, search and logical AI, epistemology, ontology, heuristics and generic programming .One or more of the previous branches can be involved together to produce one application, the most common known applications of the previous branches of A.1.are Games, search engines, neural and fuzzy networks, speech recognition, image processing and computer vision and expert systems [26]. Figure 47 shows the most known common applications of A.I.

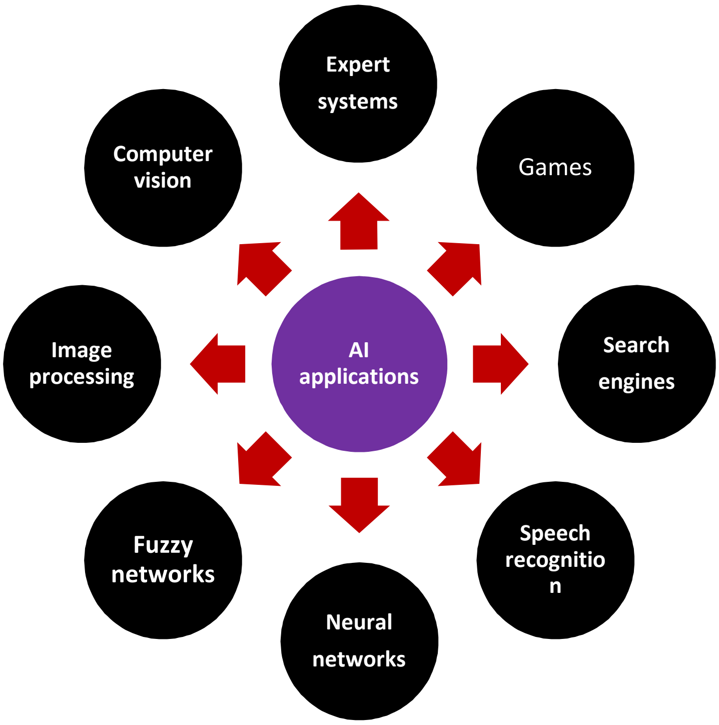


Figure 47 – Some common applications of artificial intelligence

Expert system definition and evolution

Expert system is one of the most attracting applications for the AI researchers. Expert system is a software program that tries to encapsulate the knowledge and experience of a human expert – in another words – it tries to mimic the knowledge and experience of a human expert. An expert system is designed to provide reasonable answers when given a set of conditions about the problem in hand. They are processing the symbolic information with non-algorithmic interference [27].

Expert system shell is that software program, which is encoded by a certain programming language, to contain all the knowledge or the facts that the expert feed to the system before the system run. Acquire new information from the system, process them and interfere new facts

Expert system principal appeared for the first time by researchers in the Stanford Heuristic Programming Project. Meanwhile in 1972 a programming language called Prolog invented by the French programmers have a great influence on the expert system technology, as the language tends to use first order logic to process the facts with so called rules form in a declarative form. The drawback of this language that it was not user friendly interfaced.

With the appearance of the IBM personal computer in the beginning of the nineteen eighties, the software companies started to produce more expert system software shells using programming languages like Lisp or Prolog to process symbolic facts.

Later there was no need for programming the expert system using very complicated symbolic programming language, as the companies developed their shells to use more plain text like language. New tools like Guru, Nexpert Object and VP Expert was the first to develop this kind of plain language. However the expert systems did not involve into the business operations till 1986, when at this year GSI-TECSI Software Company has introduced a more reliable expert system shell which includes its own knowledge base and use zero order logic to reason the facts and the premises.

Nowadays a lot of expert shells are used, every shell is different from the other in the way they can be programmed, and the way for reasoning the information. Some of those shells that still use till the moments are

- Prolog which has its own programming language and needs its own Prolog interpreter
- Amzi Prolog which can be programmed by C# and needs C# interpreter
- Clips and RT Expert which can be programmed by C++ and needs C++ interpreter
- Jess and Drools which can be programmed by Java and needs Java interpreter

Components of the expert system

Expert system is including many different main components [31], as it is shown in Figure 48.

The first component of the expert system is the expert human who has a lot of knowledge and experience about the system, which is going to be presented by the expert system software.

The second component is the knowledge base. This data base where all the factual and heuristic knowledge is stored.

Factual knowledge is the domain knowledge, which can be shared by all the objects with the same knowledge base domain.

Heuristic knowledge is the knowledge the expert system obtains from its experimental judgment of different situation. The learning curve of that kind of knowledge is growing exponentially due to the trend of the system, the heuristic knowledge can assist the expert to expect the system behaviour in the future, based on similar trends in similar situations of the system [32] .

One of the most common ways to present either the factual or heuristic knowledge is a unit presentation. Every unit included within the domain has a list of various properties That can be static or dynamic. Static properties are fixed values; however dynamic properties can have different values due to the behaviour of the unit. Also the unit can either be a parent of a child in the knowledge domain hierarchy, every child unit can inherit main characters from the parent unit.

For this reason, object oriented programming was a suitable tool to represent the facts inside the knowledge base as objects, using known object programming languages such as C++, C# or Java. Moreover some expert system shells support including rules inside the knowledge base.

Since the facts in the expert system knowledge base are not absolute - in other words – some facts cannot be sure completely it can happen. Thus expressing the fact in more accurate expert system will be based on the probability of every fact. One of the most famous ways used to express the uncertainty of the knowledge is the fuzzy presentation, which is a direct application of the artificial intelligence.

A knowledge acquisition subsystem will be always needed to collect the information either from the expert or from the working memory of the expert system shell

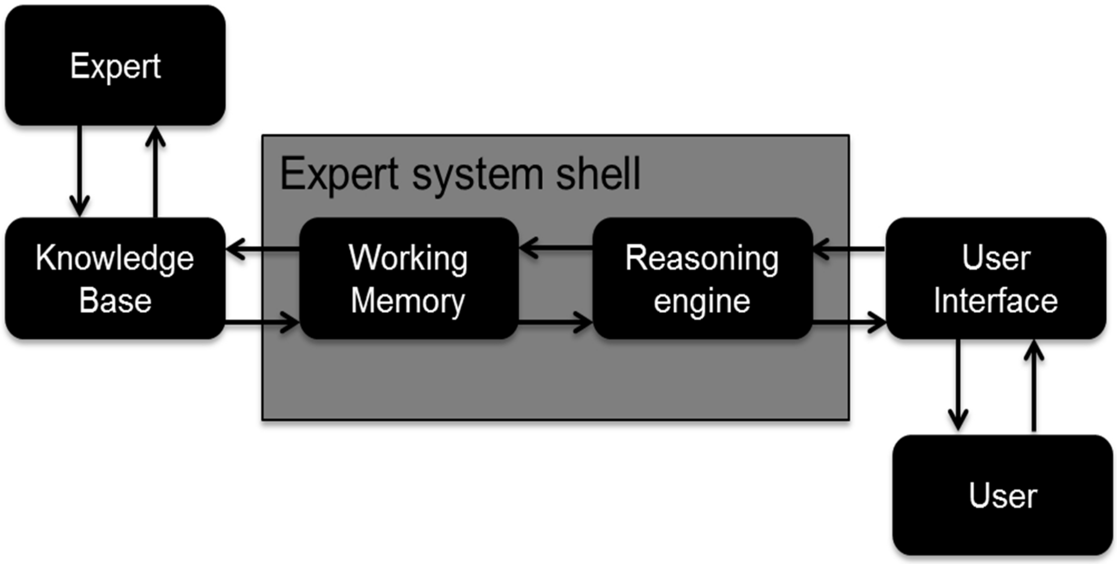


Figure 48 – Expert system components

The third component and the most important component of the expert system shell is the reasoning engine or the inference engine [32].The reasoning engine is a problem solver, which solves a certain problem based on a preprogramed problem solving model. It organises and controls the flow of the solving steps for the problem, and can approach every problem solution from different solution routes, smart reasoning engine will always try to optimize between the needed time and the processing power to find a problem solution route.

More smart way to provide a solution route using fuzzy or probability notations, as the knowledge itself in most of knowledge presentation data base can be presented on the probabilities bases, an uncertainty factor will be given for every fact and rule in the knowledge base. This can finally give many different solutions for the same problem with different probability values. The most common method the expert system shells use to drive the accumulated values of the probability of the final goal is Bayesian Belief Network [34].

The reasoning paradigm is formed from different rules. The rules can interact with one another and with the facts in the knowledge base, either to drive new facts or to find the final goal. One of the simplest and powerful common ways to present the rule is IF-THEN presentation. As the expert system acquires data from the system, in case the IF condition is matching with one of the acquired data from the expert system, it is said that the rule is fired, the THEN conclusion will be driven as a new fact. A new fact will be stored in the expert system.

All the instantaneous operations and data will be temporary stored in the expert system working memory. The reasoning engine will add or delete a fact instances from the working memory during the processing.

Three methods of reasoning are used in an expert system. Forward chaining reasoning, backward chaining reasoning and hybrid chaining reasoning.

In forward chaining reasoning, the expert system starts with the initial facts, and drives new facts whenever a rule matches a fact, the Expert system will go through a consequence of rules firing, till finding the answer of a certain goal, it is searching for.

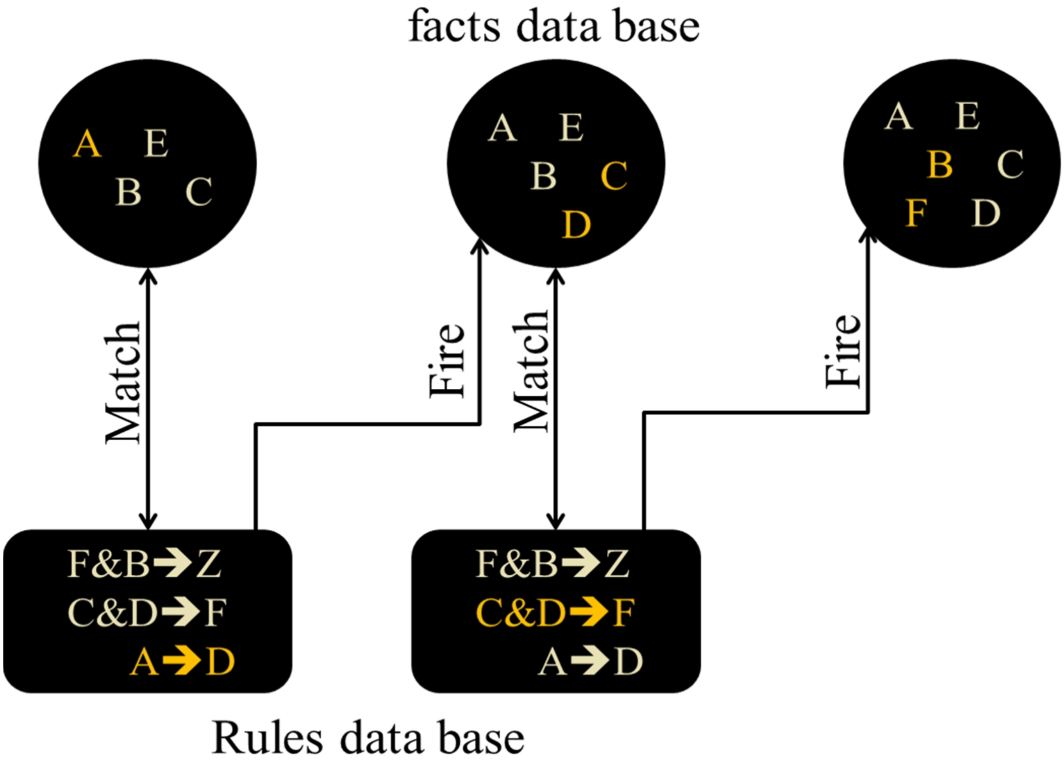


Figure 49 – Forward chaining reasoning mechanism

Figure 49 shows an example for the forward chaining reasoning mechanism, in this example. When the fact A achieved in the facts data base, that implied to fire the third rule in the rules data base, to drive a new fact D. The fact D will be stored in the facts data base again. In the next step when C and D are occurred in same time, the second rule will be fired. A new fact F will be stored in the facts data base. The reasoning engine will go through this iteration, till finding the final goal of the rules chaining.

The forward chaining reasoning assumes that all the initial facts are exists and the rules will be fired one after one till reaching one goal. However the reality is not that ideal, it can happen that either the initial facts are not exist in the very beginning or the rules will reach to a dead end track. This why another reasoning approach needed in such circumstances.

In backward chaining reasoning, the expert system will assume some hypothesis or a main goal, and then find which rule should be fired to achieve this hypothesis. It will discover a new other fact needed to reach this main goal. Thus a new sub-goal has been defined for the search engine, which will search for the rules to fire it. The system will go this way till find the very initial facts that achieve the main goal. In same time the rule path already will be known.

To figure out more about the idea of the backward chaining reasoning, Figure 50 shows an example how the mechanism works.

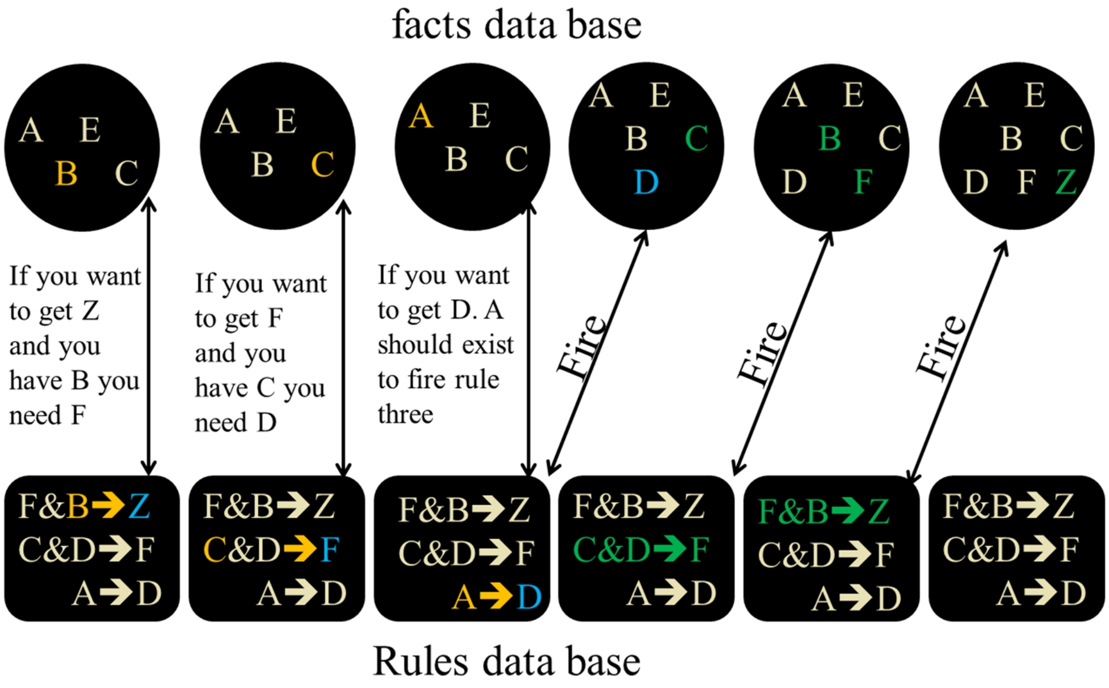


Figure 50 – Backward chaining reasoning mechanism

In the previous example, the expert system starts by defining its main goal. In our case, the main goal is the first rule conclusion which is Z. The reasoning engine will find out that the rule that imply Z needs both F and B, when the reasoning engine searches the facts data base, it will find that B is exist but not F. Thus it will create a new sub-goal, and will go to the rules data base to search for the rule which should be fired to obtain this new sub-goal. The reasoning engine will go through those iterations, till find the initial fact that will fire all the rules consequently to reach the main goal Z.

Both of the forward chaining reasoning and the backward chaining reasoning has advantages and drawbacks. Forward chaining reasoning is a rule driven track following, however the backward chaining reasoning is a goal driven track following.

Both of the two methods can work faster and more efficient, depending on the givens and the goals, the nature of the rules triggering events and the user way of thinking.

In case there are a big number of goals to be achieved in the system, in same time, the initial facts are limited and known. And there are a lot of rules paths which finally will obtain the same goals; forward chaining reasoning will work faster and efficient in this case.

However in the opposite case, if the number of goals is limited and the number of the initial facts are limited. There are definite rule paths to reach every goal; backward chaining reasoning will be faster and more efficient.

From the triggering event point of view, if the triggering of the rule will occur due to the arrival of a new fact, forward chaining reasoning will be suitable to be used. However in case of arriving of a query will trigger the rule, backward chaining will be more viable Finally the user nature of thinking will help to indicate which kind of reasoning is more suitable; majority of people will spontaneously use forward chaining reasoning to solve a problem.

Therefore, some expert system shells prefer to use both of forward and backward chaining reasoning, this called a hybrid reasoning engine. It is more complicated reasoning engine as it hides another reasoning tasks in behind. To judge which reasoning method will be used, which is not that easy task. And can waste time in some cases more than saving it [35].

The applications of expert systems

The applications of the expert systems technology can be wide, depending on the function needed by the expert system. They can expend from Microsoft word spelling and grammar correcting, to NASA design and assembly expert system to build a space shuttle The applications are implemented into three different domains. The industrial domain, the social domain and the commercial and financial domain.

Procedure planning and scheduling expert system

This can be considered one of the most common clusters of the expert systems. It can be fall also under system analysis cluster, as the expert system to define a set of interacting goals, try to find an optimum solution to reach the system goals.

Optimising the time, the effort, the distance or combinations of them, could be the goal of this expert system.

The procedure planning and scheduling expert system is more involved in commercial and social life section. Such as air plans scheduling system or the underground train or buses advising system.

Process monitoring and control expert system

This cluster of the expert systems mainly used in industrial section. The most important feature about the process monitoring and control expert system is the real time data analysis, the real time boundaries in every expert system is different from the other, depending on the target industry.

The control goal of the expert system should be to prevent the system failure, in addition of increasing the productivity of the system. The most direct applications of this expert system can be found in oil refinery, metal mining and steel manufacturing industries. Those industries are very accurate with high safety factors, which tells that process monitoring and control expert system is very reliable in this field.

Diagnosis and troubleshooting expert system

This expert system is applied either in industrial or social domain. In the industrial field most probably it will be integrated module for the process monitoring and control expert system. The expert system will always try to prognosis, diagnosis and solve the system errors, compromise the performance in case of a device malfunction to obtain hardware configurability.

However, this application of the expert system was more elite in the social domain. EMYCIN Medical expert system can be considered the first successful expert system [36], the goal of the system was to diagnosis the people disease without a doctor, especially in the low population areas of the world.

Design and manufacturing expert system

This expert system also located in the industrial field expert systems, used to help the manufacturing in the product design, starting from the high level conceptual design, ends with the detailed.

Product assembly expert system

Product assembly expert system can be either considered one form of the previous type of the design and manufacturing expert system, or an integrated module with it. It is one of the most important and famous expert system application exists, as it has been used by the McDeromtt Company starting from 1981 to facilitate the manufacture of semi-custom minicomputers, then most of the computer manufacturing companies used the same technique. Later, the technique has been implemented in other industries such as modular home building.

Financial decision making expert system

The financial market is a vigorous client of the financial decision making expert system, as the nature of the financial system is already build on the probabilities and uncertainties. The expert system can give many recommendations for people who work in stock or foreign exchange market to decide the buying and the selling decisions, for the bankers to decide the loan giving and for the insurance companies to assess the risk.

Knowledge publishing expert system

This can be considered the most new application of the expert system. The Knowledge publishing expert system is belonging to the social domain expert systems. The main goal of this expert system is to recommend and correct the user knowledge. The most famous expert system from this type is the error correction on the word programs or suggestion of the words while writing a text on a cell phone.

Advantages of the Expert system

- Speeding up the learning curve for the human, to candidate him or her to an expert in his domain, the learning factor can be start from 10 to 100 faster than the normal teaching curve
- Expert system can work around the clock and used by many users in same time
- As the main target of the expert systems is mostly to optimize the solution, they can save a big amount of money for the big firms, by saving the effort and the time
- Improving the quality, the performance and the decision making
- Capturing the instantaneous experiences from the system and feeding it to the knowledge base, that knowledge can be wasted away without existing of an expert system
- Avoiding of the experience losing in case of the human expert left his position within the firm
- In hazardous working environment, expert system will be obligoutly solution for the human expert
- Consistency as the final decision of the expert system will be far away from emotions and other factors like exhaustion for instance

Disadvantages of the Expert system

- Decision error probability due to lack of the common human sense in decision making
- Not fast respond to the unexpected situation as the human expert
- Expert system cannot investigate the original feeding knowledge in the knowledge base, which can create a wrong decision making
- Cannot adapt to the environment change, unless the knowledge base changed
- Unlikely to come up with a creative solution
- Most expert systems are menu driven which does not deal very well with ambiguous problems [37]

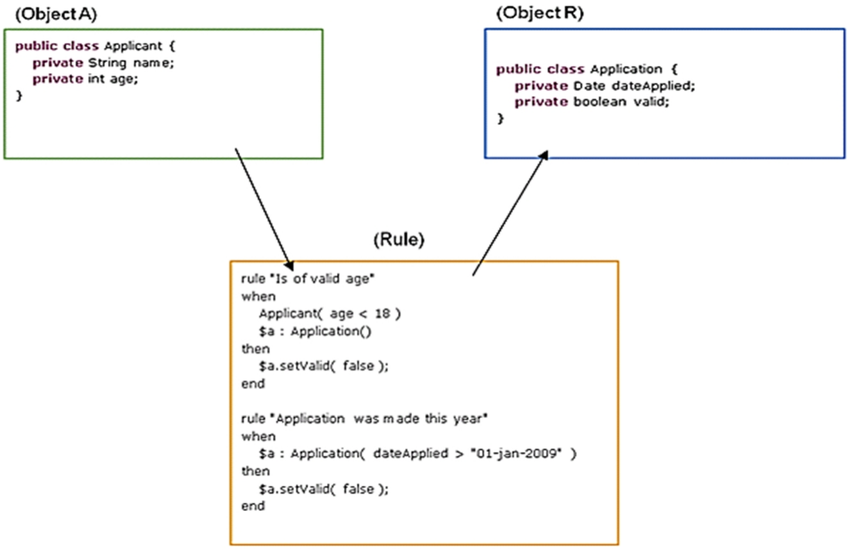
Drools expert system shell

Drools is a business rule management system (BRMS) has been developed by jboss organisation used to be known as a production rule system. A production rule system originates from formal grammars where it is described as "an abstract structure that describes a formal language precisely, i.e., a set of rules that mathematically delineates a (usually infinite) set of finite-length strings over a (usually finite) alphabet [38].

Business Rule Management Systems (BRMS) build additional value on top of a general purpose Rule Engine by providing business user focused systems for rule creation, management, deployment, collaboration, analysis and end user tools. Further adding to this value is the fast evolving and popular methodology "Business Rules Approach", which is a helping to formalize the role of Rule Engines in the enterprise.

Drools is a forward chaining interference engine. Use Rete algorithm for pattern matching The Drools Rete implementation is called ReteOO, signifying that Drools has an enhanced and optimized implementation of the Rete algorithm for object oriented systems. It combines Object Oriented (OO) Paradigm entities with rules in order to let them interact in a transparent way [39].

The following Figure 51 shows the interactivity between object entities and the objects in Drools are classes written in Java language, the rules are a text like sequence. Every rule is composed from an IF condition and THEN conclusion. Due to the value of one or more variables of any object, the rule will be fired. When the rule will be fired the conclusion part most probably will assign some value to one or more object variable value.



```
public class Applicant {
   private String name;
   private int age;
}
```

```
public class Application {
   private Date dateApplied;
   private boolean valid;
}
```

```
rule "Is of valid age"
when
   Applicant( age < 18 )
   $a : Application()
then
   $a.setValid( false );
end

rule "Application was made this year"
when
   $a : Application( dateApplied > "01-jan-2009" )
then
   $a.setValid( false );
end
```

Figure 51 – Basic interactivity between Objects and Rules

Drools implementation over Apache ServiceMix

In our project, the written rules inside Drools rule engine should be fired based on the XML messages coming from the services we created in the last chapter. For this task we will need an enterprise service bus (ESB), An ESB is a software architecture model used for designing and implementing the interaction and communication between mutually interacting software applications in service-oriented architecture.

Apache ServiceMix is an open source enterprise service bus has been used in this project to be the environment that Drools can communicate with XML messages

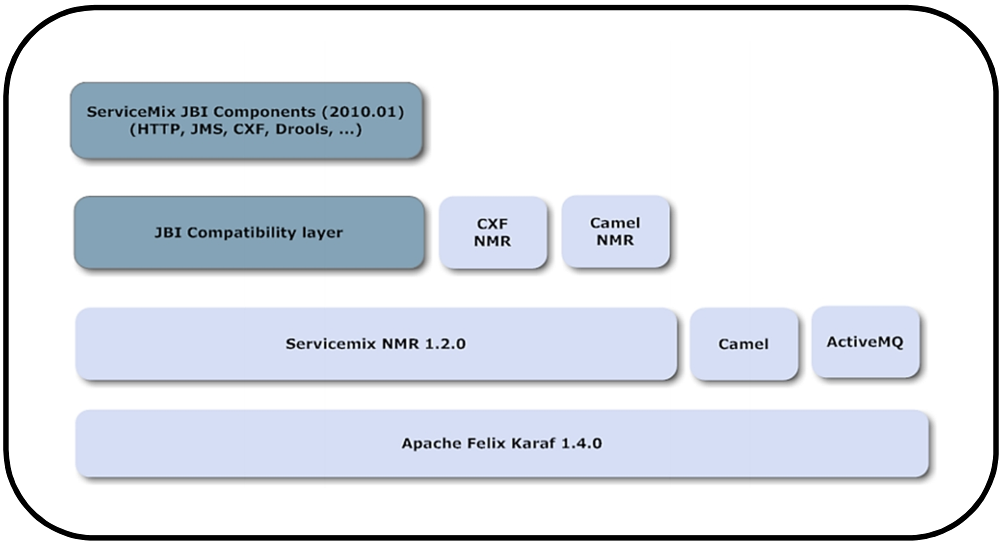


Figure 52 - Apache ServiceMix platform

Figure 52 shows the other frameworks that integrated inside Apache ServiceMix

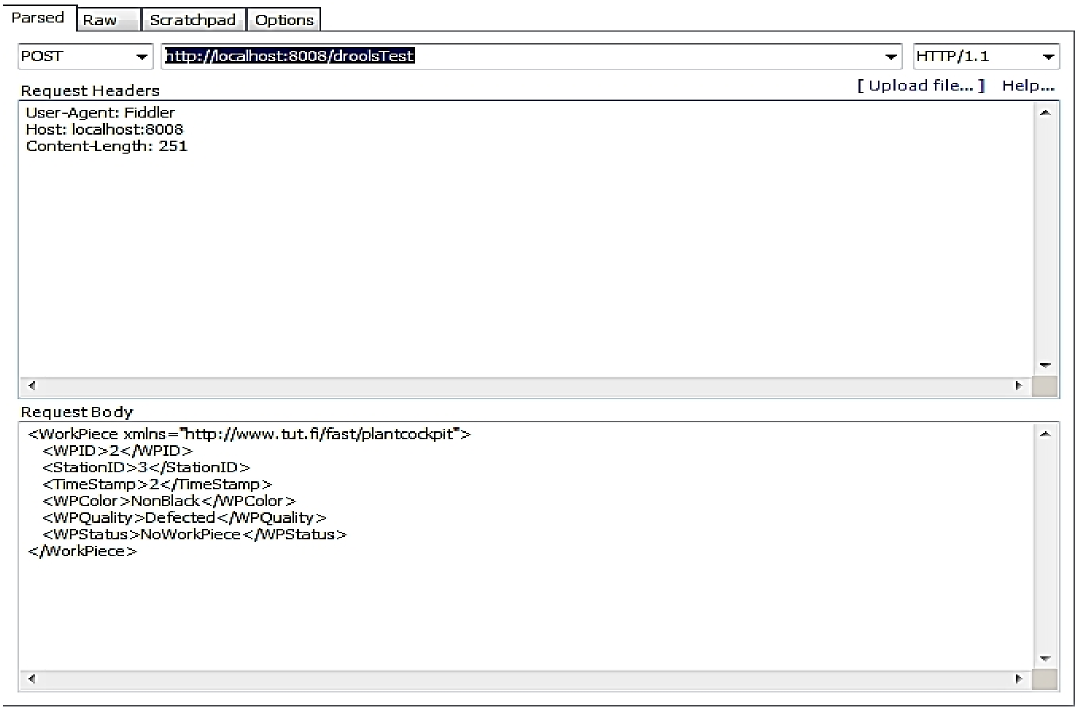
The most important two frameworks for us in this project are Apache Camel and service Java Business Integration (JBI) layer.

Apache Camel is a rule-based routing and mediation engine which provides a Java object-based implementation of the Enterprise Integration Patterns using Java DSL, spring XML DSL or Scala DSL. In our project Apache Camel will act as the gate to communicate with Drools or the event consumer end point.

The other important framework is Java Business Integration which contains Drools as one component of it. JBI is a frame using the WSDL message exchange model for system interoperation, and contains Drools as a reasoning engine for business applications.

The last component that we will deal with in implementation of the project is an even producer client, to simulate the XML messages coming from the machines, the purpose of using the client are to save the effort and the time of the process. However, the messages generated by the client is typically the same we receive from the machines in our case study

For this task Fiddler software has been, Fiddler can send/receive the XML over HTTP protocol. The interface of Fiddler client is shown in Figure 53



POST http://localhost:8008/droolsTest HTTP/1.1

Request Headers

```
User-Agent: Fiddler
Host: localhost:8008
Content-Length: 251
```

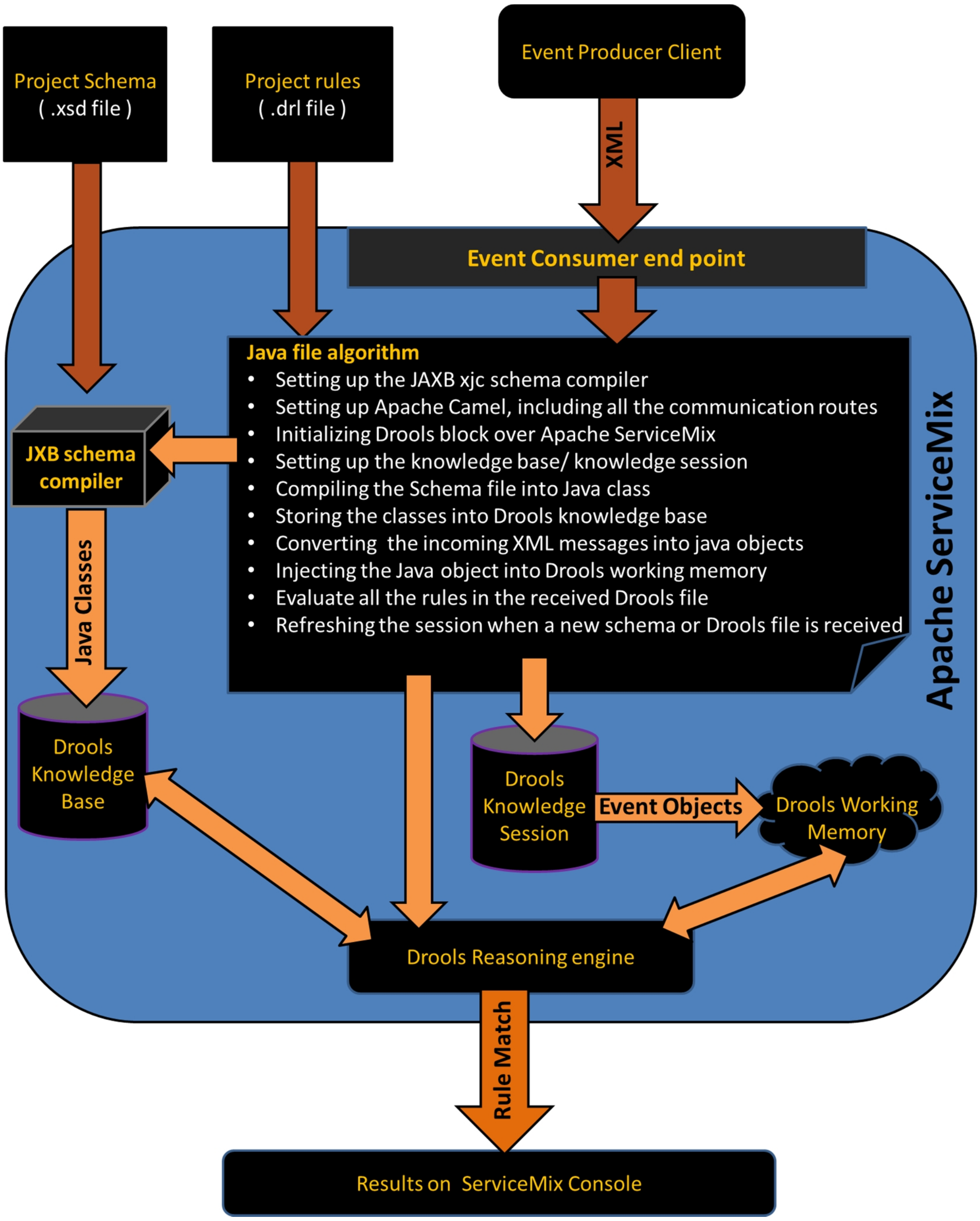
Request Body

```
<WorkPiece xmlns="http://www.tut.fi/fast/plantcockpit">
   <WPID>2</WPID>
   <StationID>3</StationID>
   <TimeStamp>2</TimeStamp>
   <WPColor>NonBlack</WPColor>
   <WPQuality>Defected</WPQuality>
   <WPStatus>NoWorkPiece</WPStatus>
</WorkPiece>
```

Figure 53 – Sending XML messages to Apache ServiceMix using Fiddler client

Figure 54 shows a flow chart to the implementation mechanism of the Drools rule engine over Apache ServiceMix.

After launching Apache ServiceMix, it will execute the algorithm included in the project.jar



Project Schema (.xsd file)
Project rules (.drl file)
Event Producer Client
XML
Event Consumer end point
Java file algorithm
• Setting up the JAXB xjc schema compiler
• Setting up Apache Camel, including all the communication routes
• Initializing Drools block over Apache ServiceMix
• Setting up the knowledge base/ knowledge session
• Compiling the Schema file into Java class
• Storing the classes into Drools knowledge base
• Converting the incoming XML messages into java objects
• Injecting the Java object into Drools working memory
• Evaluate all the rules in the received Drools file
• Refreshing the session when a new schema or Drools file is received
JXB schema compiler
Java Classes
Apache ServiceMix
Drools Knowledge Base
Drools Knowledge Session
Event Objects
Drools Working Memory
Drools Reasoning engine
Rule Match
Results on ServiceMix Console

Figure 54 – Implementation Mechanism of Drools engine over Apache ServiceMix

First the algorithm is initializing and configuring the different frameworks
The first framework that should sit up is the JAXB compiler, the rule of JAXB in our project is to convert the schema has been done for building the Web-Services schema to Java classes. This way we will be able to deal with the Web-Services messages as objects or instances from those classes.

The second task is setting up Apache Camel event consumer communication routes to be able to receive the XML messages from the events producer
The third task is to launch Drools framework inside Apache ServiceMix and prepare the knowledge basement and session of the expert system, to be ready of storing the Java classes whenever it will be converted.

In the beginning, the schema file (project.xsd) should be loaded manually to ServiceMix deployment folder, and every time a modification or change is happened to it. When the schema file will be loaded, all the objects will be complied by the JAXB complier to Java classes and stored in Drools knowledge basement.

Also the rule file (project.drl) should be manually loaded to ServiceMix deployment folder, in the beginning and every time a modification or change is happened to it. When the rules file will be loaded, all the rules should be loaded to the rule engine and ready to execute them, whenever receiving an event matching one condition part of those rules written inside.

Every time the event producer sends a XML message, the java file should convert them to Java instances of the classes have been stored in the knowledge basement, and inject them into Drools working memory crossing by the knowledge session.

When Drools reasoning engine detects a match case between one of those objects and the condition part in one of the rules, it will fire this rule and show the result on Apache ServiceMix console.

Finally the algorithm should refresh the working memory, the knowledge basement, the knowledge session and Drools reasoning engine. Every time either the schema or the rule file be loaded to Apache Web-Service deployment folder. Please refer to appendix 4 for reading the Java code that contain the previous algorithm.

Programming of the rules based on key production indicators (KPI)

Key production indicators (KPI) are a standard method to measure the performance and the quality of the production or the process. On other words, evaluation of how much the organization is successful. Usually success of an organization is defined in terms of achieving progress to a certain strategic goals. Those goals can be divided into many sub-goals; every sub-goal can has its own KPIs, this way KPIs can be divided into [39]

- **Quantitative indicators** which can be presented with a number.
- **Qualitative indicators** which can't be presented as a number.
- **Leading indicators** which can predict the future outcome of a process
- **Lagging indicators** which present the success or failure post hoc
- **Input indicators** which measure the amount of resources consumed during the generation of the outcome
- **Process indicators** which represent the efficiency or the productivity of the process
- **Output indicators** which reflect the outcome or results of the process activities
- **Practical indicators** that interface with existing company processes.
- **Directional indicators** specifying whether an organization is getting better or not.
- **Actionable indicators** are sufficiently in an organization's control to affect change.
- **Financial indicators** used in performance measurement and when looking at an operating index.

A very important point to mention, is that Drools is just an expert system software, it can be used in many different ways, and it can process many different level of details, depends on the main concept of design that it was implemented under. Our main concept of design is to use Drools as an expert system for layer 3 in ISA-95 which is concerning the manufacturing operation management.

This point will directly reflect on designing the KPIs of Drools. The same deployment of Drools can be used in level of ISA-95, in this case, different set of KPIs definition and rules should be formulated.

In our case study we selected a few of every of the mentioned KPIs, to measure the whole over performance on the management level. The main point was to proof that we can program those KPIs with Drools ES. For this reason, one object – the workpiece - in our Schema has been selected to process its related KPIs.

Those KPIs are

1- Rate of accepted production per hour = $\frac{\text{number of undefected products from the line}}{\text{hour}}$

2- Rate of accepted production per target = $\frac{\text{number of undefected products from the line}}{\text{target}}$

3- Rate of rejected production = $\frac{\text{number of defected products from the line}}{\text{hour}}$
per hour (scrap)

4- Rate of rejected production per target = $\frac{\text{number of undefected products from the line}}{\text{target}}$

5- Rate of black workpieces per hour = $\frac{\text{number of black workpieces from the line}}{\text{hour}}$

6- Rate of nonblack workpieces = $\frac{\text{number of non black workpieces from the line}}{\text{hour}}$
per hour

7- Average undefected product units time per hour =

$$\frac{\text{an hour}}{\text{number of undefected products from the line in an hour}} = \frac{1}{\text{Rate of undefected production per hour}}$$

8- Average real product unit cost per hour =

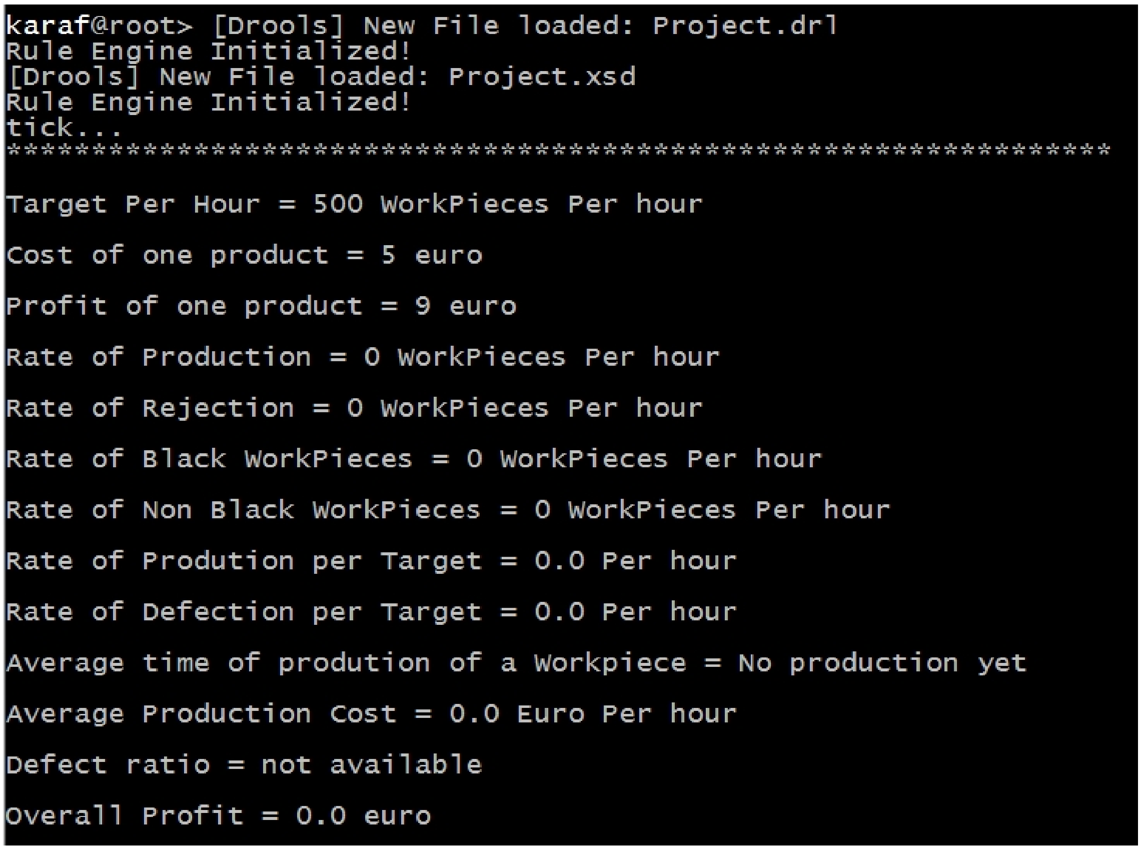
$$\text{Rate of production per hour } \times \text{cost of the product unit}$$

9- DPMO (Defects per million opportunities) or PPM (part per million) = expressing the quality of the production line = $\frac{\text{Rate of rejection per hour}}{\text{Rate of production per hour}}\ \chi\ 10^6$

10- Profit per unit per hour =
Average real product unit cost per hour – Average cost for producing the product per hour

Drools deployment over Apache ServiceMix

The initialization of Drools expert system, after loading the project schema and rule file (project.xsd and project.drl) is shown in
Figure 55

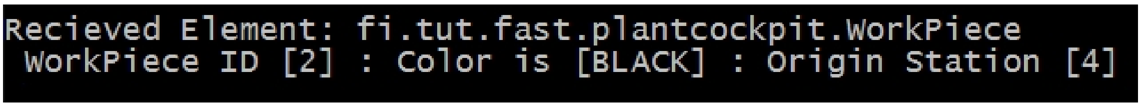


```
karaf@root> [Drools] New File loaded: Project.drl
Rule Engine Initialized!
[Drools] New File loaded: Project.xsd
Rule Engine Initialized!
tick...
*****************************************************************

Target Per Hour = 500 WorkPieces Per hour

Cost of one product = 5 euro

Profit of one product = 9 euro

Rate of Production = 0 WorkPieces Per hour

Rate of Rejection = 0 WorkPieces Per hour

Rate of Black WorkPieces = 0 WorkPieces Per hour

Rate of Non Black WorkPieces = 0 WorkPieces Per hour

Rate of Prodution per Target = 0.0 Per hour

Rate of Defection per Target = 0.0 Per hour

Average time of prodution of a Workpiece = No production yet

Average Production Cost = 0.0 Euro Per hour

Defect ratio = not available

Overall Profit = 0.0 euro
```

Figure 55 – initialization of Drools expert system

The values of the KPIs are zeros before receiving any XML message from the client, only the constant values like the cost of the product, the profit of one product and the target per hour.

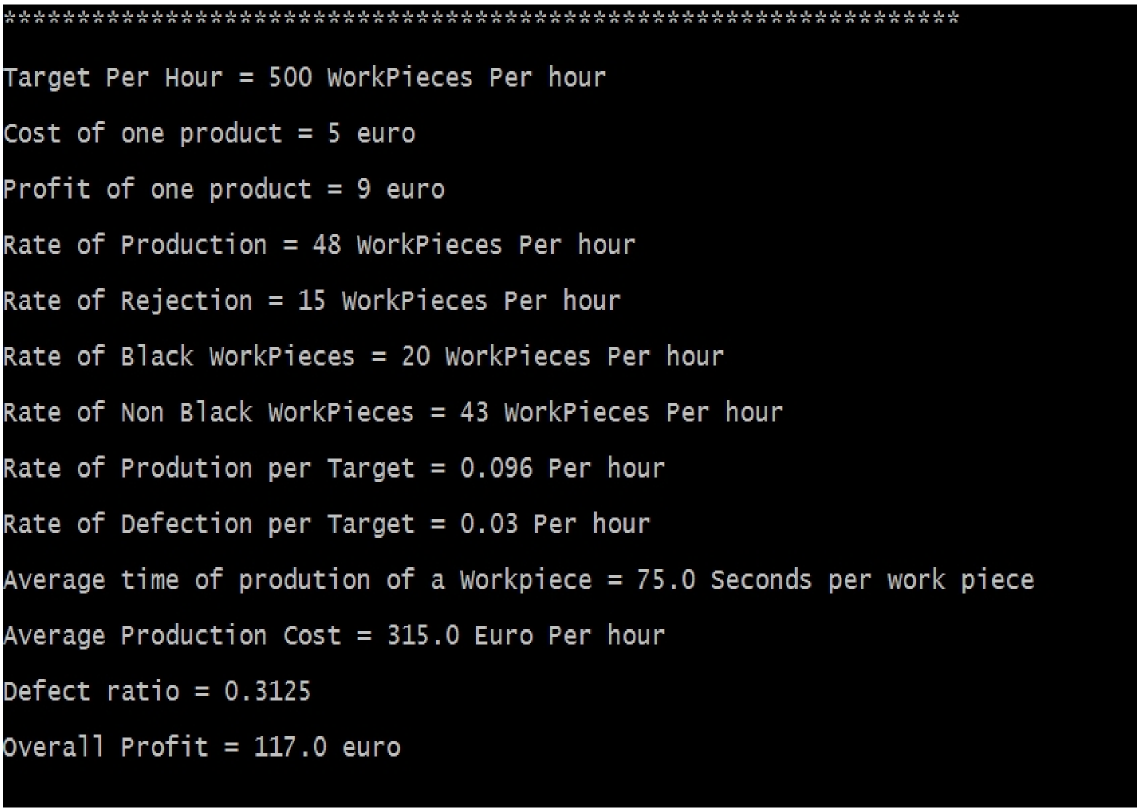
Whenever a XML message from Fiddler matches the rule condition, it will be fired as showing in Figure 56, indicating the main element in the received XML message on apache ServiceMix console



```
Recieved Element: fi.tut.fast.plantcockpit.WorkPiece
 WorkPiece ID [2] : Color is [BLACK] : Origin Station [4]
```

Figure 56 – Firing a rule after receiving of XML message from Fiddler client

Drools will use internal timer included in the project rule file to show the results of the KPIs every pre-defined period as it is shown in Figure 57.



```
*****************************************************************

Target Per Hour = 500 WorkPieces Per hour

Cost of one product = 5 euro

Profit of one product = 9 euro

Rate of Production = 48 WorkPieces Per hour

Rate of Rejection = 15 WorkPieces Per hour

Rate of Black WorkPieces = 20 WorkPieces Per hour

Rate of Non Black WorkPieces = 43 WorkPieces Per hour

Rate of Prodution per Target = 0.096 Per hour

Rate of Defection per Target = 0.03 Per hour

Average time of prodution of a Workpiece = 75.0 Seconds per work piece

Average Production Cost = 315.0 Euro Per hour

Defect ratio = 0.3125

Overall Profit = 117.0 euro
```

Figure 57 – Periodic calculations of KPIs

CH6 – DISCUSSIONS OF THE RESULTS, CONCLUSION AND FUTURE WORK

Discussion of the results

During this thesis work the following points was excessively highlighted. So it would be a good idea to discuss them to see their direct impact on the project work

ISA-95 model recommendations

From the very beginning of the thesis work, model of ISA-95 standard was the main skeleton for creating the project concept design. The standard was used basically to divide the different phases of the project work, understanding the limits of every phase either from the tasks point of view or from the time domain prospective.

ISA-95 model can be defined as a strategic project plan for most of the automation premises.

CAMX IPC- 2541 standard

The next standard has been used in an indirect way during the thesis work was CAMX IPC- 2541. Even CAMX standard meant to be dedicated for generic requirements for electronic shop floor equipment communication messages, but during this thesis work we used it as a reference guide to define the communication messages on our case study.

The idea of using CAMX analogy came from the question, why shall we use unknown new message formatting, if we with little modification to CAMX standard will be unsure about its results

Web-Service technology

During the thesis work it has been shown that, the Web-Service technology is very successful to upgrade the old PLC automation technology, to more flexible and easy to configure from hardware or software prospective.

Also it was proven by the results of chapter four, that it could be one of the best ways to conduct the industrial information to higher business processing level

Object oriented schema

A new concept has been introduced during this thesis work in designing the Web-Service XML schema, from the point of view of object oriented programming. This methodology was the main technical core of thesis work, as all the communication on the Web-Service level was based on this concept.

The project Schema has been designed once in a generic way. Every machine, workpiece or labourer in the case study was only an instance of these generic objects. That saved a lot of time and effort in customized programming and downloading the projects on the different hardware. Also the approach itself eased the communication process.

Moreover the implementation of Drools expert system depended on this analogy between the XML Schema and the Java language programming. Converting the project Schema using JAXB to Java objects and store them into Drools knowledge base was a smart way to save the programming time and effort. The same issue for converting the XML messages coming from Fiddler client into Java instances to fire the rules.

Another benefit beyond saving the programming time and effort was to guarantee the compatibility and to support integrity between the Web-Service messages and Drools reasoning engine framework.

Business rule management system – Drools BRMS

It is very important to emphasize, that we used Drools during this thesis work in layer 3 of ISA-95 model, Layer 3 of the model concern, manufacturing operation management, the level of details we need to monitor and process due to the standard recommendations ,should not step outside that domain to the next layer of the ISA-95.

While, the same implementation of Drools can be used in layer 4 of ISA-95 to give more extensive details about the business process. The main goal of using Drools was not to establish a prefect manufacturing operation management system, which is transacting every small action happened in the automation level into a conclusion. But to structure an expert system able to do that, to proof that Drools is a perfect choice for supporting this task. In the last chapter of this thesis work it has been shown that Drools BRMS is a very flexible and strong expert system, which can easily integrated with the Web-Service technology

During the implementation of the expert system, it was taken into consideration one type of the objects we have in the project Schema, which is the workpiece, success of analysis the workpiece messages on the business level can be done.

Monitoring the results of the expert system done on the real time – depending that the real time boundaries here are defined can stretch to hour, which is always less than the response of Drools to any received XML from the even producer.

The rules that have been used and the KPIs that have been taken into consideration can be more complicated and advanced, the design of the system was always putting into consideration the extendibility and the scalability of the rules without mentioned effort. By editing the project schema and the project rule file and deploying them into Apache ServiceMix, will produce more detailed analysis with different sophisticated scenarios.

Generic Web-Service format

The structure of the Web-Services has been done in a generic way, only minor customization like the machines IP-addresses needed to be modified for every machine. Plus some different modification related to the process of every machine. However, the Web-Services have been almost the same. This for saving the time and the effort plus obtaining the censurability.

Open Source concept of design

During implementation of this thesis work, an open source concept of design was always been into account.

All the software tools have been used during the thesis were open source software
The following are a brief list of most of the software tools that has been implemented and the function of every tool

- Notepad++ for editing the WSDL and the XML documents
- Web browser for monitoring the Web-Services variables for every station individually
- DPWS explorer – open source customized application implemented by Java to discover the WSDL files running over the network and subscribe the needed services, in order to monitor them
- Fiddler 2 – free HTTP client – to send XML message over HTTP application protocol – to simulate the Web-Service messages coming from the each machine in the case study

- Drools Expert – an open source business rule management system (BRMS) produced by JBoss, used to analysis the XML messages coming from the Fiddler client

- Apache ServiceMix is an enterprise-class open-source distributed enterprise service bus (ESB) and service-oriented architecture (SOA) toolkit. Apache ServiceMix is a container for Drools Expert. It receives the XML messages from the Fiddler and executes the Drools rules as a part of it. Shows the results over its console.

Summary of thesis work and Conclusion

The thesis work is offering a detailed methodology for designing and implementing of an expert system for monitoring and management of web-based industrial applications.

The work has been done in two connected phases; under ISA-95 model recommendations, the first phase of the thesis work located in level 2 of the model, concerning the manufacturing operation control.

The first task in this phase was a non-destructive upgrading of old existing PLC technology to web-service technology, by connecting a web-service module to every machine PLC.

The next task was to design the required web-services for every machine. A generic schema was designed to be able to be replicated on every machine, for the ease of downloading and programing the Web-Services.

The Web-Services were designed based on the object oriented concept, and the communication events and messages have been formed based on modification of CAMX IPC-2541 standard.

The second phase is locating in level 3 of ISA-95 model; it is more concerning in the manufacturing operation management.

The first task in this phase was to integrate the different software tools together, starting from generating the XML messages on the machine side, passing by converting them to other format and process them, and finally firing the rules over the BRMS.

The second task was to define the appropriate KPIs for our case and convert them to a rule form.

All the thesis work has been done under the umbrella of the main objectives and goals of have been mentioned in the first chapter of thesis, the main columns of the work was
To use known trusted standards either by using ANSI/ISA95 standard to define the main framework of the project or by using CAMX IPC- 2541 standard for developing the detailed structure of the Web-Service. Plus the other W3C standards have been used to write the XML messages and schema.

Some well-known concepts like object oriented programming has been used from a new prospective of the Web-Service and the rule engine, to achieve the integrity between the two platforms.

Other new concept was to convert the KPIs to business management rule, which is the main aim of the expert system to stream the experts into software format.

A generic approach was always used to cover as much as possible from similar cases for time, effort and cost saving.

And finally all the used software tools were open source to make it easier for everyone to continue working in the same subject.

Future work

On academic side

During the first task of the first phase of the thesis work, the non-destructive upgrading for the old PLC automated case study to a Web-Service technology was by parallel hard wiring of every machine PLC inputs and outputs to the Web-Service module. However, more flexible data communication either by CANopen protocol or Modbus communication protocols.

The Web-Service hardware module has been used already can communicate by those two protocols, However still the question about the ability of the old technology to connect via those protocols is not always viable.

Another point can be done in the future by adding more objects for the XML schema; that would be very easy task, as from the first place the schema has been designed in a generic way.

The third point can be enhanced is defining new KPIs, with more complicated formulas and converting them to rule form. Trying to build more intelligence in the BRMS by adding new rules to understand the trend of the business process and may recommend some suggestions. Furthermore, using the same deployment of Drools we discussed in the chapter 5 in level 4 of ISA-95, level 4 will require more business oriented rules to be processed as it recommended in ISA-95 model.

The last point can be done in the future work is providing a more handy graphical user interface for Drools end user, which can be done using Java programming, Drools results should contain more graphs and charts about the production process.

On Personal side

In February 2013, I became one of the founders of NovoGenie project; NovoGenie is an online expert Platform for entrepreneurs and companies to develop their capabilities for creativity and innovations. It helps the individuals to focus on their needs to be developed, get the best out of their talent and time through our personalized informal development process.

NovoGenie has two main levels of services. The first level is public (B2C) which serves individuals, teams and coaches. The second level is a professional cloud service for organization (B2B) functioning as their intranet service and is closed within the organization.

On the individual side, it helps the users to interact with other members to build balanced and dynamic teams matched with the end mission, optimizing team collaboration, especially innovative teams where the team members come from different locations, organizations or disciplined. NovoGenie helps the teammates to quickly identify their roles in the team with the least conflict and bring the best of their ideas.

On the B2B side, NovoGenie helps companies to accelerate their business and generate quality and quantity innovations. It provides more disciplined approach for the manger to predict the outcomes of the different teams, plan their resources, minimize the risk of innovation's investment and motivate their subordinates.

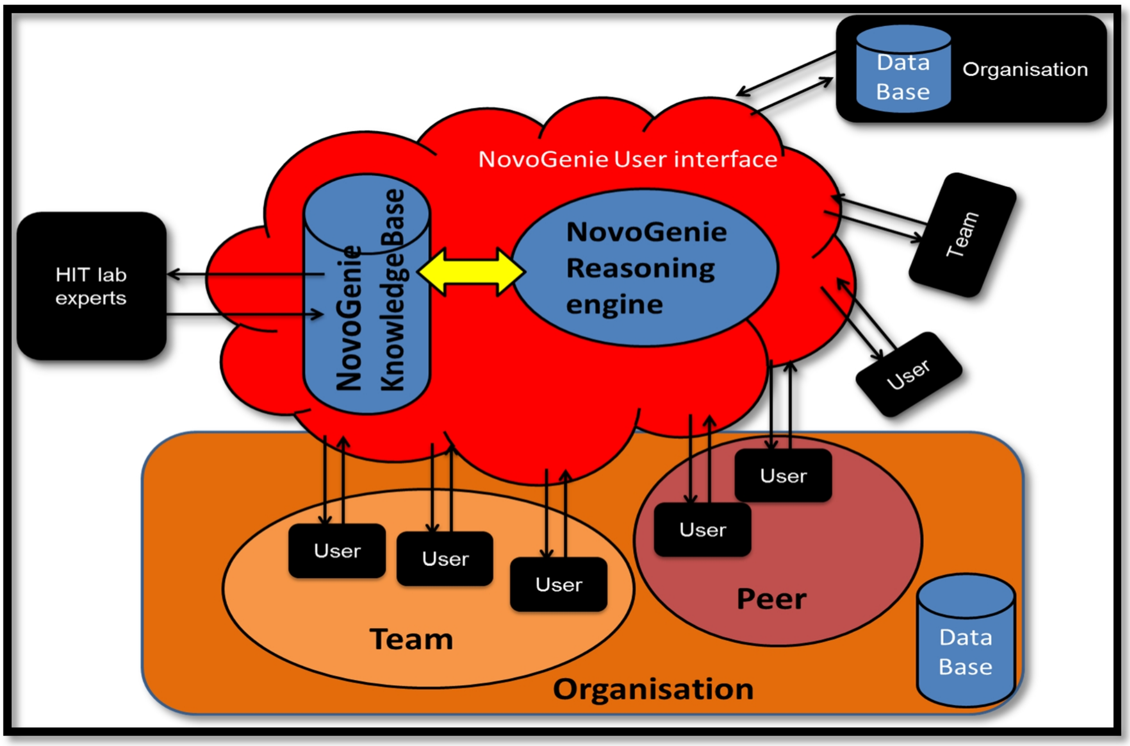


Figure 58 – NovoGenie expert platform core

The main idea of the NovoGenie expert platform to stream the experiences of the experts in Carl Jung field in an online from, the smartness in the system come from the reasoning engine of the software shell which is shown in Figure 58,

The reasoning engine is combination of two branches of artificial intelligent techniques; the first technique is an expert system which is responsible of verification of the data coming from the user, converting them into a weighted knowledge, giving to draw a road path for the user by recommending him/her by the advices and consultations.

The next technique is an cognitive system which should always follow the reaction of the user with the others inside the site, correct the previous results of the cognitive test

HIT (human information technology) Lab is a Finnish company based in Tampere, responsible for feeding the platform with the needed experience and science

The project still in the developing phase, the existing platform is Beta version. In the current time, we are coordinating now with a lot of international companies and Finnish universities to produce a complete version

REFERENCES

[1] K. Darlington, The essence of expert systems, Prentice Hall, 2000.

[2] B. Q. a. H. B. Gooi, “Web-Based SCADA Display Systems (WSDS),” *IEEE TRANSACTIONS ON POWER SYSTEMS,* vol. 15, no. MAY 2000, pp. 681-686, 2000.

[3] I. c. R. A. GE Fanuc Ameriicas, “Standards for Manufacturing Systems Integration,” 2006.

[4] ANSI, “ISA95.00.03 - Enterprise-Control System and Integration part 3: Model of Manufacturing Operation Management,” 2005.

[5] S. M. W. Deon Reynders, Practical Industrial DataCommunications – Best Practice Techniques, IDC Technologies, 2005.

[6] “Plantcockpit - Production Logistics and sustainability cockpit,” 31 8 2012. [Online]. Available: http://plantcockpit.eu/.

[7] G. Rathwell, “ISA-95 Tool for Enterprise Modelling,” *IEEE Conference ,* pp. 113 - 114, 23-23 Nov. 2006.

[8] V. D. Goran Simic, “Building an intelligent system using modern Internet technologies,” *expert systems with applications - Elsevier Science ,* pp. 231 - 246, 2003.

[9] J. Robert J. St. Jacques, “XESS: The XML Expert System Shell,” 2008.

[10] F. D. GmbH, CIROS ADVANCED MECHATRONICS, Germany , 2008.

[11] T. K. R. S.-J. S. S. W. Ueli Wahli, WebSphere Version 6 – Web Services Handbook Development and Deployment, Germany : IBM, July 2005.

[12] O. Z. Olaf, “Architectural Decisions as Service Realization Methodology in Model Methodology in Model--Driven SOA Driven Construction,” in *4th IEEE European Conference on Web Services*, Zurich, Switzerland, December 4 – 6, 2006.

[13] S. H. Seungjin Choi, “WS-BPEL Monitoring System,” *International Journal of Web Services Practices,* vol. 6, pp. 18-20, 2011.

[14] W3C, “W3C Working Group Note,” 11 February 2004. [Online]. Available: http://www.w3.org/TR/ws-arch/.

[15] Y. V. R. N. Manan Shah, “AN AUTOMATED END-TO-END MULTI-AGENT QOS BASED ARCHITECTURE FOR SELECTION OF GEOSPATIAL WEB SERVICES,” in *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Melbourne, Australia, 01 September 2012.

[16] J. L. L. -. I. M.Delamer, Factory Information Systems in Electronic Production, Tampere - Finland : JuvensPrint, 2005.

[17] T. organisation, “The XML Version of the TEI Guidelines – A Gentle Introduction to XML,” [Online]. Available: http://www.tei-c.org/release/doc/tei-p4-doc/html/SG.html.

[18] T. Myer, “A Really Good Introduction to XML,” 24 August 2005. [Online]. Available: http://www.sitepoint.com/really-good-introduction-xml/.

[19] w3school, “Introduction to XML Schema,” [Online]. Available: http://www.w3schools.com/schema/schema_intro.asp.

[20] Erik Christensen-Francisco Curbera-Greg Meredith-Sanjiva Weerawarana, “Web Services Description Language(WSDL) 1.1,” MicroSoft - IBM , 15 March 2001.

[21] M. P. Papazoglou, Web Services: Principles and Technology, Pearson Education Limited 2008.

[22] N. U. o. I. -E. Curry, “Message-Oriented Middleware,” John Wiley & Sons, Ltd, Galway, Ireland, 2004.

[23] INICO, S1000 user manual datasheet.

[24] I. I. f. I. a. P. E. Circuit, “IPC-2541 - Generic Requirements for Electronics Manufacturing Shop-Floor Equipment Communication Messages (CAMX),” October 2001 .

[25] W. A. S. George F. Luger, Artificial intelligence - structures and strategies for complex problem solving, Benjamin/Cummings Pub. Co. (Redwood City, Calif.) , 1993.

[26] P. H. Winston, Artificial Intelligence, Addison Wesley, 1992.

[27] R. Kurzweil, The age of intelligent machines, MIT Press (Cambridge, Mass.) , 1990.

[28] R. J. Schalkoff, Artificial Intelligence Engine, McGraw-Hill, 1990.

[29] R. A. Brooks, Intelligence Without Reason, Sydney, Australia: Morgan Kaufmann, 1991.

[30] Y. Kumar, “Research Aspects of Expert System,” *International Journal of Computing & Business Research,* 2012.

[31] E. R. &. K. Knight, Artificial Intelligence, McGraw Hill, Second Edition, 1991.

[32] R. A. Frost, Introduction to Knowledge Base Systems, Collins, 1986.

[33] F. v. Harmelen, Meta-level Inference Systems, Pitman & Morgan Kaufmann, 1991.

[34] H. L. D. &. S. E. Dreyfus, Mind over Machine, The Free Press, 1986.

[35] M. Kerber, “Introduction to AI,” 24 April 2005. [Online]. Available: http://www.cs.bham.ac.uk/~mmk/Teaching/AI/.

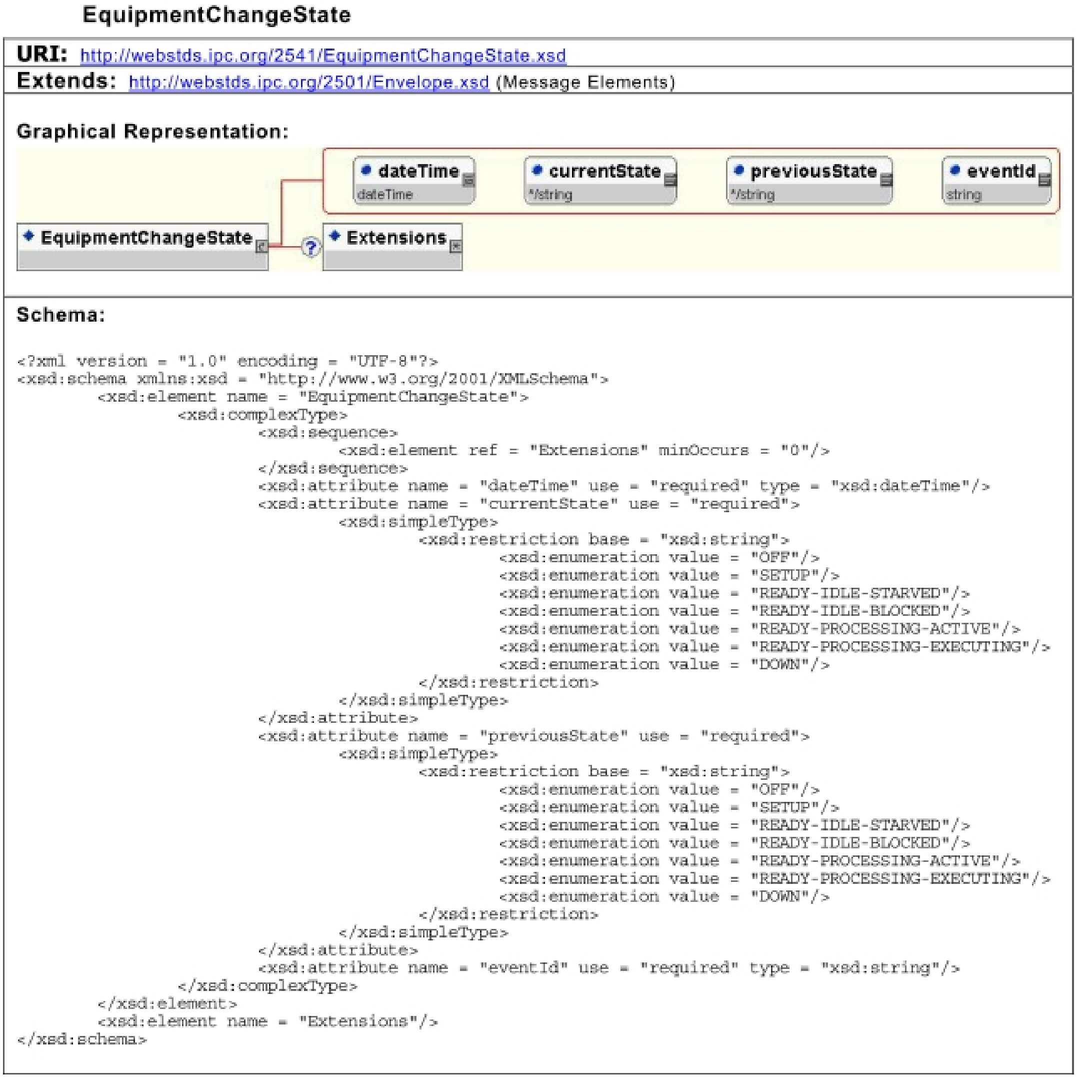
[36] E. H. Shortliffe, Computer-based medical consultations, MYCIN, 1976.

[37] “Teach-ICT,” [Online]. Available: http://www.teach-ict.com/as_a2_ict_new/ocr/ A2_G063/334_applications_ict/expert_systems/miniweb/pg9.htm.

[38] J. team, “Drools Expert User Guide,” jboss, [Online]. Available: http://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/html_single/index.html.

[39] airgroup.com.ar, “airgroup.com.ar,” [Online]. Available: http://wiki.aigroup.com.ar/ci/expert-systems/a-brief-introduction-about-expert-system-rules-in-drools.

APPENDIX 1– XML SCHEMAS FROM CAMX IPC-2541



EquipmentChangeState

URI: http://webstds.ipc.org/2541/EquipmentChangeState.xsd

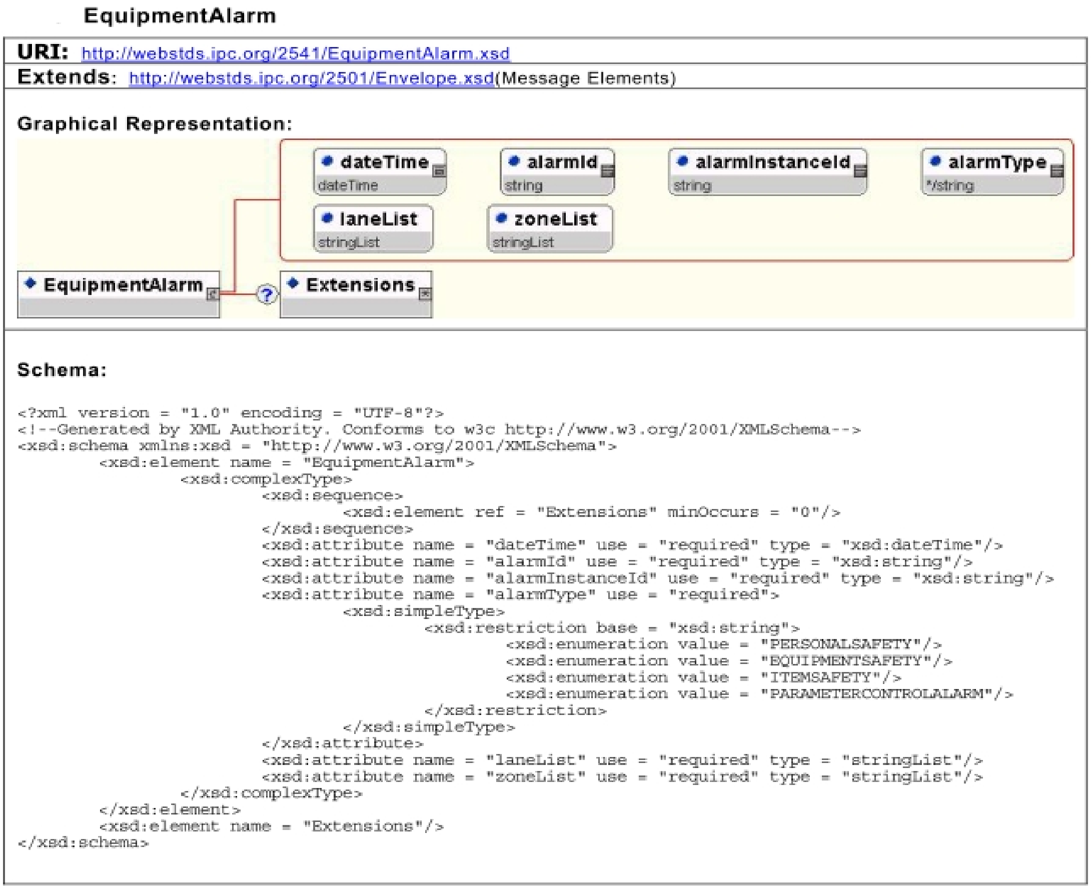
Extends: http://webstds.ipc.org/2501/Envelope.xsd (Message Elements)

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
        <xsd:element name = "EquipmentChangeState">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref = "Extensions" minOccurs = "0"/>
                        </xsd:sequence>
                        <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                        <xsd:attribute name = "currentState" use = "required">
                                <xsd:simpleType>
                                        <xsd:restriction base = "xsd:string">
                                                <xsd:enumeration value = "OFF"/>
                                                <xsd:enumeration value = "SETUP"/>
                                                <xsd:enumeration value = "READY-IDLE-STARVED"/>
                                                <xsd:enumeration value = "READY-IDLE-BLOCKED"/>
                                                <xsd:enumeration value = "READY-PROCESSING-ACTIVE"/>
                                                <xsd:enumeration value = "READY-PROCESSING-EXECUTING"/>
                                                <xsd:enumeration value = "DOWN"/>
                                        </xsd:restriction>
                                </xsd:simpleType>
                        </xsd:attribute>
                        <xsd:attribute name = "previousState" use = "required">
                                <xsd:simpleType>
                                        <xsd:restriction base = "xsd:string">
                                                <xsd:enumeration value = "OFF"/>
                                                <xsd:enumeration value = "SETUP"/>
                                                <xsd:enumeration value = "READY-IDLE-STARVED"/>
                                                <xsd:enumeration value = "READY-IDLE-BLOCKED"/>
                                                <xsd:enumeration value = "READY-PROCESSING-ACTIVE"/>
                                                <xsd:enumeration value = "READY-PROCESSING-EXECUTING"/>
                                                <xsd:enumeration value = "DOWN"/>
                                        </xsd:restriction>
                                </xsd:simpleType>
                        </xsd:attribute>
                        <xsd:attribute name = "eventId" use = "required" type = "xsd:string"/>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 59 – Equipment Change State XML schema due to CAMX IPC-2541



EquipmentAlarm

URI: http://webstds.ipc.org/2541/EquipmentAlarm.xsd

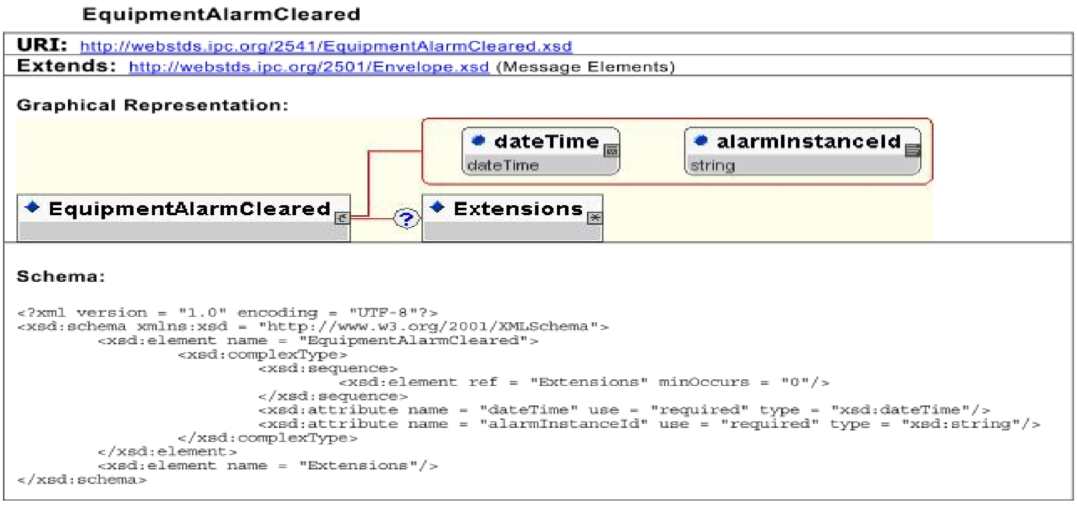
Extends: http://webstds.ipc.org/2501/Envelope.xsd(Message Elements)

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!--Generated by XML Authority. Conforms to w3c http://www.w3.org/2001/XMLSchema-->
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
        <xsd:element name = "EquipmentAlarm">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref = "Extensions" minOccurs = "0"/>
                        </xsd:sequence>
                        <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                        <xsd:attribute name = "alarmId" use = "required" type = "xsd:string"/>
                        <xsd:attribute name = "alarmInstanceId" use = "required" type = "xsd:string"/>
                        <xsd:attribute name = "alarmType" use = "required">
                                <xsd:simpleType>
                                        <xsd:restriction base = "xsd:string">
                                                <xsd:enumeration value = "PERSONALSAFETY"/>
                                                <xsd:enumeration value = "EQUIPMENTSAFETY"/>
                                                <xsd:enumeration value = "ITEMSAFETY"/>
                                                <xsd:enumeration value = "PARAMETERCONTROLALARM"/>
                                        </xsd:restriction>
                                </xsd:simpleType>
                        </xsd:attribute>
                        <xsd:attribute name = "laneList" use = "required" type = "stringList"/>
                        <xsd:attribute name = "zoneList" use = "required" type = "stringList"/>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 60 – Equipment Alarm XML schema due to CAMX IPC-2541



EquipmentAlarmCleared

URI: http://webstds.ipc.org/2541/EquipmentAlarmCleared.xsd

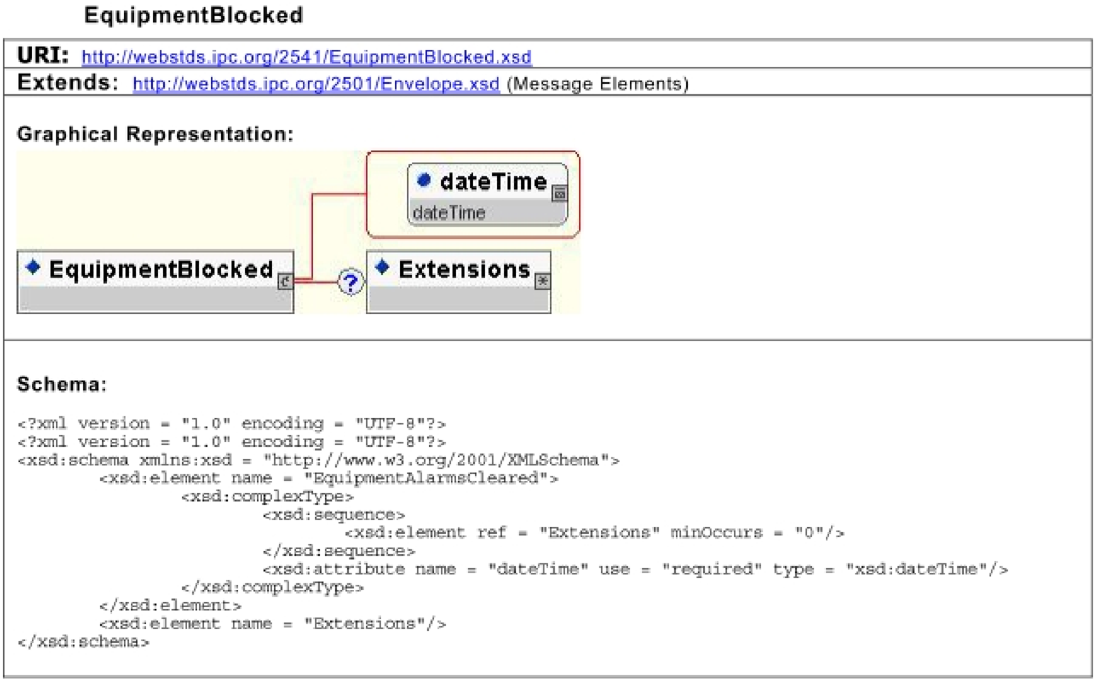
Extends: http://webstds.ipc.org/2501/Envelope.xsd (Message Elements)

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
        <xsd:element name = "EquipmentAlarmCleared">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref = "Extensions" minOccurs = "0"/>
                        </xsd:sequence>
                        <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                        <xsd:attribute name = "alarmInstanceId" use = "required" type = "xsd:string"/>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 61 – Equipment Alarm Clear XML schema due to CAMX IPC-2541



EquipmentBlocked

URI: http://webstds.ipc.org/2541/EquipmentBlocked.xsd

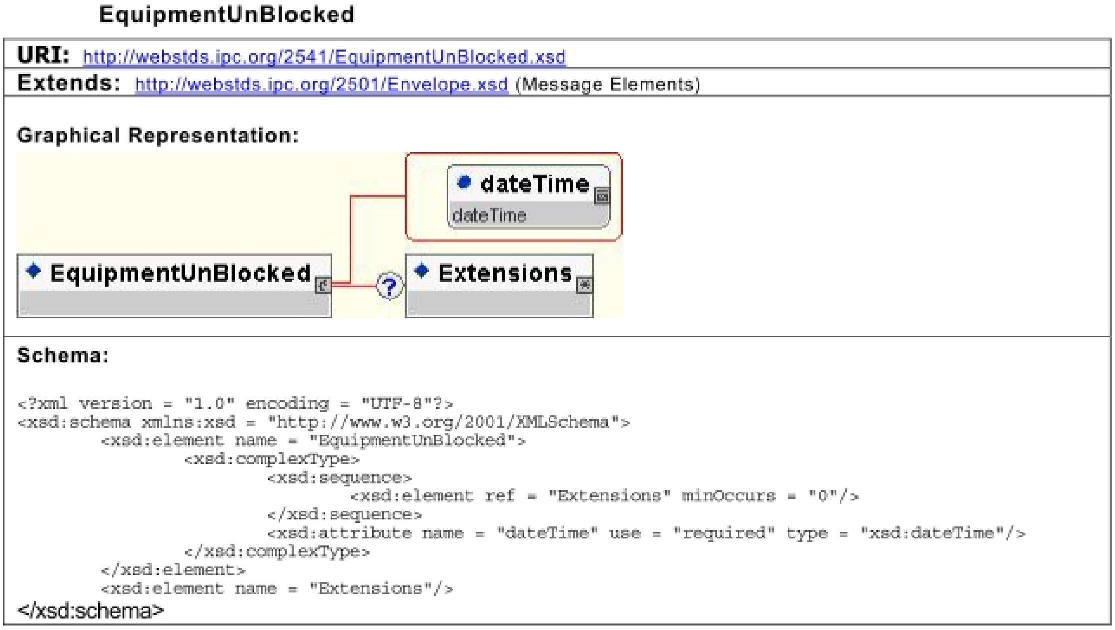
Extends: http://webstds.ipc.org/2501/Envelope.xsd (Message Elements)

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
        <xsd:element name = "EquipmentAlarmsCleared">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref = "Extensions" minOccurs = "0"/>
                        </xsd:sequence>
                        <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 62 – Equipment Blocked XML schema due to CAMX IPC-2541



EquipmentUnBlocked

URI: http://webstds.ipc.org/2541/EquipmentUnBlocked.xsd

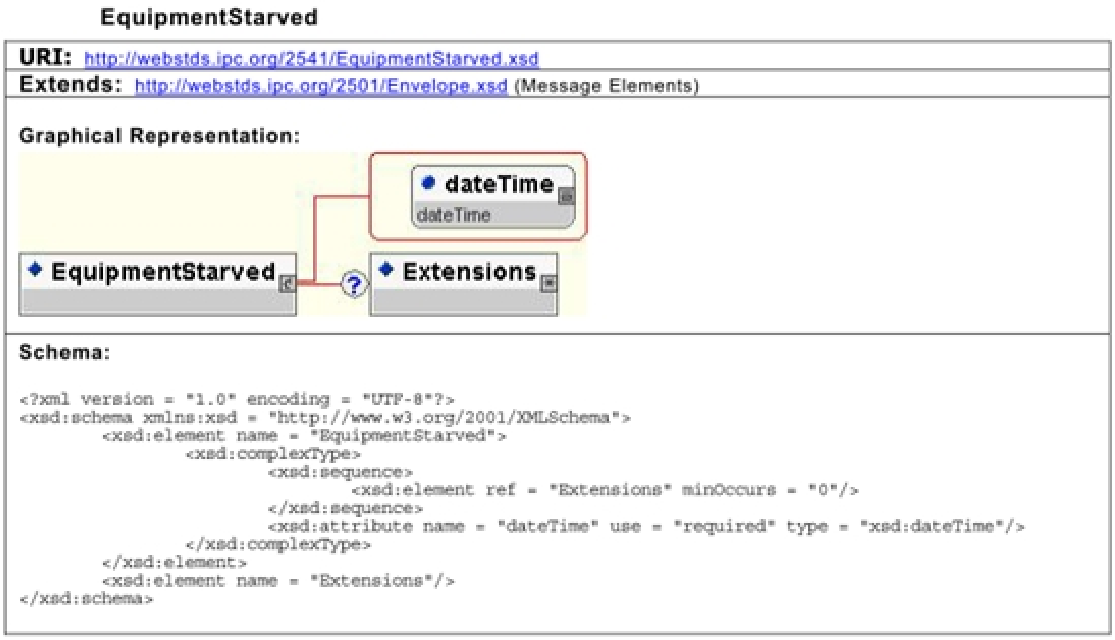
Extends: http://webstds.ipc.org/2501/Envelope.xsd (Message Elements)

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
        <xsd:element name = "EquipmentUnBlocked">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref = "Extensions" minOccurs = "0"/>
                        </xsd:sequence>
                        <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 63 – Equipment Unblocked XML schema due to CAMX IPC-2541



EquipmentStarved

URI: http://webstds.ipc.org/2541/EquipmentStarved.xsd

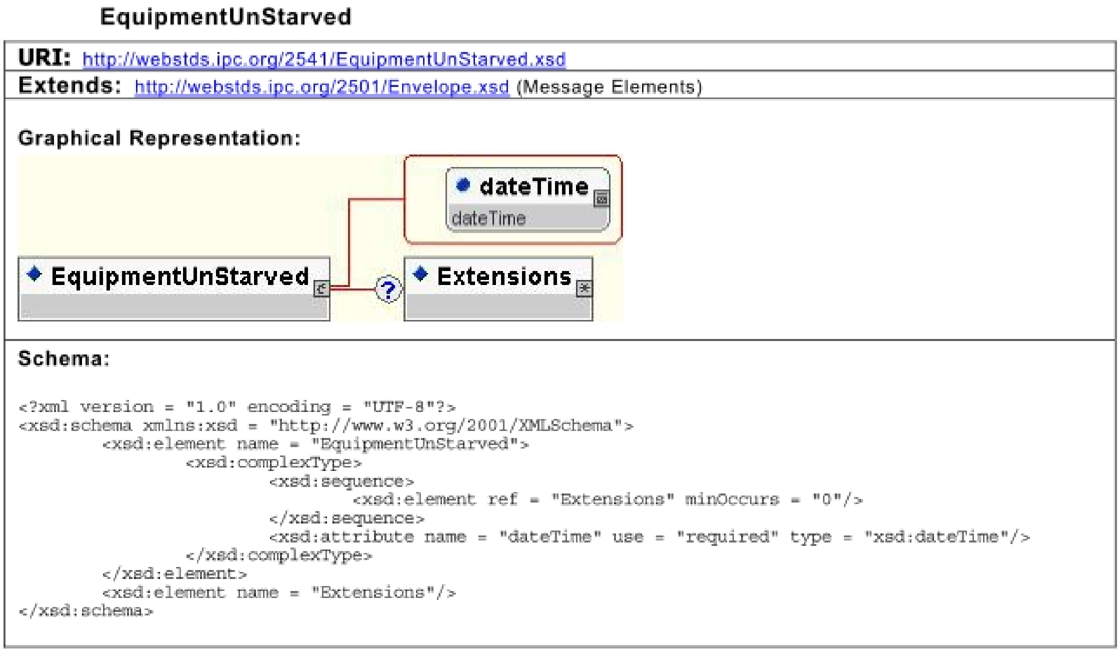
Extends: http://webstds.ipc.org/2501/Envelope.xsd (Message Elements)

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
        <xsd:element name = "EquipmentStarved">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref = "Extensions" minOccurs = "0"/>
                        </xsd:sequence>
                        <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 64 – Equipment Starved XML schema due to CAMX IPC-2541



EquipmentUnStarved

URI: http://webstds.ipc.org/2541/EquipmentUnStarved.xsd

Extends: http://webstds.ipc.org/2501/Envelope.xsd (Message Elements)

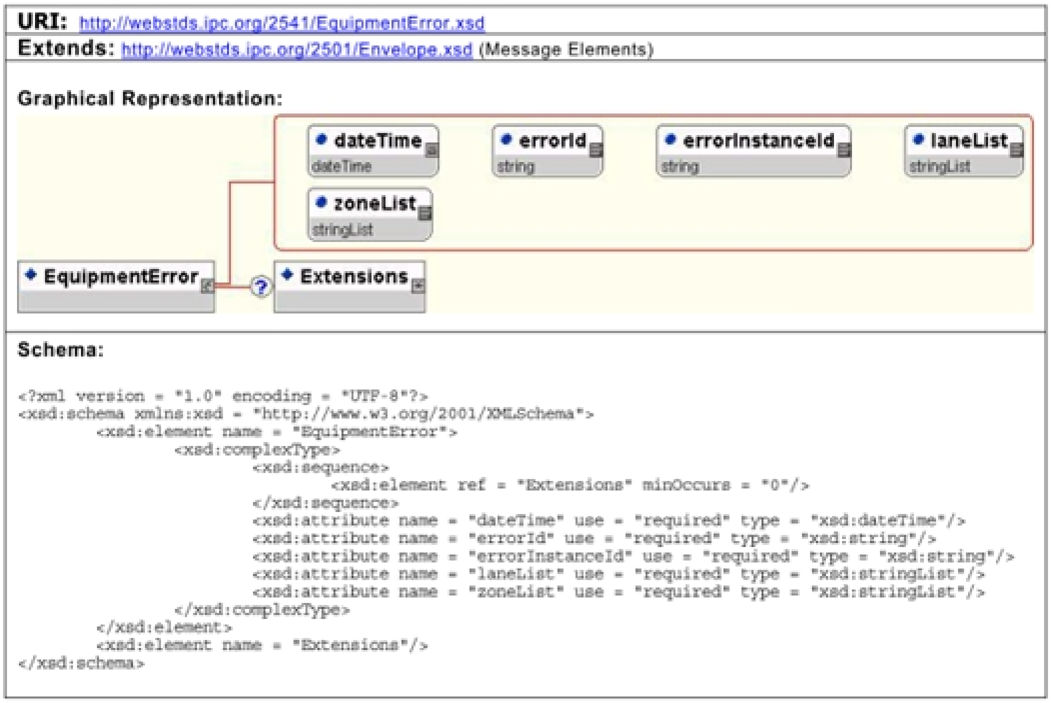
Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
        <xsd:element name = "EquipmentUnStarved">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref = "Extensions" minOccurs = "0"/>
                        </xsd:sequence>
                        <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 65 – Equipment UnStarved XML schema due to CAMX IPC-2541

EquipmentError



URI: http://webstds.ipc.org/2541/EquipmentError.xsd

Extends: http://webstds.ipc.org/2501/Envelope.xsd (Message Elements)

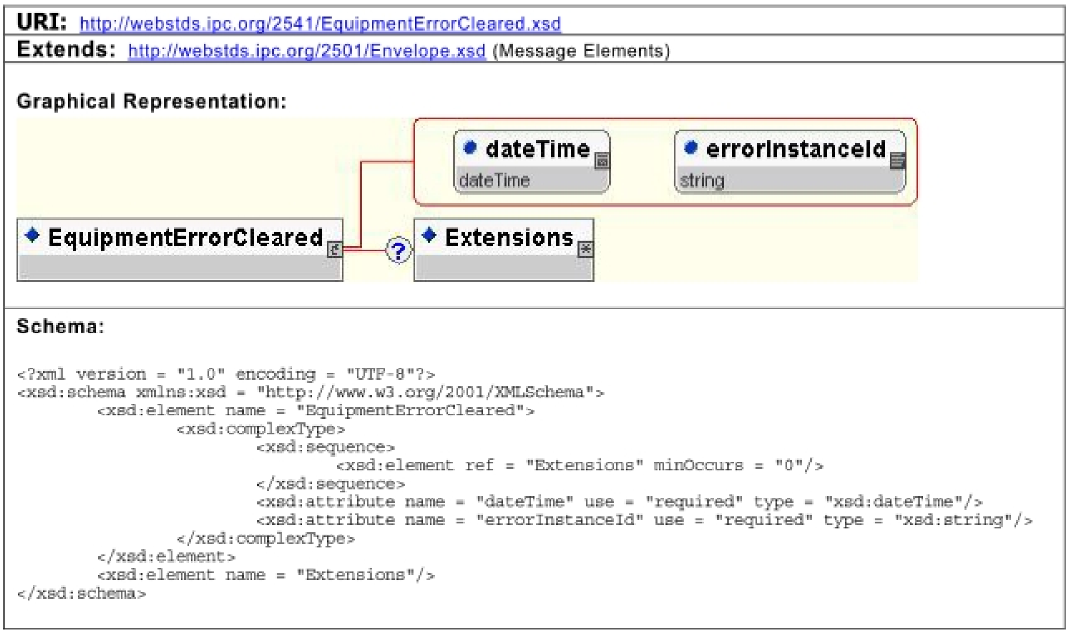
Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
        <xsd:element name = "EquipmentError">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref = "Extensions" minOccurs = "0"/>
                        </xsd:sequence>
                        <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                        <xsd:attribute name = "errorId" use = "required" type = "xsd:string"/>
                        <xsd:attribute name = "errorInstanceId" use = "required" type = "xsd:string"/>
                        <xsd:attribute name = "laneList" use = "required" type = "xsd:stringList"/>
                        <xsd:attribute name = "zoneList" use = "required" type = "xsd:stringList"/>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 66 – Equipment Error XML schema due to CAMX IPC-2541

EquipmentErrorCleared



URI: http://webstds.ipc.org/2541/EquipmentErrorCleared.xsd

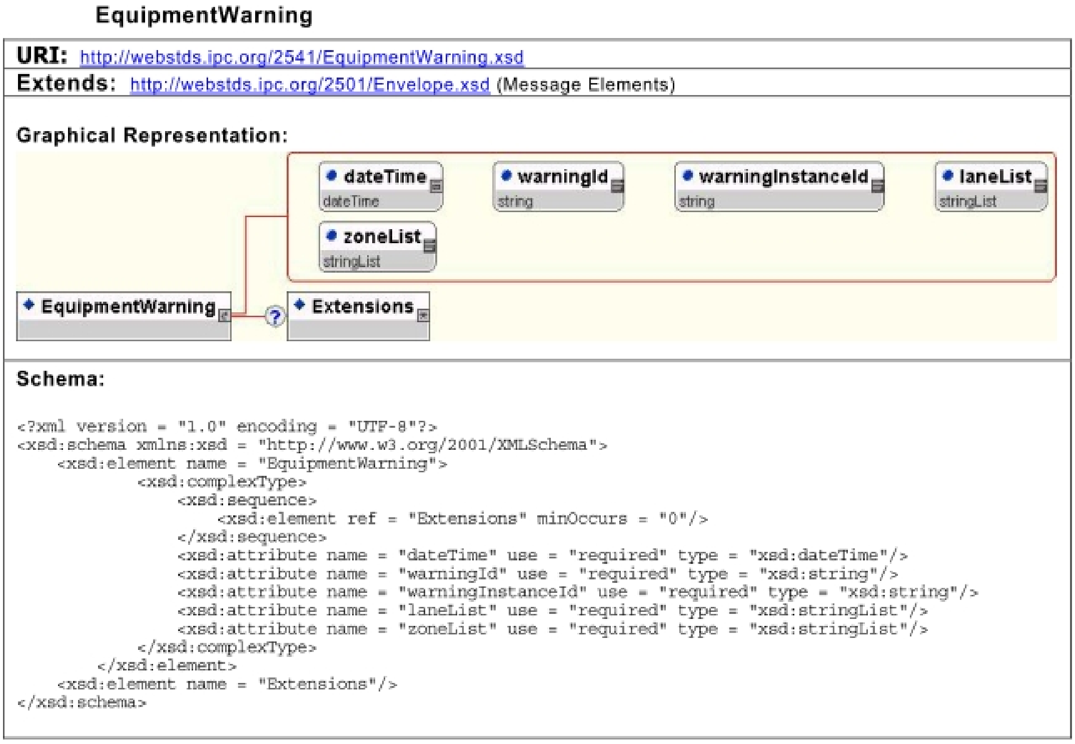
Extends: http://webstds.ipc.org/2501/Envelope.xsd (Message Elements)

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
        <xsd:element name = "EquipmentErrorCleared">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref = "Extensions" minOccurs = "0"/>
                        </xsd:sequence>
                        <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                        <xsd:attribute name = "errorInstanceId" use = "required" type = "xsd:string"/>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 67 – Equipment Error Cleared XML schema due to CAMX IPC-2541



EquipmentWarning

URI: http://webstds.ipc.org/2541/EquipmentWarning.xsd

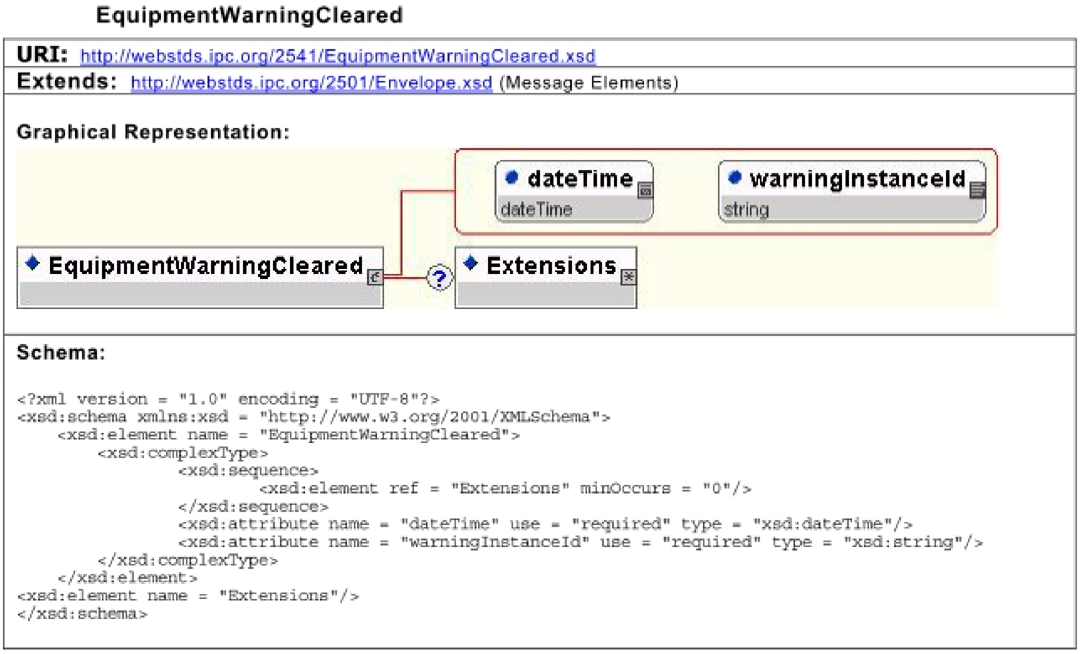
Extends: http://webstds.ipc.org/2501/Envelope.xsd (Message Elements)

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
    <xsd:element name = "EquipmentWarning">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element ref = "Extensions" minOccurs = "0"/>
                </xsd:sequence>
                <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                <xsd:attribute name = "warningId" use = "required" type = "xsd:string"/>
                <xsd:attribute name = "warningInstanceId" use = "required" type = "xsd:string"/>
                <xsd:attribute name = "laneList" use = "required" type = "xsd:stringList"/>
                <xsd:attribute name = "zoneList" use = "required" type = "xsd:stringList"/>
            </xsd:complexType>
        </xsd:element>
    <xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 68 – Equipment Warning XML schema due to CAMX IPC-2541



EquipmentWarningCleared

URI: http://webstds.ipc.org/2541/EquipmentWarningCleared.xsd

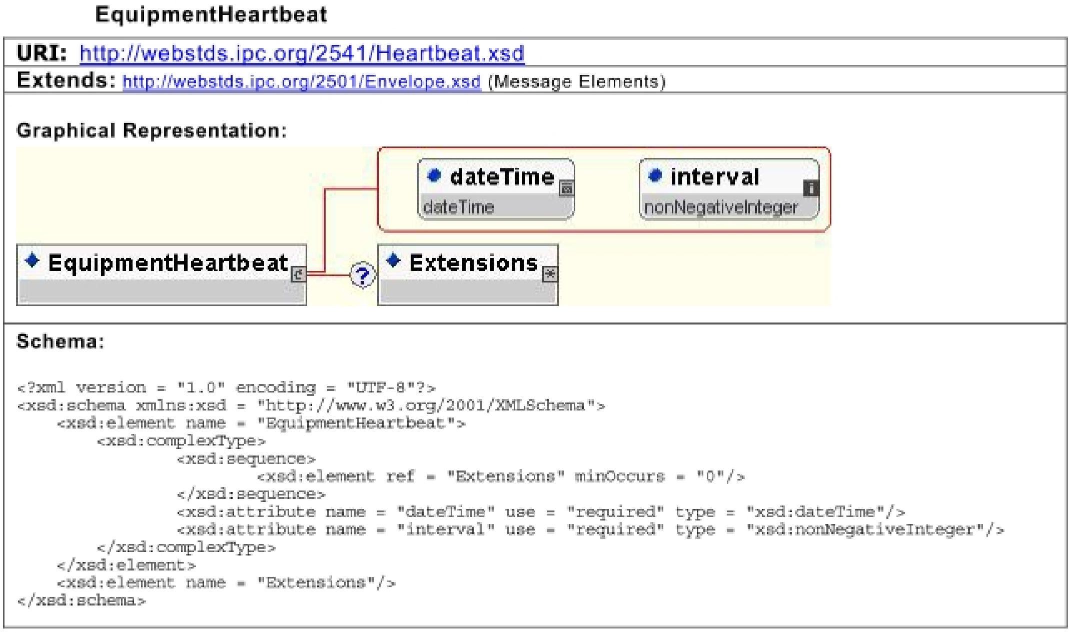
Extends: http://webstds.ipc.org/2501/Envelope.xsd (Message Elements)

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
    <xsd:element name = "EquipmentWarningCleared">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element ref = "Extensions" minOccurs = "0"/>
                </xsd:sequence>
                <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                <xsd:attribute name = "warningInstanceId" use = "required" type = "xsd:string"/>
        </xsd:complexType>
    </xsd:element>
<xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 69 – Equipment Warning Cleared XML schema due to CAMX IPC-2541



EquipmentHeartbeat

URI: http://webstds.ipc.org/2541/Heartbeat.xsd

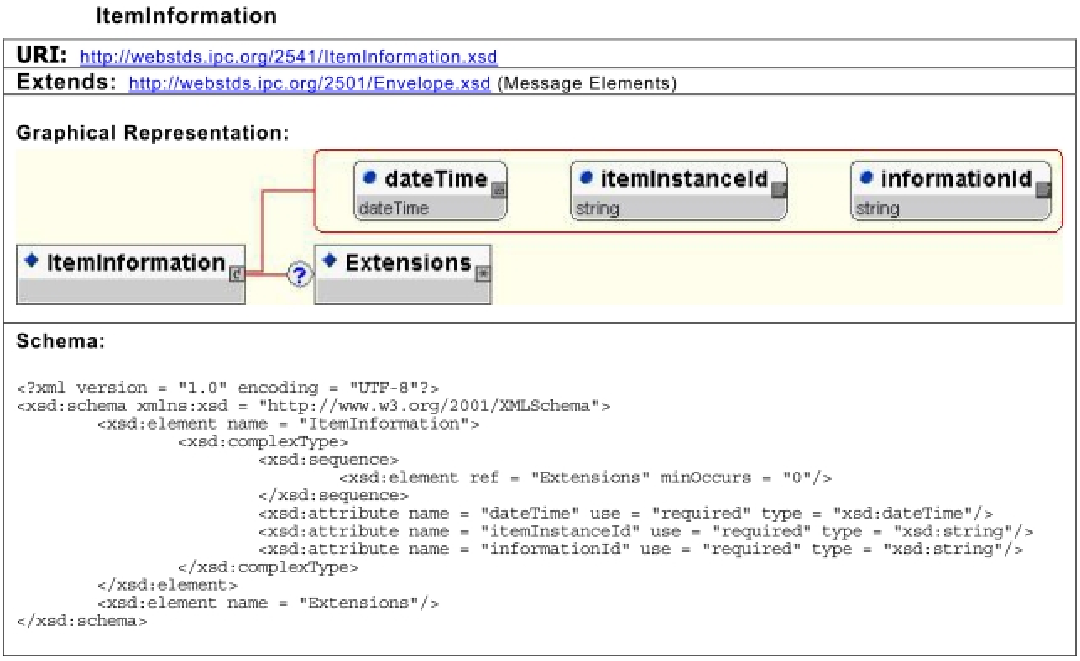
Extends: http://webstds.ipc.org/2501/Envelope.xsd (Message Elements)

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
    <xsd:element name = "EquipmentHeartbeat">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element ref = "Extensions" minOccurs = "0"/>
                </xsd:sequence>
                <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                <xsd:attribute name = "interval" use = "required" type = "xsd:nonNegativeInteger"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 70 – Equipment Heart Beat XML schema due to CAMX IPC-2541



ItemInformation

URI: http://webstds.ipc.org/2541/ItemInformation.xsd

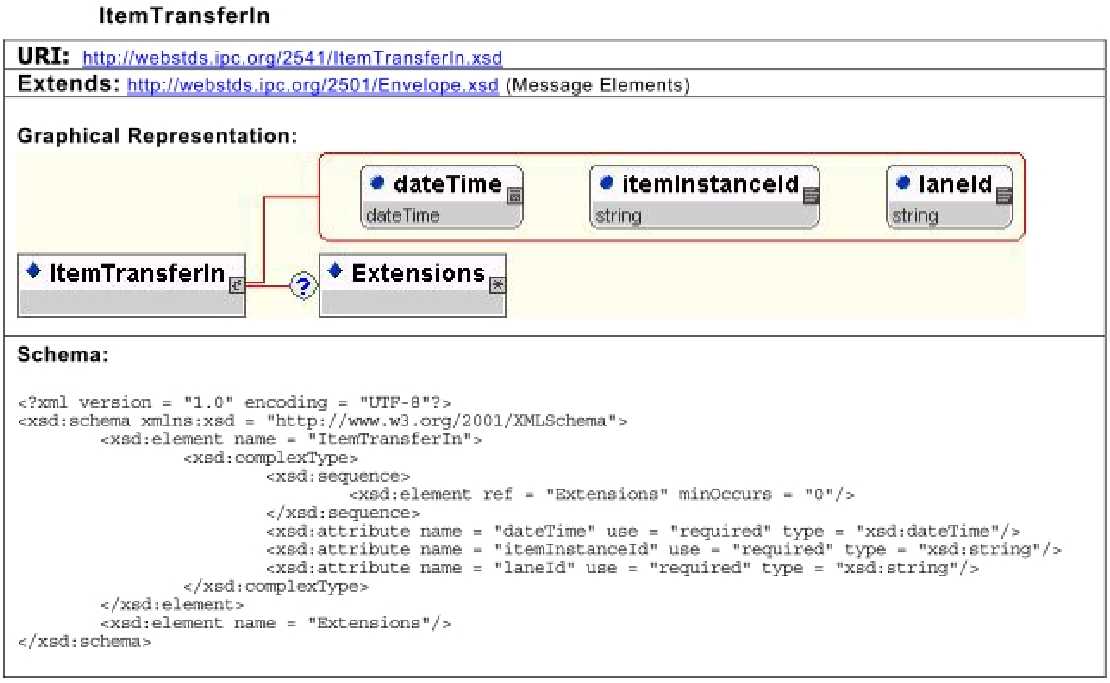
Extends: http://webstds.ipc.org/2501/Envelope.xsd (Message Elements)

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
        <xsd:element name = "ItemInformation">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref = "Extensions" minOccurs = "0"/>
                        </xsd:sequence>
                        <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                        <xsd:attribute name = "itemInstanceId" use = "required" type = "xsd:string"/>
                        <xsd:attribute name = "informationId" use = "required" type = "xsd:string"/>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 71 – Item Information XML schema due to CAMX IPC-2541



ItemTransferIn

URI: http://webstds.ipc.org/2541/ItemTransferIn.xsd

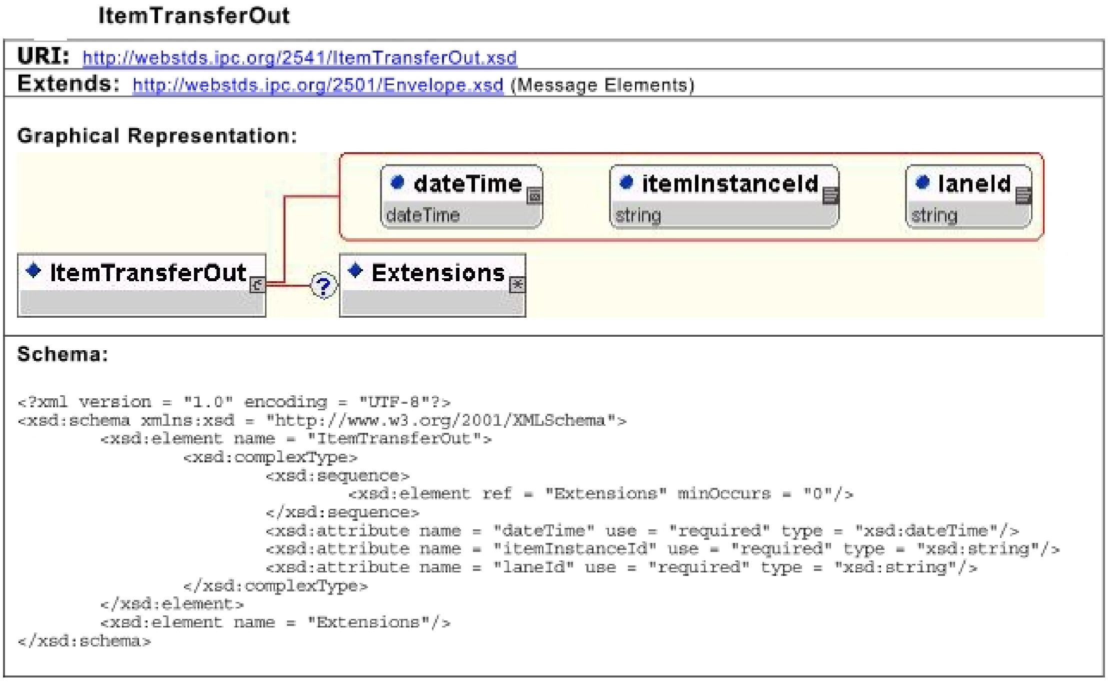
Extends: http://webstds.ipc.org/2501/Envelope.xsd (Message Elements)

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
        <xsd:element name = "ItemTransferIn">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref = "Extensions" minOccurs = "0"/>
                        </xsd:sequence>
                        <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                        <xsd:attribute name = "itemInstanceId" use = "required" type = "xsd:string"/>
                        <xsd:attribute name = "laneId" use = "required" type = "xsd:string"/>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 72 – Item Transfer in XML schema due to CAMX IPC-2541



ItemTransferOut

URI: http://webstds.ipc.org/2541/ItemTransferOut.xsd

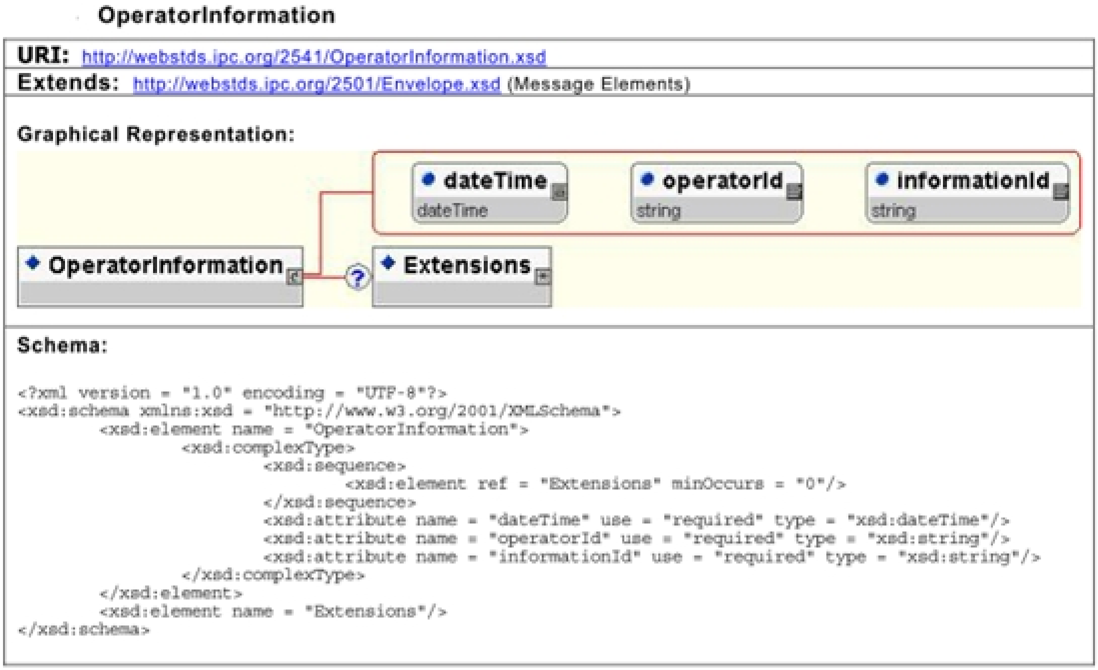
Extends: http://webstds.ipc.org/2501/Envelope.xsd (Message Elements)

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
        <xsd:element name = "ItemTransferOut">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref = "Extensions" minOccurs = "0"/>
                        </xsd:sequence>
                        <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                        <xsd:attribute name = "itemInstanceId" use = "required" type = "xsd:string"/>
                        <xsd:attribute name = "laneId" use = "required" type = "xsd:string"/>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 73 – Item TransferOut XML schema due to CAMX IPC-2541



OperatorInformation

URI: http://webstds.ipc.org/2541/OperatorInformation.xsd

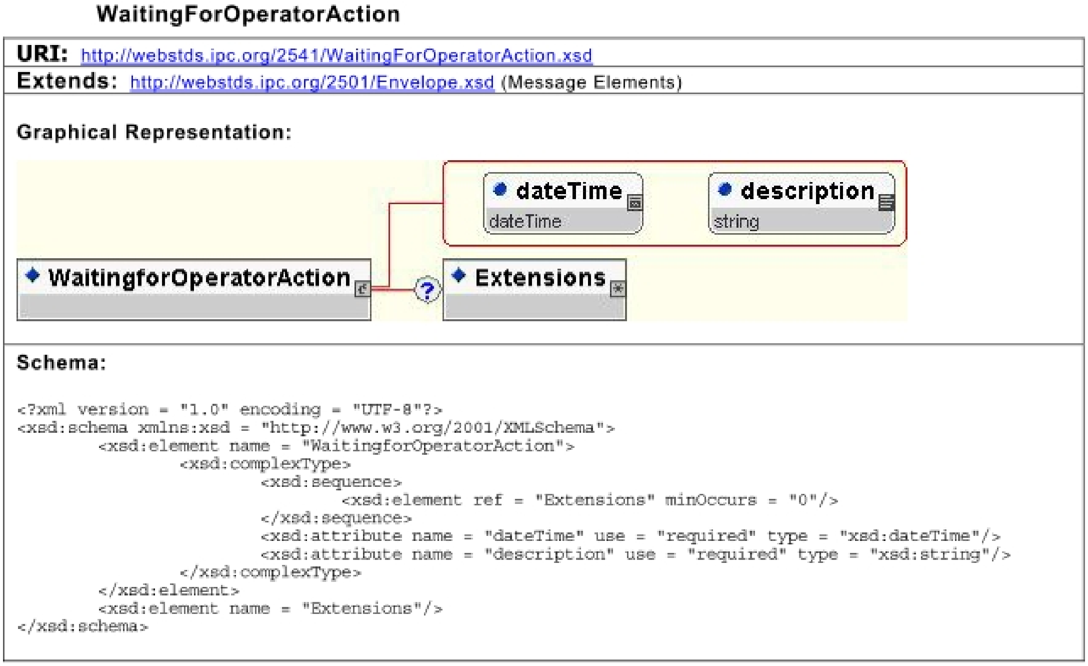
Extends: http://webstds.ipc.org/2501/Envelope.xsd (Message Elements)

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
        <xsd:element name = "OperatorInformation">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref = "Extensions" minOccurs = "0"/>
                        </xsd:sequence>
                        <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                        <xsd:attribute name = "operatorId" use = "required" type = "xsd:string"/>
                        <xsd:attribute name = "informationId" use = "required" type = "xsd:string"/>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 74 – Operator information XML schema due to CAMX IPC-2541



WaitingForOperatorAction

URI: http://webstds.ipc.org/2541/WaitingForOperatorAction.xsd

Extends: http://webstds.ipc.org/2501/Envelope.xsd (Message Elements)

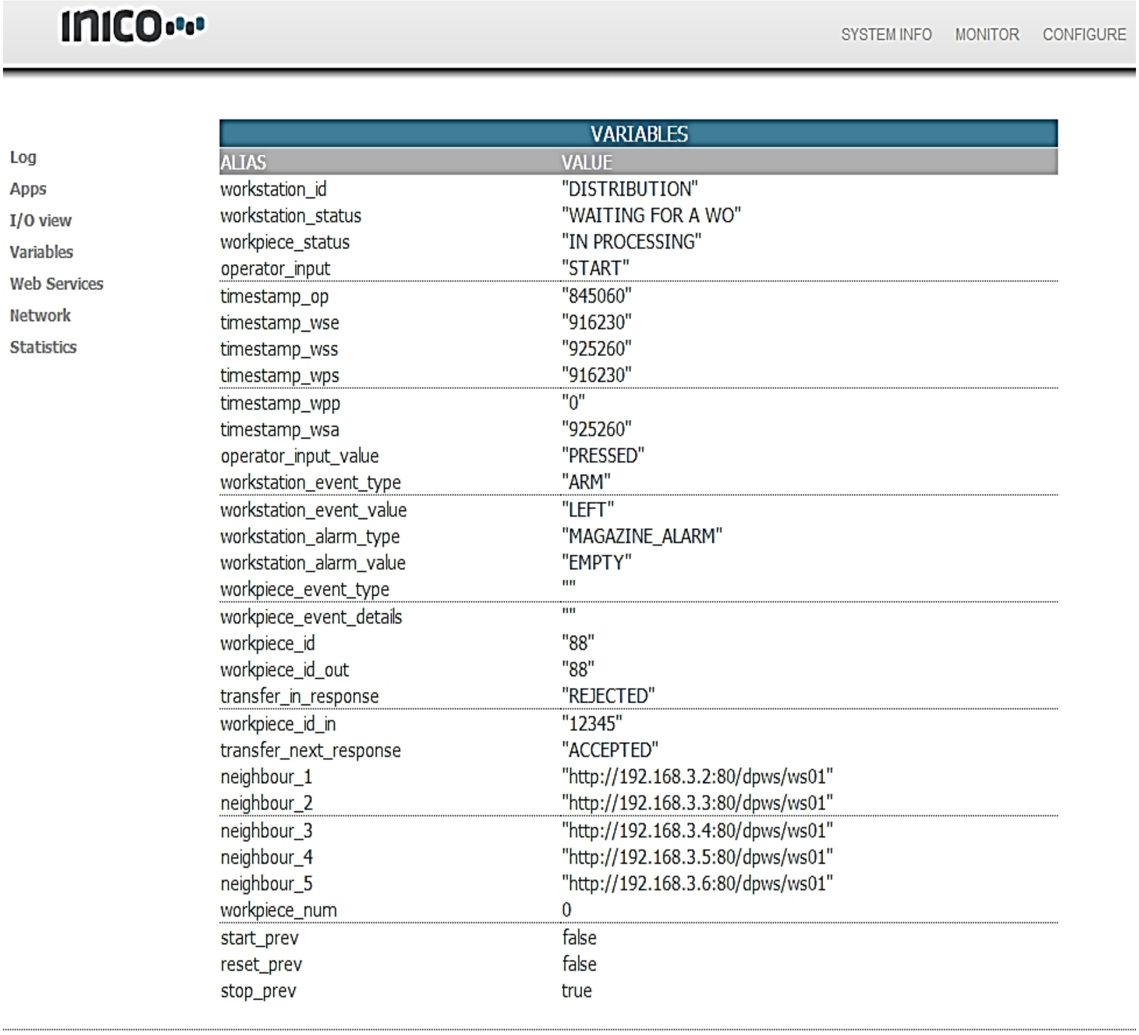
Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
        <xsd:element name = "WaitingforOperatorAction">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref = "Extensions" minOccurs = "0"/>
                        </xsd:sequence>
                        <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
                        <xsd:attribute name = "description" use = "required" type = "xsd:string"/>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name = "Extensions"/>
</xsd:schema>
```

Figure 75 – Operator Action XML schema due to CAMX IPC-2541

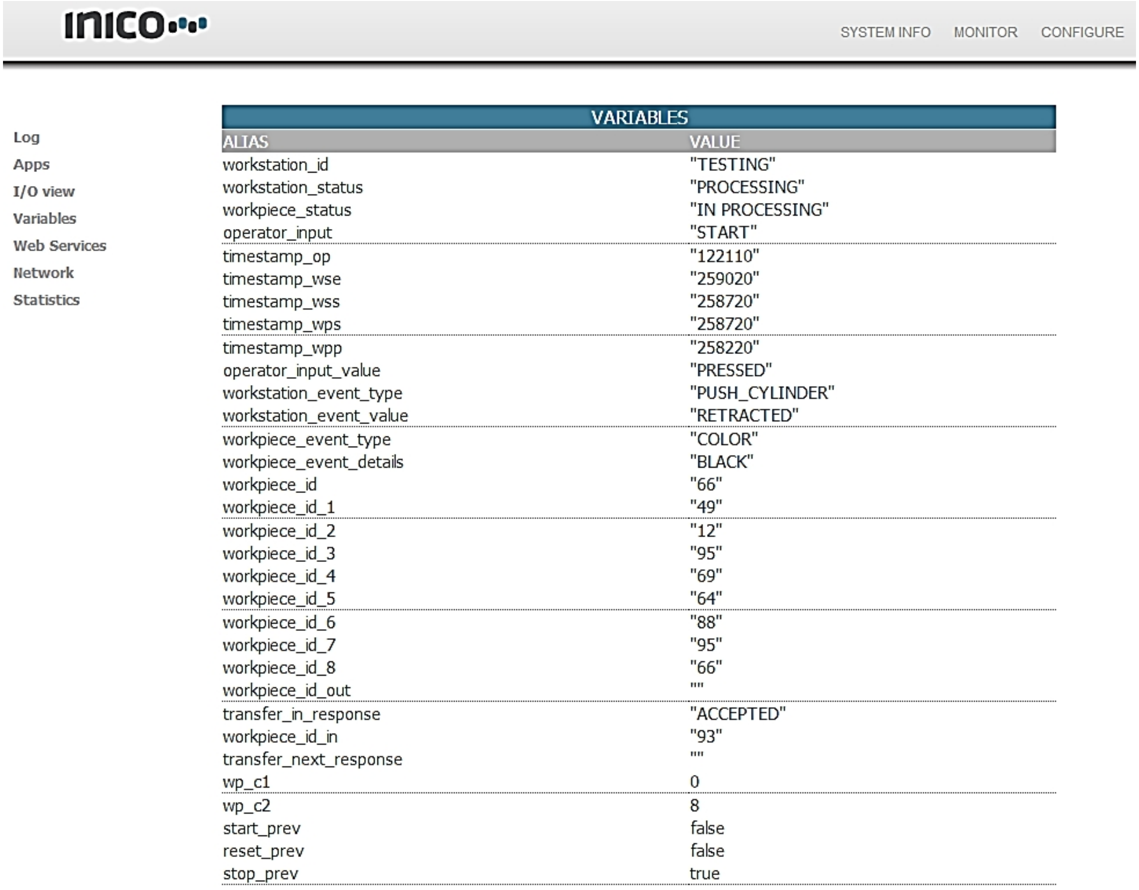
APPENDIX 2– THE RESULTS OF CHAPTER 3



VARIABLES

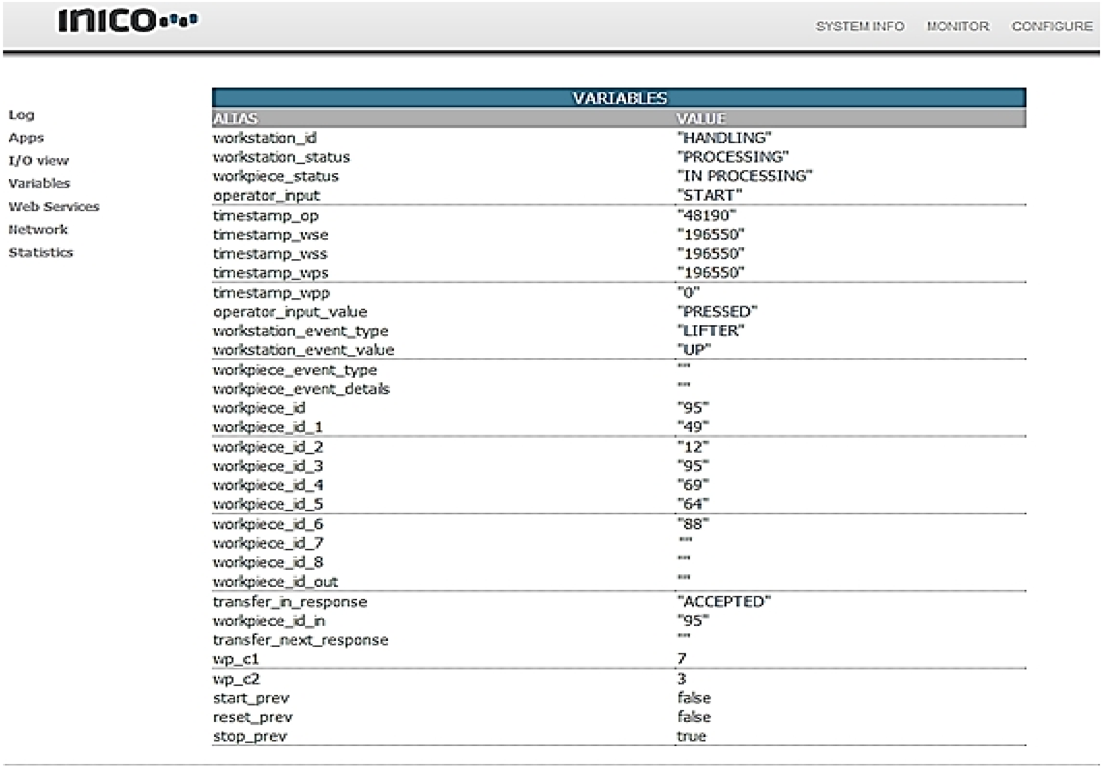
ALIAS	VALUE
workstation_id	"DISTRIBUTION"
workstation_status	"WAITING FOR A WO"
workpiece_status	"IN PROCESSING"
operator_input	"START"
timestamp_op	"845060"
timestamp_wse	"916230"
timestamp_wss	"925260"
timestamp_wps	"916230"
timestamp_wpp	"0"
timestamp_wsa	"925260"
operator_input_value	"PRESSED"
workstation_event_type	"ARM"
workstation_event_value	"LEFT"
workstation_alarm_type	"MAGAZINE_ALARM"
workstation_alarm_value	"EMPTY"
workpiece_event_type	""
workpiece_event_details	""
workpiece_id	"88"
workpiece_id_out	"88"
transfer_in_response	"REJECTED"
workpiece_id_in	"12345"
transfer_next_response	"ACCEPTED"
neighbour_1	"http://192.168.3.2:80/dpws/ws01"
neighbour_2	"http://192.168.3.3:80/dpws/ws01"
neighbour_3	"http://192.168.3.4:80/dpws/ws01"
neighbour_4	"http://192.168.3.5:80/dpws/ws01"
neighbour_5	"http://192.168.3.6:80/dpws/ws01"
workpiece_num	0
start_prev	false
reset_prev	false
stop_prev	true

Figure 76 – The values for the distribution station Web-Service monitoring variables



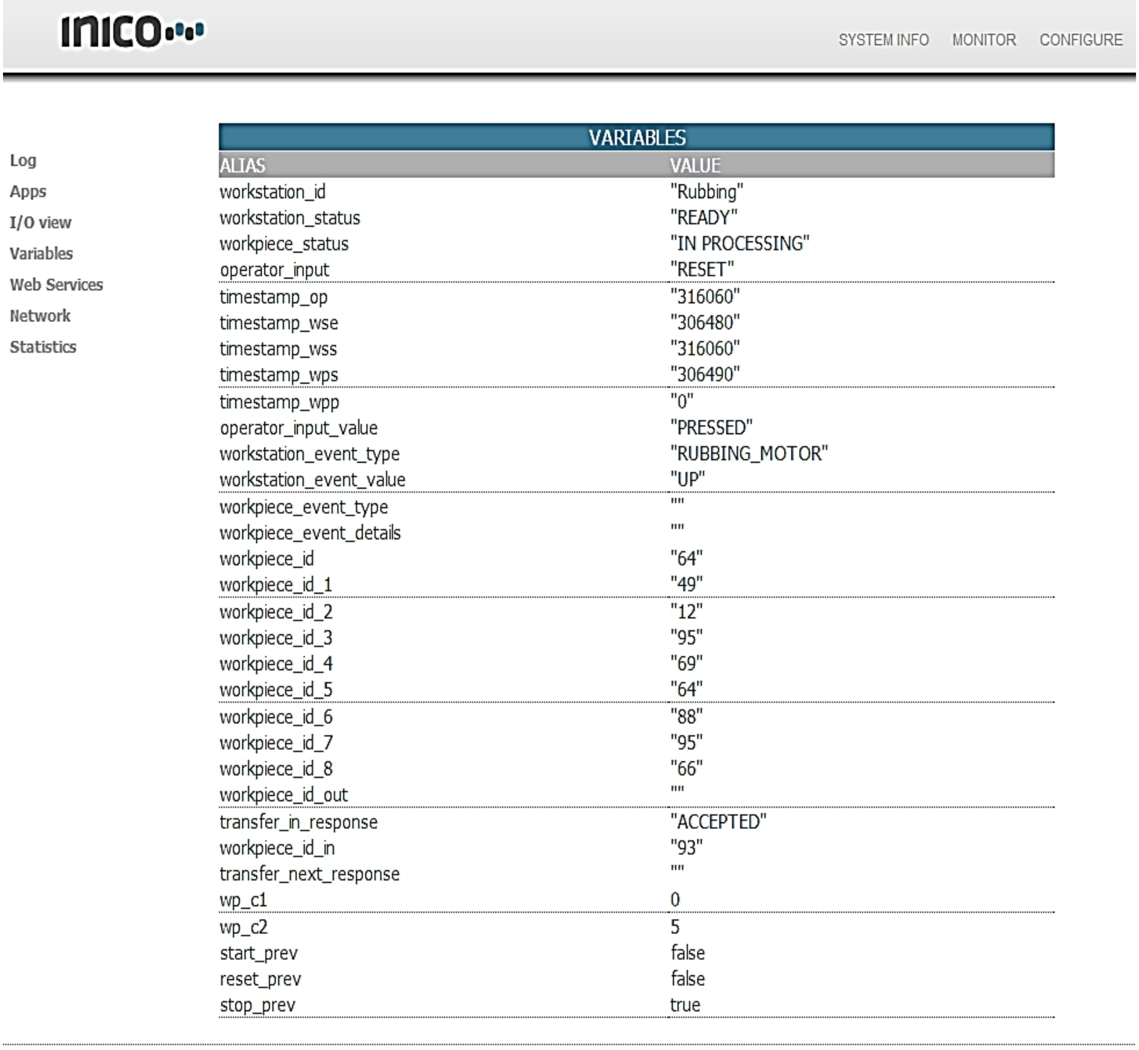
VARIABLES	
ALIAS	VALUE
workstation_id	"TESTING"
workstation_status	"PROCESSING"
workpiece_status	"IN PROCESSING"
operator_input	"START"
timestamp_op	"122110"
timestamp_wse	"259020"
timestamp_wss	"258720"
timestamp_wps	"258720"
timestamp_wpp	"258220"
operator_input_value	"PRESSED"
workstation_event_type	"PUSH_CYLINDER"
workstation_event_value	"RETRACTED"
workpiece_event_type	"COLOR"
workpiece_event_details	"BLACK"
workpiece_id	"66"
workpiece_id_1	"49"
workpiece_id_2	"12"
workpiece_id_3	"95"
workpiece_id_4	"69"
workpiece_id_5	"64"
workpiece_id_6	"88"
workpiece_id_7	"95"
workpiece_id_8	"66"
workpiece_id_out	""
transfer_in_response	"ACCEPTED"
workpiece_id_in	"93"
transfer_next_response	""
wp_c1	0
wp_c2	8
start_prev	false
reset_prev	false
stop_prev	true

Figure 77– The values for the testing station Web-Service monitoring variables



VARIABLES	
ALIAS	VALUE
workstation_id	"HANDLING"
workstation_status	"PROCESSING"
workpiece_status	"IN PROCESSING"
operator_input	"START"
timestamp_op	"48190"
timestamp_wse	"196550"
timestamp_wss	"196550"
timestamp_wps	"196550"
timestamp_wpp	"0"
operator_input_value	"PRESSED"
workstation_event_type	"LIFTER"
workstation_event_value	"UP"
workpiece_event_type	""
workpiece_event_details	""
workpiece_id	"95"
workpiece_id_1	"49"
workpiece_id_2	"12"
workpiece_id_3	"95"
workpiece_id_4	"69"
workpiece_id_5	"64"
workpiece_id_6	"88"
workpiece_id_7	""
workpiece_id_8	""
workpiece_id_out	""
transfer_in_response	"ACCEPTED"
workpiece_id_in	"95"
transfer_next_response	""
wp_c1	7
wp_c2	3
start_prev	false
reset_prev	false
stop_prev	true

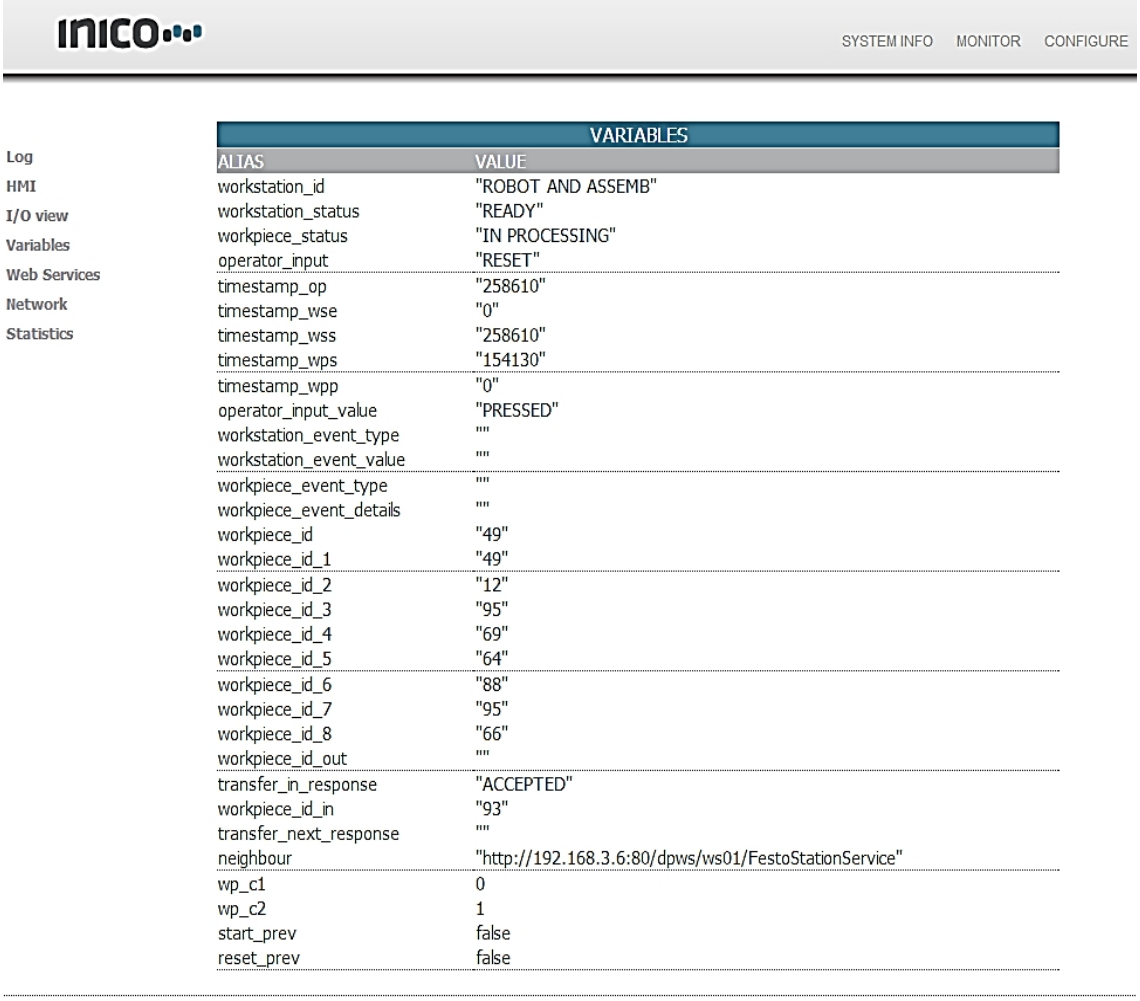
Figure 78 – The values for the handseling station Web-Service monitoring variables



VARIABLES

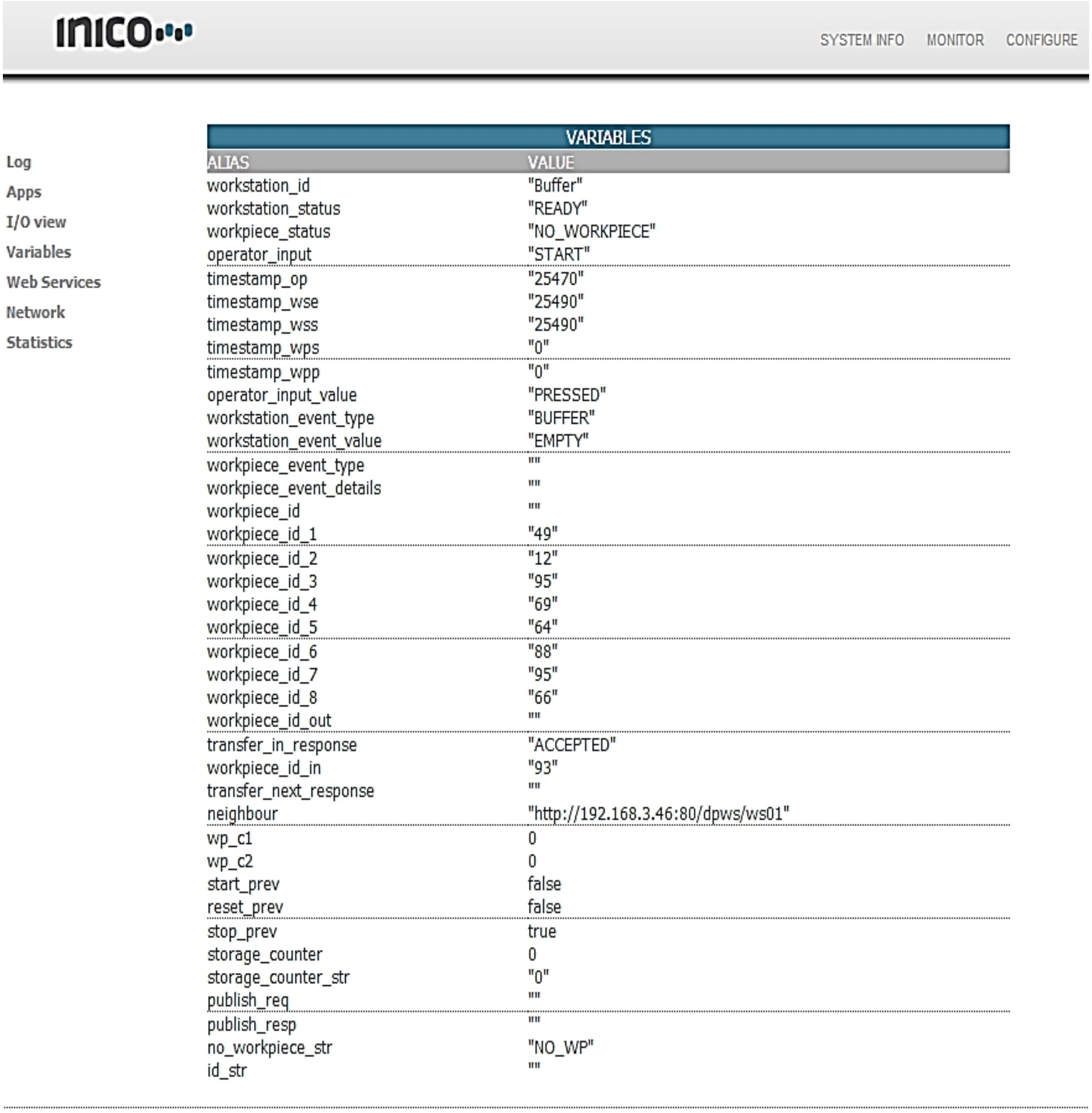
ALIAS	VALUE
workstation_id	"Rubbing"
workstation_status	"READY"
workpiece_status	"IN PROCESSING"
operator_input	"RESET"
timestamp_op	"316060"
timestamp_wse	"306480"
timestamp_wss	"316060"
timestamp_wps	"306490"
timestamp_wpp	"0"
operator_input_value	"PRESSED"
workstation_event_type	"RUBBING_MOTOR"
workstation_event_value	"UP"
workpiece_event_type	""
workpiece_event_details	""
workpiece_id	"64"
workpiece_id_1	"49"
workpiece_id_2	"12"
workpiece_id_3	"95"
workpiece_id_4	"69"
workpiece_id_5	"64"
workpiece_id_6	"88"
workpiece_id_7	"95"
workpiece_id_8	"66"
workpiece_id_out	""
transfer_in_response	"ACCEPTED"
workpiece_id_in	"93"
transfer_next_response	""
wp_c1	0
wp_c2	5
start_prev	false
reset_prev	false
stop_prev	true

Figure 79 – The values for the rubbing station Web-Service monitoring variables



VARIABLES	
ALIAS	VALUE
workstation_id	"ROBOT AND ASSEMB"
workstation_status	"READY"
workpiece_status	"IN PROCESSING"
operator_input	"RESET"
timestamp_op	"258610"
timestamp_wse	"0"
timestamp_wss	"258610"
timestamp_wps	"154130"
timestamp_wpp	"0"
operator_input_value	"PRESSED"
workstation_event_type	""
workstation_event_value	""
workpiece_event_type	""
workpiece_event_details	""
workpiece_id	"49"
workpiece_id_1	"49"
workpiece_id_2	"12"
workpiece_id_3	"95"
workpiece_id_4	"69"
workpiece_id_5	"64"
workpiece_id_6	"88"
workpiece_id_7	"95"
workpiece_id_8	"66"
workpiece_id_out	""
transfer_in_response	"ACCEPTED"
workpiece_id_in	"93"
transfer_next_response	""
neighbour	"http://192.168.3.6:80/dpws/ws01/FestoStationService"
wp_c1	0
wp_c2	1
start_prev	false
reset_prev	false

Figure 80 – The values for the robot and assembly station Web-Service monitoring variables



VARIABLES

ALIAS	VALUE
workstation_id	"Buffer"
workstation_status	"READY"
workpiece_status	"NO_WORKPIECE"
operator_input	"START"
timestamp_op	"25470"
timestamp_wse	"25490"
timestamp_wss	"25490"
timestamp_wps	"0"
timestamp_wpp	"0"
operator_input_value	"PRESSED"
workstation_event_type	"BUFFER"
workstation_event_value	"EMPTY"
workpiece_event_type	""
workpiece_event_details	""
workpiece_id	""
workpiece_id_1	"49"
workpiece_id_2	"12"
workpiece_id_3	"95"
workpiece_id_4	"69"
workpiece_id_5	"64"
workpiece_id_6	"88"
workpiece_id_7	"95"
workpiece_id_8	"66"
workpiece_id_out	""
transfer_in_response	"ACCEPTED"
workpiece_id_in	"93"
transfer_next_response	""
neighbour	"http://192.168.3.46:80/dpws/ws01"
wp_c1	0
wp_c2	0
start_prev	false
reset_prev	false
stop_prev	true
storage_counter	0
storage_counter_str	"0"
publish_req	""
publish_resp	""
no_workpiece_str	"NO_WP"
id_str	""

Figure 81 – The values for the buffer station Web-Service monitoring variables

APPENDIX 3 – XML INSTANCES USED FOR SIMULATING THE MACHINES WEB-SERVICES MESSAGES

```
<Labourer xmlns="http://www.tut.fi/fast/plantcockpit">
  <LabourerID>1234</LabourerID>
  <StationID>5678</StationID>
  <TimeStamp>46556</TimeStamp>
  <LabourerAction>ManualReset</LabourerAction>
</Labourer>

<WorkPiece xmlns="http://www.tut.fi/fast/plantcockpit">
  <WPID>1</WPID>
  <StationID>1</StationID>
  <TimeStamp>4</TimeStamp>
  <WPColor>NonBlack</WPColor>
  <WPQuality>Defected</WPQuality>
  <WPStatus>Pause</WPStatus>
</WorkPiece>

<Station xmlns="http://www.tut.fi/fast/plantcockpit">
  <StationID>1</StationID>
  <TimeStamp>4</TimeStamp>
  <StationEvent>Alarm</StationEvent>
  <EventSourceComponent>arm</EventSourceComponent>
  <StationStatus>Idle</StationStatus>
</Station>


<tns:OperatorInput xmlns:tns="http://www.plantcockpit.eu/fast/festo">
  <tns:StationId>BU</tns:StationId>
  <tns:Type>STOP_BUTTON</tns:Type>
  <tns:Value>anything</tns:Value>
  <tns:Timestamp>41256</tns:Timestamp>
</tns:OperatorInput>


<tns:WorkpieceStatus xmlns:tns="http://www.plantcockpit.eu/fast/festo">
  <tns:StationId>PU</tns:StationId>
  <tns:WorkpieceId>8</tns:WorkpieceId>
  <tns:Status>TRANSFERIN</tns:Status>
  <tns:Timestamp>42073</tns:Timestamp>
</tns:WorkpieceStatus>
```

APPENDIX 4 – JAVA CODE USED FOR CHAPTER 5

```
package fi.tut.fast.smx;

import java.io.File;
import java.io.FileFilter;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.StringWriter;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBElement;
import javax.xml.bind.JAXBException;
import javax.xml.bind.JAXBIntrospector;
import javax.xml.bind.Marshaller;
import javax.xml.bind.Unmarshaller;
import javax.xml.namespace.QName;
import org.apache.camel.Exchange;
import org.apache.camel.Processor;
import org.apache.commons.io.IOUtils;
import org.drools.KnowledgeBase;
import org.drools.KnowledgeBaseConfiguration;
import org.drools.KnowledgeBaseFactory;
import org.drools.KnowledgeBaseFactoryService;
import org.drools.builder.KnowledgeBuilder;
import org.drools.builder.KnowledgeBuilderError;
import org.drools.builder.KnowledgeBuilderFactory;
import org.drools.builder.KnowledgeBuilderFactoryService;
import org.drools.builder.ResourceType;
import org.drools.builder.help.KnowledgeBuilderHelper;
import org.drools.conf.EventProcessingOption;
import org.drools.io.ResourceFactory;
import org.drools.io.ResourceFactoryService;
import org.drools.runtime.Channel;
import org.drools.runtime.StatefulKnowledgeSession;
import org.osgi.framework.BundleContext;
import org.xml.sax.InputSource;
import com.sun.tools.xjc.Language;
import com.sun.tools.xjc.Options;

public class DroolsTest implements Processor {

   private static final transient Logger logger = Logger.getLogger(DroolsTest.class.getName());// System ID used for compiling
schema types
                public static final String JAXBSYSTEM_ID = "DROOLSXSD";
                // Constants to define expected file extensions for schema and rules files
                public static final String XSD_FILE_EXT = "xsd";
                public static final String DRL_FILE_EXT = "drl";
                //  Output Channel Name
                public static final String OUTPUT_CHANNEL_NAME = "PC_OUTPUT_CHANNEL";
```

```
private StatefulKnowledgeSession ksession;
private Marshaller marshaller;
private Unmarshaller unmarshaller;
private JAXBIntrospector introspector;// For interacting with servicemix private BundleContext context;

// Shutdown DroolsTest
public void destroy() throws Exception {
logger.info("OSGi Bundle Stopping.");
ksession.dispose();
}
//  Setting up JAXB Schema Compiler Options
private Options getOptions() throws IOException{
// Configure XJC options
//  Simple Binding for adding XMLRootElement annotations to Elements.
URL bindingUrl = context.getBundle().getResource("/types/binding.xjb");
InputSource bind = new InputSource(ResourceFactory.newUrlResource(bindingUrl).getInputStream());
bind.setSystemId(JAXBSYSTEM_ID);

Options xjcOpts = new Options();
    xjcOpts.setSchemaLanguage( Language.XMLSCHEMA );
    xjcOpts.compatibilityMode = Options.EXTENSION;
    xjcOpts.addBindFile(bind);

    return xjcOpts;
}

// Initialize the drools Block
public void init() throws Exception {

    System.out.println("Rules Block Started...");
    try {
                    refreshRulesSession();
    } catch (JAXBException e) {
                    System.out.println("Error Parsing Schema. Check logs...\n" + e.getMessage() );
                    logger.log(Level.SEVERE,"Error Parsing Schema. Check logs...\n" + e.getMessage() , e);
    } catch (Exception e) {
                    System.out.println("Exception Caught while refreshing rules. Check logs...\n" + e.getMessage() );
                    logger.log(Level.SEVERE,"Exception Caught while refreshing rules. Check logs...\n" + e.getMessage() , e);
    }

}

//  Process an incoming XML message from the HTTP Endpoint
@Override
public void process(Exchange exchange) throws Exception {

    // Convert XML to Java object
    Object o = unmarshaller.unmarshal(exchange.getIn().getBody(InputStream.class));

    // Inject the object into the Drools Working Memory
    if(o instanceof JAXBElement){
                    System.out.println("Recieved JAXB Element: " + ((JAXBElement)o).getDeclaredType().getCanonicalName());
                    ksession.insert(((JAXBElement)o).getValue());
    }else{
                    System.out.println("Recieved Element: " + o.getClass().getCanonicalName());
                    ksession.insert(o);
    }

    // Evaluate all rules
    ksession.fireAllRules();
```

```
            // Send output response
            exchange.getOut().setBody("Accepted.");

        }


        // Refresh rules session when new XSD or DRL files are received.
        private void refreshRulesSession() throws IOException, JAXBException, Exception {

            // Dispose of the old session if it exists.
            if(ksession != null){
                            ksession.dispose();
                            ksession = null;
            }

            // Get all the cached files
            List<URL> modelFiles = getCachedFiles(XSD_FILE_EXT);
            List<URL> rulesFiles = getCachedFiles(DRL_FILE_EXT);

            //  If there is no schema, do nothing.
            if(modelFiles.isEmpty()){
                            System.out.println("No schemas loaded.");
                            return;
            }

            // If there are no rules, do nothing.
            if(rulesFiles.isEmpty()){
                            System.out.println("No Rules loaded.");
                            return;
            }

        // Loading XSDs with Type Definitions
KnowledgeBuilder builder = KnowledgeBuilderFactory.newKnowledgeBuilder();
List<String> classList = new ArrayList<String>();
for(URL url : modelFiles){
            String[] classNames = KnowledgeBuilderHelper.addXsdModel(  ResourceFactory.newUrlResource(url),

                            builder,

                            getOptions(),

                            JAXBSYSTEM_ID );
            classList.addAll(Arrays.asList(classNames));
}

// Set Up Knowledge Base
KnowledgeBaseConfiguration kbaseConfig = KnowledgeBaseFactory.newKnowledgeBaseConfiguration();
kbaseConfig.setOption( EventProcessingOption.STREAM );
            KnowledgeBase knowledgeBase = KnowledgeBaseFactory.newKnowledgeBase(kbaseConfig);

            // Add Rules Definitions Files
for(URL url : rulesFiles){
            builder.add(ResourceFactory.newUrlResource(url), ResourceType.DRL);
            // Check Errors
            for(KnowledgeBuilderError err : builder.getErrors()){
                            System.err.println(err);
            }
}

// Set up knowledge session
knowledgeBase.addKnowledgePackages(builder.getKnowledgePackages());
            ksession = knowledgeBase.newStatefulKnowledgeSession();
            // Get Marshaller, Unmarshaller, and Introspector.
  JAXBContext jcontext = KnowledgeBuilderHelper.newJAXBContext(classList.toArray(new String[]{}), knowledgeBase);
  introspector = jcontext.createJAXBIntrospector();
            marshaller = jcontext.createMarshaller();
            marshaller.setProperty( Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE );
            unmarshaller = jcontext.createUnmarshaller();

            System.out.println("Rule Engine Initialized!");

            // Set up Output Channel to recieve and marshall objects and messages from rules
            ksession.registerChannel("PC_OUTPUT_CHANNEL", new Channel(){
                    @Override
                    public void send(Object obj) {

                            if(obj instanceof String){
                                    // Simple String
                                    System.out.println("Message Receieved on Output
Channel: " + obj);
                            }else{
                                    // Object to be unmarshalled
                                    StringWriter writer = new StringWriter();
                                    QName name = introspector.getElement-
Name(obj);
                                    if(name == null){
                                            name = new
QName(obj.getClass().getPackage().getName(),obj.getClass().getSimpleName());
                                    }
                                     try {
                                            marshaller.marshal(new JAX-
BElement(name,obj.getClass(), obj), writer);
                                            System.out.println("Object
Recieved on Output Channel:");
                                            Sys-
tem.out.println(writer.toString());
                                    } catch (JAXBException e) {
                                            logger.log(Level.SEVERE,
"Cannot Unmashall object on Output Channel.", e);
                                    }
                            }
                    }
            });

            ksession.fireAllRules();

        }


        // Store Schemas and rules in folder
        private void cacheFile(Exchange exchange) throws FileNotFoundException, IOException{
//          System.out.println(Arrays.toString(exchange.getIn().getHeaders().keySet().toArray(new String[]{})));
            File rulesFile = context.getDataFile(exchange.getIn().getHeader(Exchange.FILE_NAME, String.class));
            if(rulesFile.exists()){
                    rulesFile.delete();
            }
            InputStream fileIs = exchange.getIn().getBody(InputStream.class);
            IOUtils.copy(fileIs, new FileOutputStream(rulesFile));
            fileIs.close();
        }

        // Handles new DRL and XSD files dropped into the hot folder
        public void newFileDropped(Exchange exchange) {

            try {
                    cacheFile(exchange);
            } catch (FileNotFoundException e1) {
                    System.out.println("FileNotFoundException caught while caching file. Check logs...\n"+
e1.getMessage() );
                    logger.log(Level.SEVERE,"FileNotFoundException caught while caching file. Check
logs...\n" + e1.getMessage() , e1);
                    return;
            } catch (IOException e1) {
                    System.out.println("IOException caught while caching file. Check logs...\n" + e1.getMes-
sage() );
                    logger.log(Level.SEVERE,"IOException caught while caching file. Check logs...\n" +
e1.getMessage() , e1);
                    return;
            }

            System.out.println("[Drools] New File loaded: " + exchange.getIn().getHeader(Exchange.FILE_NAME,
String.class));
            try {
                    refreshRulesSession();
            } catch (JAXBException e) {
                    System.out.println("Error Parsing Schema. Check logs...\n" + e.getMessage() );
                    logger.log(Level.SEVERE,"Error Parsing Schema. Check logs...\n" + e.getMessage() , e);
            } catch (Exception e) {
                    System.out.println("Exception Caught while refreshing rules. Check logs...\n" + e.getMes-
sage() );
                    logger.log(Level.SEVERE,"Exception Caught while refreshing rules. Check logs...\n" +
e.getMessage() , e);
            }
        }

        // Search cache for files of a certian extension
        private List<URL> getCachedFiles(final String ext){
            List<URL> files = new ArrayList<URL>();
            File dir = context.getDataFile("");

            for(File f : dir.listFiles(new FileFilter(){
                    @Override
                    public boolean accept(File pathname) {
                            return pathname.getName().endsWith(ext);
                    }
            })){
                    try{
                            files.add(f.toURI().toURL());
                    }catch(MalformedURLException ex){
                            logger.log(Level.WARNING, "Ignoring File: " , ex);
                    }
            }

            return files;
        }

        // Clear the file cache
        public void clearCachedFiles(Exchange ex){
            clearFileCache();
            ex.getOut().setBody("CACHE CLEARED.");
        }

        private void clearFileCache(){
            File dir = context.getDataFile("");

            for(File f : dir.listFiles(new FileFilter(){
                    @Override
                    public boolean accept(File pathname) {
```

```
                                    return pathname.getName().endsWith(XSD_FILE_EXT)
                                                            ||          pathname.getName().endsWith(DRL_FILE_EXT) ;
                        }
            })){
                        f.delete();
            }
        }

        public BundleContext getContext() {
            return context;
        }

        public void setContext(BundleContext context) {
            this.context = context;
        }

}
```