TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

SERGII IAROVYI

SEMANTIC MODELLING FOR DYNAMIC SYSTEM RECOGNITION IN NON-INTRUSIVE INDUSTRIAL MONITORING SYSTEMS

Master of Science Thesis

Thesis topic approved in the Faculty of Engineering Sciences meeting on 8th of May 2013.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Machine Automation
IAROVYI, SERGII: Semantic modelling for dynamic system recognition in non-intrusive industrial monitoring systems
Master of Science Thesis, 62 pages, 12 Appendix pages
March 2014
Major: Factory Automation
Examiner: Prof. José Luis Martínez Lastra
Supervisor: Prof. José Luis Martínez Lastra, Jorge Garcia Izaguirre Montemayor
Keywords: Semantics, Device Profile for Web Services, semantic annotations, monitoring system, smart shop floor devices, enterprise integration, ontology, knowledge-driven configuration, function blocks.

Industrial monitoring systems play important role in decision making on all levels of a factory from the shop floor to ERP systems influencing overall efficiency of production. Together with a trend for mass customization and constantly increasing tempo of introduction of new products, equipment and technologies to manufacturing, contemporary monitoring systems should provide enough flexibility to be up to date with manufacturing system. Such monitoring systems as the one offered in European Commission project PLANTCockpit, offer the approach of extensively reconfigurable, loosely coupled systems. Unfortunately, configuration of the monitoring system which could work on all levels of automation hierarchy requires the knowledge of all those levels together with knowledge of integration technologies and tedious work related with creation of configuration itself.

Present thesis work offers an approach which automates the configuration process employing knowledge bases. This approach includes employment of SOA on device level, with semantically enhanced services descriptions (and possibility to employ the gateway devices for non-intrusiveness), definition of the metrics to be monitored by the system in the knowledge base, as well as set of algorithms and standards required to create configuration of the monitoring system. Reusability of knowledge defined on devices and in knowledge base simplifies the process of introduction of new devices, metrics or other reconfiguration of the monitoring system. The system implementing proposed approach has been developed in this thesis and was able to configure monitoring system for a test bed.

PREFACE

After several months of search, research, implementation, reimplementation, writing and rewriting, finally of procrastination this work is ready to be published. I would not manage to make this thorny path alone, so I would like to mention the ones who helped me the most.

I am grateful to prof. José Luis Martínez Lastra for possibility to develop myself in Factory Automation domain as a student and as a researcher. As well I am thankful to all the people who have been teaching during courses I have had during my Master studies.

My very special thanks should go to Jorge, with whom I worked in PLANTCockpit, during last two years, who gave me plenty of important advices in work and study, supervised this work and occasional encounters with whom in Kultainen Apina were fun.

I would like to mention the friends from almost all possible continents whom I found in TUT and who made my leisure, study and work full of joy. So cheers to Borja and Amalia, Ahmed, Manu, Oscar, Zoran, Carlos, Gerardo.

Also I would like to thank my parents and whole family in Ukraine who cared about me, supported me and asked me hundreds of times about advance of my thesis.

Finally, my biggest possible gratitude is dedicated to my wife Daryna, who encouraged me to apply and get through Master Studies, as well, she inspired me on multiple other important deeds, and who simply makes me happy.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ANSI	American National Standards Institute
B2MML	Business To Manufacturing Markup Language
CAEX	Computer Aided Engineering Exchange
CEP	Complex Event Processing
CFBNO	Configuration Function Block Network Ontology
DCS	Distributed Control System
DPWS	Device Profile for Web Services
EAI	Enterprise Application Integration
EDA	Event-Driven Architecture
EIP	Enterprise Integration Patterns
EIS	Enterprise Information System
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
EU	European Union
FB	Function block
FBN	Function Block Network
FIS	Factory Information System
GDP	Gross Domestic Product
HMI	Human Machine Interface
HTTP	Hyper Text Transfer Protocol
ICT	Information Communication Technologies
IEC	International Electrotechnical Commission
ISA	International Society of Automation
ISO	International Organization for Standardization
IT	Information Technologies
JMEDS	Java Multi Edition Device Profile for Web Services Stack
KB	Knowledge Base
MES	Manufacturing Execution System
OPC-UA	OPC - Unified Architecture
OWL	Web Ontology Language
PLANTCockpit	Production Logistics and Sustainability Cockpit
RDF	Resource Description Framework
SAWSDL	Semantic Annotations for Web Service Description Language
SCADA	Supervisory Control And Data Acquisition
SOA	Service-Oriented Architecture
SOA4D	Service-Oriented Architecture for Devices
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SWRL	Semantic Web Rule Language
UDDI	Universal Description, Discovery and Integration
UDP	User Datagram Protocol

W3C	World Wide Web Consortium
WS	Web Service
WSDL	Web Service Description Language
WSMO	Web Service Modelling Ontology
XML	Extendable Markup Language
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformations
WS4D	Web Services for Devices

1. INTRODUCTION

The chapter provides a motivation for this thesis work, defines the problem, which is addressed in this research, specifies research objectives, limitations and methodology. Additionally structure of the document is described in this section.

1.1. Motivation

Manufacturing is a vitally important sector of economy of European Union. Before the financial crisis, in 2007 share of manufacturing in GDP of EU was 17.1%. Taking into account corresponding energy generation, construction and business services the aggregated share in GDP increases to 37%. In general, about 47% of GDP of European Union are closely related with manufacturing, by estimations of European Commission's Directorate-General for Enterprise and Industry [1]. During crisis the share of manufacturing reduced, and in 2010 the manufacturing contributed 16.5% of GDP [2] of European Union. Considering non-financial business economy, it is the biggest share of value added 26.8%, employing 22.6% of workers [3]. Hence the manufacturing is still an effective and powerful locomotive of European economy.

On-going saturation of the markets and results of financial crisis constantly escalates the competition between the manufacturers. Contemporary enterprise should execute the customized orders in short terms with maximal resource efficiency and labour productivity. In [4] among five main factors of production efficiency the *technology* and closely related *energy intensity* had been defined; also the *labour productivity* has been selected as one of two competitiveness indicators of the EU economy. The labour productivity per employee in manufacturing has been annually increasing since 1995 at average by 2% (as for 2009), which is about twice higher than average in economy. Moreover, in 2010 the labour productivity per hour worked in manufacturing had grown by 8.5% [4]. In long term improvement of the worker productivity and even more obvious in the growth of productivity per hour the technology is a key determinant. These facts leads us to the conclusion, that technology plays important role in European manufacturing, increases the competitiveness of economy as whole and thus ensuring prosperity.

The new technologies offer more effective means to produce the conventional products or even introduce the innovative ones. Another important idea employed to maximize efficiency of enterprises is concept of information systems, which can reduce of paperwork, help to optimize manufacturing and predict the flow of industrial processes. This approach also allows to detect and to solve some of the problems before they affected the production process.

1.2. Problem Definition

Currently, the benefits provided by the implementation of new technologies and related equipment in the manufacturing are reduced by expenses required for reconfiguration. The lower reconfiguration costs are, the more likely factories would faster adapt new technologies and equipment, and hence the manufacturing will be more effective, sustainable and competitive.

Contemporary factory information systems are, usually, customized to manufacturing system and change of the last one leads to reimplementation of information system. This process significantly increases time and cost of introduction of the new technologies, thus reducing benefits from both new technologies and manufacturing systems.

The advance in microcontroller technology, as well as, in IT and ICT domains which can be observed during last decades, introduces new potential to factory automation, especially in domain of factory information systems. The concepts, technologies and tools already developed in the named domains can be applied to manufacturing, reducing the cost of implementation of the new technologies.

Summing up, the currently sufficient efforts to set up monitoring system are required on introduction of new equipment, products or technologies to factory shop floor. Monitoring systems employing modern software concepts and technologies can be dynamically configured, but still require abundant amount of data about shop floor devices and layout. Contemporary technologies and smart devices are providing opportunities for exposure shop floor device descriptions by device itself.

1.2.1. Justification for the work

Re-configurability of the manufacturing systems is closely related to reusability of the system parts. Service Oriented Architecture (SOA) plays an important role in reusability of contemporary software modules in IT [5]. Introduction of this approach to manufacturing domain had been going under several projects and initiatives [6]–[10] starting from 2000s. In this endeavours the functional implementation of SOA based Web Services (WS) technology had been developed – Device Profile for WS (DPWS). The DPWS-enabled devices are currently available on the market [11]. These devices can be employed on the factory shop floor as controllers, or as a gateway to existing equipment, allowing non-intrusive creation of device SOA.

Currently, Event Driven Architecture (EDA), concept of Function Blocks (FB) and loosely coupled integration are enhancing the reusability to systems in IT domain. Same set of concepts is lying in the basis of the layered cockpit architecture, which is being implemented in the PLANTCockpit EU project [12]. This approach makes factory information system dynamically configurable, e.g. no customization or recompilation of the information system applications is needed, once the business functionality of the last one has to be changed. Moreover, DPWS-enabled devices, mentioned in previous paragraph, can be bridged to factory information system as shown in [13].

In order to automate the reconfiguration process, on changes in the shop floor, semantic enrichment and reasoning can be used. For semantic enrichment DPWS devices can contain annotations in the service descriptions. Service descriptions with annotations may define the role of device in manufacturing system. The role of device can be used to deduce the meaning of data that this device can deliver to monitoring system. Moreover, usage of standardized industrial taxonomy for named annotations, will allow unambiguous definition of the integrated data. Later the reasoning mechanisms may be employed to create a new configuration for dynamically reconfigurable system, so that information system will be synchronized with changes on the factory shop floor. This approach will reduce the efforts to adjust monitoring system on introduction of new equipment, products or technologies to factory shop floor.

1.2.2. Problem statement

As it was mentioned above the automated configuration process of monitoring systems can reduce both costs of monitoring system maintenance and of new equipment introduction. The approach to achieve of this goal is dynamic system recognition. This approach needs following questions to be answered:

- How to synchronize the configuration of a monitoring system with the real life layout of the manufacturing equipment?
- How to calculate the metrics to be displayed in monitoring system minimising configuration efforts of monitoring system?
- How to define metrics for the monitoring system so that configuration of such can be automated?
- How to identify the current state of the system and transform it into a dynamic monitoring model?

1.3. Work description

1.3.1. Objectives

The objectives of this thesis work are defined as follows:

- Definition of a generic method for system model exposure and discovery:
  - This task consists on the creation of a methodology for exposure of new components into an integration framework based on the literature review. It can be seen as a blueprint that specifies the steps to follow to semantically expose newly added components with different communication protocols.
- Semantic description of the components of an industrial scenario:
  - Modeling of the industrial components. Each component should expose its functionality in a standard way. (Similar to WSDL for WS) How other protocols can expose their functionality while being more expressive than WSDL (e.g. SAWSDL).

- Extension of Semantic Configuration model of a SOA framework:
  - Integration of the approach and models into the whole SOA integration platform. This work should include:
    - Definition of messages for integration with the SOA platform
    - Sequence diagrams describing interaction between components
    - Inclusion example of the information exposed by the newly added component to the system
- Implementation of the system in the test bed:
  - Testing the whole integrated solution in an industrial scenario.
- Validating results in an industrial scenario:
  - Outcome should prove the generation of component networks using dynamic system recognition. It should as well demonstrate how the dynamic recognition allows to adequate the calculations of a certain variable in a system. (e.g. if a system is monitoring process throughput, how adding a new component affects the calculation and how the framework can expand this calculation adding the new component)

1.3.2. Methodology

Current research will be conducted as presented further. At first, the review of related works will be performed. This review will allow to form a theoretical background for the thesis work. Based on this background the methodology for achievement of the stated goal will be defined. After this stage the generic method for system model exposure and discovery and the approach for semantic description of components have to be defined.

Once the technologies, tools models and techniques will be selected the implementation of the solution will be conducted. On implementation stage the algorithms and interactions will be defined and implemented as software. Resulted tool will be tested in industrial scenario. On this testing the test bed system will be configured according to technological decisions defined in methodology.

The capabilities of the tool will be analysed based on results of testing. At last, the developed approach and research process will be evaluated.

1.3.3. Assumptions and limitations

Assumptions made in this work are following:

- Equipment contains its operation descriptions where the final user can define its role and data descriptions.
- Manufacturers enrich the device descriptions with semantic description of data sources, employing the standardized taxonomy. Roles of the devices in manufac-

turing system are to be defined by the end users of equipment (for example experts of manufacturing process).

- Expert creates ontological description of KPIs and metrics to be calculated, employing standardized taxonomy.

1.4. Thesis Outline

This thesis is organized in 6 Chapters. In Chapter 2 theoretical background required for research is presented. Based on theoretical background and thesis objectives Chapter 3 defines methodological approach for current research. Chapter 4 describes implementation of the project and its testing in a test bed. The results obtained during research, implementation and testing. Chapter 6 concludes the thesis work and defines future work.

2. THEORETICAL BACKGROUND

In this chapter the review of the concepts and technologies, related to this thesis will be provided.

2.1. Industrial Monitoring Systems

Industrial monitoring is a process of retrieval, analysis and presentation of the data related to the industrial processes. Industrial monitoring systems are the subdivision of the Factory Information Systems (FIS), which are subset of Enterprise Information Systems (EIS) applied for manufacturing industry. Contemporary EIS are commonly implemented as legacy applications with poor cross-vendor compatibility.

2.1.1. Enterprise application layers

The most employed hierarchy of the applications is based on ISA-95 standard. ISA-95 (also known as ANSI/ISA-95) is a standard dedicated to integration interfaces for the enterprise and control systems for factories. This standard had emerged in 2000s as a result of cooperation of experts under the ISA Committee. ISA-95 also standardized as IEC/ISO 62264.

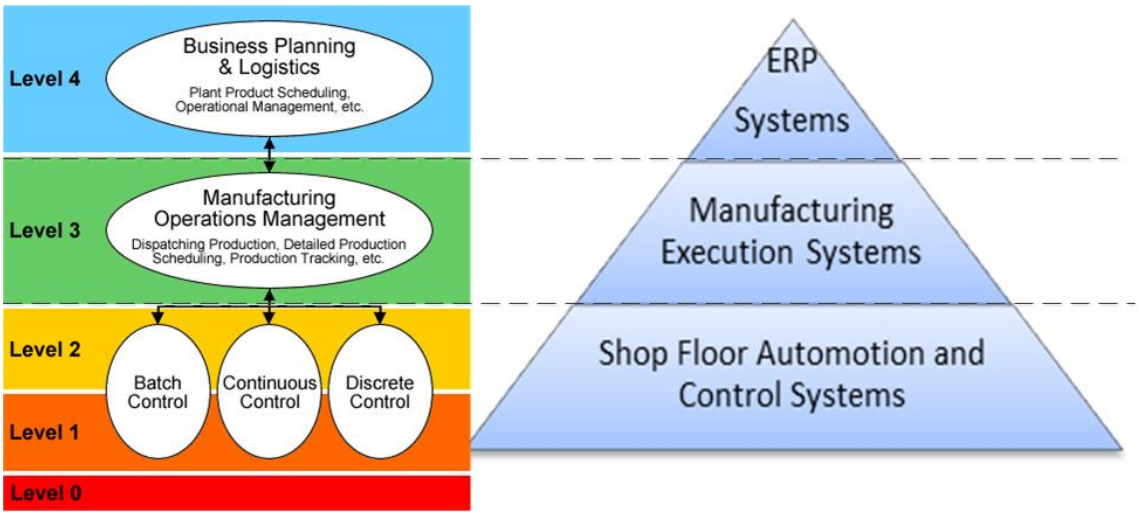

Figure 1: ISA-95 activity hierarchy and corresponding applications hierarchy [14]

ISA-95 is divided in parts and the third part of it described the hierarchy of the activity domains. As activities are executed primarily by the applications this hierarchy is applicable for related applications. Level 4 of activity hierarchy corresponds to layer of Enterprise Resource Planning (ERP) systems, level 3 – to layer of Manufacturing Execution Systems (MES), and levels 0, 1, 2 are matching with shop floor automation and

control systems such as Supervisory Control and Data Acquisition (SCADA) and Distributed Control Systems (DCS) layers. The hierarchy of domain activities proposed by ISA-95 and corresponding hierarchy of applications are depicted on Figure 1.

2.1.2. Flush-up Enterprise Integration Paradigm

As it was mentioned FIS are usually realized as a set of independent applications, employing dissimilar software, approaches, technologies and networking protocols on different levels of hierarchy, presented on Figure 1. These dissimilarities are restricting access to data by layers. However, in such applications as energy management (see [15]) or asset utilization (see [16]) data from different levels of hierarchy is required, and currently this access is achieved by sporadic customized integration solutions.

Generalized approaches for enterprise integration are described in Enterprise Integration Patterns (EIP) by G. Hophe and B. Woolf in[17]. The book emphasize on importance of reliability, scalability and reusability of integration. Concept of loose coupling is assumed by the authors as most beneficial approach for integration. This concept can be implemented employing the philosophy of Service Oriented Architecture (SOA). Web Services (WS) technology is a particular implementation of SOA and is currently most employed in industry.

Another important concept which can be employed to implement loosely coupled integration of the enterprise applications is concept of Function Blocks (FB). Particular realization of idea of FB is IEC-61499.

2.1.2.1 Service Oriented Architecture

Service Oriented Architecture had been defined in 1990s by Sun targeting the EIS domain. SOA is a software design which is based on the services – the independent and interoperable modules of software. SOA services implement some defined functionality which can be called by other services, so the loosely coupled set of services can implement some business logic. This approach allows to reuse the software component (conceptually, even encapsulate non SOA software) thus it reduces development and maintenance costs of SOA enterprise applications.

The concepts lying behind the SOA are modularity and interoperability of services and separated from service functionality algorithms of service discovery and binding. For interoperability the business logic of the service has to be separated from its implementation. To access the business functionality of the service the interfaces are used. The interfaces are separated from the service implementation and this allows modification of each service implementation (business logic) independent of whole service network of SOA application[18].

SOA as a concept has proved its benefits to IT domain especially in the enterprise application integration (EAI). The main benefits of SOA paradigm are summed up in Table 1.

Table 1: Main benefits of the SOA paradigm [5], [19]

Property	Benefit
Interface separated from implementation	Simplification of maintenance of system
Dynamic services discovery	Simplification of extension of the system
Loosely coupled aggregated services	Increased reusability of system parts

Web Services

Currently, there exist several dissimilar technologies implementing SOA concept. Among the technologies implementing SOA concept Web Service (WS) is particularly important. Web Service provider offers its functionality to network, allowing web clients (or service requesters) to use this functionality to process the information.

Operations provided by providers and interfaces required to invoke those operations are defined in the description files. Description files follow Web Service Description Language (WSDL) standard. Messaging between requesters and providers is implemented using SOAP messages. Having the WSDL file describing the service provider the requester can generate the SOAP message to invoke the operation and will receive the response SOAP message schema of which is defined in WSDL. This allows to decouple the implementation of business logic of service providers and requesters. Universal Description Discovery and Integration (UDDI) is designed as a general service registry system, but it was not widely adapted in industry[20], hence several alternatives are created. All mentioned standards are open and XML based. [17]

The WS architecture and main interactions are presented on Figure 2. The actions on Figure 2 are numbered. Initially discovery service obtains the service descriptions from all services it can discover, this is presented on figure as "0.Discovery". Discovery service is a logical role and is usually implemented in requester, provider or other implementation of service. Later once some requester needs to use certain service it can get WSDL describing this service from discovery service ("1.Get WSDL" on Figure 2). This WSDL request can use some criteria (e.g. WSDL location or availability of certain operation) to select the proper description file. If the WSDL matching the criteria is found it will be returned to requester ("2.WSDL" on Figure 2). Once service requested has a WSDL of the provider it can create and parse SOAP messages to communicate and use the service of provider. [19]

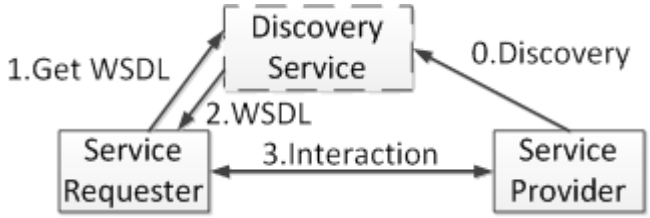

Figure 2: WS architecture and interaction

In following subsections the overview of standards implemented for discovery, description and interaction with services will be performed.

WS-Discovery

Main goal for WS discovery is to provide consumers with requested service descriptions. In W3C standard for web services no single technology is defined for discovery, but three approaches are offered. These approaches are: registry, index and peer-to-peer approach.

Generalizing, registry approach requires discovery service to store all service descriptions in it, index – to store links on descriptions located elsewhere and peer-to-peer discovery approach allows Web Services to discover each other dynamically in WS network. These approaches differ in its robustness and speed of discovery. Peer-to-peer approach is considered as more applicable for dynamic networks, while registry provide most benefits in static ones. [21]

All the mentioned approaches require the discovery protocol to search for service descriptions in the network. This protocol is specified in OASIS WS-Dynamic Discovery (WS-Discovery). WS-Discovery employs multicast SOAP-over-UDP messages to discover the services. This protocol defines messages and their flow to find service descriptions in a local network. On top of this protocol any of approaches from WS standard from W3C can be implemented. [22]

WSDL

There are several versions of WSDL, but version 1.1 is still most employed in industry. WSDL 1.1 is a XML file of a standard schema. W3C standard for WSDL defines the meanings for WSDL elements present in Table 2.

Table 2: WSDL elements according to standard [23]

Name	Meaning
Types	Container for data type definitions using some type system
Message	Abstract, typed definition of the data being communicated
Operation	Abstract description of an action supported by the service
PortType	Abstract set of operations supported by one or more endpoints
Binding	Concrete protocol and data format specification for a particular port-type
Port	Single endpoint defined as a combination of a binding and a network address
Service	Collection of related endpoints

In WSDL the operations are defined as a set of input and output messages. This part of service description defines the message flow required to perform some action and receive the result from it if required. According to message flow there are four types of operations which are presented in Table 3.

In case service provider should send an outbound message to client, it should know the endpoint where to send it. For request-response type of operation the response message will be delivered to requester. But if a requester needs to receive notification type of operation to monitor status of a service a mechanism of registering of requester endpoint is needed.

Table 3: WS operation types [24]

Type	Definition	Pattern
One-way	Operation only receives a message	Input-only
Request-response	Operation receives a request and returns a response	Input-output
Solicit-response	Operation sends a request and waits for a response	Output-input
Notification	Operation only sends a message	Output-only

WS-Eventing

The mechanism of registering interest in the notifications is defined in WS-Eventing. This mechanism is named subscription. To subscribe for notification a requester should send a SOAP message defined in the standard. This message should define which notification to subscribe to, where to send the messages of notification, until when the subscription should be maintained, filters and some other options. Once subscribed, unless the subscription expires or requester had unsubscribed, requester will receive the notifications from the service. [25]

2.1.2.2 Event-Driven Architecture

In [26] J. Van Hoof emphasizes the importance of publish-and-subscribe communication pattern, which should introduce higher level of autonomy of the system components. This communication pattern is associated with a different architecture - Event-Driven Architecture (EDA). In the article SOA is suggested to be used mainly for vertical and EDA for horizontal integration.

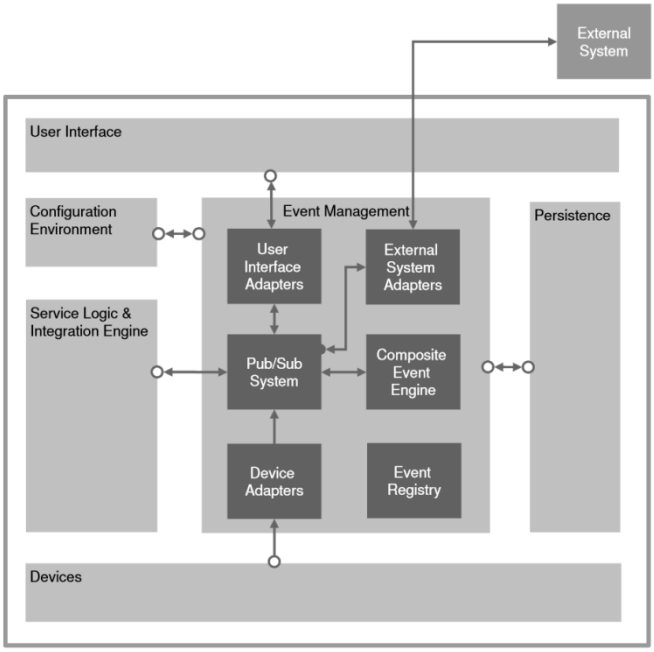

Figure 3: Unified event management architecture by Walzer et al. [27]

K. Walzer et al. in [27] are presenting an architecture for cross-layer enterprise integration, based on EDA concept. This architecture was named unified event management architecture and can be observed on Figure 3. The concept of abovementioned architecture allows to reduce complexity of the integration of dissimilar applications which be-

long to different layers of application hierarchy. Similar approach has been employed in PLANTCockpit project which is described in section 2.1.3.

More advantages for EDA provides Complex Event Processing (CEP), which allows to process event to generate high level events based on most basic ones and sets of rules. Being decoupled from generation of the events itself, CEP provides a flexible method to analyse the status of the system consisting of heterogeneous modules, such as SOA and EDA based ones. [28]

Also Van Hoof in his article notes, that the concept of Enterprise Service Busses (ESB) combines benefits of both SOA and EDA providing loosely coupled integration of services. S. Ortiz Jr. in [29] defines the high reusability of ESB based integration solutions, allowing incremental evolutionary development of enterprise information eco-system. Additionally ESB provides means for further decentralization of the EIS and hence increase robustness and reduce maintenance time and costs.

2.1.2.3 Function Blocks

Another important concept for flush-up enterprise integration is the idea of function blocks. Function blocks can simplify configuration of integration system and introduction of new functionality in it modularising and decoupling business logic from functional logic. Extensive configurability of the FB and separation of functional and business logics allows to avoid customization of the function blocks based system

The most employed concepts of FB are specified in IEC-61131 and IEC-61499. The goal of IEC-61131 is to define common programming languages for automation domain. Among others this standard specifies most basic concept of FB Diagram (FBD) programming language, for application in PLC. Hence IEC-61131 defines the basic concept of FB and their interactions [30].

In more details the concept of FB is developed in IEC-61499. The FB defined in the IEC-61499 is an independent program module, which can be loosely coupled with other FB employing EDA and messaging [31]. To introduce a data from external source, which may not employ FB approach, in IEC-61499 a service interface block type is defined. The concept of FB, defined in IEC-61499 provides scalability, connectivity, reusability and re-configurability of system modules and hence tends to simplify loosely coupled integration of the system [32], [16], [33].

2.1.3. PLANTCockpit

Employing the contemporary technologies and concepts Vasyutynskyy et al. in [16] are proposing a generic approach for data integration in enterprise information system. This approach is based on addition of a parallel level to standard automation ISA-95 based hierarchy. This additional level is employing adapters to retrieve data from the enterprise applications or other relevant sources and can processed retrieved data.

Creation of generic access to dissimilar information systems would simplify the process of integration. This approach is being developed in PLANTCockpit project and can be observed on Figure 4.

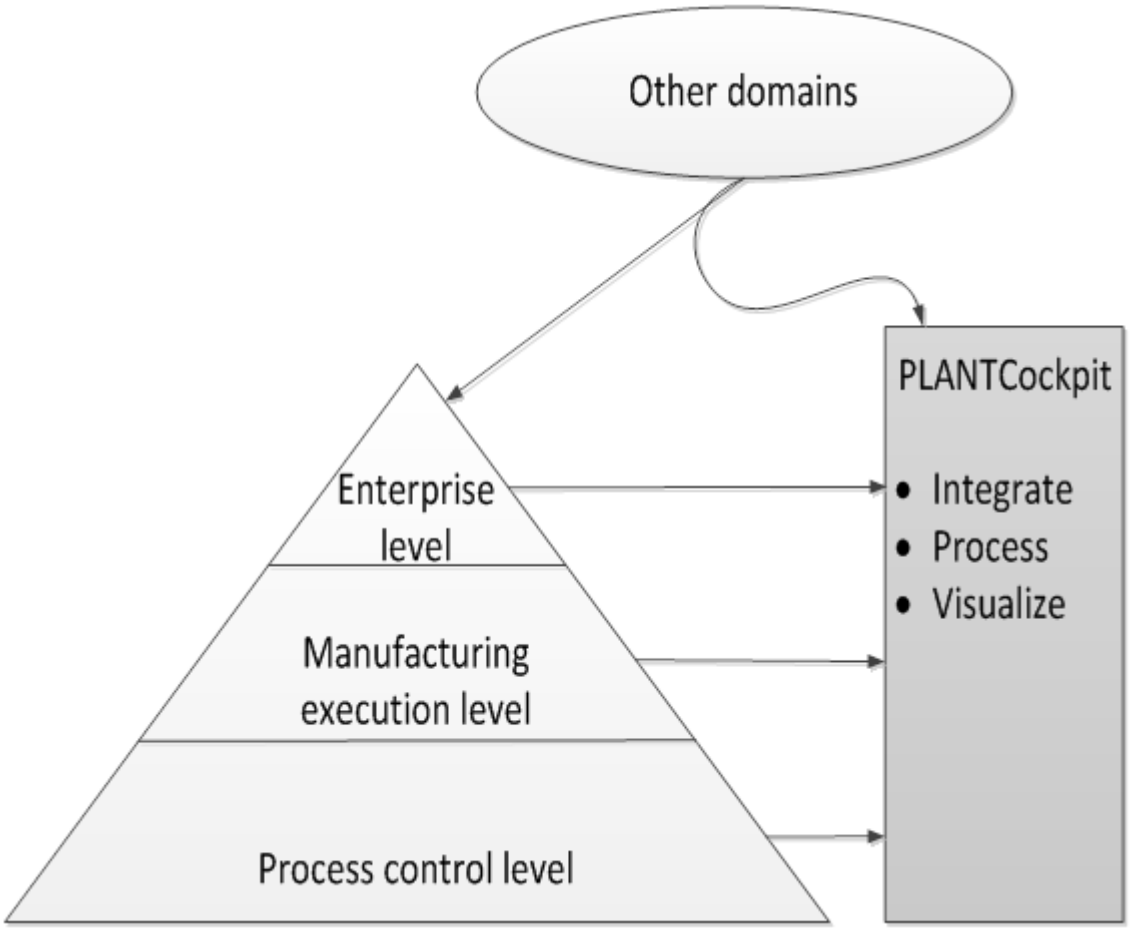

Figure 4: Integration of FIS with PLANTCockpit [15]

The principles on which the project is based are:

- Loose coupling
- Standardized messaging
- Configurability
- Independence
- Extendibility

To address the named principles and to allow high reusability PLANTCockpit employs the concept of IEC-61499. The data extraction, processing and presentation are performed by the FB Network (FBN), which consists of the instances of FB Types. Among FB Types there are adapters capable to extract data from external systems and processors capable to process data. The most important FB Types for this thesis work are DPWS adapter, which allows the system to subscribe to notifications from device level and Esper Complex Event Processing (CEP) which enables asynchronous event processing.

The FBN besides configuration of individual FBs describes connections between function blocks, message including transformations. For message transformation Exten-

sible Stylesheet Language (XSL) Transformation (XSLT) is employed. The XSLT file is a XML of a specific format, which allows transform one XML to another by place values of XML elements and attributes to elements and attributes of another XML.

The configuration of the FB and FBN is defined in XML file (see Appendix 1), containing the business logic of FB instance, description of the routes between FB, which includes required transformations [15, 24]. This XML file is being processed by Function Block Engine Configurator (FBEC) which parses the configuration file and interacts with PLANTCockpit framework utilities such as Route Manager, Function Block Manager, Persistence Manger in order to deploy the scenario, described in file. FBEC also provides GUI for creation and edition of the FBNs.

2.2. SOA at device level

The benefits provided by SOA EDA are being adapted to Factory Automation domain in several researches. A.W. Colombo et al. in [35] and [36], I. Delamer and J.L.M. Lastra in [37] and [38], Lobov et al. in [39] have emphasized on importance of SOA compatible devices to provide full benefits of loosely coupled automation systems.

2.2.1. DPWS and OPC-UA

As SOA has exhibited its benefits being implemented in enterprise systems the endeavours to introduce SOA approach to manufacturing domain appeared. Currently, two most employed implementations of WS-based SOA technologies for devices are OPC-UA and DPWS.

In 2010 Candido et al. are comparing named technologies [40]. The comparison of the features provided in the article is presented in Table 4. The authors note that OPC-UA is assumed to be not fully SOA compliant, partly employing WS protocols, and that OPC-UA is limiting extensibility of the system, restricting user to predefined Web Services. Main advantages of OPC-UA are mature binary encoding UA Binary and metadata model UA Object Model [40].

On the other hand main advantages of DPWS are in flexibility and openness. Due to these properties the lacking metadata can be added to DPWS services employing dissimilar semantic enrichment technologies and a relevant taxonomy which will be described in 2.3.

Table 4: Features comparison of OPC-UA and DPWS [40]

Feature	OPC-UA	DPWS
Infrastructure		
General-purpose transport	HTTP 1.1	HTTP 1.1
General-purpose messaging	SOAP 1.2 WS-Addressing	SOAP 1.2 WS-Addressing
General-purpose encoding	XML	XML
Security	WS-Security	WS-Security

	WS-Trust WS-SecureConversation	WS-Trust WS-SecureConversation
Optimized transport	UA Native UA SecureConversation	None (open)
Optimized encoding	UA Binary	Binary XML
Discovery	WS-Inspection WS-Discovery UDDI	WS-Discovery
Architecture		
Software architecture	Client-server Layered client-server	Peer-to-peer Client-server
Targeted hardware platform	Gateways / SCADA / HMI	Devices
Modelling		
Meta model	UA Object Model	None (open)
Management		
Session management	SecureChannel service set Session service set	None required (WS-SecureConversation may be used)
Resource discovery and selection	View service set Query service set	WS-Enumeration
Resource discovery and management	NodeManagement service set Attribute service set	WS-Enumeration
Eventing	MonitoredItem service set Subscription service set	WS-Eventing
Operation invocation	Method service set	Standard Web Services

2.2.2. State of the art of DPWS related research

In recent decade the number of researches dedicated to implementation of factory wide SOA employing DPWS technology was performed. Most important among them will be described below.

2.2.2.1 ITEA-SIRENA

ITEA-SIRENA can be decrypted as Service Infrastructure for Real-time Embedded Networked Applications. This project took place in 2003-2005 under European cooperative R&D programme ITEA, pioneering introduction of the SOA on device level for industrial, automotive, telecommunication and home automation domains. SOA on the device level is required to provide interoperability for heterogeneous devices with each other and with information systems of the factory thus providing cross-layer and cross-domain interconnectivity [6], [41].

During SIRENA project the framework for implementation of SOA on the device level had been defined. Comparing the technologies capable to introduce SOA on the device level the DPWS technology has been selected and tested [41].

2.2.2.2 WS4D

Web Services for Devices or WS4D initiative is a non-profit project which aims to extend the result of the SIRENA project. Former partners of the SIRENA projects are cooperating within this initiative. WS4D had implemented the toolkits to allow integration of the DPWS implantations with different programming languages, hence ensuring the interoperability of DPWS devices with software following WS standards [8].

WS4D initiative has currently created four toolkits targeting different languages and domains where DPWS can be implemented. These toolkits are WS4D-gSOAP, WS4D-uDPWS, WS4D-Axis2 and WS4D-J2ME. WS4D-gSOAP is based on gSOAP toolkit used to implement SOAP/XML WS with small footprint, thus applicable for resource constrained devices. WS4D-uDPWS aims to provide DPWS functionality to devices with a small amount of memory (few hundreds of kilobytes [42]) such as wireless sensors or other constrained networked embedded systems. WS4D-Axis2 is based Apache Axis2 SOAP processor for WS. WS4D-J2ME or Java Multi-Edition DPWS Stack (JMEDS) is Java based implementation of DPWS [8], [42].

2.2.2.3 SODA and SOA4D

Another project ITEA-SODA, also had been based on the results of SOA introduction to factory devices achieved in ITEA-SIRENA. Aim of SODA project was to develop an ecosystem for high-level SOA-based communication between WS-enabled devices, employing the basis designed by SIRENA. Within the project the low-cost WS-enabled devices had been created and the results of the research had been used in OASIS standardization process for DPWS. The project took place in 2006-2008.

During the project in 2007 the SOA for Devices (SOA4D) Forge had been setup in order to promote an ecosystem for development of SOA-based software components for resource constrained devices [10].

2.2.2.4 SOCRADES

In extension of the results achieved in the field of introduction of SOA to manufacturing (mainly in SIRENA project) SOCRADES project took place in 2006-2010. Name SOCRADES stands for Service Oriented Cross-layer Infrastructure for Distributed smart Embedded devices [43]. This project aims to develop a platform for design, management and execution for industrial automation systems based on smart devices, employing SOA paradigm on device and application layers [9].

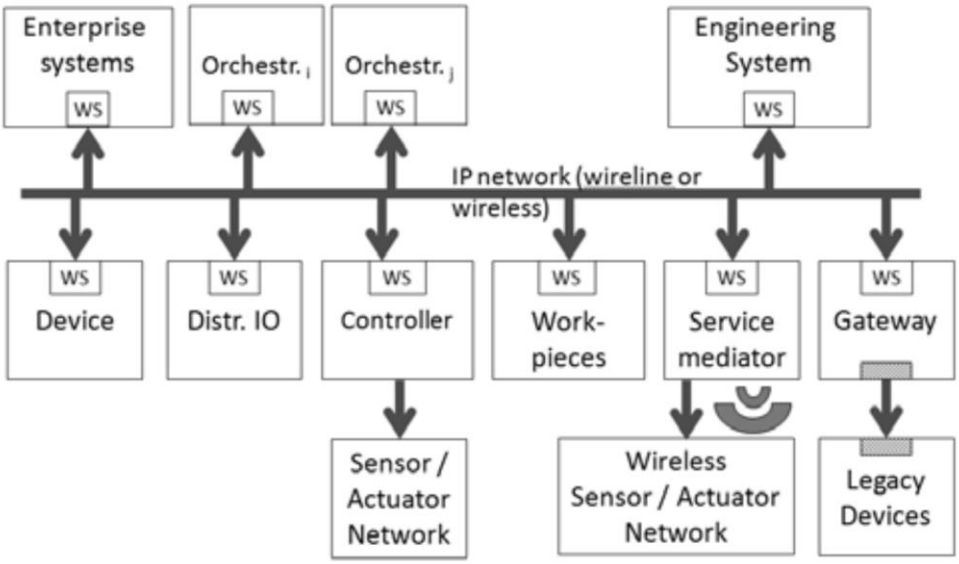

Figure 5: General architecture of SOCRADES framework [43]

Implementation of SOA paradigm on device and application layers would allow cross-layer loosely coupled seamless integration of the devices and information systems through the enterprise. For SOA framework WS technology and correspondently DPWS on device level technologies had been employed. The framework architecture utilized in the project can be found on Figure 5. The reference implementation of the concept developed in the project was presented in [44]. This implementation employed the project framework to integrate of smart devices with an ERP system, hence proving the capabilities of developed infrastructure for seamless integration from enterprise to device level applications.

The reference implementation for cross component communication employed the important concept of Enterprise Service Bus, already mentioned in 2.1.2.2.

2.3. Semantics

Semantics is usually defined as a study of meaning. Employment of semantics in enterprise information systems allows delegate part of the tasks which were performed by personnel to some reasoning algorithms. This approach is especially beneficial if big amount of data should be processed. The EIS seamlessly connected to shop floor manufacturing devices providing multiple benefits described before contain numerous data sources hence require semantic to keep it usable.

The machine readable Knowledge Base (KB) as a storage for the semantic meaning s which can be processed by computer provides a concept for semantically rich environment. In conjunction with defined rules KB can automate some decisions or at least to provide decision support. Ontologies allow to describe the KB structure and to store the knowledge. Zhonghua Yang et al. in [45] are defining two core technologies for semantic rich environment: ontologies and standard language for ontology development.

2.3.1. Ontology

The term ontology is originating from philosophy. In context of this thesis ontology refers the definition provided by T. R. Gruber in [46]:

"An ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an Ontology is a systematic account of Existence. For AI systems, what "exists" is that which can be represented."

Some other definitions that describes ontology from different perspectives are summarized by A. Gómez-Pérez et al. in [47]. Finally, detailed encyclopaedic definition of ontology in correspondence to computer science is provided by Gruber in [48].

The classification of ontologies can be performed based on subject of conceptualization as defined by Gómez-Pérez et al. in [47]. On the other hand considering semantics i.e. level of formalization ontologies S. Borgo et al. in [49] specifies ontological systems with weak and rich semantics. The first ones are presented by thesauri, taxonomies and other terminological ontologies. These ontological systems are beneficial in classification and categorization and not provide strong reasoning support. The ontologies with reach semantics are implementing rich logical relations between the described classes, hence providing extensive base for reasoning [49].

Application of ontologies in manufacturing domain is being studied in multiple researches for description of related entities as in [50] and [51] and extended further for knowledge driven manufacturing frameworks [52], [53]<f. F. de B. Ramis Ferrer in [54] has developed interoperable Manufacturing System and Configuration Function Block Network Ontologies (MSO and CFBNO), concept of which could be employed as a basement for EIS semantics, once it will be enhanced with appropriate manufacturing taxonomy.

2.3.2. Manufacturing taxonomies

Currently, several solutions for manufacturing industry integration are employed. These solutions are based on some classification. This classification can be used as a standard language for integration of the heterogeneous systems from shop floor to ERP systems.

2.3.2.1 CAEX or IEC 62424

CAEX or Computer Aided Engineering Exchange defines an application neutral object-oriented data format for hierarchical object description. This standard is used for static features of the manufacturing systems such as plant topology. The product-process-resource is employed in CAEX model (see Figure 6) [55].

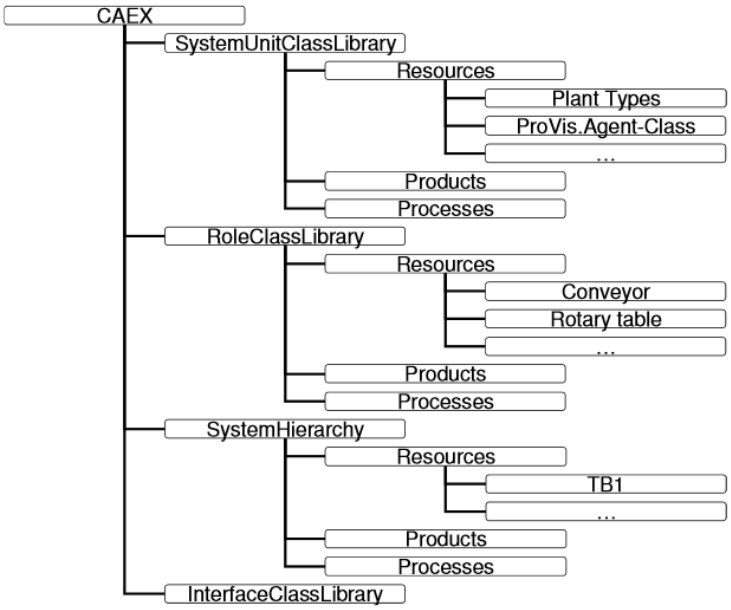

Figure 6: CAEX product-process-resource model [55]

CAEX has to be extended to represent dynamic data from device level.Most often it is used with OPC data models for dynamic data components [56].

2.3.2.2 AutomationML

AutomationML is an open, XML based standard for data format which aims to improve interoperability for heterogeneous engineering tools [57]. AutomationML employs CAEX to describe plant topology, COLLADA to describe geometry and kinematics of the objects and PLCOpen XML for behavior description [58]. In [59] Faltinski et al. are stating the benefits of AutomationML for planning, testing and integration of dissimilar engineering tools. Particularly important benefit for EIS is possibility to extract and neutralized the engineering data from the automation tools.

2.3.2.3 ISO 15926

ISO 15926 is a standard dedicated for oil and gas industry to model and exchange the plant data. This standard has been compared with CAEX by T. Holm et al. in [60]. Originally ISO 15926 had been based on ISO 10303 or STEP (STandard for the Exchange of Product modelling data) and was oriented towards databases. Later this standard had been reformulated in OWL, but still is defined by W. Marquardt in [61] as incoherent and with limited reusability. Moreover S. Berger in [62] states that this standard did not solved interoperability problem.

2.3.2.4 ANSI/ISA-95, IEC 62264, B2MML

As it was mentioned in 2.1.1 ANSI/ISA-95 aims to define integration interfaces for the enterprise and control systems for factories and is also known as IEC/ISO 62264. In the first part of the standard the models and terminology employed in enterprise information

systems are described. This standard offers the taxonomy for all levels in the enterprise thus is most suitable for enterprise wide integration [63].

Employment of ISA-95 for manufacturing intelligence for lean manufacturing and its benefits has been described on example of a manufacturing operation centre by H.O. Unver in 2012 in his paper [64].

B2MML or Business to Manufacturing Markup Language implements all of ISA-95 data models as a set of XML Schemata, hence making implementation for XML based communications easier. Also as XML is a widely accepted standard in EI, B2MML is compliant with modern integration solutions. In [65] D.I. Nastasie et al. define B2MML as the single existing international standard which covers all three main management domains of the enterprise – engineering, business and information management layers.

2.3.3. Semantic enrichment of Web Services

Semantic enrichment is attachment of the metadata to some content [66]. In case of WS, the metadata should be coupled with service description modules. More precisely, in context of the current project, the metadata should be provided to the descriptions of the data in messages.

Currently semantic enrichment is based on semantic annotation of the content. For some domains exist algorithms which provide automatic annotations of the content (such as text), but even best of those do not provide high level of precision in big enough knowledge domains [67]. Hence in scope of this thesis work only manual semantic enrichment is studied.

In [68] J. Cardoso et al. define three main approaches to bring semantics to Web Services: SAWSDL (Semantic Annotations for WSDL), OWL-S (Web Ontology Language for Services) and WSMO (WS Modelling Ontology). Only first from the list provides direct encapsulation of semantics within WSDL, hence providing bottom-up semantic description and decentralizing efforts to bring semantics to WS [68]. SAWSDL is based on WSDL-S standard [69]. SAWSDL had been developed for semantic enrichment of WSDL 2.0, but it was also adapter for WSDL 1.1 as the last one is wider used [53].

3. METHODOLOGY

The main technologies and approaches currently available for the implementation of non-intrusive dynamic system recognition for monitoring purposes are presented in previous section. Based on the technology overview, in this section the technologies, tools and approaches to be employed in this thesis work will be selected and described.

3.1. Tasks and criteria

The process of development of the solution to the problem defined in this thesis can be separated in set of subtasks. Firstly, the shop floor devices should be bridged to EIS, the technology that allows dynamic connection and interoperability of the devices is requires. For non-intrusiveness of the system, the business logic independent gateway devices should be used. Secondly, the shop floor devices should provide the semantic meaning of the data being exchanged by the devices. These semantic meanings have to be standardized, to allow automation of dynamic and automated configuration of the system. The next step is to define a technology to discover the shop floor devices and react on the changes in shop floor. Further the semantic meanings of the discovered devices should be extracted from description. At fifth step, the extracted metadata should be mapped to the KB capable to present explicit descriptions of the data available in the system, including all the data required to integrate the data sources and calculate generic predefined metrics. Lastly, the technologies and algorithms of the analysis of the knowledge base should be defined. All these steps are summarized in Figure 7 and described with criteria in more details further in this section.

Figure 7: Overview of project tasks

*Subtask 1: Shop floor connectivity***.** The shop floor factory devices should be interoperable with EIS. This task includes definition of the technology for shop floor to EIS interoperability and selection of the device that implements the technology. To increase the benefits provided by the solution should be non-intrusive as it is noted in problem definition. The criteria are assembled in Table 5.

Table 5: Criteria for selection of shop floor device connectivity technology and device

Select	Criteria
Technology for introduction of the device	▪ Connectivity with EIS standard

constrained data to EIS	technologies
Device implementing technology	▪ Non-intrusiveness ▪ Industrial applicability

Subtask 2: Semantic enrichment. This task requires definition of technology to attach semantic values to the elements of the system and to define the taxonomy of the semantic values. The attachment of the semantic description should be encapsulated in the devices to allow true re-configurability of the shop floor. The taxonomy in its turn, should be applicable on all the levels of the EIS, to unambiguous define the meaning of the data and to loosely reference this data in the metrics formulae. Criteria for selection of semantic enrichment technology and taxonomy are summed up in Table 6.

Table 6: Criteria for device description semantic enrichment technology and taxonomy selection

Select	Criteria
Technology to introduce semantic meaning to the device description	▪ Connectivity with technology from subtask 1 ▪ Capability to be encapsulated in device
The taxonomy to for semantics	▪ Applicability on shop floor and EIS levels ▪ Industrial applicability

Subtask 3: Shop floor discovery. Development of a tool which will employ connectivity of the shop floor devices requires selection of appropriate technologies, approaches and algorithms. This tool (hence technology as well) should enable discovery triggered by the changes in shop floor and also retrieval of the description of the data sources available in devices, which includes all the integration related information. Subtask criteria are summed up in Table 7.

Table 7: Criteria for technology selection for shop floor device discovery

Select / Develop	Criteria
Select technology to discover devices and develop a tool	▪ Dynamic discovery ▪ Capability to retrieve device description

Subtask 4: Semantics extraction. In this subtask the technology for parsing of the semantically rich device description should be defined. Criteria for this selection are presented in Table 8.

Table 8: Criteria for semantic extraction technology selection

Select / Develop	Criteria
Select technology to extract semantics from shop floor device description	▪ Capability to extract device description defined in subtask 2

Subtask 5: Mapping to KB. The subtask includes definition and realization of the technologies, approaches and algorithms, which will map the semantic annotation from devices to the knowledge base. Criteria for decisions in subtask 5 can be found in Table 9.

Table 9: Criteria for selection of technologies and algorithms to map the device descriptions to KB

Select / Develop	Criteria
Select language to present KB	▪ Capable to express relations between the manufacturing entities
Select technology to edit KB	▪ Capable to parse and edit the KB of defined format
Select or develop KB to model shop floor	▪ Capable to handle all data required to map the shop floor ▪ Providing enough relations to efficient querying
Develop a tool to map data from device to KB	▪ Capable to parse device description and retrieve semantic meaning ▪ Capable to read and edit the ontology, based on semantic data ▪ Capable to analyse edited ontology

Subtask 6: Analysis of KB. Develop the algorithm to analyse the updated knowledge base to generate possible component network configurations which can be used to calculate predefined metrics. Selection of technology for KB analysis and development of algorithm of it should satisfy the criteria presented in Table 10.

Table 10: Criteria for technology and algorithm selection to analyse the KB

Select / Develop	Criteria
Select technology and define algorithm for KB analysis	▪ Capability to provide support on data integration

3.1.1. Shop floor connectivity

The concept of SOA on device level has been chosen to be employed in this thesis, due to benefits provided by the approach in manufacturing, especially in EI. As it was mentioned in section 2.2, main implementations which are introducing SOA on shop floor level are DPWS and OPC-UA. OPC-UA provides comprehensive set of technologies and standards for factory SOA, but requires efforts of experienced personnel to be implemented due to pervasive specification. DPWS, on the other hand, provides higher flexibility and openness. Hence DPWS has been employed to implement SOA in current work.

Among currently available, applicable for industrial purposes DPWS-enabled devices Inico S1000 RTU of Inico Technologies is, allegedly, most suitable for purposes of

this thesis. The Inico S1000 RTU is compatible with industrial standards and can be employed as a controller. The RTU is capable on one hand to handle SOAP Web Services, and on another to be connected via digital and analogue inputs and outputs to manufacturing system, hence bridging the manufacturing devices with information systems of enterprise, such as monitoring systems.

The DPWS, implementing WS specification, allows devices to host the services and interact with clients. The monitoring system should only receive the messages from the shop floor devices. Hence, the notification type of operations is expected to be implemented on shop floor side. To provide required information for monitoring system these notifications should contain descriptions of current status of the shop floor equipment. Following the EDA concepts each time that the status of device changes the notification should be sent to monitoring system.

The DPWS-enabled devices, such as Inico S1000, on one hand can be connected by its inputs the shop floor devices and process the date received from these connections. The devices can generate the notifications containing the information about shop floor status generated as a result of processing the inputs. On the other hand, as it was noted before DPWS is WS compatible and implements WS-Eventing standard allowing service requester to subscribe and receive notifications from the service provider. Thus DPWS compliant devices allow to discover and subscribe to their notifications to DPWS-based applications. Hence, if monitoring system possesses a DPWS adapter it can be loosely integrated with the shop floor. Moreover, such system get all benefits of EDA as publish-subscribe pattern will be employed.

3.1.2. Semantic enrichment

As description of the service in DPWS-enabled device is realized employing a WSDL file, and standard WSDL do not contain any functional or data description of operations and messages, extension for WSDL is required. This extension should attach a semantic meaning to service description.

SAWSDL format is most employed and mature standard to enrich WSDL with semantic meanings. SAWSDL provides a “model reference” attribute for each element of the WSDL described in section 2.1.2.1. The model reference attribute can define the semantic meaning of the element. Being attached to the primitive data types, this semantic value can describe the meaning of the data. As factory metrics are based on certain formulae, including these data the mapping of the service in the enterprise ontology can be performed in order to define monitoring system configuration from device level up to calculate the metric. The SAWSDL is schematically represented on Figure 8.

Need for common definitive language to describe the services on the shop floor is already discussed in section 2.3. Hence, certain manufacturing taxonomy has to be selected in order to provide this common language in EIS. B2MML based descriptions are employed in this thesis due to its high descriptive capability and acceptance on shop floor and business (to define metrics and KPIs) levels.

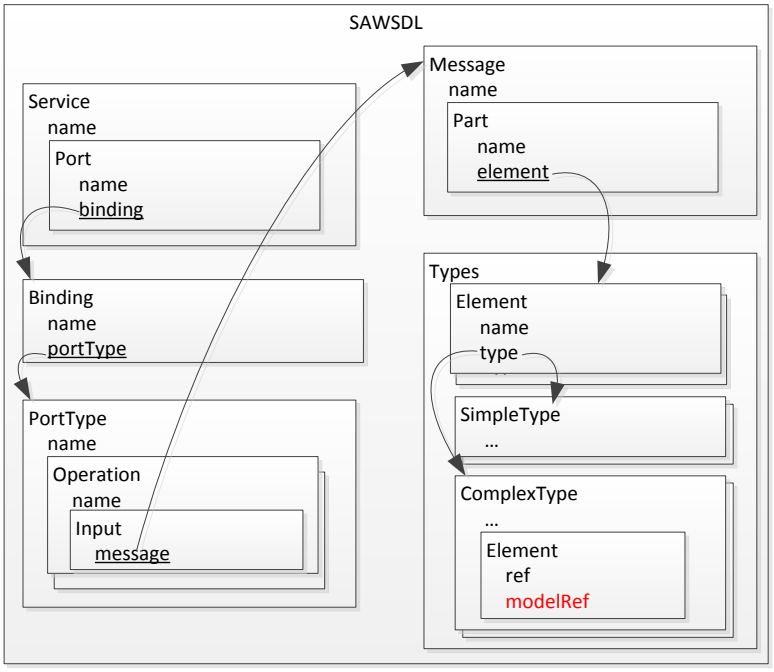

Figure 8: Schematic presentation of SAWSDL

3.1.3. Shop floor discovery

For dynamicity and flexibility of the developed solution, it should be synchronized with shop floor level. For this synchronization the changes in the shop floor layout should be mapped to the model of the factory. Employment of DPWS introduces possibility to discover DPWS-enabled devices as DPWS standard strongly requires WS-Discovery compatibility [71]. WS-Discovery also offers an approach to dynamically track the changes in connected devices using Hello and Bye messages once device is connected and disconnected respectively. This allows to make a system developed employing DPWS to by synchronized with shop floor devices.

The algorithms of multicast discovery, Hello and Bye message listeners are implemented in the WS4D (see section 2.2.2.2). Java programming language is used in this thesis, hence JMEDS as realization of WS4D for Java can be employed to discover DPWS-enabled devices and get descriptions of the related services.

3.1.4. Semantics Extraction

As it was defined in previous sections SAWSDL had been chosen in this thesis to provide metadata into device description. The metadata elicitation from the SAWSDL can be performed employing various WSDL parsers. These parsers are generally speaking XML parsers adapted for WSDL schema for XML.

Unfortunately, as SAWSDL had been initially developed to extend WSDL 2.0 and in this thesis WSDL 1.0 is employed as more accepted in industry and due to current device restrictions. SAWSDL has an adaptation for earlier version of WSDL, but not all SAWSDL parser extensions can be used in this work. The most robust solution found for semantic annotated complex data types of WSDL 1.0 is to use EasyWSDL parser

with extension for complex data types with specific algorithms. More details on SAWSDL parsing used in this work can be found in Implementation section.

3.1.5. Mapping to Knowledge Base

Next step required for automatic system recognition is to map the semantics extracted from device description to the KB. This KB should contain all information about the manufacturing system, providing single source for data integration information. The KB should contain description of the metrics to be calculated as well as required data to configure data extraction and processing if it is possible to calculate a metric in the system.

As it was mentioned in section 2.3 ontologies are among most developed realization of the machine-readable knowledge bases. Among the ontology languages in its turn most employed currently are OWL and its dialects. Currently, there exist several OWL ontology editors such as Protégé which makes the human processing of the ontologies easier, allowing to introduce the new metric definitions manually if required.

The ontology should have a model, which will define the classes, properties and relations between them employed to represent the knowledge and enable its analysis later. The model of ontology which can handle integration information has been developed by Borja Ramis Ferrer in [54], and will be employed in this thesis with some modifications. The ontology is named Configuration Function Block Network Ontology. Original model of CFBNO, offered by Borja Ramis Ferrer can be observed on Figure 9. CNFBO ontology defines main classes, their relations and properties of dissimilar entities of data integration. For more details about modified CFBNO see section 4. This ontology allows to define the business and manufacturing metrics and KPIs as the relation of the data sources.

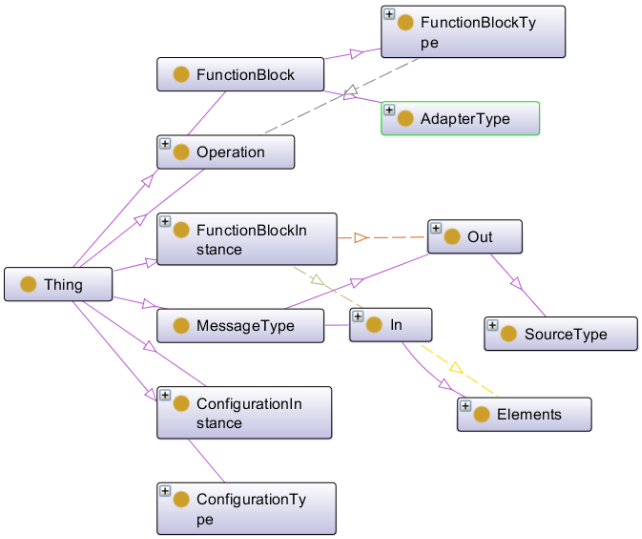

Figure 9: Original CFBNO model offered in [54]

For the purposes of reading and editing of the ontology, Apache Jena has been used. Jena provides Java API to parse OWL files and add elements to ontologies. In more

details the algorithms of population of the ontologies will be described in Integration section.

3.1.6. Analyse the Knowledge Base

The ontology employed in this thesis contains not only device descriptions, but also the definitions of the metrics. These definitions are the representation of formulae in which the values are defined by semantic meanings. The semantic meaning of the data sources should follow same taxonomy as one used for semantic definition of the data produced by external systems and devices, such as shop floor DPWS-enabled devices. This standard taxonomy allows loose assembling of the data available from shop floor devices to more abstract metrics.

The metrics could require the information from all levels of automation, hence the taxonomy should be applicable to whole automation system. Moreover, metric itself should have a semantic meaning as it may be required to calculate more complex ones.

The goal of analysis of KB is to define which metrics can be calculated and to provide to monitoring system information required for data integration related to metric calculation. The algorithms of data analysis can be implemented employing Apache Jena. These algorithms should find all metrics defined in CFBNO, check if all data sources required for the metric calculation are available and finally, on user request to generate data integration configuration for monitoring system to calculate the metrics which have all required data sources available. The last part should be decoupled from the approach to make the solution applicable for multiple monitoring systems.

3.2. Use case

The implementation of the system, which employs semantic models for dynamic system recognition in industrial monitoring systems, should be able to monitor the changes on device level of factory shop floor and propose information required to reconfiguration of the monitoring system according to new status of the system. To prove the functionality of the approach proposed in this thesis the tool implementing the concept will be developed and tested.

For testing of the tool capabilities it will be employed for data integration for monitoring of the industrial system. The implementation scenario is based on the Festo Modular Production System MPS® 500-FMS test bed, PlantCockpit EI tools, Inico S1000 DPWS-enabled RTUs and the tool developed in this thesis.

3.2.1. MPS® 500-FMS

MPS® 500-FMS is an educational modular manufacturing system. This system is contains the independent and interoperable modules, which are controlled by SIMATIC Siemens PLCs.

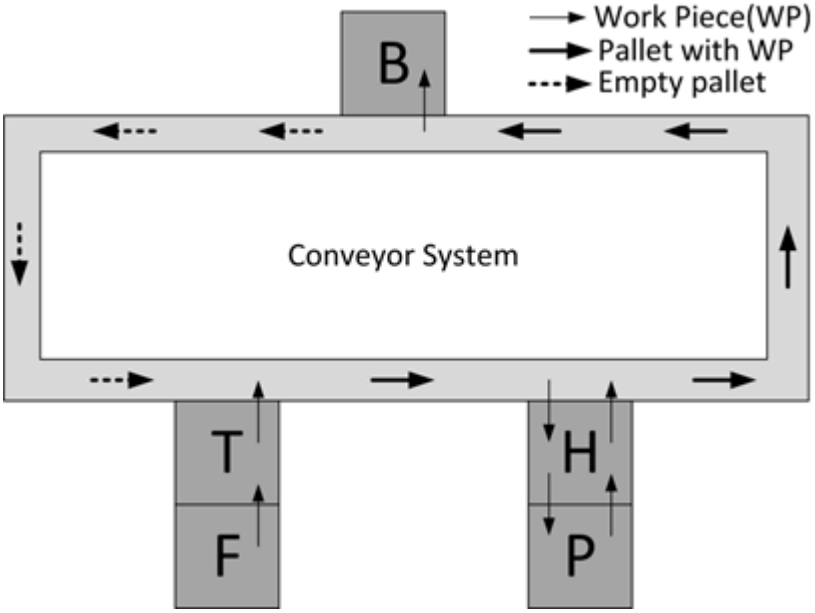

Figure 10: Festo FMS-500 layout and flow of work pieces and pallets in system

In current configuration there are 5 manufacturing modules and a conveyor system. The layout of the system can be observed on Figure 10. The manufacturing modules are: feeding (F), testing (T), handling (H), processing (P) and buffering (B on Figure 10) stations. Feeding and testing stations as well as handling and processing stations are connected in pairs. Translation of the work piece between the stations of the named pairs is performed immediately, while all other work piece transitions are executed by the conveyor pallets.

The work piece being processed in the system is a plastic cylinder. This cylinder is being introduced in the system by feeder station from magazine. The work piece it transmitted to testing station, where parameters of cylinder are tested and cylinders with wrong parameters are rejected. Not rejected cylinders are transferred on conveyor pallet, once one is available. The pallet with work piece is delivered by conveyor to handling station. Handling station delivers the cylinder to processing station. In the processing station some manufacturing actions are performed on the cylinder, and once cylinder is ready it is returned to handling station. The handling station, once it had received a complete work piece places it on conveyor pallet. The pallet with finished cylinder is delivered by controller to buffer station, where it is placed in storage.

3.2.2. Gateway device

Based on the status of the workstation equipment and history of its changes it is possible to establish the status of the workstation. It was mentioned each workstation is con-

trolled by a controller. Based on status and history of changes of inputs and outputs of controller it is possible to define the current status of the workstation. It was stated in 0 that to provide dynamic shop floor device connectivity to monitoring system DPWS will be used in this thesis.

As SIMATIC Siemens S300 controllers, employed in Festo MPS® 500-FMS, do not provide DPWS, some other devices should be employed as a gateway. Moreover, it would be beneficial to reduce amount of messages in network, as SOAP-messages employed in DPWS can provide sufficient load. Hence the gateway device should process status of controller inputs and outputs and provide messages describing workstations and work pieces status changes. Inico S1000 RTU can be employed as such device, as it provide required computing power to define workstation status and also are DPWS-enabled.

3.2.3. PLANTCockpit

The PlantCockpit project is already partly described in section 2.1.3. The project provides the tools for enterprise integration. Among other utilities PlantCockpit provides set of adapters and processors, which can be used to create the system for data processing required for monitoring system. As well PlantCockpit has a Visualization Engine, which can listen to user defined endpoints in configured data processing system and represent values received from the endpoints in a web browser. These tools can be employed for monitoring tool in the thesis.

Important benefit provided by PlantCockpit tools is configurability. Integration scenario in PlantCockpit is defined by XML files, which can be parsed and deployed by FBEC.

Generation of these configuration files based on KB analysis can provide dynamical configurability of the system, triggered by changes on the shop floor. To generate the configuration file the business logic of adapters and processers required in the FBN, message flow between them and required message transformations to adjust message types should be defined. All information required for creation of the configuration file for metric calculation should be received from shop floor devices and the KB describing the metric.

As well it should be mentioned that configuration XML files contain sufficient amount of information and thus are complex. Even for a specialist, who knows both production process and data integration, manual creation of FBN configuration is a complex task as size of configuration XML file can exceed 200 KB (i.e. could include more than 200,000 characters). Creation of FBN configuration employing UI of a FBEC, is easier and not require to interact with thousands of lines, but still requires interaction of user with repetitive low level tasks. Hence employment of the automated configuration based on KB and semantic descriptions of the shop floor devices can simplify this task.

3.3. Summary

At the first stage the tool should be developed. The tool development includes definition and implementation of algorithms capable to discover semantically rich descriptions on shop floor, algorithms to extract the semantic values from descriptions, algorithms to populate KB accordingly to the service descriptions of shop floor devices and to analyse the updated KB in search for metrics which can be calculated.

In second phase the tool should be employed in industrial scenario to prove its functionality. For this purpose the selected manufacturing system should be extended with DPWS-enabled devices which will provide the web services. The WS should be triggered by the manufacturing line equipment and generate the messages, which will describe the change of the system status. Semantic definition of the messages should be also provided in devices employing the technologies and models specified in methodology. In the end, transformation of the information obtained by tool to the format readable by the integration tools has to be developed.

Summing up methodology section Table 11 Table 4can be formed.

Table 11: Methodology

<table>
<tr><td>Manufacturing line</td><td>Festo MPS® 500-FMS</td><td rowspan="10">II. Industrial System</td></tr>
<tr><td rowspan="2">Shop Floor Connectivity</td><td>Inico S1000 RTU</td></tr>
<tr><td>DPWS</td></tr>
<tr><td rowspan="2">Semantic Enrichment</td><td>SAWSDL</td></tr>
<tr><td>B2MML</td></tr>
<tr><td>Shop Floor Discovery</td><td>JMEDS</td><td rowspan="6">I. Tool</td></tr>
<tr><td rowspan="3">Mapping to KB</td><td>EasyWSDL</td></tr>
<tr><td>OWL</td></tr>
<tr><td>CFNBO</td></tr>
<tr><td rowspan="2">Analysis of KB</td><td rowspan="2">Apache Jena</td></tr>
<tr><td>Integration</td><td>PlantCockpit</td></tr>
</table>

4. IMPLEMENTATION

This section will describe implementation of the tool which enables the dynamic system recognition for industrial monitoring systems and realization of the implementation scenario, which aims to test the tool. The development of the tool had been performed employing Java programming language in Eclipse IDE.

4.1. Implementation of the tool

The developed tool should perform four main actions in response to user requests. These actions are to discover devices in the network and listen to changes on shop floor, to populate ontology with data received on discovery, search for metrics which have all required data sources and finally to create the FBN for monitoring system configuration. These actions are summed up on Figure 11.

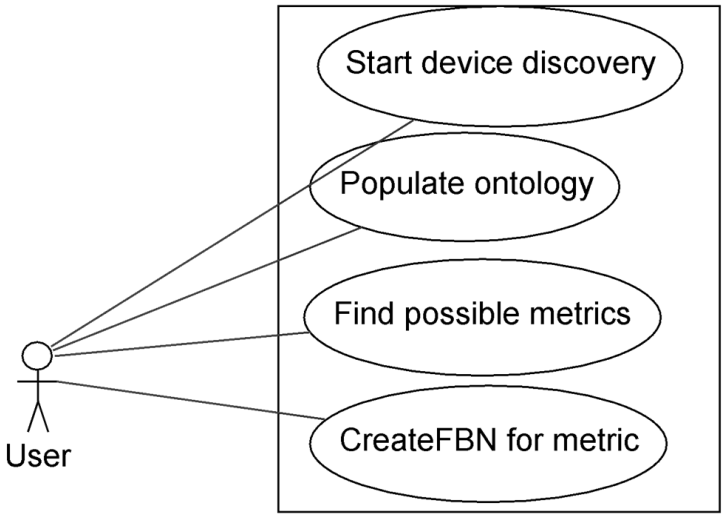

Figure 11: Use case diagram for the developed tool

In more details the interaction of user and modules of the tool are presented on sequential diagram (see Figure 12). On this diagram it can be observed that once user selects start discovery action UI triggers discovery activity on corresponding module. When the services are found their WSDL files retrieved by discovery module and user can start populating the ontology. First action triggered by UI in response to user request is parsing of the WSDLs performed by Semantics Extraction Module. When the parsing of the service descriptions is finished, extracted data is being placed in ontology on “populateOntology” action of Mapping Module.

Once population of ontology is finished user can initialize the search for metrics which can be calculated on current system configuration. This action is performed by Analysis Module, and when the search is finished the module returns the list of names of metrics which can be calculated.

As a next step the user can select the metric by name to create FBN XML. This action, firstly, requires extraction of the data integration information from ontology. Once all required information to create XML is extracted the next action of creation of the XML file can be started.

In more details all the behaviour of the modules is presented in following sections.

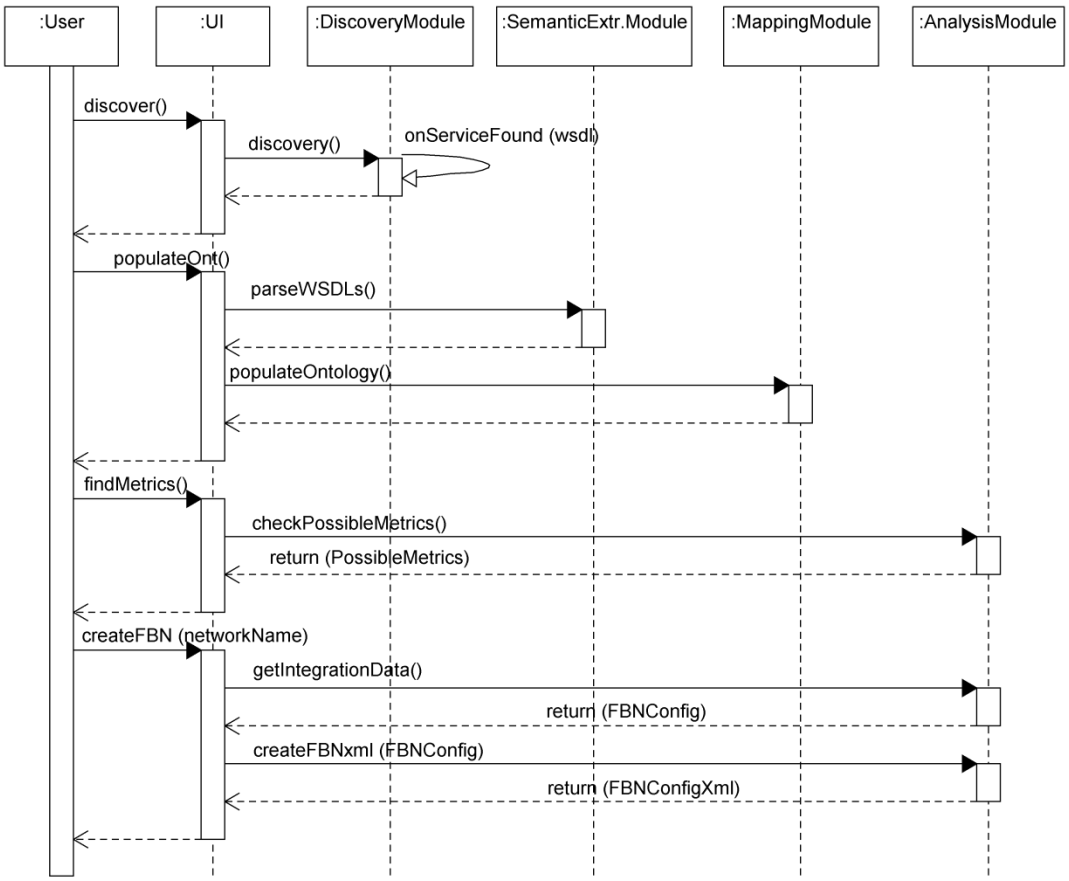

Figure 12: Generalized sequential diagram of the tool use

4.1.1. UI overview

In order to trigger execution of operation of the modules the GUI presented on Figure 13 have been developed. “Discover DPWS Devices” button triggers DPWS discovery and starts hello/bye message listeners. This process includes operations of discovery module. Once the process of DPWS Discovery is finished user can put data about services found in the network to CFBNO using “Populate CNFB Ontology”. This action employs functionality of semantic extraction and mapping modules. “Find Possible Metrics” button triggers analysis of KB, and populates drop down list on the right side of interface with names of possible metrics. On this activity KB analysis module is used

If it is possible to select a metric “Create FB Network” button can be pressed and XML configuration file will be generated. The generation of XML is defined by format of the monitoring system. In this thesis PLANTCockpit enterprise integration framework had been employed, hence the format of XML follows framework requirements. Generation of this XML was included in KB analysis module of the program.

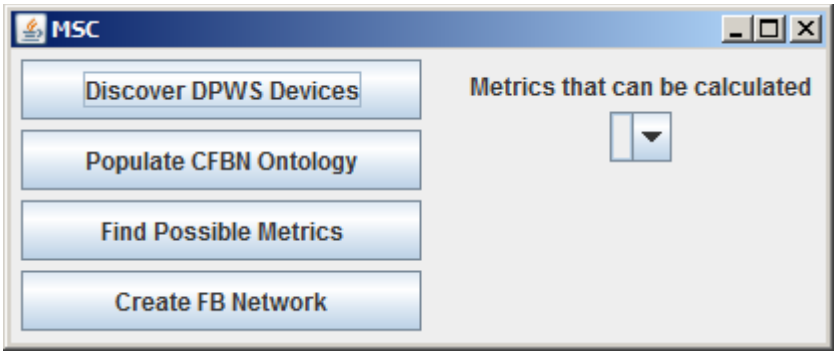

Figure 13: GUI of developed tool

4.1.2. Discovery Module

The aim of the discovery module of the program is to enable synchronization of the knowledge base describing shop floor with the shop floor itself. The functionality of DPWS is depicted on Figure 14. Once the tool is started the information about shop floor devices should be placed in knowledge base. On the other hand once some changes are performed in configuration of the shop floor it should initiate required changes in the knowledge base to keep the system synchronized with real world.

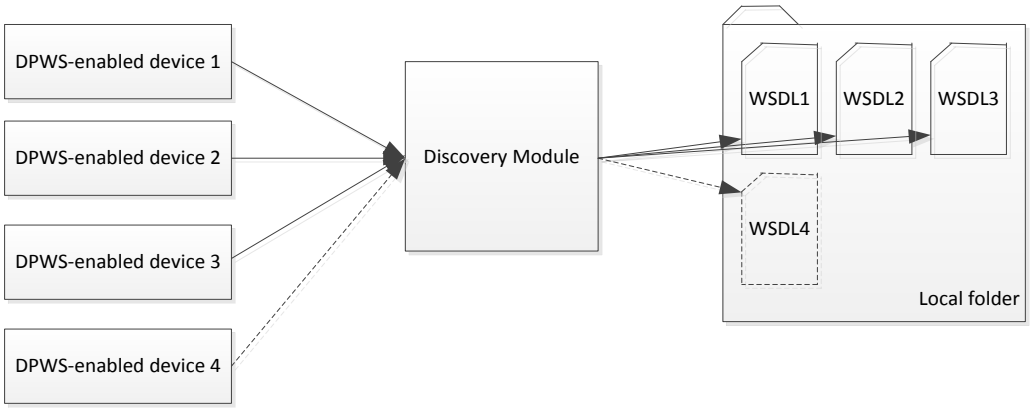

Figure 14: Concept of discovery module

The information about the devices is placed in WSDL files hosted on devices. The set of WSDL files contains all the information required to create a model of the factory shop floor. Hence the discovery module should update the set of the WSDL files according to devices available in the network. Each WSDL file itself will be processed by other module of the program. In order to avoid the problems caused by networking (such time delays, access rights limitations, communication failures and others) the WSDL files should be retrieved from devices and stored locally on the hard disk of the user’s computer. Hence, the aim of the discovery module is to keep local storage of WSDL files synchronized with the WSDL files available in the network.

The discovery module is implemented in this thesis as DPWS Client employing JMEDS stack. The DPWS Discovery Client should trigger discovery once it is connected to the network of shop floor devices, to get initial configuration of the shop floor. This discovery is client-triggered. On the other hand once the DPWS Client is already connected to the network it should be able to notice the changes in the shop floor, thus the device-triggered discovery should be implemented as well in DPWS Client.

4.1.2.1 Client-triggered Discovery

Client-triggered discovery is performed in two phases. At first phase the DPWS-enabled devices are discovered on second the services hosted on them are found. The services are related to WSDL files and define the path to them. Hence, discovering services allows to download the WSDL files from the shop floor device.

DPWS client is able to multicast the discovery probe over UDP to all devices available in the network. This probe can have the filtering by several parameters such as Universally Unique Identifier (UUID), which is part of WS standard. On the first stage no filtering is employed, so each DPWS-enabled device will respond on such probe. This is the first phase.

On the second phase immediately after the new device is found the DPWS Client starts to search for services hosted on the device. Once the service is found it sends a response. The response will contain the location of WSDL file which describes services. Employing standard Java functionality the WSDL file can be downloaded and saved locally in predefined location.

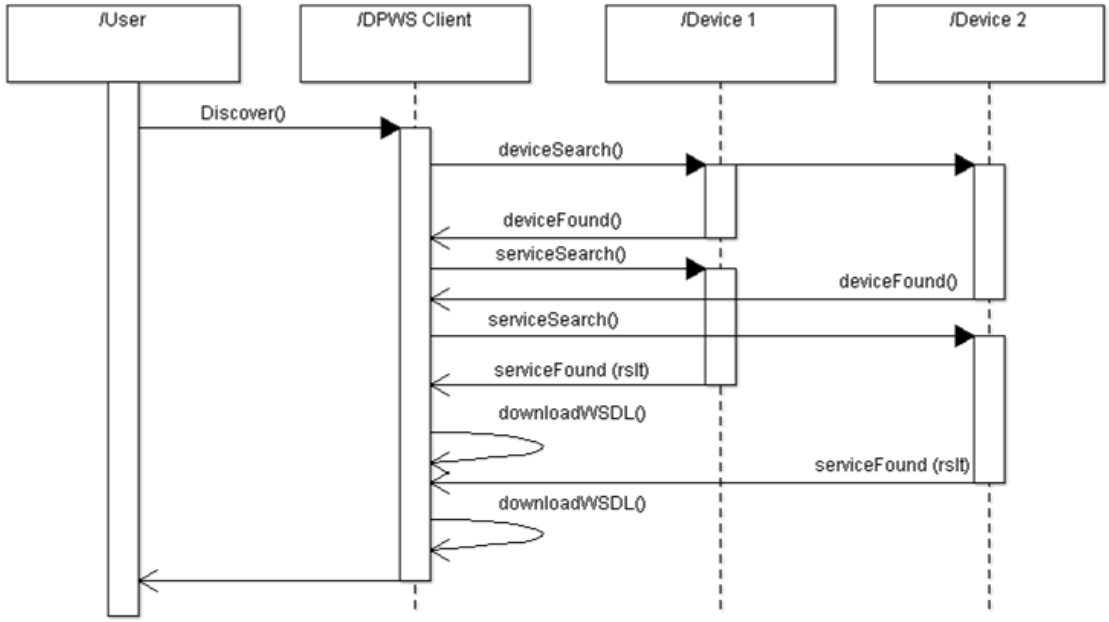

Figure 15: Sequential diagram of DPWS discovery process triggered by DPWS client

First and second phases may overlap as the communication between the DPWS entities is asynchronous. The sequential diagrams for both phases can be found onFigure 15. The “deviceSearch()” message on the Figure 15 corresponds to device search probe which is sent to all devices in the network. It may take different time for devices in the

network to respond to this message so as it is presented on the figure the search for services can go same time as search for devices. Search for services is triggered by the "deviceFound()" response from devices, and is presented in figure as "serviceSearch()". The respond for this message "serviceFound(rslt)" contains the service description. Once this description is parsed the location of WSDL file can be estimated, allowing to download WSDL for local repository ("downloadWSDL()" on Figure 15).

The name of the WSDL file saved locally should be unique and thus formed by concatenation of device IP address and the WSDL file location on device, replacing "/" characters of file path by "-", to satisfy the file naming conventions of all main operating systems (e.g. "192.168.3.1-dpws-ws01").

4.1.2.2 Device-triggered Discovery

Device-triggered discovery is employed to handle two possible cases: new shop floor device is introduced in the network, after client-triggered discovery is finished, and device is removed from the network after it had been discovered.

In first case the added DWPS-enabled device multicasts a Hello message in the network. Once the DPWS Client receives such message it can start same discovery algorithm as in 4.1.2.1. Once the client finds already discovered device it will not try to get WSDL from it, but if a new device is discovered the WSDL will be retrieved from the device.

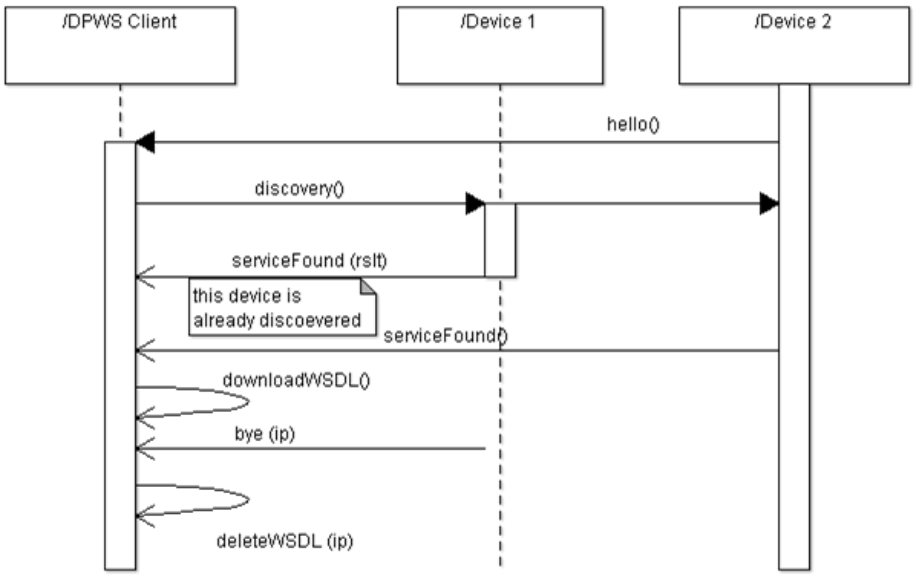

Figure 16: Sequential diagram of DPWS discovery process triggered by device

In second case the on device disconnection the Bye message will be distributed in network. This message contains information about the device being disconnected, including its IP address. As the IP address is a part of name employed to store the WSDL file locally, all service descriptions related to device can be removed from local folder. Hence, the device-triggered discovery will update the list of available WSDL files to match available shop-floor devices.

The sequential diagrams of the both cases of device-triggered discovery activities are presented on Figure 16. As it can be observed, in response to Hello message from DPWS-enabled device, the same process of discovery as defined in previous section takes place. Only difference is that the WSDLs are not overwritten if they are found for second time. In response to Bye message the DPWS client triggers “deleteWSDL(ip)” action which removes the WSLDs related to the IP of disappeared device from local WSDL repository.

4.1.3. Semantic Extraction Module

The running discovery module copies the WSDL files, making them accessible for other program modules. The semantic extraction module aims to parse these WSDL files to retrieve the information required for further data integration and computer support of related configuration and save it as a Java objects which describe the shop floor. As it was mentioned before, in section 3.1.4, the description of the services should contain the semantic meaning of the data provided by the services, in order to enable configuration support. Attachment of metadata should be made employing SAWSDL according to methodology. Hence, extraction of data integration information (parsing WSDL) and semantic meanings (parsing Semantic Annotations) is the aim of this module of a program. The extracted data should be stored in an object model, which later can be easily transformed to ontology form. The concept of the module is depicted on Figure 17.

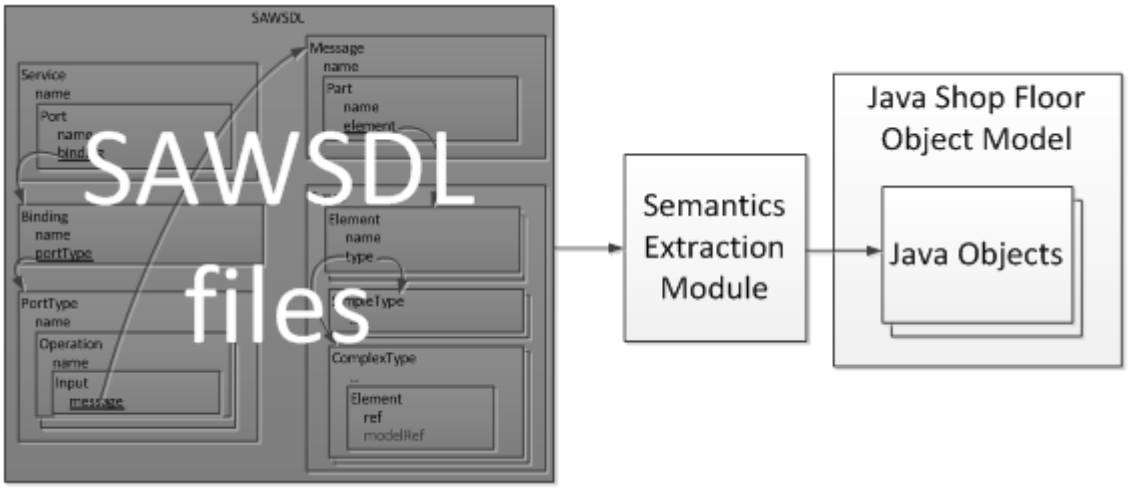

Figure 17: Concept for semantics extraction module

Considering the ontology model selected for description of the system, the information which has to be presented in the object model should include:

- type of adapter function block, required to obtain the information (for future use in once the tool will support other WS implementations but DPWS);
- configuration of the adapter instance, required to perform subscription;
- message type associated with instance;
- data sources, with XPath (for mapping);
- semantic values (for configuration support).

Considering named properties describing shop floor devices the object model had been created. The class diagram of the model can be observed on Figure 18.

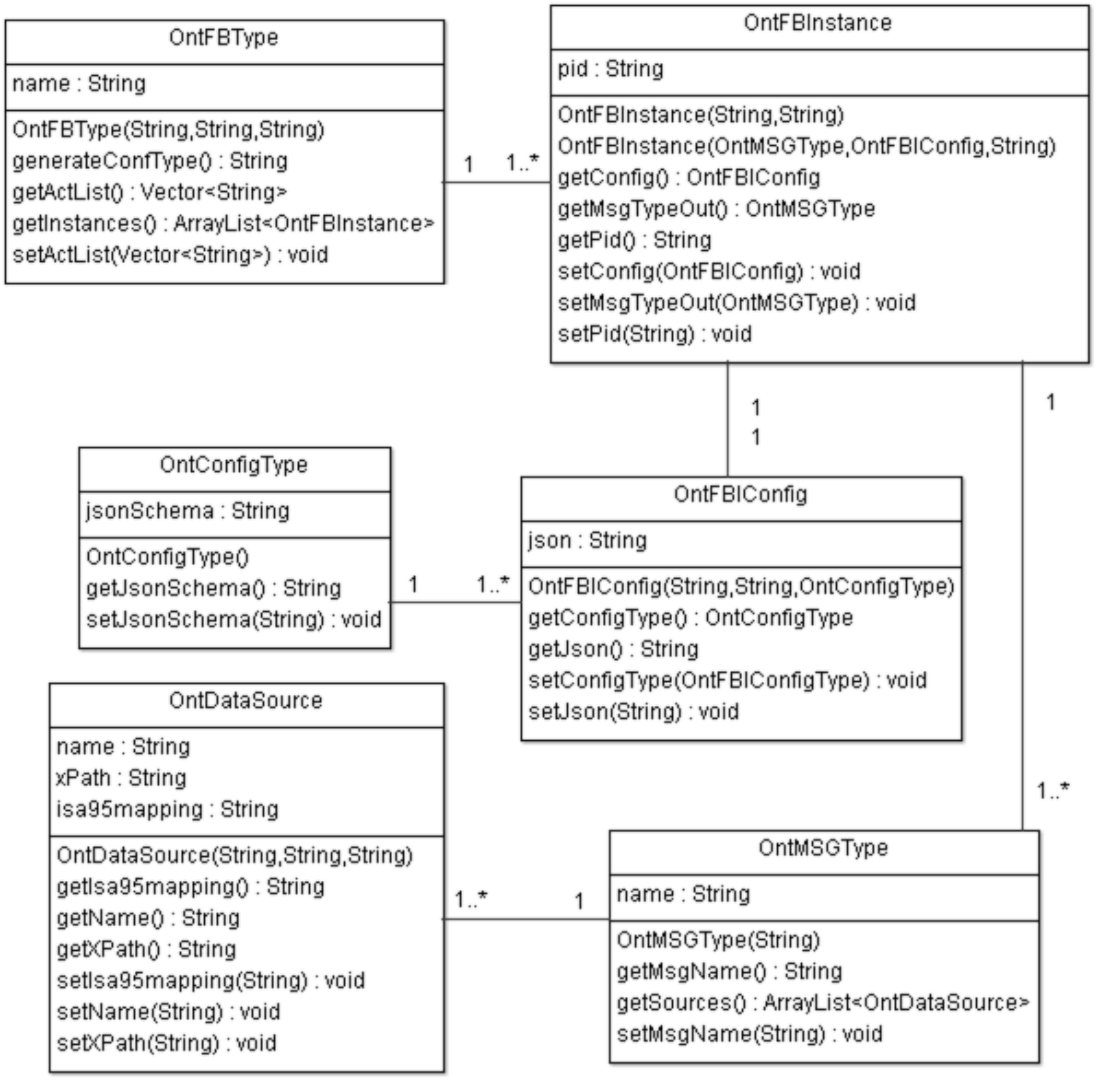

Figure 18: Class diagram of factory shop floor devices object model

In order to parse WSDL files with semantic annotations EasyWSDL Toolbox had been employed. This programming library implements the methods to parse standard WSDL files of versions 1.1 and 2.0 both. It also includes possible extensions to parse complex data types and SAWSDL annotations. Unfortunately, current version of EasyWSDL – 2.1 did not support parsing of WSDL 1.1 with SAWSDL annotations and complex data types. For this reason extraction of semantic annotation had been implemented using properties of only one extension – complex WSDL parsing, and extracting the annotation as an attribute of XML element.

The algorithm developed to parse the WSDL with semantic annotations is provided on Figure 19. In order to parse the semantic annotations for WSDL another recursive sub-algorithm had been employed. This algorithm is depicted on Figure 20. The recursive approach allows to parse the data types of arbitrary level of granularity.

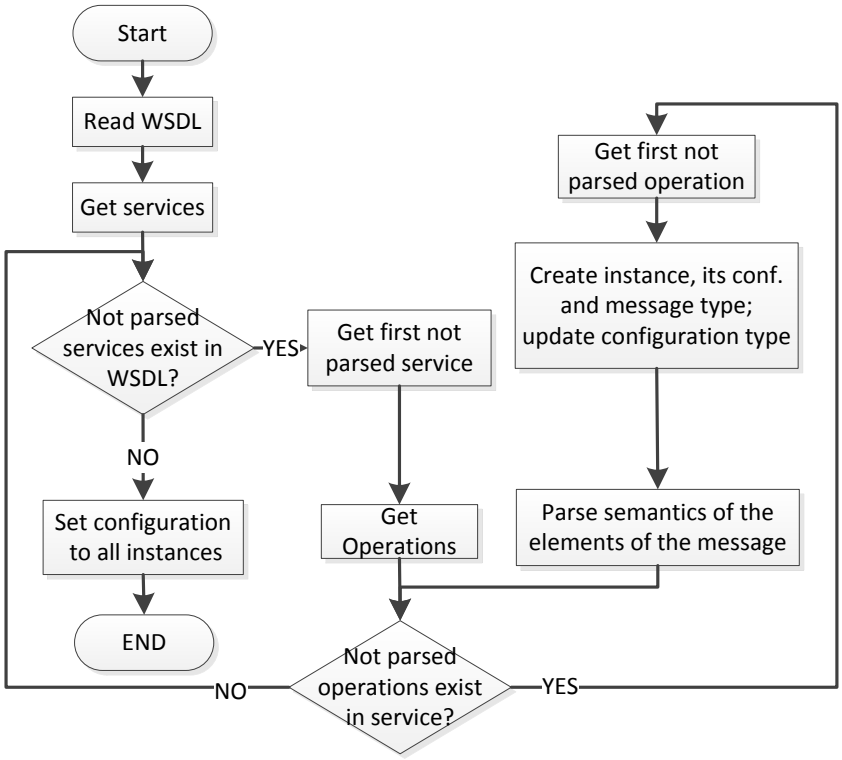

Figure 19: Algorithm for WSDL parsing

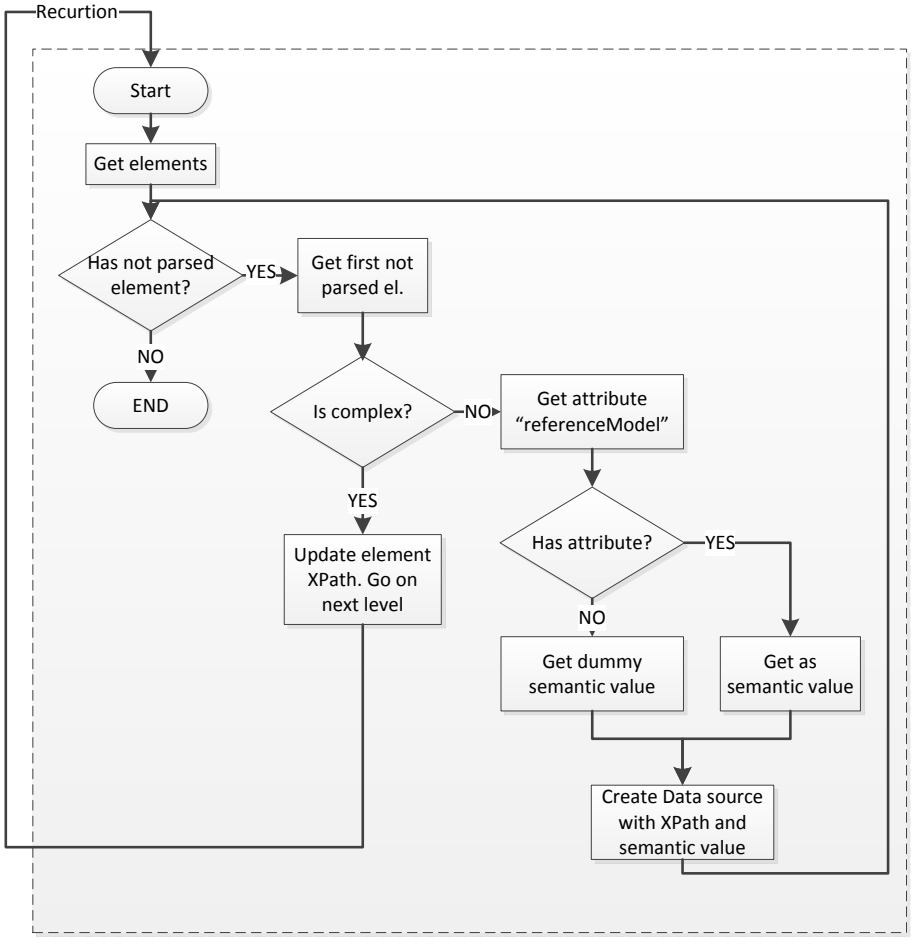

Figure 20: Recursive parsing of elements in SAWSDL

4.1.4. Mapping Module

The information stored in the Java object has to be transferred to the ontology, which already contains the defined formulae for metrics and KPIs. These formulae are employing semantics to define types of data used in calculations. The CFBNO ontology developed by Borja Ramis Ferrer includes the classes required to describe the shop floor device generated data bridging semantics and data integration related information. Thus a formula of a metric through semantics explicitly defines the configuration required for data integration. This ontology model does not consider some required data properties hence. Needed data properties were added to modified CFBNO(see Figure 21).

The mapping module is employed to transform the Java objects generated by parsing module to modified CFBNO ontology. The modifications had been made to provide efficient and unambiguous description of adapter based data sources. Data properties and relations between the classes of modified CFBNO can be observed on Figure 21.

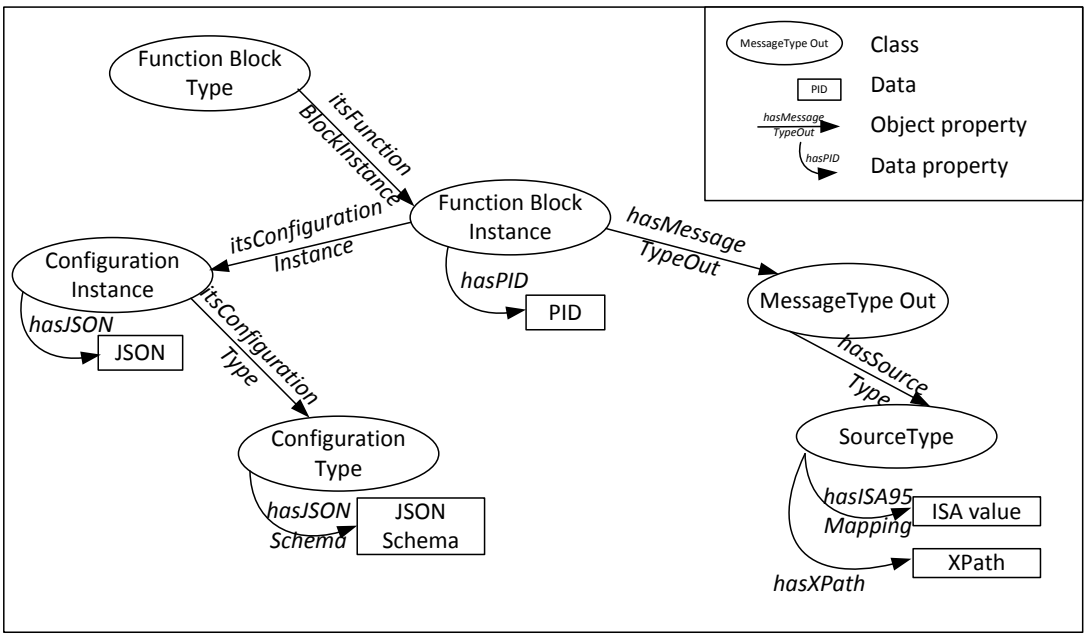

Figure 21: CFBNO: relations between classes and data properties of adapters

On Figure 21 can be observed that in CFBNO there is a class Function Block Type, which has a property "itsFunctionBlockInstance". This property is referring to one or more individuals of Function Block Instance class. FB Instance individual is describing a possible FB of adapter in PlantCockpit, which has certain configuration and generates certain messages. The configuration of FB and generated message types are defined by the properties of the individual.

FB Instance class individuals have a data property "hasPID" which refers to the PID of a function block. PID of a function block is unique identifier of textual type defining the type of a PlantCockpit FB. Also individual of FB Instance class describing adapter has two properties "itsConfigurationInstance" and "hasMessageTypeOut". Both of these properties are referring to other individuals.

“ItsConfigurationInstance” property refers to individual Configuration Instance class. This individual defines configuration of business logic of the instance. It has a property “itsConfigurationType” which refers to individual of a Configuration Type class defining the format of possible configurations of FB. The individual of Configuration Type class has a data property “hasJSONSchema”, which defines a schema of JSON file describing a business configuration of FB. The JSON which fits the schema defined in schema, is referred by individual of Configuration Instance class by data property “hasJSON”. The data referred by this property is a JSON string describing business configuration of the PlantCockpit FB.

As was noted before the individual of FB Instance class describing adapter also has “itsMessageTypeOut” property. This property refers to individual of a Message Type Out class. The individuals of Message Type Out class describe the message type of the notifications generated by adapter. To describe the elementary data particles of the messages generated by the adapter of defined business logic Message Type individual can have one or more “hasSource” properties.

The “hasSource” property refers to individual of Source Type class. Source Type individuals are describing semantic meaning and XPath location of elementary data units of the messages generated by adapter. Semantic meaning of the elementary data unit is defined by textual ISA-95 meaning of the data which is referred by a data property “hasISA95Meaning”. XPath location of the data is referred by “hasXPath” data property. XPath is also defined by a textual variable compatible with XPath standard.

To populate this individuals and their data properties describing available shop floor data adapters this module uses Apache Jena. Populating the ontology the individuals of the predefined classes are injected in the CFBNO ontology. The individuals in OWL ontology can be named and not named. A named instance should have a unique name, while not named individual created by Jena has an automatically generated unique ID. This makes the not named individuals preferable for ontologies with unpredictable amount of dissimilar instances, such as generated in the application of the thesis.

The algorithms of population of CFNBO developed for mapping module are presented on Figure 22. Firstly, as seen from Figure 22a, CFBNO ontology file should be loaded and a Configuration Type instance configuration should be generated. For DPWS adapter, the one employed in this thesis, the configuration schema of the adapter should include the list of all possible events to be subscribed. This Configuration Type Individual will be referred by all adapter Configuration individuals in ontology. Once the Configuration Type individual is created, the system should iterate over possible discovered configurations and map the FB Instances to the ontology. After all possible FB Instances are created in the ontology the OWL file should be saved.

On Figure 22b, algorithm of creation of a single FB Instance is presented. In this algorithm starts with creation of FB Instance individual, then the “hasPID” data property is attached to individual, with value corresponding to the FB. On the next stage Individuals of Configuration and Message Type Out classes have to be created and attached to FB Instance individual by appropriate properties.

The algorithm of creation of Configuration individuals is depicted on Figure 22c. According to this algorithm, firstly, the unnamed individual of Configuration class is created. For created individual the "hasJSON" data property is defined with values describing business logic of a FB instance. Also predefined individual of Configuration Type class is attached as "itsConfigurationType" property.

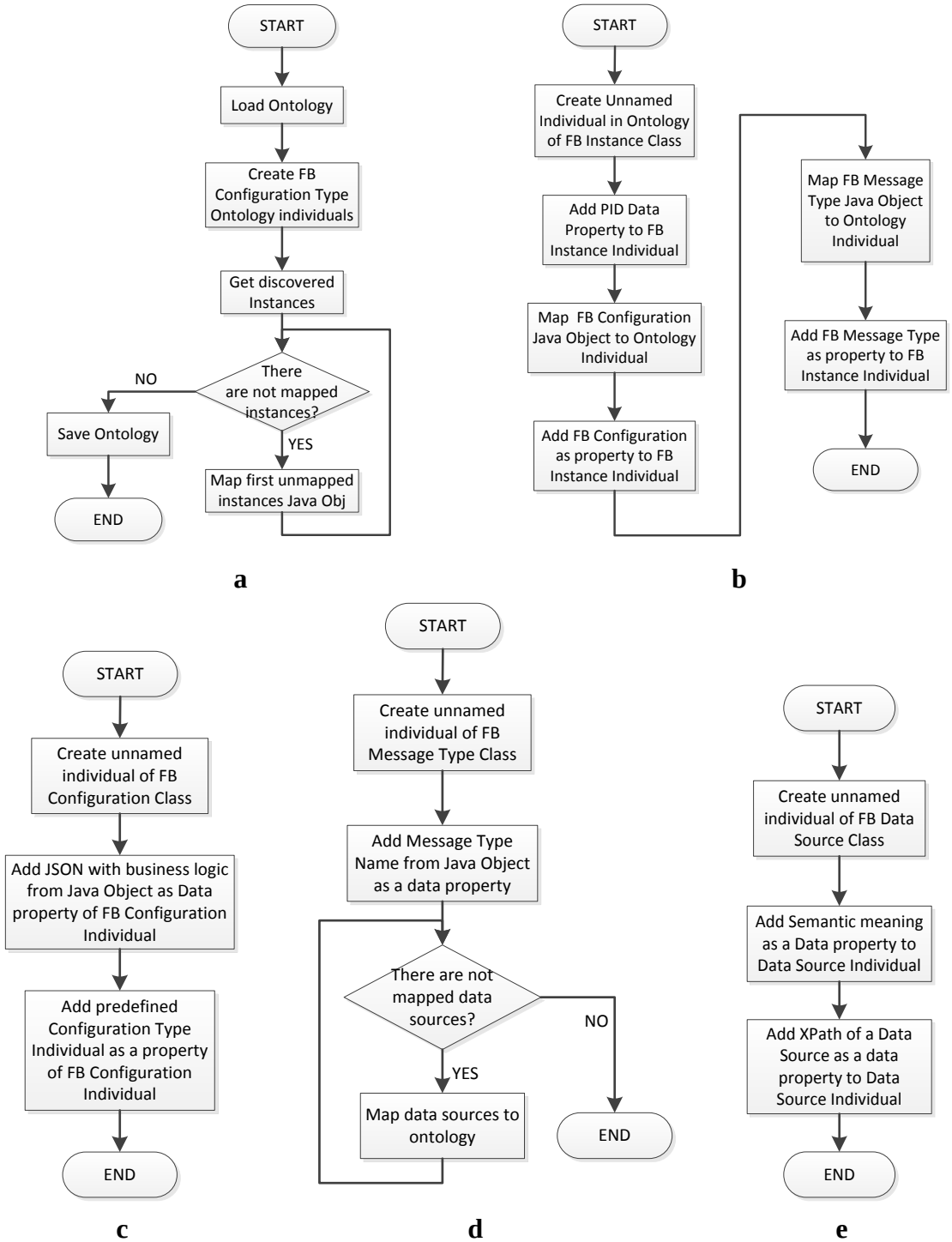

Figure 22: Algorithm for CFBNO population; a – algorithm of mapping of found FB instances to ontology; b – algorithm of mapping of a single FB instance; c – algorithm of mapping of a FB Instance configuration and configuration type to ontology; d – algorithm of mapping of a message type of FB instance; e – algorithm of mapping of a data source to ontology

On Figure 22d the algorithm of mapping of Message Types to proper class individuals is presented. In this algorithm after creation of unnamed individual of Message Type class the “hasName” data property is populated with message type name. After this step the elements of the message type should be described iteratively by mapping them to Source Type class individuals.

Finally on Figure 22e presents mapping of message elementary data units to Source Type class individuals. In this algorithm the firstly unnamed individual of Source Type individual is created, then its properties “hasISA95Meaning” and “hasXPath” are populated.

4.1.5. KB Analysis Module

To automate the configuration of the monitoring system the analysis of shop floor devices is required. The analysis should provide required data for data integration which will allow to calculate the metrics and KPIs.

This analysis is possible because CFBNO contains on the one hand the information about metrics and KPIs defining which data sources they require and how to manipulate them to obtain the metric value; on the other hand it contains the descriptions available data sources, which include data integration required information. Assembling this information full configuration of the monitoring system can be generated.

To perform the analysis the program at first should discover all metrics defined for the system in CFBNO. Then it should check each metric on presence of required data sources in the system (as the data sources of the system are mapped to CFBNO, this task can also be performed as ontology parsing). For the metrics having all data sources in the system the procedure of configuration assembly can be applied. On this procedure the information required for data integration and processing for one metric should be extracted from CFBNO. All the previous tasks are employing Apache Jena to parse CFBNO.

The data extracted from CFBNO is placed in a developed Java Object structure of which is presented on Figure 24. This Java object describes generic configuration required for data integration. The information required for data integration should include configuration of adapters to get information from external systems, configuration for function blocks of monitoring system, to process and visualize data and configuration for transformation between the blocks. This configuration should be transformed to the configuration format acceptable by certain monitoring system, which in this case is a PLANTCockpit.

The algorithms of this program module can be found on Figure 23.

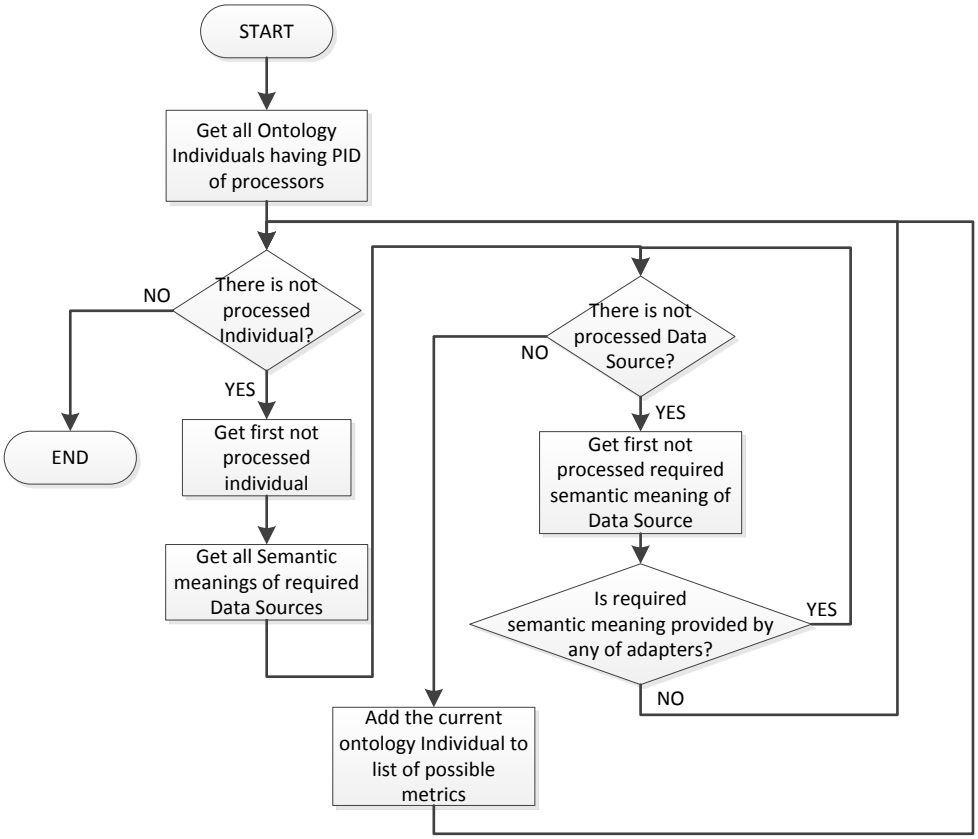

Figure 23: Algorithm for analysis of KB

On Figure 23 it can be observed that the algorithm of analysis starts with extraction of all individuals from ontology, which have PID referring to data processor. Each of extracted processors is able to calculate some metric provided it receives its predefined inputs. Hence if the data processor can receive all required data from other FBs the metric can be calculated. Current implementation includes the assumption that all metrics can be calculated employing one data processor FB and several adapter FBs. This limitation can be removed employing recursive analysis – the output of processors which have all required data sources should be added to list of available data sources and the analysis algorithm should be repeated until no more new metrics become available.

In the algorithm the processors inputs are being checked to be present as outputs of other FBs one by one, until all are processed. Input data sources are compared with output ones by their semantic meanings. As can be seen from Figure 23 if all of the input data sources of processor have at least one matching by semantic meaning output data source in the system, it is considered that this metric can be calculated in current system configuration. The metrics which can be calculated are displayed in UI.

4.1.6. FBN Creation

Creation of FBN configuration includes two main steps: firstly, the data integration information required for metric calculation should be extracted from ontology, secondly,

this information should be formatted in XML document compatible with monitoring system.

4.1.6.1 Extraction of data integration information

The data extraction aims to create the Java Object which will represent all data integration information related to configuration of the Function Block Network. This Java Object can be observed on Figure 24.

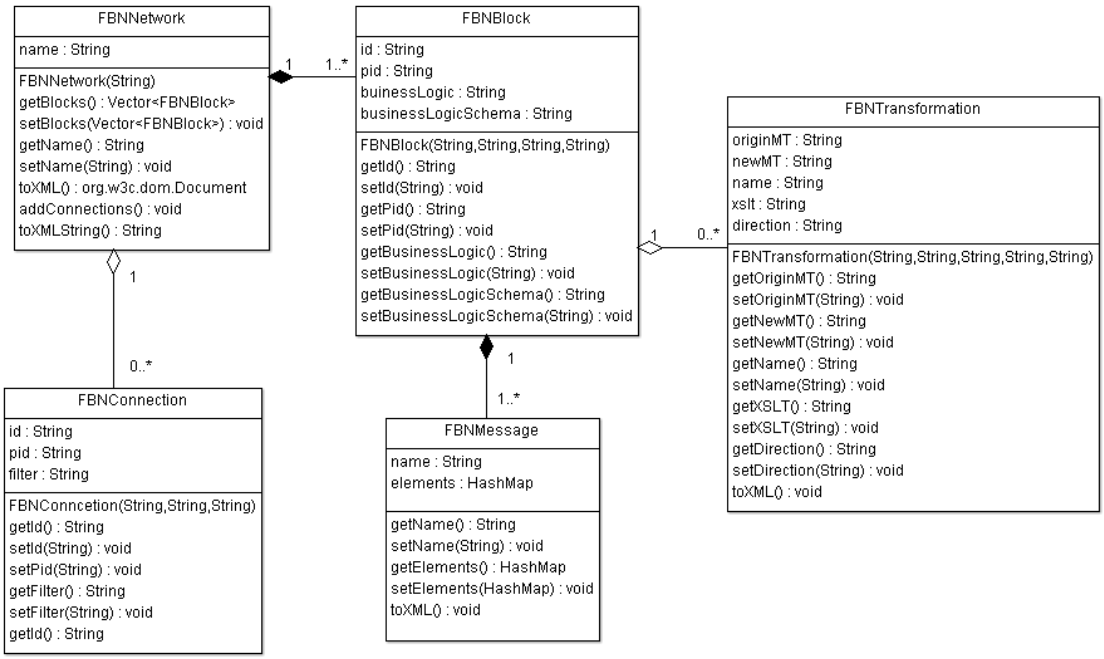

Figure 24: Java Object for FBN configuration

The class diagram presented on Figure 24 is presenting 5 classes. The Function Block Network is described by Java Object of FBNetwork Class. This object has a textual property which represents name of the network XML. Also the object describing FBN can include one or more objects of FBNConnection class. FBNConnection object describes the message flow between the FB, it has ID and PID of source or destination Function Block and message filter, which are required to create connection between FBs. FBNConnection objects are grouped in input connections and output connections of function blocks which can be assembled in single connection descriptor by matching the filtering message.

FBN is composed of one or more Java Objects of FBNBlock Class. The FBNBlock object has textual properties which describe FB ID, PID, businessLogic and businessLogicSchema. The FBNBlock object is also composed of one or more objects of FBNMessage class, which describe the message types of the messages related to FB. Among others FBNMessage objects includes the name of message type and mapped list of message elements and their semantic meanings.

FBNTransofrmation class objects may be owned by the objects of FBNBlock objects. FBNTransformation object describes the message transformations. This descrip-

tion defines the message type to be transformed, message type of transformed message, XSLT transformation and its name and direction of the message to be transformed.

All the classes mentioned above have the methods required to populate the new objects.

Process of creation of FBNNetwork Java Object based on the information from knowledge base has two phases. Firstly, FBNBlock describing the data processor required to metric calculation and related FBNConnection objects should be created. Then FBNBlock objects describing adapters required for calculation the metric presented by processes. Corresponding algorithms are presented on Figure 25 and Figure 26.

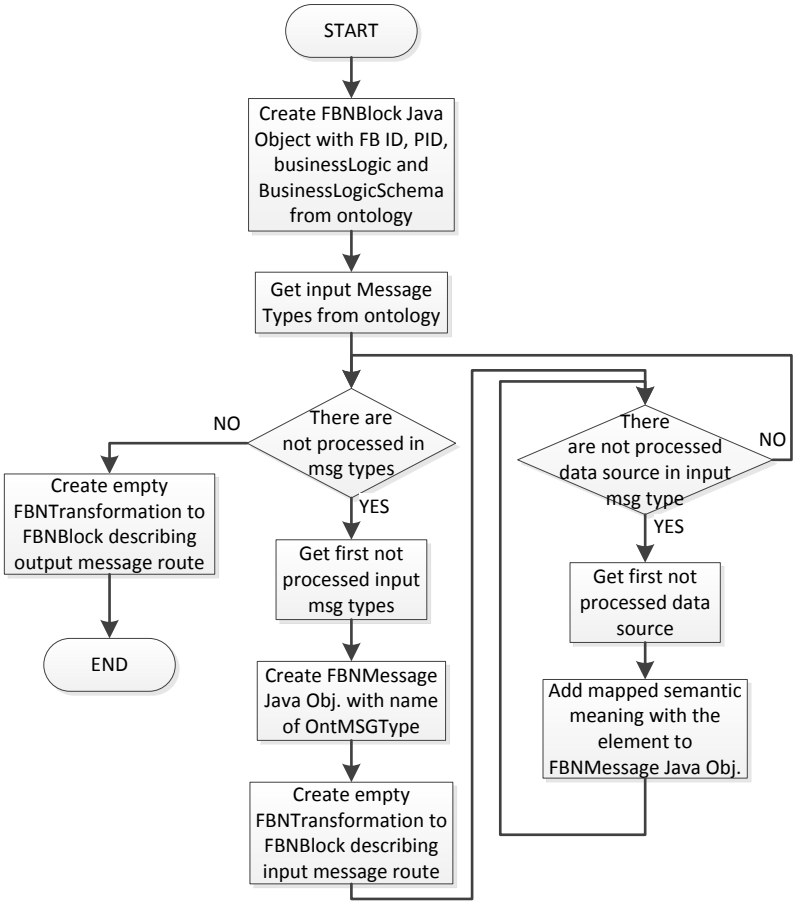

Figure 25: Creation of FBNBlock Java Object of processor

On Figure 25 algorithm employed for creation of FBNBlock of data processor is presented. The algorithm starts with creation of FBNBlock object representing the processor FB with generation of ID and extraction of PID, businessLogic and businessLogicSchema form ontology individual.

As the processor requires one or more data sources to be processed, the FB representing the processor should have one or more inputs. These inputs should be described by FBNMessage objects which characterize the message type of the inputs. To create these objects the Message type individuals should be extracted from ontology. Once the input message types are extracted from ontology the message types should be processed one-by-one and FBNMessage objects should be created in FBNBlock object. Each

FBNMessage object should have corresponding FBNTransformation Java Object. In case of input messages of processors no transformations are required as all transformations should be implemented on outbound messages of the function blocks. Next step on creation of the FBNMessage describing the message types is description of its parts by creation of the map between the message parts and their semantic meanings.

Finally when all the input message types of the FBNBlock are created, the FBN-Transformation describing output transformation should be created.

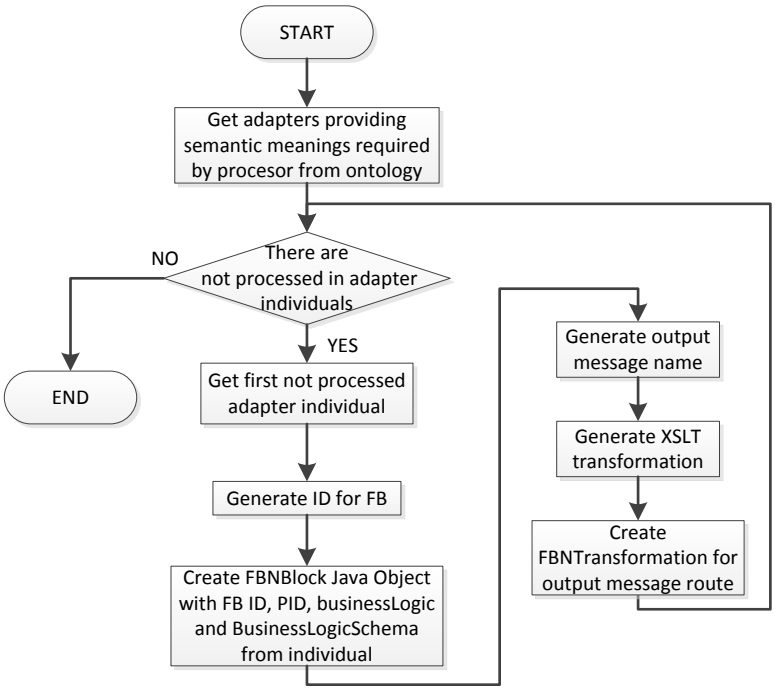

Figure 26: Creation of FBNBlocks Java Objects of adapters

On Figure 26 the creation algorithm of FBNBlocks describing adapters is depicted. From the algorithm it can be seen that firstly the list of adapters required for calculation of the metric should be extracted from ontology. This extraction is performed by search for adapters providing the semantic meanings which were defined on creation of FBNBlock describing the processor.

Once the list of ontology individuals describing adapters is formed, the adapter FBNBlock should be created. To create FBNBlock objects the iteration through the list of adapters is performed. On each of the iterations similar steps are performed as the ones required to creation of the processor FBN, with the difference related to presence of output messages and related transformations.

On creation of output transformations which are required to map the data from adapter to message format of processor inputs, XSLT transformation template is required. This transformation is generated based on the data about location of the value with required semantic meaning in the output message of adapter and expected location of this value in input message of the processor.

4.1.6.2 Formatting of data integration information in PlantCockpit FBN XML format

Second phase of creation of the FBN XML includes the process of transformation of data integration information to a specific format of XML file which can be processed by the FBEC in order to generate a FBN calculating a metric. The schema of FBN XML is presented on Figure 27.

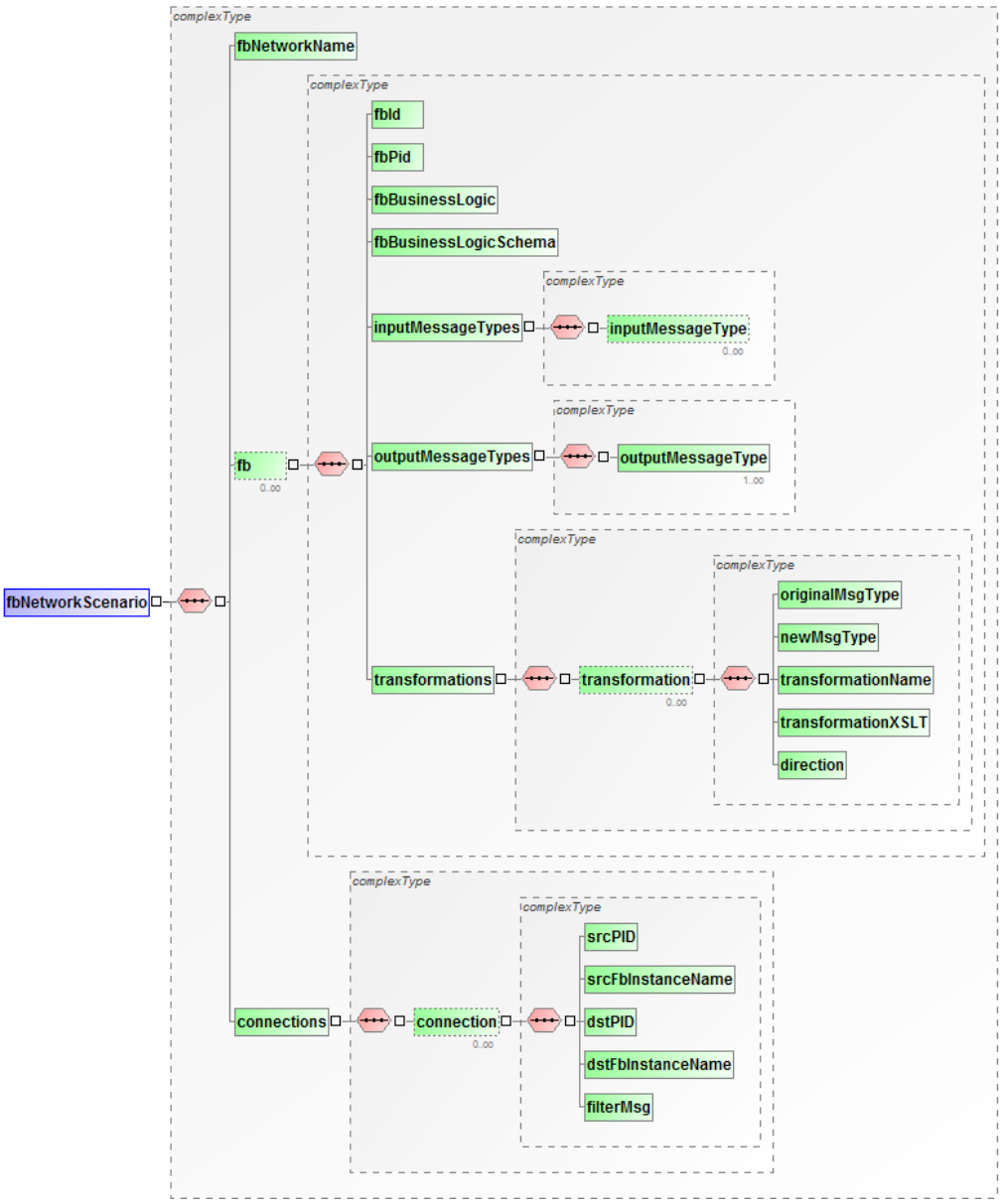

Figure 27: XSD schema of configuration FBN

As can be observed on figure above, the root element of FBN XML is fbNetworkScenario, which includes three child elements: fbNetworkName, fb and connections. First element fbNetworkName is of string type and defines the name to be related to the FBN in PlantCockpit system. Element fb is an element of a complex type which defines the configuration of individual function block. In FBN XML the amount of fb

elements is unbounded. Finally connections element is a parent element for the connection description elements of a complex type.

The data type of fb includes four elements of plain string type and three complex type elements. The elements of a string type are “fbId” which should define unique id of function block instance in the system; “fbPid” defines to which type of the function blocks curecnt instance belongs; “fbBusinessLogic” should specify the business logic of the FB instance as a JSON string and “fbBusinessLogicSchema” defines the format of a business logic JSON string. Among the complex type elements there are “inputMessageTypes” and “outputMessageTypes” which enlists the names of input message types and output message types correspondingly. The last element of FB description “transformations” enlists the transformation definitions presented in XML as the “transformation” elements of a complex type. The “transformation” element type includes the “originalMsgType” and “newMsgType” child elements which specify the message names. “direction” element specifies if the transformation should be applied to input or output messages. Finally, “transformationXSLT” defines the transformation itself and “transformationName” is used to identify it.

The child elements of “connections” element - “connection” are employed to define the message flow in the system. Each “connection” element has 5 child elements: source and destination IDs (“srcFbInstanceName” , “dstFbInstanceName”), source and destination PIDs (“srcPID”, “dstPID”) to define the connection explicitly and a “filterMsg” which defines messages of which type should be delivered from source to destination. All child elements of the “configuration” are of string data type.

The FBNNetwork Java Object and its parts are implementing a method, which transforms the FBNNetwork object to XML of the format defined above.

4.2. Use Case implementation

For the use case calculation of production time metric had been employed. This metric represents duration of manufacturing of a work piece. To calculate this metric the timestamp of start of the process should be subtracted from timestamp of end of the process. The formula to calculate the metric is presented in Equation 1.

$$Production\ Time = End\ timestamp - Start\ timestamp \quad (1)$$

This metric definition should be placed in the CFBNO, together with data required to configure the FBN for data processing. Employing PLANTCockpit framework, to calculate production time metric two instances of adapters which will retrieve start and end timestamps from manufacturing line and one instance of event processor which will subtract the timestamps will be required. For retrieval of the timestamps the instances of DPWS adapter will be used and as event processor Esper Function Block will be employed. Both DPWS adapter and ESPER CEP function blocks are provided by PLANTCockpit framework.

As it is mentioned in assumptions the configuration of the processor should be defined manually in the ontology. Same time no configuration for DPWS adapters will be placed in CFBNO, only references on taxonomy defined semantic values. Section 4.2.1 will describe the metric definition.

This metric will be applied to Festo MPS® 500-FMS. For simplicity in this use case the production process will be limited with feeder and tester stations. The time when the work piece enters the feeder station should be mapped in system as start timestamp, and time when the work piece leaves the tester station should be the end timestamp. As was mentioned in methodology section, the Inico S1000 devices should process their the input values obtained from the shop floor devices and generate the notifications containing information for monitoring system (such as start and end timestamps in this case). The notifications generated by gateway devices should be described in WSDL file of Inico, and provide semantic descriptions to the message elements employing SAWSDL specification. In section 4.2.2 the configuration of S1000 required to use them as gateway devices for the system will be presented.

Based on information defined in ontology and semantic descriptions attached to gateway devices, the developed tool will be able to generate the configuration of FBN for PLANTCockpit framework. Configured system will be able to calculate the defined metric employing the information delivered to system in notifications from the shop floor.

4.2.1. Metric definition

To configure the metric in CFBNO the ontology should be created. Protégé 4.2 has been employed as editor for ontology. Manually in ontology had been defined the FB instance of Esper CEP processor. This configuration included the source types required to calculate the metric. This source types had attached data properties which specified taxonomy defined semantic meanings of data. Configured metric named “calcTime”, which corresponds to Production Time metric, can be observed in Figure 28.

From Figure 28 it can be observed that “calcTime” metric has a reference on configuration instance “calcTimeConfiguration” which contains business logic of the Esper CEP FB instance. Also “calcTime” has a reference on two element instances named “Tester” and “Feeder” which in their turn have references on source type instances “TSTesterOUT” and “TSFeederIN”. These source types have data properties which define the semantic meaning of the data which is required to calculate the metric.

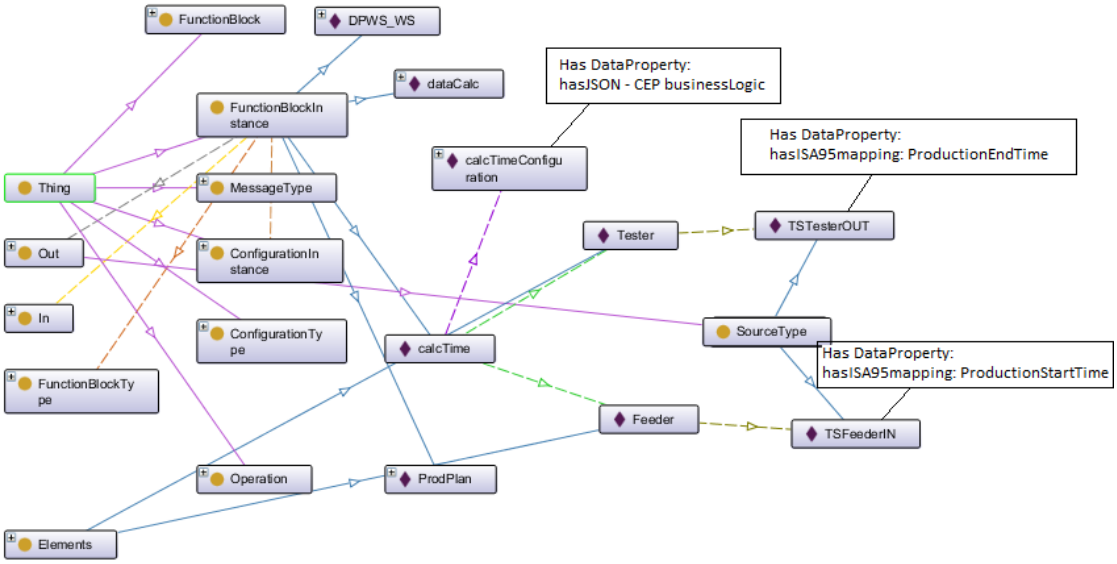

Figure 28: Ontology with definition of the production duration metric

4.2.2. Configuration of gateway devices

Inico S1000 the gateway devices selected for this system has to be configured in order to provide the monitoring system with information from the shop floor. This configuration includes configuration of controller logic which should transform input signals from the manufacturing equipment to data about manufacturing process. Then this data should be placed in the messages and sent to the subscribers. To make notification services available for monitoring systems it should be defined in WSDL of the device. Also this WSDL should contain semantic information about transmitting data in order to provide grounds for reasoning for monitoring system.

4.2.2.1 WSDL configuration

The structure of WSDL is already described in Section 2.1.2.1. The service description should define the all the stated values. In current section only most important decisions will be outlined. Whole WSDL developed for the use case can be found in Appendix 2.

On Figure 29 the most important parts of created WSDL file of one of the stations are presented. Going on the bottom of the WSDL file the definition on the port type and its operations can be found. The operation which provides notification should declare only output message. In this work for both feeder and tester stations the operation required to calculate duration of manufacturing of a work piece is named “TransferEvent”. The production starts when the work piece is provided to first station and finishes once it is removed from last one, hence the “TransferEvent” will contain both timestamps of the beginning of the process and timestamp of its end. The definition of operation contains the reference to the corresponding message.

As can be observed on Figure 29, the message definition provides the link on the element of the message which in its turn defined message schema. In case of “TransferMessage” through “TransferMsg” the “TransferMsgElementType” can be declared.

This element type schema defines format of the message sent with notification. The message will contain such elements as “Direction”, “StationID” and a “Timestamp”. The timestamp contains additional semantic meaning, which in case of feeder station defines that this timestamp refers to start of production:

```
<definitions ...>

    <types>
        <schema ...>
            ...
            <complexType name="TransferMsgElementType">
                <sequence>
                    <element name="Direction" type="string"/>
                    <element ref="tns:StationID"/>
                    <element ref="tns:Timestamp" sawsdl:modelReference="isa95:ProductionStartTime"/>
                </sequence>
            </complexType>
            ...
            <element name="TransferMsg" type="TransferMsgElementType"/>
            ...
        </schema>
    </types>
    ...
    <message name="TransferMessage">
        <part name="TransferMessagePart" element="TransferMsg"/>
    </message>
    ...
    <portType name="FeederServicePortType" wse:EventSource="true">
        <operation name="TransferEvent">
            <output message="TransferMessage"/>
        </operation>
        ...
    </portType>
    ...
</definitions>
```

Figure 29: Parts of WSDL file

<element ref="tns:Timestamp" sawsdl:modelReference="isa95:ProductionStartTime"/>

For the tester station in this use case the semantic value would be ProductionEnd-Time.

4.2.2.2 Device logic for notification generation

For realization of the use case scenario Inico S1000 devices should provide the notifications on work piece transition in and out of workstation, which represent production start time and end time. The RTU devices were directly connected to the sensors of the workstations the sensors changes of which are corresponding to the changes of status of the work pieces. This allowed notification generation without any complex logic.

The state diagram representing notifications generation actions in reference to states of the workstation can be observed on Figure 30.

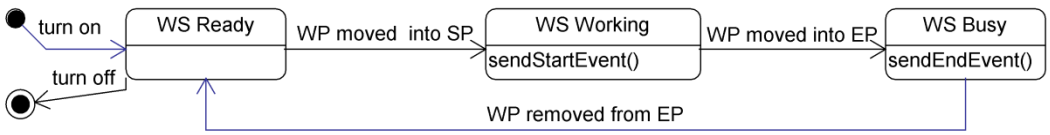

Figure 30: State diagram of the WorkStation

On state diagram Workstations (WS) can be in three main states: Ready, Working and Busy. State Ready represents the system being able to receive a new work piece

(WP) and start to process it. After system is started it automatically is assumed to be in Ready state. Once the WP is received in start position (SP) by the WS it changes its status to Processing and sends the Production start notification. After the processing of WP is finished the work piece is moved in end position (EP) of the WS and the station changes its state to Busy. When system becomes Busy it sends notification representing end of processing of a work piece in WS. As soon as WP will be removed from the EP of workstation the station returns to Ready state.

5. RESULTS

In this chapter the results of the use case implementation and research in general will be presented.

5.1. Use Case

During execution of the use case in the thesis project the manufacturing line attached Inico devices has been successfully discovered. On changes of the configuration of the shop floor the repository of WSDL files which represents the shop floor configuration had been updated correspondingly. WSDL file for the feeder station as example of WSDL with semantics employed in use case can be found in Appendix 2.

The WSDL semantics had been successfully extracted, parsed, processed and mapped to ontology (which did not have any instances of data source class). The SPARQL query which shows the data sources in populated CFBNO with corresponding semantic meanings and XPath of data source in parent messages is presented in Table 12. The results of query contain two data sources with defined semantic meanings (ProductionStartTime, ProductionEndTime). These semantic meanings were retrieved from semantic annotations of WSDL and are used in analysis for possible metrics in current system configuration.

Table 12: SPARQL query of populated CFBNO ontology and part of corresponding result

SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX onto: <http://www.plantcockpit.eu/ontologies/functionblock_network.owl#>
SELECT ?Data_Source ?Semantic_Meaning
        WHERE { ?Data_Source onto:hasISA95Mapping ?Semantic_Meaning }
```

Data_Source	Semantic_Meaning	XPath
<urn:AnonId:ccbbff6c_9ad7_49ad_8ab7_53272e25efe4>	"NO SEMANTIC VALUE"	"/TransferMsg/StationID"
<urn:AnonId:7ba3e9a0_a344_4152_87fd_86503ec33039>	"NO SEMANTIC VALUE"	"/TransferMsg/Direction"
<urn:AnonId:9d21b870_c7d4_4323_a8d2_8f6e5ca1d070>	"NO SEMANTIC VALUE"	"/TransferMsg/Direction"
<urn:AnonId:8253bd72_7ef7_46e0_8ff7_c4c8987e8f13>	"ProductionEndTime"	"/TransferMsg/Timestamp"
<urn:AnonId:47d34a71_b321_4f79_803a_e4996c1ce33a>	"NO SEMANTIC VALUE"	"/WorkStationStatusMsg/StationID"
<urn:AnonId:e8d6d698_bc82_4bae_97b6_683d9ea290b0>	"NO SEMANTIC VALUE"	"/WorkStationStatusMsg/Timestamp"
<urn:AnonId:7c1708d2_02ff_4174_bb6c_38d0c6feb4e4>	"ProductionStartTime"	"/TransferMsg/Timestamp"
<urn:AnonId:1ef682f5_91a0_41b0_a958_191c2af980fb>	"NO SEMANTIC VALUE"	"/AlarmMsg/StationID"

Analysis of the ontology provided correct list of available metrics which were available for calculation in current configuration of the shop floor devices. Finally, genera-

tion of the FBN XML has been performed successfully, providing the proper configuration for the PLANTCockpit system, for metric calculation. Automatically generated FBN XML can be found in Appendix 1.

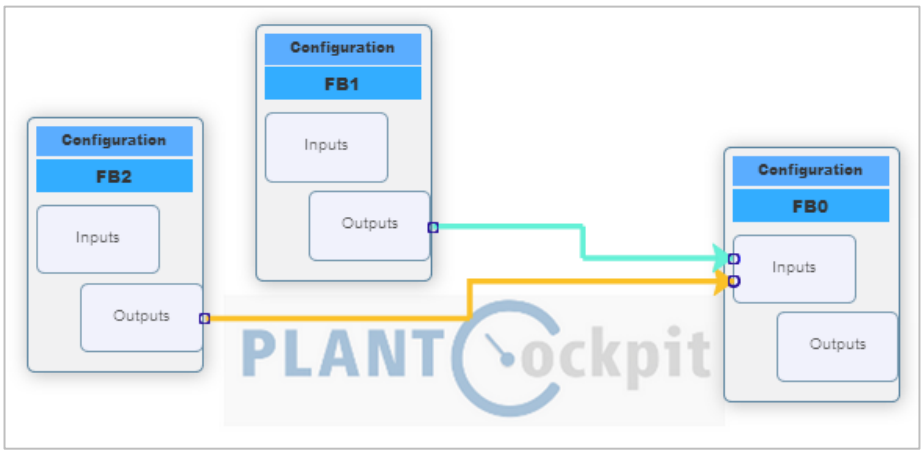

Figure 31: FBEC with deployed network for use case

On Figure 31 which depicts the deployed FBN for metric calculation three named above function blocks can be observed. FB0 is Esper CEP data processor and FB1 and FB2 are DWPS adapters. Automatically generated function block network was able to calculate the duration of manufacturing of a work piece based on notifications received from DPWS devices as it was expected.

5.2. Concepts and learnings

As a result of this thesis work a solution for a problem defined in 1.2 has been developed. Most importantly this solution responds to the stated questions.

- ***How to synchronize monitoring system with shop floor?***

For synchronization of the monitoring system with shop floor in this thesis following technical solution is offered. On shop floor level either devices immediately or employing gateway devices should implement DPWS technology. DPWS technology offers dynamic, event triggered expression of changes in industrial data sources. Implementation of DPWS notification listeners can provide the data required to map the state of the shop floor to a knowledge base. The ontology has been used in the offered solution as a knowledge base. This set of tools, technologies and approaches allows to synchronize the status of the shop floor with the world model used in monitoring system.

- ***How to calculate the metrics to be displayed in monitoring system minimising customization of monitoring system?***

Conceptually, to remove a need for customization the system should provide deep configuration options. Especially is important to separate business logic from functional logic, and allow user to configure the business logic without a need of system recompilation. In terms of reusability of the system the functionality should be distributed between the loosely-coupled modules. This approach can be achieved employing FB concept. Hence to minimize the system customization and thus to reuse more the system

component and simplify their maintenance function blocks concept should be used in the monitoring system.

- ***How to define metrics for the monitoring system so that configuration of the monitoring system can be performed automatically?***

Employing FB concept (provided all required FB types are created) any metric calculation can be realized using FB network. FB network is the function block instances and connections between them, hence creation of the function block network, which uses existing FB types requires only configuration of the FBs and connections. Therefore, definition of the metric should unambiguously describe the FBN configuration.

Firstly, this definition is proposed to provide description of the data processor FB, which should contain metadata about configuration of the FB and describe required data sources. In this thesis for simplicity the metadata about configuration included the configuration itself, but obviously the metadata can be more abstract.

Secondly, following description of adapters is offered in this thesis: the adaptor FB description should contain technical details about data extraction and semantic meaning of the data provided by the adapter. Employing semantic description in the devices which can be exposed to the system the available adapter descriptions can be generated automatically.

Finally, it is offered to generate the connections based on the matching pairs of semantic meanings of adapter generated data and the ones of data sources of data processors.

With proposed approach, automated generation of the FBN configuration is possible employing definition of the metrics and the metadata placed in the shop floor devices. As storage for extracted data about adapters and metrics definition the ontology is proposed to be used, as it provides high expressivity, reasoning mechanisms and can be dynamically extended for more complex data processing.

- ***How to analyse current status of the system in order to provide configuration for monitoring system?***

As the status of the shop floor devices is mapped in the ontology, analysis of system status can be performed using data from ontology. This analysis should provide the system with a list of the predefined metrics which can be calculated in current status of the system. For metric to be calculated it should have all required data sources which are provided by the shop floor devices. Mapping of data sources required and available semantic meanings of the data is employed. Hence the analysis of the current status should be analysis of the ontology for metrics for which all data sources are available in the ontology. To extract data from ontology to Java Objects Apache JENA had been used and later the analysis logic had been implemented in the program.

6. CONCLUSIONS

In this chapter the research will be concluded and possible future work to extend benefits of proposed solution will be offered. Also final thoughts about this thesis work will be presented.

6.1. Thesis conclusions

Contemporary enterprise information systems, such as industrial monitoring systems, require extensive amount of manual configuration on set up. Standardization and unification of data access partially simplifies technical side of configuration process, but still high level of expertise is required from the person who will configure the system. This expertise is in combination of knowledge about production process and data integration. Employing attachment of device description from manufacturing process point of view and algorithms to process these descriptions, one can automate the configuration process. The approach for such automatic configuration was developed in this thesis.

From analysed theoretical background, implementation and use case it was concluded that for the dynamic system recognition in non-intrusive industrial monitoring system knowledge extraction, representation and analysis are required. Recapitalization of main decisions is offered in next paragraph.

Knowledge extraction from factory shop floor devices requires dynamic, expressive and acceptable on industrial device level set of technologies to store metadata about device and extract data from it on demand. DPWS as a technology bringing WS functionality to device level, is providing dynamic discovery of devices, protocol for data and metadata extraction. Extensibility of WSDL files which are lying in the base of WS allows introduction of more detailed, device specific metadata about devices employing SAWSDL. This set of technologies covers data and metadata extraction from shop floor devices. Knowledge representation requires a KB, which will contain the representation of real world system. This KB should hold knowledge about metrics which theoretically can be required in the system and also information about available data sources (for example from shop floor level). OWL ontologies can be employed as a KB for the developed system. As ontology model the CFBNO ontology offered by Borja Ramis Ferrer in his thesis was modified and employed. To populate and analyse the OWL ontology Apache JENA API was used.

Set of technologies and models mentioned above can be employed in contemporary factories and taking into account the trend of increasing capabilities and decreasing of smart devices this approach should be even more applicable in future. Described approach requires creation definitions of metrics to be calculated in the system as well as

to describe the shop floor devices. These tasks can be executed by different specialists on demand, can be reused in different manufacturing systems or their configurations and most importantly, drastically, decreases amount of work and level of expertise required on industrial monitoring system set up.

6.2. Future work

The solution defined in this work had been implemented as a software tool and tested on a real system. As the result of testing the system was able to connect to the shop floor devices and dynamically track the changes on manufacturing system configuration. The tool was able to parse the ontology and to find all metrics which are defined in the system. On analysis the developed tool was capable to find all metrics which can be calculated in following configuration of the shop floor. Furthermore, the FBN configuration file was generated based on predefined metric definition and ontology instances created on synchronization with manufacturing system. This configuration was successfully deployed to PLANTCockpit monitoring system and the metric was calculated correctly.

The goal of the thesis work was achieved, thou there are number of issues where the solution can be improved.

- Firstly, more abstract definition of the metric will be preferable, as it will allow to implement the mechanism of generation of business logic for dissimilar data processors not only Esper CEP ones. This improvement will offer more reusability of the system in cost higher computation resources employed.
- Secondly, analysis of the ontology can be performed employing SPARQL querying language. Usage of SPARQL can offer more readability for the analysis algorithm, reducing its maintenance costs. Also querying will allow to modify the analysis algorithm without need to recompile code, thou for current task the analysis algorithm does not need to be modified as long as ontology model is not changed. Hence usage of SPARQL can introduce more reusability and related benefits of cost reduction, while slightly reducing performance of the system.
- Thirdly, the ontology side can be improved, to be able to analysis more details about shop floor devices such as their topology, hierarchy and so on. This approach can provide more possibilities to describe data sources and thus to define the data for metrics more explicitly. Perspective from this point of view is Semantic Web Rule Language (SWRL).

6.3. Final considerations

In this thesis work the contemporary studies applicable to research topic had been studied, analysed and based on results methodology dynamic system recognition for non-intrusive monitoring systems based on semantic models had been offered. According to

this methodology the system employing the dynamic system recognition had been developed. Developed system was tested in real test bed and has proved that it provides expected functionality. Hence the task of research had been achieved.

During research in theoretical background required for this thesis dissimilar scientific and technical sources had been studied. This study allowed to create general approach for development of dynamic system recognition tool, thou this approach was corrected on development as new technical problems took place. Finally after application of the concept and developed tool to real manufacturing system on testing provided even more ideas for improvement of the concept, which had been concluded in section 6.2.

REFERENCES

[1] "EU Manufacturing Industry: What are the Challenges and Opportunities for the Coming Years?" 26-Apr-2010.

[2] "Manufacturing, value added (% of GDP) | Data | Graph." [Online]. Available: http://data.worldbank.org/indicator/NV.IND.MANF.ZS/countries/1W-EU-US-JP-CN?display=graph. [Accessed: 30-Jul-2013].

[3] "Manufacturing statistics - NACE Rev. 2 - Statistics Explained." [Online]. Available: http://epp.eurostat.ec.europa.eu/statistics_explained/index.php/Manufacturing_statistics_-_NACE_Rev._2. [Accessed: 30-Jul-2013].

[4] European Union, European Commission, and Directorate-General for Enterprise and Industry, *EU industrial structure 2011: trends and performance*. Luxembourg: Publications Office of the European Union, 2011.

[5] "SOA and Web Services." [Online]. Available: http://www.oracle.com/technetwork/articles/javase/soa-142870.html. [Accessed: 30-Jul-2013].

[6] "Home - Sirena 2003 - 2005." [Online]. Available: http://www.sirena-itea.org/. [Accessed: 15-Mar-2013].

[7] "Home - Soda - 2006." [Online]. Available: http://www.soda-itea.org/. [Accessed: 15-Mar-2013].

[8] "Web Services for Devices (WS4D)." [Online]. Available: http://ws4d.e-technik.uni-rostock.de/. [Accessed: 15-Mar-2013].

[9] "Home - Socrades." [Online]. Available: http://www.socrades.eu/Home/default.html. [Accessed: 15-Mar-2013].

[10] "SOA4D Forge: Welcome." [Online]. Available: https://forge.soa4d.org/. [Accessed: 15-Mar-2013].

[11] "Inico Technologies." [Online]. Available: http://www.inicotech.com/about.html. [Accessed: 30-Jul-2013].

[12] "PlantCockpit: Home." [Online]. Available: http://www.plantcockpit.eu/. [Accessed: 15-Mar-2013].

[13] S. Iarovyi, J. Garcia, and J. L. M. Lastra, "An approach for OSGi and DPWS interoperability: Bridging enterprise application with shop-floor," in *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, 2013, pp. 200–205.

[14] "BatchControl.com: Now This is Exciting!" [Online]. Available: http://www.batchcontrol.com/s95/s95.shtml. [Accessed: 30-Jul-2013].

[15] A. Florea, J. A. G. I. Montemayor, C. Postelnicu, and J. L. M. Lastra, "A cross-layer approach to energy management in manufacturing," in *2012 10th IEEE International Conference on Industrial Informatics (INDIN)*, 2012, pp. 304–308.

[16] V. Vasyutynskyy, C. Hengstler, J. McCarthy, K. G. Brennan, D. Nadoveza, and A. Dennert, "Layered architecture for production and logistics cockpits," in *2012 IEEE 17th Conference on Emerging Technologies Factory Automation (ETFA)*, 2012, pp. 1–9.

[17] G. Hohpe, *Enterprise integration patterns: designing, building, and deploying messaging solutions*. Boston: Addison-Wesley, 2004.
[18] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: State of the Art and Research Challenges," *Computer*, vol. 40, no. 11, pp. 38–45, 2007.
[19] M. H. Valipour, B. Amirzafari, K. N. Maleki, and N. Daneshpour, "A brief survey of software architecture concepts and service oriented architecture," in *2nd IEEE International Conference on Computer Science and Information Technology, 2009. ICCSIT 2009*, 2009, pp. 34–38.
[20] "Microsoft, IBM, SAP To Discontinue UDDI Web Services Registry Effort | SOA World Magazine." [Online]. Available: http://soa.sys-con.com/node/164624. [Accessed: 17-Oct-2013].
[21] "Web Services Architecture." [Online]. Available: http://www.w3.org/TR/ws-arch/. [Accessed: 15-Mar-2013].
[22] "OASIS Web Services Dynamic Discovery (WS-Discovery) Version 1.1." [Online]. Available: http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html. [Accessed: 17-Oct-2013].
[23] "Web Service Definition Language (WSDL)." [Online]. Available: http://www.w3.org/TR/wsdl. [Accessed: 17-Oct-2013].
[24] "WSDL portType." [Online]. Available: http://www.w3schools.com/wsdl/wsdl_ports.asp. [Accessed: 02-Sep-2013].
[25] "Web Services Eventing (WS-Eventing)." [Online]. Available: http://www.w3.org/Submission/WS-Eventing/. [Accessed: 17-Oct-2013].
[26] "SOA and EDA: Using events to bridge decoupled service boundaries." [Online]. Available: http://searchsoa.techtarget.com/tip/SOA-and-EDA-Using-events-to-bridge-decoupled-service-boundaries. [Accessed: 30-Jul-2013].
[27] K. Walzer, J. Rode, D. Wunsch, and M. Groch, "Event-driven manufacturing: Unified management of primitive and complex events for manufacturing monitoring and control," in *IEEE International Workshop on Factory Communication Systems, 2008. WFCS 2008*, 2008, pp. 383–391.
[28] G. I. Montemayor and J. Andres, "A Complex Event Processing System for Monitoring of Manufacturing Systems," Mar. 2012.
[29] S. Ortiz, "Getting on Board the Enterprise Service Bus," *Computer*, vol. 40, no. 4, pp. 15–17, 2007.
[30] A. Otto and K. Hellmann, "IEC 61131: A general overview and emerging trends," *IEEE Ind. Electron. Mag.*, vol. 3, no. 4, pp. 27–31, 2009.
[31] J. L. M. Lastra, *Function Blocks for Industrial-process Measurement and Control Systems: IEC-61499 Introduction and Run-time Platforms*. Tampere University of Technology, 2004.
[32] C. Sunder, A. Zoitl, J. H. Christensen, V. Vyatkin, R. W. Brennan, A. Valentini, L. Ferrarini, T. Strasser, J. L. Martinez-Lastra, and F. Auinger, "Usability and Interoperability of IEC 61499 based distributed automation systems," in *2006 IEEE International Conference on Industrial Informatics*, 2006, pp. 31–37.
[33] V. Vyatkin, "IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review," *IEEE Trans. Ind. Inform.*, vol. 7, no. 4, pp. 768–781, 2011.
[34] A. Dennert, A. Gossling, J. Krause, M. Wollschlaeger, and A. M. Henao Montoya, "Vertical data integration in automation based on IEC 61499," in *2012 9th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2012, pp. 99–102.

[35] A. W. Colombo, F. Jammes, H. Smit, R. Harrison, J. L. M. Lastra, and I. M. Delamer, "Service-oriented architectures for collaborative automation," in *31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005*, 2005, p. 6 pp.–.

[36] J. M. L. AW Colombo, "An approach to develop flexible & collaborative factory automation systems (FLEXCA)," *Teti Red Proc. 4th CIRP Int. Semin. Intell. Comput. Manuf. Eng. CIRP ICME3904 June 30-July 2 2004 Sorrento Italy*, 2011.

[37] I. M. Delamer and J. L. M. Lastra, "Service-Oriented Architecture for Distributed Publish/Subscribe Middleware in Electronics Production," *IEEE Trans. Ind. Inform.*, vol. 2, no. 4, pp. 281–294, Nov. 2006.

[38] I. M. Delamer and J. L. M. Lastra, "Loosely-coupled Automation Systems using Device-level SOA," in *2007 5th IEEE International Conference on Industrial Informatics*, 2007, vol. 2, pp. 743–748.

[39] A. Lobov, J. Puttonen, V. V. Herrera, R. Andiappan, and J. L. M. Lastra, "Service oriented architecture in developing of loosely-coupled manufacturing systems," in *6th IEEE International Conference on Industrial Informatics, 2008. INDIN 2008*, 2008, pp. 791–796.

[40] G. Candido, F. Jammes, J. B. de Oliveira, and A. W. Colombo, "SOA at device level in the industrial domain: Assessment of OPC UA and DPWS specifications," in *2010 8th IEEE International Conference on Industrial Informatics (INDIN)*, July, pp. 598–603.

[41] H. Bohn, A. Bobek, and F. Golatowski, "SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains," in *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006*, 2006, pp. 43–43.

[42] E. Zeeb, G. Moritz, D. Timmermann, and F. Golatowski, "WS4D: Toolkits for Networked Embedded Systems Based on the Devices Profile for Web Services," in *2012 41st International Conference on Parallel Processing Workshops*, Los Alamitos, CA, USA, 2010, vol. 0, pp. 1–8.

[43] A. Cannata, M. Gerosa, and M. Taisch, "SOCRADES: A framework for developing intelligent systems in manufacturing," in *IEEE International Conference on Industrial Engineering and Engineering Management, 2008. IEEM 2008*, 2008, pp. 1904–1908.

[44] L. M. S. de Souza, P. Spiess, D. Guinard, M. Köhler, S. Karnouskos, and D. Savio, "SOCRADES: A Web Service Based Shop Floor Integration Infrastructure," in *The Internet of Things*, C. Floerkemeier, M. Langheinrich, E. Fleisch, F. Mattern, and S. E. Sarma, Eds. Springer Berlin Heidelberg, 2008, pp. 50–67.

[45] Z. Yang, J.-B. Zhang, R. Gay, L. Zhuang, and H. M. Lee, "Building a Semantic-Rich Service-Oriented Manufacturing Environment," in *Web Information Systems Engineering – WISE 2005*, A. H. H. Ngu, M. Kitsuregawa, E. J. Neuhold, J.-Y. Chung, and Q. Z. Sheng, Eds. Springer Berlin Heidelberg, 2005, pp. 623–632.

[46] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?," *Int. J. Hum.-Comput. Stud.*, vol. 43, no. 5–6, pp. 907–928, Nov. 1995.

[47] "Theoretical Foundations of Ontologies," in *Ontological Engineering*, Springer London, 2004, pp. 1–45.

[48] L. LIU and M. T. ÖZSU, Eds., "Ontologies," in *Encyclopedia of Database Systems*, Springer US, 2009, pp. 1959–1959.

[49] S. Borgo and P. Leitão, “Foundations for a Core Ontology of Manufacturing,” in *Ontologies*, R. Sharman, R. Kishore, and R. Ramesh, Eds. Springer US, 2007, pp. 751–775.
[50] J. L. M. L. Omar J. López Orozco, “Using semantic web technologies to describe automation objects.,” *IJMR*, vol. 1, pp. 482–503, 2006.
[51] J. L. M. Lastra and I. M. Delamer, “Ontologies for Production Automation,” in *Advances in Web Semantics I*, T. S. Dillon, E. Chang, R. Meersman, and K. Sycara, Eds. Springer Berlin Heidelberg, 2009, pp. 276–289.
[52] J. L. M. Lastra, I. M. Delamer, and F. Ubis, *Domain Ontologies for Reasoning Machines in Factory Automation*. ISA, 2010.
[53] A. Lobov, F. U. Lopez, V. V. Herrera, J. Puttonen, and J. L. M. Lastra, “Semantic Web Services framework for manufacturing industries,” in *IEEE International Conference on Robotics and Biomimetics, 2008. ROBIO 2008*, 2009, pp. 2104–2108.
[54] B. Ramis Ferrer, “An ontological approach for modelling configuration of factory-wide data integration systems based on IEC-61499,” Jun. 2013.
[55] M. Schleipen, R. Drath, and O. Sauer, “The system-independent data exchange format CAEX for supporting an automatic configuration of a production monitoring and control system,” in *IEEE International Symposium on Industrial Electronics, 2008. ISIE 2008*, 2008, pp. 1786–1791.
[56] O. Sauer, “Trends in Manufacturing Execution Systems,” in *Proceedings of the 6th CIRP-Sponsored International Conference on Digital Enterprise Technology*, G. Q. Huang, K. L. Mak, and P. G. Maropoulos, Eds. Springer Berlin Heidelberg, 2010, pp. 685–693.
[57] R. Drath, “Let’s talk AutomationML what is the effort of AutomationML programming?,” in *2012 IEEE 17th Conference on Emerging Technologies Factory Automation (ETFA)*, 2012, pp. 1–8.
[58] R. Drath, A. Luder, J. Peschke, and L. Hundt, “AutomationML - the glue for seamless automation engineering,” in *IEEE International Conference on Emerging Technologies and Factory Automation, 2008. ETFA 2008*, 2008, pp. 616–623.
[59] S. Faltinski, O. Niggemann, N. Moriz, and A. Mankowski, “AutomationML: From data exchange to system planning and simulation,” in *2012 IEEE International Conference on Industrial Technology (ICIT)*, 2012, pp. 378–383.
[60] T. Holm, L. Christiansen, M. Goring, T. Jager, and A. Fay, “ISO 15926 vs. IEC 62424 #x2014; Comparison of plant structure modeling concepts,” in *2012 IEEE 17th Conference on Emerging Technologies Factory Automation (ETFA)*, 2012, pp. 1–8.
[61] P. D. W. Marquardt, J. M. D. Ing, A. W. D. Ing, and D. A. Yang, “Related Work on Ontologies for Engineering Applications,” in *OntoCAPE*, Springer Berlin Heidelberg, 2010, pp. 369–390.
[62] S. Berger, G. Grossmann, M. Stumptner, and M. Schrefl, “Metamodel-Based Information Integration at Industrial Scale,” in *Model Driven Engineering Languages and Systems*, D. C. Petriu, N. Rouquette, and Ø. Haugen, Eds. Springer Berlin Heidelberg, 2010, pp. 153–167.
[63] P. T. PhD and C. G. Dr, “Trends in Automation,” in *Springer Handbook of Automation*, P. S. Y. Nof, Ed. Springer Berlin Heidelberg, 2009, pp. 127–143.
[64] H. O. Unver, “An ISA-95-based manufacturing intelligence system in support of lean initiatives,” *Int. J. Adv. Manuf. Technol.*, vol. 65, no. 5–8, pp. 853–866, May 2012.

[65] D. L. Nastasie, A. Koronios, and A. Haider, “Integration Through Standards – An Overview of Internal Information Standards for Engineering Asset,” in *Definitions, Concepts and Scope of Engineering Asset Management*, J. E. Amadi-Echendu, K. Brown, R. Willett, and J. Mathew, Eds. Springer London, 2010, pp. 239–258.

[66] “IBM Emerging Technologies - jStart - On The Horizon - Semantic Enrichment,” 19-Jan-2012. [Online]. Available: http://www-01.ibm.com/software/ebusiness/jstart/semantic/. [Accessed: 07-Aug-2013].

[67] A. Hinze, R. Heese, A. Schlegel, and M. Luczak-Rösch, “User-Defined Semantic Enrichment of Full-Text Documents: Experiences and Lessons Learned,” in *Theory and Practice of Digital Libraries*, P. Zaphiris, G. Buchanan, E. Rasmussen, and F. Loizides, Eds. Springer Berlin Heidelberg, 2012, pp. 209–214.

[68] J. Cardoso, J. A. Miller, and S. Emani, “Web Services Discovery Utilizing Semantically Annotated WSDL,” in *Reasoning Web*, C. Baroglio, P. A. Bonatti, J. Małuszyński, M. Marchiori, A. Polleres, and S. Schaffert, Eds. Springer Berlin Heidelberg, 2008, pp. 240–268.

[69] “Semantic Annotations for WSDL Working Group.” [Online]. Available: http://www.w3.org/2002/ws/sawsdl/. [Accessed: 08-Aug-2013].

[70] J. Kopecky, T. Vitvar, C. Bournez, and J. Farrell, “SAWSDL: Semantic Annotations for WSDL and XML Schema,” *IEEE Internet Comput.*, vol. 11, no. 6, pp. 60–67, 2007.

[71] “OASIS Web Services Discovery and Web Services Devices Profile (WS-DD) TC | OASIS.” [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-dd. [Accessed: 15-Mar-2013].

APPENDIX 1: USE CASE FBN XML

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<fbNetworkScenario>
  <fbNetworkName>calcTime</fbNetworkName>
  <fb>
    <fbId>FB0</fbId>
    <fbPid>pid_eu.plant.plant_fb_Esper</fbPid>
    <fbBusinessLogic>{ "EsperConfigura-
tion":{"CEPServiceName":"FB0","EventProcessingQuery":{"queryStatement":"SELEC
T\nTester.TSTesterOUT - Feeder.TSFeederIN as Dura-
tion\nFROM\nTester.win:length(1) as Tester,\nFeeder.win:length(1) as Feed-
er"},"InputMessageDefinitions":{"InputMessage":[{"elements":{"Element":[{"Name":"
TSTesterOUT","SampleValue":"123123123123112"}]},"rootElementName":"Tester"},
{"ele-
ments":{"Element":[{"Name":"TSFeederIN","SampleValue":"123123123123121"}]},"r
ootElementName":"Feeder"}]}}}</fbBusinessLogic>
    <fbBusinessLogicSchema>{"type":"object","$schema": "http://json-
schema.org/draft-03/schema","id": "#","required":false,"properties":{ "EsperConfigura-
tion": { "type":"object", "id": "EsperConfiguration", "required":false, "properties":{
"CEPServiceName": { "type":"string", "id": "CEPServiceName", "required":false },
"EventProcessingQuery": { "type":"object", "id": "EventProcessingQuery", "re-
quired":false, "properties":{ "queryStatement": { "type":"string", "id": "queryState-
ment", "format":"text", "_inputex": { "rows":3, "cols":70, "description": "Event Pro-
cessing Language Query expected " } , "required":false } } }, "InputMessageDefini-
tions": { "type":"object", "id": "InputMessageDefinitions", "required":false, "proper-
ties":{ "InputMessage": { "type":"array", "id": "InputMessage", "required":false,
"_inputex": {"description": "Input message declaration " } , "items": { "type":"object",
"id": "0", "required":false, "properties":{ "elements": { "type":"object", "id": "ele-
ments", "required":false, "properties":{ "Element": { "type":"array", "id": "Element",
"required":false, "items": { "type":"object", "id": "0", "required":false, "properties":{
"Name": { "type":"string", "id": "Name", "required":false }, "SampleValue": {
"type":"string", "id": "SampleValue", "required":false } } } } } }, "rootElementName":
{ "type":"string", "id": "rootElementName", "required":false } } } } } } } }
}}</fbBusinessLogicSchema>
    <inputMessageTypes>
      <inputMessageType>Feeder</inputMessageType>
      <inputMessageType>Tester</inputMessageType>
    </inputMessageTypes>
    <outputMessageTypes>
      <outputMessageType>FB0_OUT</outputMessageType>
    </outputMessageTypes>
    <transformations>
```

```
    <transformation>
      <originalMsgType>Feeder</originalMsgType>
      <newMsgType/>
      <transformationName/>
      <transformationXSLT/>
      <direction>INPUT</direction>
    </transformation>
    <transformation>
      <originalMsgType>Tester</originalMsgType>
      <newMsgType/>
      <transformationName/>
      <transformationXSLT/>
      <direction>INPUT</direction>
    </transformation>
    <transformation>
      <originalMsgType>FB0_output</originalMsgType>
      <newMsgType/>
      <transformationName/>
      <transformationXSLT/>
      <direction>OUTPUT</direction>
    </transformation>
  </transformations>
 </fb>

 <fb>
  <fbId>FB1</fbId>
  <fbPid>pid_eu.plant.plant_fb_dpws_adapter</fbPid>
  <fbBusinessLog-
Log-
ic>{"DPWSConfig":{"subscribe":{"string":["192.168.1.4;http://www.fast.tut.fi/iarovyi/
thesis/TesterServicePortType/TransferEvent"]}}}</fbBusinessLogic>
  <fbBusinessLogicSchema>{"type":"object","$schema":"http://json-schema.org/draft-
03/schema","id":"#","required":false,"properties":{"DPWSConfig":{"type":"object","id
":"DPWSConfig","title":"DPWSConfig","required":false,"properties":{"subscribe":{"ty
pe":"object","id":"subscribe","required":false,"properties":{"string":{"type":"array","titl
e":"Subscribe","id":"stringArray","required":false,"items":{"type":"string","title":"Actio
n","_inputex": { "description": "NOTE! Currently only one action can be subscribed by
each instance of the adapt-
er"},"choices":[{"value":"192.168.1.4;http://www.fast.tut.fi/iarovyi/thesis/TesterService
Port-
Type/TransferEvent"},{"value":"192.168.1.4;http://www.fast.tut.fi/iarovyi/thesis/Tester
ServicePort-
```

Type/OperatorEvent"},{"value":"192.168.1.4;http://www.fast.tut.fi/iarovyi/thesis/TesterServicePort-
Type/EnergyEvent"},{"value":"192.168.1.4;http://www.fast.tut.fi/iarovyi/thesis/TesterServicePort-
Type/AlarmEvent"},{"value":"192.168.1.4;http://www.fast.tut.fi/iarovyi/thesis/TesterServicePort-
Type/WorkStationStatusEvent"},{"value":"192.168.3.45;http://www.fast.tut.fi/iarovyi/the-
sis/FeederServicePortType/TransferEvent"},{"value":"192.168.3.45;http://www.fast.tut.fi/iarovyi/thesis/FeederServicePortType/OperatorEvent"},{"value":"192.168.3.45;http://www.fast.tut.fi/iarovyi/thesis/FeederServicePortType/EnergyEvent"},{"value":"192.168.3.45;http://www.fast.tut.fi/iarovyi/thesis/FeederServicePortType/AlarmEvent"},{"value":"192.168.3.45;http://www.fast.tut.fi/iarovyi/thesis/FeederServicePortType/WorkStationStatusEvent"}],"id":"string","required":false}}}}}}}</fbBusinessLogicSchema>
<inputMessageTypes/>
<outputMessageTypes>
<outputMessageType>192-168-1-4_TransferEvent</outputMessageType>
</outputMessageTypes>
<transformations>
<transformation>
<originalMsgType>192-168-1-4_TransferEvent</originalMsgType>
<newMsgType>Tester</newMsgType>
<transformationName>192-168-1-
4_TransferEvent2Tester13</transformationName>
<transformationXSLT><?xml version="1.0"?> <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/"><Tester><TSTesterOUT><xsl:value-of select="/TransferMsg/Timestamp"/></TSTesterOUT></Tester></xsl:template></xsl:stylesheet>
<direction>OUTPUT</direction>
</transformation>
</transformations>
</fb>

<fb>
<fbId>FB2</fbId>
<fbPid>pid_eu.plant.plant_fb_dpws_adapter</fbPid>
<fbBusinessLog-
Log-
ic>{"DPWSConfig":{"subscribe":{"string":["192.168.3.45;http://www.fast.tut.fi/iarovyi/thesis/FeederServicePortType/TransferEvent"]}}}

<fbBusinessLogicSchema>{"type":"object","$schema":"http://json-schema.org/draft-03/schema","id":"#","required":false,"properties":{"DPWSConfig":{"type":"object","id":"DPWSConfig","title":"DPWSConfig","required":false,"properties":{"subscribe":{"type":"object","id":"subscribe","required":false,"properties":{"string":{"type":"array","title":"Subscribe","id":"stringArray","required":false,"items":{"type":"string","title":"Action","_inputex": { "description": "NOTE! Currently only one action can be subscribed by each instance of the adapter"},"choices":[{"value":"192.168.1.4;http://www.fast.tut.fi/iarovyi/thesis/TesterServicePortType/TransferEvent"},{"value":"192.168.1.4;http://www.fast.tut.fi/iarovyi/thesis/TesterServicePortType/OperatorEvent"},{"value":"192.168.1.4;http://www.fast.tut.fi/iarovyi/thesis/TesterServicePortType/EnergyEvent"},{"value":"192.168.1.4;http://www.fast.tut.fi/iarovyi/thesis/TesterServicePortType/AlarmEvent"},{"value":"192.168.1.4;http://www.fast.tut.fi/iarovyi/thesis/TesterServicePortType/WorkStationStatusEvent"},{"value":"192.168.3.45;http://www.fast.tut.fi/iarovyi/thesis/FeederServicePortType/TransferEvent"},{"value":"192.168.3.45;http://www.fast.tut.fi/iarovyi/thesis/FeederServicePortType/OperatorEvent"},{"value":"192.168.3.45;http://www.fast.tut.fi/iarovyi/thesis/FeederServicePortType/EnergyEvent"},{"value":"192.168.3.45;http://www.fast.tut.fi/iarovyi/thesis/FeederServicePortType/AlarmEvent"},{"value":"192.168.3.45;http://www.fast.tut.fi/iarovyi/thesis/FeederServicePortType/WorkStationStatusEvent"}],"id":"string","required":false}}}}}}}}</fbBusinessLogicSchema>
<inputMessageTypes/>
<outputMessageTypes>
<outputMessageType>192-168-3-45_TransferEvent</outputMessageType>
</outputMessageTypes>
<transformations>
<transformation>
<originalMsgType>192-168-3-45_TransferEvent</originalMsgType>
<newMsgType>Feeder</newMsgType>
<transformationName>192-168-3-45_TransferEvent2Feeder13</transformationName>
<transformationXSLT><?xml version="1.0"?> <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/"><Feeder><TSFeederIN><xsl:value-of select="/TransferMsg/Timestamp"/></TSFeederIN></Feeder></xsl:template></xsl:stylesheet></transformationXSLT>
<direction>OUTPUT</direction>
</transformation>

```
  </transformations>
 </fb>

 <connections>
  <connection>
   <srcPID>pid_eu.plant.plant_fb_dpws_adapter</srcPID>
   <srcFbInstanceName>FB2</srcFbInstanceName>
   <dstPID>pid_eu.plant.plant_fb_Esper</dstPID>
   <dstFbInstanceName>FB0</dstFbInstanceName>
   <filterMsg>Feeder</filterMsg>
  </connection>
  <connection>
   <srcPID>pid_eu.plant.plant_fb_dpws_adapter</srcPID>
   <srcFbInstanceName>FB1</srcFbInstanceName>
   <dstPID>pid_eu.plant.plant_fb_Esper</dstPID>
   <dstFbInstanceName>FB0</dstFbInstanceName>
   <filterMsg>Tester</filterMsg>
  </connection>
 </connections>

</fbNetworkScenario>
```

APPENDIX 2: FEEDER WORK STATION WSDL

```
<?xml version="1.0" encoding="UTF-8" ?>
 <definitions name="FestoStationService" target-
Namespace="http://www.plantcockpit.eu/fast/festo"
 xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://www.plantcockpit.eu/fast/festo"
 xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
 xmlns:sawsdl="http://www.w3.org/ns/sawsdl"
 xmlns:isa95="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder">
 <types>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.plantcockpit.eu/fast/festo"
        xmlns:tns="http://www.plantcockpit.eu/fast/festo"
        elementFormDefault="qualified">

 <xs:element name="StationId" type="xs:string"/>
 <xs:element name="Timestamp" type="xs:long"/>
 <xs:element name="WorkpieceId" type="xs:string"/>
 <xs:element name="Response" type="xs:string"/>
 <xs:element name="Status" type="xs:string"/>

 <xs:simpleType name="WorkstationStatusCode">
       <xs:restriction base="xs:token">
             <xs:enumeration value="READY" />
             <xs:enumeration value="STARTING" />
             <xs:enumeration value="STARTED" />
             <xs:enumeration value="IDLE" />
             <xs:enumeration value="PROCESSING" />
             <xs:enumeration value="WAITING" />
             <xs:enumeration value="STOPPING" />
       </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="WorkpieceStatusCode">
       <xs:restriction base="xs:token">
             <xs:enumeration value="TRANSFERIN" />
             <xs:enumeration value="PROCESSING" />
             <xs:enumeration value="WAITING" />
             <xs:enumeration value="TRANSFEROUT" />
```

```
            <xs:enumeration value="NO_WORKPIECE" />
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="OperatorInputCodeType">
        <xs:restriction base="xs:token">
            <xs:enumeration value="START" />
            <xs:enumeration value="STOP" />
            <xs:enumeration value="RESET" />
            <xs:enumeration value="EMERGENCY_STOP" />
            <xs:enumeration value="AUTO_MANUAL_SWITCH" />
        </xs:restriction>
</xs:simpleType>

 <xs:complexType name="WorkstationStatusType">
  <xs:sequence>
   <xs:element ref="tns:StationId"/>
   <xs:element ref="tns:Status"/>
   <xs:element ref="tns:Timestamp"/>
  </xs:sequence>
 </xs:complexType>

 <xs:complexType name="OperatorInputType">
  <xs:sequence>
   <xs:element ref="tns:StationId"/>
   <xs:element name="Type" type="tns:OperatorInputCodeType"/>
   <xs:element name="Value" type="xs:string"/>
   <xs:element ref="tns:Timestamp"/>
  </xs:sequence>
 </xs:complexType>

 <xs:complexType name="WorkstationEventType">
  <xs:sequence>
   <xs:element ref="tns:StationId"/>
   <xs:element name="EventType" type="xs:string"/>
   <xs:element name="Value" type="xs:string"/>
   <xs:element ref="tns:Timestamp"/>
  </xs:sequence>
 </xs:complexType>

<xs:complexType name="WorkstationAlarmType">
  <xs:sequence>
```

```
      <xs:element ref="tns:StationId"/>
      <xs:element name="AlarmType" type="xs:string"/>
      <xs:element name="Value" type="xs:string"/>
      <xs:element ref="tns:Timestamp"/>
     </xs:sequence>
   </xs:complexType>


   <xs:complexType name="WorkpieceStatusType">
    <xs:sequence>
      <xs:element ref="tns:StationId"/>
      <xs:element ref="tns:WorkpieceId"/>
      <xs:element name="Status" type="tns:WorkpieceStatusCode"/>
      <xs:element ref="tns:Timestamp"
sawsdl:modelReference="isa95:ProductionStartTime"/>
     </xs:sequence>
   </xs:complexType>

    <xs:complexType name="WorkpiecePropertiesType">
     <xs:sequence>
      <xs:element ref="tns:StationId"/>
      <xs:element ref="tns:WorkpieceId"/>
      <xs:element name="EventType" type="xs:string"/>
      <xs:element name="Details" type="xs:string"/>
      <xs:element ref="tns:Timestamp"/>
     </xs:sequence>
   </xs:complexType>

  <xs:element name="WorkstationStatus" type="tns:WorkstationStatusType"/>
  <xs:element name="OperatorInput" type="tns:OperatorInputType"/>
  <xs:element name="WorkstationEvent" type="tns:WorkstationEventType"/>
  <xs:element name="WorkpieceStatus" type="tns:WorkpieceStatusType"/>
  <xs:element name="WorkpieceProperties" type="tns:WorkpiecePropertiesType"/>
  <xs:element name="WorkstationAlarm" type="tns:WorkstationAlarmType"/>
</xs:schema>

  </types>

   <message name="WorkstationStatusMessage">
    <part name="status" element="tns:WorkstationStatus"/>
   </message>
```

```
<message name="OperatorInputMessage">
  <part name="input" element="tns:OperatorInput"/>
 </message>

 <message name="WorkpieceStatusMessage">
  <part name="status" element="tns:WorkpieceStatus"/>
 </message>

<message name="WorkstationEventMessage">
  <part name="event" element="tns:WorkstationEvent"/>
 </message>

<message name="WorkstationAlarmMessage">
  <part name="event" element="tns:WorkstationAlarm"/>
 </message>

<message name="WorkpiecePropertiesMessage">
  <part name="event" element="tns:WorkpieceProperties"/>
 </message>

<message name="TransferInMessage">
  <part name="tin" element="tns:WorkpieceId"/>
 </message>

<message name="TransferInResponseMessage">
  <part name="tin" element="tns:Response"/>
 </message>

<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="WorkstationAlarm">
  <output message="tns:WorkstationAlarmMessage"/>
 </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>

<binding name="FestoStationServicePortType"
type="tns:FestoStationServicePortType">
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />

 <operation name="TransferIn">
   <soap:operation style="document" />
   <wsdl:input>
    <soap:body use="literal" />
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal" />
   </wsdl:output>
 </operation>

 <operation name="WorkstationStatus">
   <soap:operation style="document" />
   <wsdl:output>
    <soap:body use="literal" />
   </wsdl:output>
 </operation>

 <operation name="WorkstationEvent">
   <soap:operation style="document" />
```

```
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </operation>

  <operation name="WorkstationAlarm">
    <soap:operation style="document" />
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </operation>

  <operation name="OperatorInput">
    <soap:operation style="document" />
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </operation>

  <operation name="WorkpieceStatus">
    <soap:operation style="document" />
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </operation>

  <operation name="WorkpieceProperties">
    <soap:operation style="document" />
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </operation>

</binding>

 <service name="FestoStationService" sawsdl:modelReference="isa95:FEEDER">
   <port name="FestoStationServicePortType" bind-
ing="tns:FestoStationServicePortType">
    <soap:address location="http://192.168.2.62:80/dpws/ws01" />
  </port>
  </service>
  </definitions>
```