TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

FRANCESC DE BORJA RAMIS FERRER

AN ONTOLOGICAL APPROACH FOR MODELLING CONFIGURATION OF FACTORY-WIDE DATA INTEGRATION SYSTEMS BASED ON IEC-61499

Master of Science Thesis

Thesis topic approved in the Automation, Mechanical and Materials Engineering Faculty Council meeting on 8th of May 2013.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master’s Degree Programme in Machine Automation
RAMIS FERRER, FRANCESC DE BORJA: An ontological approach for modelling configuration of Factory-Wide data integration systems based on IEC-61499
Master of Science Thesis, 76 pages, 7 Appendix pages
May 2013
Major: Factory Automation
Examiner: Prof. José Luis Martínez Lastra
Supervisor: Jorge Garcia Izaguirre Montemayor
Keywords: system modelling, ontological approach, ontology, query, key performance indicator, function block, IEC-61499, service oriented architecture, factory automation, OWL, SPARQL.

The comprehensive study of Key Performance Indicators (KPIs) in nowadays manufacturing systems has become a need for improving the efficiency of production processes, pressed by the market demand. To achieve this management, industrial control systems are being modelled robustly by using standards.

This causes developers to use semantic technologies to deal with the complexity of data integration and modelling of systems. On the other hand, ISA-95 is an international standard that reduce human efforts by helping directly on the business logistics of manufacturing systems. Then, ISA-95-based implementations can be developed for data integration.

Moreover, the current trend on systems modelling is the use of ontologies which provide models to be more descriptive, allowing knowledge to be decoupled from business logic, and extending modularity of the domain knowledge. In addition, the application of AI to industrial control systems is being developed by the interoperability between a knowledge base, created by ontologies, and a reasoner.

This thesis proposes a methodology for modelling configuration for heterogeneous data integration while considering the various information models contained in the systems of the modelled manufacturing systems. The implementation of this work not only presents how to model a production line as an example of manufacturing system, but also allows carrying out the configuration of a Function Block Network (FBN) by specifying a KPI. Then, this work pretends to present a possible manner of KPI classification and its calculation by configuring a FBN, all of this by the use of ontologies.

PREFACE

I could not imagine a few years ago that today, at this time, I was going to be writing a thesis for a Master of Science degree. It has been a great challenge for me to leave my hometown with my couple and make a new life in Finland which is 3475 Km away from Mallorca.

However, this is not an illusion. I have learned a lot thanks to the Master of Science implementation. In fact, this thesis has been the result of a lot of effort and time dedication. But obviously, the success in these studies could not be done without some persons to whom I would want to express my grateful in this preface.

First of all, I would like to thank Prof. José Luis Martínez Lastra for the opportunity of learning in FAST laboratory environment. I tried to do my best in my studies for giving back all the confidence on me, which always has been demonstrated. I am sure that without his help I would not be able to reach the objectives of the work, studies and thesis. I think that all of his advices will be useful in my following career objectives.

In the same way, I want to thank all the staff members of FAST laboratory that also teach me some courses and were there for solving any doubt.

On the other hand, I would thank also specially to Jorge. He has not only been my supervisor in thesis and work but also a friend that has taught me a lot during my stay in FAST.

I want to thank also to all of the good friends that I have met here. Thus, thanks to Sergii and Dasha, Ahmed, Hector, Miri, Carlos, Johannes and, of course thanks to the Mexican community: Oscar, Gerardo, Luis, and Angélica. Finally, thanks to my childhood friends that have helped me from distance, which I know that it is not easy.

I want to thank to my father José María and my mother Magdalena all the faith, guidance and support that always I have received from them. It is impossible for me to remember all the times that their advices and comments help me to choose the right way to do the things in my life. Also, I want to thank the rest of my family, my sister Constança and my grandmothers Francisca and Maria.

Finally, and most important, I would like to thank Amalia for all her love, help, and support that she gives to me each day. I could not imagine this adventure without her, since she is the person who makes me to enjoy the life, even in the worst moments.

Tampere, May 18th, 2013
Francesc de Borja Ramis Ferrer

CONTENTS

1. Introduction 1
   1.1. Background 1
   1.2. Problem Definition 3
      1.2.1. Justification for the work 3
      1.2.2. Problem statement 3
   1.3. Work description 4
      1.3.1. Objectives 4
      1.3.2. Process 5
      1.3.3. Assumptions and limitations 5
   1.4. Thesis Outline 6
2. Theoretical background 7
   2.1. Ontologies 7
      2.1.1. Types of ontologies 7
      2.1.2. Ontology components 9
      2.1.3. Methodologies for design of ontologies 10
      2.1.4. Ontology languages 12
      2.1.5. Protégé 14
      2.1.6. Reasoning of ontologies 14
      2.1.7. Query definition 14
   2.2. Key Performance Indicators 16
      2.2.1. The two top-level metrics 16
      2.2.2. KPI definition and relevance in manufacturing systems 17
      2.2.3. ISA-95 Standard 18
      2.2.4. B2MML 22
   2.3. IEC-61499 23
      2.3.1. FB representation 23
      2.3.2. Composition of FBs 25
3. Methodology approach 28
   3.1. Manufacturing systems modelling system selection 28
      3.1.1. Support tool for OWL 28
   3.2. ISA-95 based implementation 29
   3.3. Required ontologies for thesis work objectives 29
      3.3.1. Interoperability between ontologies 30
      3.3.2. Reasoning 32
   3.4. IEC-61499-based function block network application 32
      3.4.1. Function block engine configurator 33
   3.5. Technologies and architecture review 34
   3.6. Use case definition 35
4. Implementation 37

4.1. FESTO line 37
4.1.1. Assembly process on Multi-FMS 40
4.2. Production ontology 41
4.2.1. Implementation of the production ontology 41
4.2. FBN configuration ontology 48
4.2.1. Requirements of the FBEC supported by the CFBNO 49
4.2.2. Implementation of the CFBNO 50
4.3. Interoperability between MSO and CFBNO 54
4.3.1. MSO classes description for interoperability 55
4.3.2. CFBNO classes description for interoperability 56
4.3.3. Ontologies connection through defined classes 56
4.4. Reasoning ontologies 57
5. Results 60
5.1. Sequence for FBN performance 60
5.1.1. Reasoning of MSO 61
5.1.2. Reasoning of CFBNO 62
5.1.3. Final steps for FBN performance 63
5.2. Resulting FBN for KPI calculation 64
6. Conclusions 66
6.1. Conclusions of implementation and results 66
6.2. Further work 67
References 70
Appendix A – Modelled multi-FMS station components 77
Appendix B – Possible semantic scenario 80

LIST OF FIGURES

Figure 1: Reusability-Usability depending on ontology types..8
Figure 2: Components of ontologies by an example of workstation elements 10
Figure 3: A class hierarchy example...11
Figure 4: OWL layers..13
Figure 5: SPARQL basic query example..15
Figure 6: Simple query for finding subclasses..16
Figure 7: ISA-95 areas 19
Figure 8: ISA-95 levels [46] 19
Figure 9: ISA-95 production elements [45]..20
Figure 10: Function block characteristics [53] 24
Figure 11: Execution model for basic Function Blocks [53]..24
Figure 12: Conveyor test station FBD [53]..26
Figure 13: MSO and CFBNO description..30
Figure 14: Interoperability of ontologies through ISA-95 taxonomy 31
Figure 15: Required domains of experts for defining the ontological approach............32
Figure 16: FBN model generated in the FBEC [40] 33
Figure 17: Architecture for data integration by using the thesis ontological approach ..35
Figure 18: FESTO Multi-FMS on TUT FAST Laboratory facilities [61]......................37
Figure 19: Multi-FMS configuration 38
Figure 20: Venn diagram of MSO classes 42
Figure 21: MSO class hierarchy..43
Figure 22: MSO object properties..44
Figure 23: MSO instances..45
Figure 24: Logic for naming the MSO instances...45
Figure 25: Selected KPIs and formulas...46
Figure 26: MSO data type properties...46
Figure 27: Example of both properties definition in the same instance.........................47
Figure 28: Verifying the MSO design with the *'OntoGraf'* window of Protégé 48
Figure 29: Connection of FBI thought XSL transformation of message types...............50
Figure 30: Venn diagram of CFBNO classes...50
Figure 31: CFBNO class hierarchy..51
Figure 32: CFBNO object properties...52
Figure 33: CFBNO instances 53
Figure 34: CFBNO data type properties 53
Figure 35: Verifying the CFBNO design with the *'OntoGraf'* window of Protégé 54
Figure 36: Interoperability between MSO and CFBNO implementation 55
Figure 37: Query for retrieving information of a FB type 57
Figure 38: Results in Protégé of a query..58
Figure 39: Interactions of the configuration support with the ontologies and user 60
Figure 40: Connections between generated FBIs..63

Figure 41: XML model of the FBN scenario generation 64
Figure 42: Resulting FBN for KPI calculation using the ontological approach 65
Figure 43: Higher implementation [14] assisted by the ontological approach 69

LIST OF TABLES

Table 1: Types of ontologies.....8
Table 2: Possible resulting table by querying a use case.....15
Table 3: Another possible resulting table by querying a use case.....16
Table 4: Examples of KPIs [47].....21
Table 5: Timing phases of the execution model for basic Function Blocks [53].....25
Table 6: Multi-FMS stations [62].....39
Table 7: Example of workpieces assembling sequence in Multi-FMS.....40
Table 8: Requirements of the FBEC supported by the CFBNO.....49
Table 9: First three queries of the sequence for the MSO reasoning.....61
Table 10: Queries of the sequence for the CFBNO reasoning.....62
Table 11: Remaining steps for the FBN implementation.....63
Table 12: Mill 105 station components.....77
Table 13: Robot station (RV-1A) components.....77
Table 14: Parts Buffer station components.....77
Table 15: AFB pallet transport system components.....77
Table 16: Distributing station components.....77
Table 17: Testing station components.....78
Table 18: Handling station components.....78
Table 19: Processing station components.....78
Table 20: Robot station (RV-2AJ) components.....78
Table 21: Assembling station components.....79
Table 22: Storing station components.....79
Table 23: Warehouse (ASRS20) station components.....79

ACRONYMS

AI	Artificial Intelligence
AS	Automation Systems
ASRS	Automatic Storage and Retrieval System
B2MML	Business to Manufacturing Markup Language
BMIR	Stanford Center for Biomedical Informatics Research
BPEL	Business Process Execution Language
CDL	Choreography Description Language
CFB	Composition Function Block
CFBNO	Configuration Function Block Network Ontology
DARPA	Defense Advanced Research Projects Agency
DAML	DARPA Agent Markup Language
DL	Description Language
DPWS	Devices Profile for Web Services
DSS	Decision Support System
EC	European Commission
ES	Enterprise System
ESB	Enterprise Service Bus
EU	European Union
FAST	Factory Automation Systems and Technologies
FB	Function Block
FBEC	Function Block Engine Configurator
FBI	Function Block Instance
FBN	Function Block Network
FBNI	Function Block Network Instance

FOL	First-Order Logic
FMS	Flexible Manufacturing System
FoF PPP	“Factories of the Future” Public-Private Partnership
IEC	International Electro-technical Commission
ISA	Instrumentation, System and Automation Society
IT	Information Technology
JSON	JavaScript Object Notation
KB	Knowledge Base
KPI	Key Performance Indicator
KR	Knowledge Representation
MOM	Manufacturing Operations Management
MPS	Modular Production System
MSO	Manufacturing System Ontology
OEE	Overall Equipment Effectiveness
OIL	Ontology Inference Layer
OWL	Web Ontology Language
PLC	Programmable Logic Controller
QL	Query Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SCADA	Supervisory Control And Data Acquisition
SE	Software Engineering
SOA	Service Oriented Architecture
SPARQL	Simple Protocol and RDF Query Language
SWRL	Semantic Web Rules Language

TEEP	Total Effective Equipment Performance
TUT	Tampere University of Technology
UML	Unified Modelling Language
W3C	World Wide Web Consortium
WBF	World Batch Forum
WS	Web Service
WSDL	Web Service Definition Language
WS-CDL	Web Service Choreography Description Language
XMI	eXtensible Markup Language Metadata Interchange
XML	eXtensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

1. INTRODUCTION

This section exposes a background on the thesis work domain for citing the concepts that any reader of this document should know to understand the objectives, tasks, and results of this Master of Science thesis.

1.1. Background

In the last decades, the industrial automation technologies have been evolving passing by several generations for achieving the actual situation. The main reason of this evolution is the need of using renewed technologies for achieving new goals that cannot be reached by using previous ones [1]. This causes that technologies and manufacturing systems must grow and suffer changes rapidly. Note that this trend not only gets more complicated in terms of fast adaptability to system upgrades, but also due to the market pressure. Systems are nowadays required to be adaptable and efficient since this feature can be directly translated in economic benefits and savings for the industry. In fact, system integration is another factor to take into account in the cost for projects on the manufacturing industry [2, 3]. In addition, [4] indicates that the high cost of integration can be avoided by using distributed intelligence system technologies, so once again the investment and improvement of system integration development is pointed as a priority.

The implementation of Service Oriented Architectures (SOA) paradigms to industrial Automation Systems (AS) has been a tendency in last decade projects [5] because this approach facilitate control systems to be reusable, reconfigurable, and flexible, which are three of the required characteristics for an actual production line, as an example of an industrial control system.

In this evolution, the appearance of standards, as ANSI/ISA-95 [6, 7, 8, 9], has been a basic fact for developers to build modelling and control systems reducing human efforts. Normally, standards are based on semantic languages as the eXtensible Markup Language (XML) that specifies rules for facilitating its codes to be readable for humans and machines. By means of standards, systems get more accessibility and interoperability due to the capability to incorporate devices that does not have same protocols.

The International Electro-technical Commission (IEC) 61499 is a component-based international standard used for modelling distributed systems. IEC-61499 uses a Function Block (FB) as a main element that encapsulates the business logic and can be implemented as software and/or hardware. Then, by using this standard, the architecture for distributed AS can be made by developers as it is shown in [10]. In addition, IEC-61499 models are composed by interconnections of FBs, via event and data flows, configuring a Function Block Network (FBN). A solution based on IEC-61499 standard has been used in the European Union (EU) project called PLANTCockpit. This research project was launched by the European Commission (EC) within the 7th Framework Programme, under the umbrella of “Factories of the Future” Public-Private Partnership (FoF PPP) [11]. This project develops a platform for monitoring and controlling production processes [12]. The integration approach of PLANTCockpit is presented on [13], and its solution can be extended as it is explained in [14]. Supported by a comparison between the conceptual and operational perspective of PLANTCockpit, [14] depicts the functionality of a FBN on an implementation which performs the calculation and monitoring of Key Performance Indicators (KPIs) of a shop floor integration.

Through all of this, actual systems data integration becomes a complex task to be faced by developers in manufacturing systems due to several facts as; the variety of functionality in production lines and the tough process of configuration, derived on controlling different protocols, complexity of Enterprise Service Bus (ESB) solutions [14, 15], and definition of Web Services Description Language (WSDL) [16]. Then, semantic technologies are used in nowadays manufacturing systems projects to standardize a human readable lingo in certain domain, which is also a goal of this thesis work.

During this decade, there has been a noticeable tendency to apply Knowledge Representation (KR) for modularizing the knowledge into domains and decouple business logic from knowledge. KR is part of the various Artificial Intelligence (AI) approaches which makes use of human-readable semantic languages for interoperability and expressivity of systems. This objective is possible by definition of system domain ontologies, using XML-based KR languages and standards as Resource Description Framework (RDF) [17], or Web Ontology Language (OWL) [18]. Research on this field combined with semantic definition, bring closer the distribution and self-management of industrial AS, as the presented ontology-based KR for Flexible Manufacturing Systems (FMS) in [19]. Then, ontologies implementation allows systems to share common KR between heterogeneous devices, by defining the functionalities of the factory model and describing the semantics of overall manufacturing capability [20]. The main objective of this thesis work is to allow a system to configure a network of heterogeneous systems to perform a metric calculation by the use of ontologies.

To sum up, the SOA paradigm facilities the reusability, reconfigurability, and flexibility to industrial AS because they can be modelled by developers within semantic language based standards. This encourages the present projects to develop platforms for control systems, as manufacturing systems, by modelling the systems and integrate to them AI as the most recent tendency. The main goals of these integrations are to remove the Information Technology (IT) knowledge requirements for metric calculation, and lower the integration capabilities down to the production manager level. In addition, the efficiency of production systems and its self-dependency by AI is obtained.

1.2. Problem Definition

Current manufacturing systems are exhaustively controlled and updated to be more efficient than previous ones because of the nowadays production and market demands. This encourages the management of processes metrics for improving the performance of products in the industry.

On the other hand, given that the complexity of systems is increasing, systems must be standardly modelled. Thereby, production systems are endowed with reusability, reconfigurability, and flexibility, among other advantages. To achieve these objectives, SOA and semantic technologies are useful to be utilized for modelling complex systems and data management between different layers of an integration system.

Due to this, the need to define an approach for combining manufacturing models with enterprise integration patterns becomes a must to reach the actual technological goals.

1.2.1. Justification for the work

Present-day industrial control systems are so complex that, in their integration, a language with large expressivity capabilities is required. This description helps to the integration by using the semantic value of the values that each system provides.

By the use of ontologies, a manufacturing system can be modelled. Note that, depending on the domain of the ontology, the model can include several shop floor integration parts in order to reduce complexity of the representation. In addition, ontologies add description to models because they are capable to indicate the relation between different integration layers, even roles of model instances.

Moreover, by using ISA-95 based implementations, [20] presents an operation and KPI metrics assessment doing a KPIs classification by categories. Thus, by using ontologies, equipment of a production line should be able to be related with KPIs, taking into account KPIs category.

Knowing that KPIs are calculations of systems, [14] depicts conceptually how the KPIs can be calculated and visualized by a FBN. As a solution for the composition of services depicted on [14], ontologies could be used for modelling the interconnections between FBs.

Then, the main reason of this thesis work is to use ontologies for supporting the integration of a manufacturing system, because this allows the automatic configuration of a FBN by querying developed ontologies with the objective of manufacturing system KPIs calculation.

1.2.2. Problem statement

Following the problem definition and the justification of the work, there is a need for using manufacturing system models as source of information during the configuration of heterogeneous system interoperability. In addition, these integrations have to be standardized and well defined to approach semantic knowledge to the manager level of a production system. Thus, a methodology for retrieving the information from ontologies is required. All these cited statements, and the ones glimpsed on the previous sections of the thesis introduction, prompt the following questions:

- How to integrate manufacturing models during heterogeneous system integration and metric calculation support?
- How KPI calculation description be assembled in production ontologies?
- How a FBN configuration for KPI calculation can be semantically modelled?
- How to design a generic methodology to query the previous ontologies types?

1.3. Work description

The main task is to integrate a manufacturing model for heterogeneous system integration and metric calculation support. This model has to include the process for describing the extraction of KPIs from an industrial control system and the configuration of its FBN. This means that at least two different, but relational, ontologies are required to be done; while the first one is needed for describing the components of the system and KPI equation definitions, the second one is required for describing how the different FBs of a FBN for the KPI calculation will be configured.

Once the required methodology is developed, the reasoning and data extraction from ontologies must be described and demonstrated on a use case by this thesis work. This means that the needed data to calculate the KPIs of a manufacturing system and the semantic configuration necessary for the FBN configuration has to be queried from the ontologies.

1.3.1. Objectives

By following the previous work description explication, the incoming list presents the objectives of this thesis work:

- Design an ontology schema for modelling manufacturing systems:
  - It has to be generic; the solution must approach the implementation to a standard-base manufacturing system model.
- Develop the two ontologies described on the work description for the different following proposes of a manufacturing use case:
  - For describing the equipment and KPI domains from a production line by using the designed generic ontology schema of previous objective.
  - For describing a FBN semantic configuration.
- Design the interoperability between Knowledge Bases (KBs):

    - To design the interoperability between ontologies to achieve a common goal. In this case, the objective is a KPI calculation.
- Design a reasoning process for retrieving the required data for FBN semantic configuration.

1.3.2. Process

- Study and implementation of methods for manufacturing systems modelling:
    - A research on current trend for manufacturing systems modelling is done. This thesis is purposed to implement an ontological approach, so then ontologies is the selected formalization to be used.
- Selection of a tool for implementing the required ontology models:
    - A search on available ontology platforms is done for implementing the thesis manufacturing system ontologies. Then, languages used by the selected platform are needed to be studied.
- Study, design and implementation of the interoperability between ontologies:
    - A research for the relation between ontologies is studied and designed. Then, an implementation of that interoperability within the selected platform is done.
- Study, design and implementation of the reasoning process:
    - A research for reasoning is done. The process must be compatible with the platform used to model the manufacturing system. Once the reasoning process is defined, its implementation by using required languages is done.
- Experimental study:
    - The experimental study has been performed modelling a module of a FESTO Modular Production System (MPS), as an example of a manufacturing system. This assembly line consists on workstations that have at least one task for accomplish a differentiation and assembly of cylinders. The FESTO MPS is presented in Section 4 of this thesis work documentation.

1.3.3. Assumptions and limitations

The performed work in this thesis is meant to satisfy the stated objectives but the following assumptions must be noted:

- Assumption 1: The configuration of FBN modelled by ontologies, presents results that require tools to be executed.
- Assumption 2: The demonstration of the two methodologies (modelling systems by ontologies and data extraction from ontologies) implementation is done by concrete cases since the large amount of possible KPIs to be tested.
- Assumption 3: A reasoner is needed for process the results of queries. The supporting tool for ontology design can also facilitate this requirement.
- Assumption 4: There is not a starting statement for using any specific software, tool or languages for reach the solution of this thesis work. Besides this free choice, the ontological approach must be usable for the PLANTCockpit [11] data integration implementations [12, 13, 14, 40, 58 and 66].

1.4. Thesis Outline

This thesis work is structured as follows: Chapter 2 presents a theoretical background defining and describing the concepts, technologies and tools used to this work. Then, Chapter 3 presents the methodology approach. Chapter 4 presents the implementation of standard-based ontologies on a Multi-FMS as an example of an industrial manufacturing system. Chapter 5 presents the results of reasoning the modelled system, and the extraction of FBN data configuration for a desired KPI calculation. Finally, Chapter 6 presents conclusions and future work.

2. THEORETICAL BACKGROUND

This chapter is a literature and technology review which defines and describes the concepts, technologies and tools used to this thesis work. Beyond the previous *"Introduction"* section, this chapter explains the essential cited concepts that are in relation with different performed tasks for this thesis work. Note that this information is focused on manufacturing systems, since this is the environment in which the applications of the thesis implementation are planned to be applied.

2.1. Ontologies

As is stated on the *"Introduction"* section of this thesis work, the next generation of manufacturing systems are being implemented with knowledge management tools to approach the AI for developing production processes. For this reason, in the early 1990's ontologies started to be developed as a KR language [21].

Tom Gruber defines ontology as *"an explicit specification of a conceptualization"* [22]. In fact, this term is retrieved from philosophy, which is the study of the nature of existence, its categories and their interrelations. For knowledge-based systems, ontologies describe the existence of things and their relations between them defining objects and their properties, respectively.

Then, the concept of ontology is defined in nowadays systems as a formal representation of knowledge [23]. Actually, ontologies can be used in any domain but this thesis deals with the applications of ontologies in manufacturing systems. A large amount of implementations on this field with a deeper conceptual definition of ontologies can be found in [24, 25, and 26] Master Thesis of Science, developed in the Tampere University of Technology (TUT). These thesis works presents different applications using ontologies as the KR language on the manufacturing domain. In addition, many publications about similar implementations can be found as, for instance, [27 and 28] which presents ontology implementations for flexible and adaptive agents for complex manufacturing systems.

2.1.1. Types of ontologies

There is not only one classification of ontologies since authors categorize them in a different manner. In any case, ontologies can be classified from general to specific as in [29], in a more defragmented classification based on the subject of conceptualization as it is presented in [24], and/or doing a differentiation between the issue of the conceptualization and the content of ontologies as it is depicted on [30].

A classification of ontologies is done by contrasting above cited published classifications is presented in following Table 1:

Table 1: Types of ontologies

Type	**Description**
General/Common ontologies	Represents general/common sense that can be used in different domains as things, events, or functions
KR ontologies	Represents primitives for formalising the knowledge. They are "*a conceptualization of KR formalisms*" [30]
Top/Upper level ontologies	Represents very general concepts. These ontologies are domain independent
Generic domain ontologies	Is composed by specific terminology of a domain. In addition, describe the interrelations between the ontology concepts
Task domain ontologies	Is composed by terminology of certain task or activity. These ontologies are domain dependent
Application domain ontologies	Is composed by required declarations for the KR for certain application. These ontologies are domain and task dependent

Different ontology types can also be analyzed, and some features of this classification can be found as the reusability-usability that depends directly on the ontology type. In fact, Figure 1 is based on a figure shown in [31] which depicts how the usability and reusability in ontologies is indirectly proportional.

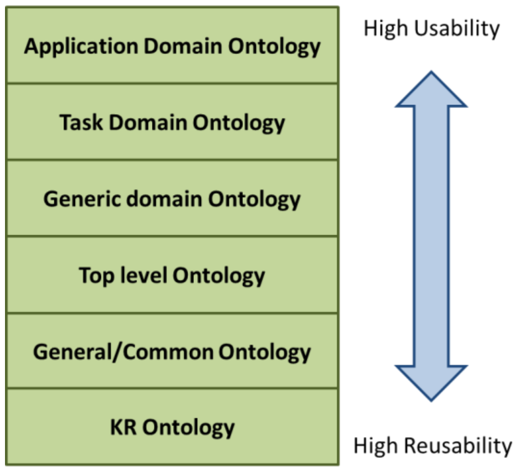


Figure 1: Reusability-Usability depending on ontology types

Note that besides this briefly classification of ontologies, an exhaustive categorization can be found in [31], starting from the evolution of this classification and presenting several instances for exemplify each ontology type.

On the other hand, it has to be stated that one of the main reasons of using ontologies, from explicit specifications, is that they can be applied to any domain and/or sector because anything can be described from general to concrete. As it is explained along this thesis work documentation, the implemented ontologies belong to the industrial manufacturing domain.

Note that the use of different types of ontologies can determine characteristics of the system model as the depicted reusability of the ontology for using it in another domain. This means that general ontologies can be applied in different domains due to their high reusability but, for example, task ontologies cannot be reused since they describe concrete activities in certain domains.

As it can be seen in further chapters, this thesis work presents two different ontology types. The first one is a generic ontology for manufacturing system domain so it is reusable for modelling production lines, but not enough for being used on different domains because as it is described on Table 1, it contains specific terminology for a certain domain. On the other hand, the second ontology is composed by specific terminology of a domain, and describes the interrelations between the ontology concepts for the KR of a FBN configuration task, so that means that is highly usable for this case but it cannot be reused in other domain or distinct task. Then, the relevance of the different ontology types can be understood as a crucial characteristic that determines the domain and scope of implementation of ontologies.

2.1.2. Ontology components

There are five main components of an ontology as it is explained on [32]. Conceptually, (1) expresses an ontology as the composition of concepts, relations, functions, instances, and axioms.

$$O = \{\boldsymbol{C}, \boldsymbol{R}, \boldsymbol{F}, \boldsymbol{A}, \boldsymbol{I}\} \tag{1}$$

- **C**oncepts are compositions of objects and they are organized in taxonomies
- **R**elations are a set of connections between objects.
- **F**unctions represent the defined operations in concepts
- **A**xioms are veritable sentences
- **I**nstances are real elements that belong to an object

Note that these are the ontology components by definition but in following sections can be seen how ontology languages use same type of components but defined differently, following its own Description Language (DL). This means, for example, that an object can be named as class or even an entity, a relation can be defined as a property, and an instance can be defined as an individual, but the above definitions are applied to these “synonyms”.

2.1.2.1 An example of components by using graphical notation

The previous described components can be depicted by the following Figure 2, showing the elements represented by a graphical notation based on the one explained in [33].

Note that functions and axioms are not represented in this example since an axiom is a sentence and functions are descriptions of concept operations.

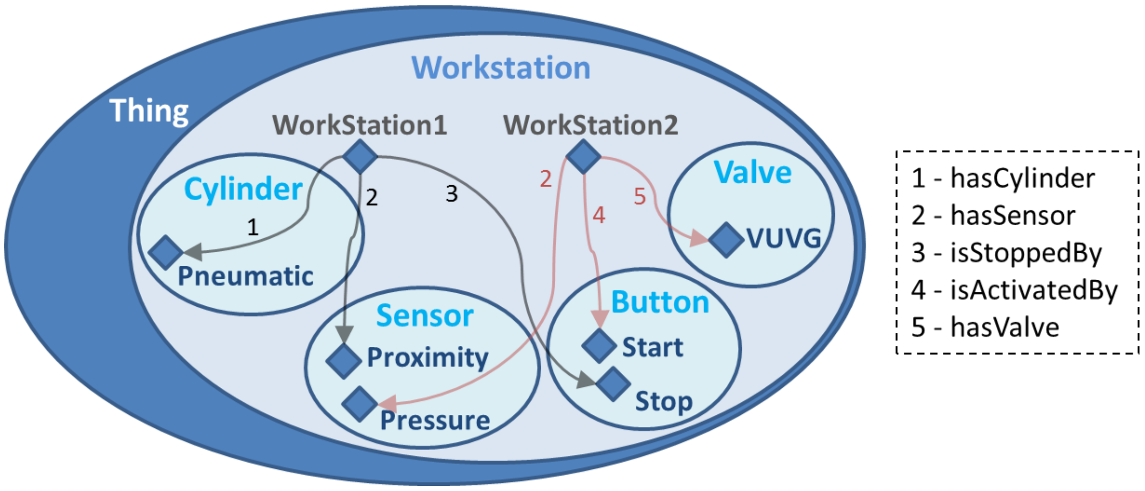


Figure 2: Components of ontologies by an example of workstation elements

Concretely, Figure 2 shows a concept as a composition of objects that describes a hierarchy of other objects, which contains real interrelated elements. Note that the name for the represented concept in the presented example is *'Thing'* since is the usual word used for concepts on several ontology languages, as it can be seen in further sections of these thesis work documentation.

Then, it is demonstrated how a concept is a higher object that includes other objects represented by ovals, instances are real elements of certain objects represented by rhombus, and relations describe type of connection existing between instances represented by arrows.

2.1.3. Methodologies for design of ontologies

Modelling systems is a general task that requires following some sub-tasks in DLs. The use of methodologies guarantees a satisfactory performance of models without missing any system description. A knowledge-engineering methodology for developing ontologies is described in [34] presenting seven steps to design an ontology. This section summarizes each step to notify its relevance. For having a graphical description, the case presented on Figure 2 will be used for exemplifying each step of the methodology. Note that this thesis work has used this methodology for designing the developed ontologies.

2.1.3.1 Domain/scope of the ontology

The first step is the decision of the domain of the ontology. The easier method to achieve the required domain and scope is by answering four questions presented in [34]:

- *What is the domain that the ontology will cover?*
- *For what we are going to use the ontology?*

- *For what types of questions the information in the ontology should provide answers?*
- *Who will use and maintain the ontology?*

Then, by answering the previous questions, the domain of the ontology that is being designed is achieved. By using the Figure 2 case as example, the resulting domain would be the representation of workstation components.

2.1.3.2 Reusing other ontologies

Once the domain and scope is defined, the research of existing ontologies on same domain is a recommended task since, as it is explained in *"Types of ontologies"* section, ontologies permit knowledge reuse. This could help the designer to use the information and/or structure of already designed ontologies because, for instance, if there is any ontology with the same domain of Figure 2 case, it could be merged with that one.

2.1.3.3 Relevant terms of the ontology

Thirdly, the enumeration of the relevant terminology of ontologies is a critical task on ontology development. Once the list is done, the determination of component type in the ontology is defined. This means that the selected terms will be defined in the model as concepts, objects, and relations of the ontology.

As it can be seen in Figure 2, the terminology is composed by all the terms that are written on the picture and each one has a role in the diagram as, for example, *'workstation'* as an object, *'Pneumatic'* as an instance or *'hasValve'* as a relation.

2.1.3.4 Defining objects/classes

Once the terminology is established and the components are identified, the definition of an object/class hierarchy must be done. The result of this task is the organizations of object terms in classes, and sub-classes of classes as is depicted by Figure 3.

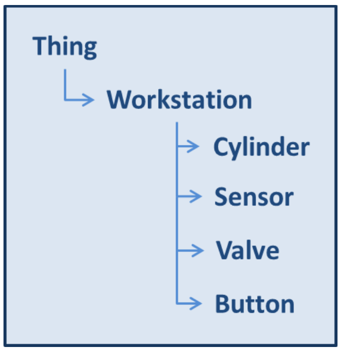


Figure 3: A class hierarchy example

As it can be seen, previous figure is a class hierarchy example using the Figure 2 components. The upper class is *'Thing'* that has a sub-class named *'Workstation'*, which in turn has four sub-classes; *'Cylinder'*, *'Sensor'*, *'Valve'*, and *'Button'*.

2.1.3.5 Defining relations/properties of classes

Fourthly, the relation between classes must be defined. In this step, the objects get the description of their connection between other objects. Note that these relations are adopted by instances, so that in Figure 2 it can be seen properties as *'isActivatedBy'* which indicates that *'Workstation1'* is activated by button *'start'*.

2.1.3.6 Defining additional properties for defined properties

Once relations of classes are defined, the properties or characteristics of those relations must be defined. There are several additional properties that can be defined as domain, range, functional, or inverse. The declaration of these properties is defined in the tool used for modelling a system by ontologies.

2.1.3.7 Creating instances/individuals

The seventh step indicated by [34] is to create the instances. Once again, instances are real elements that belong to an object. This is the last step of the methodology because once all the system is coherent and the classes are well hierarchized and related are able to accept instances for modelling a real use case. In Figure 2 it can be seen how individuals are defined on classes as *'Stop'* on *'Button'* or *'VUVG'* on *'Valve'* which composes a real scenario through the defined object composition. Note that, once the instances are defined, the steps related to properties for classes must be done again but for defining the relations between individuals, named as *'Data Type properties'*, explained in [33].

2.1.3.8 Next steps

Once the steps are accomplished, the result is a modelled system by ontology. From the last step, the objective is to enrich the ontology with all the use case instances and test real situations. This task will for sure force the definition of new relations and/or even a restructuring of the class hierarchy.

This process of ontology improvement can help developers to minimize the amount of classes and avoiding redundancy on modelled systems among other problems that a modelled system can have.

Summing up this section, it has been demonstrated the importance of following a methodology for developing ontologies. As it is remembered on [34], there is no one 'correct' manner to design ontologies but this general steps are the basics to achieve a satisfactory result, and it has been the way to go on the performance of ontologies on this thesis work.

2.1.4. Ontology languages

Throughout ontology evolution different types of ontology languages have been performed due to the high develop of ontologies application on Software Engineering (SE), AI, and semantic web. This produces a large amount of choices to use for ontology definition and several ontology languages reviews are done as in [35], which presents a classification and main features of existing frequently used languages. The decision of the ontology language is a crucial task because of the differences between language types. This does not means only differences of semantics, but also means that there are languages that have support tools to implement them as OWL in Protégé. As it can be seen in this document, the performed ontologies on this thesis work are described by OWL.

2.1.4.1 OWL

OWL is a semantic markup language derived from the combination of the Defense Advanced Research Projects Agency (DARPA) Agent Markup Language and Ontology Inference Layer (DAML+OIL) and it is a language based on XML, XML Schema, RDF and RDF Schema (RDFS). OWL is used for ontology definition and is a recommendation of the World Wide Web Consortium (W3C). OWL is divided in three layers; OWL Full, OWL DL, and OWL Lite, as it is explained in [31], discussed in [23], and depicted by following Figure 4:

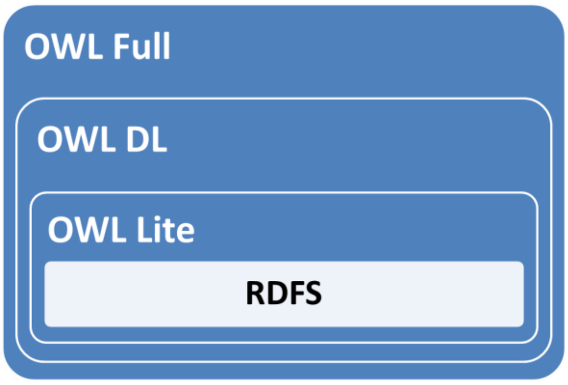


Figure 4: OWL layers

From bottom to top of Figure 4 and following the description given in [31], OWL Lite is an extension of RDFS and is meant to be used by developers that needs to design basic taxonomies with simple constraints due to be composed for the most general features of OWL. Secondly, OWL DL is composed by the entire OWL terminology, described on [18]. Thirdly, OWL Full is the higher part of the OWL layer distribution since it allows more flexibility than OWL DL because allows more primitives as it is described on [31].

The description of OWL is fully covered and revised in [18]. In addition, an analysis of this language and others is described on [31]. Note that there are other languages for ontology development as the use of the Unified Modelling Language (UML) [36] because it is widely used on academic and industry fields. The main reason for working with OWL instead of UML on this thesis work is because UML is a modelling language for object oriented software artifacts; meanwhile, OWL is a notation for KR which is the precise modelling language type that is required by this work implementation. Note that besides this explanation, the support tool decided to use for the performance of ontologies in this thesis work has been Protégé and this tool permits the user to develop ontologies in RDF/OWL format.

2.1.5. Protégé

As it has been stated by previous sections, Protégé is the support tool used for the ontologies definition in this thesis work. Protégé is an open source ontology editor and a KB framework [37] and was developed by the Stanford Center for Biomedical Informatics Research (BMIR) at the Stanford University School of Medicine. Protégé project webpage offer a large amount of manuals and tutorials for designing ontologies by using this tool and latest version can be freely downloaded.

Note that implemented ontologies of this thesis work have been done by Protégé 4.3 that is the actual version of this tool. Besides Protégé project documentation [33] is a useful manual for learning how to use Protégé 4 and to understand OWL language implementation within this tool.

2.1.6. Reasoning of ontologies

As it is explained on this documentation, ontologies are developed to be a KR language. This means that ontologies must be used for reasoning the knowledge that it is described by the defined information. For that reason, there is a need to have something capable to reason the data provided by an ontology.

A reasoner is an entity that is able to give logical results by asserting rules or axioms to an ontology. Reasoners are also known as reasoning and/or rule engines. There are many different semantic reasoners and each one have different features [38]. Then, reasoning engines can be differentiated by OWL DL entailment, support of expressivity for reasoning, availability of reasoning algorithm and ontology consistency checking among other features [38].

This thesis work has used Pellet for reasoning the implementation performed on Protégé 4.2. The results given by a reasoner are descriptive but one of the problems that have to be faced is the managing of the resulting data. This thesis work is done in connection with another project as it is explained on [40] in which an interface manages the data retrieved by the reasoner.

2.1.6.1 First-Order Logic

First-Order Logic (FOL) is a formal system with a powerful expressiveness used in computer science for representing semantics and reasoning. The automated reasoning is a nowadays demand and [41] discusses this fact and presents how to implement FOL.

FOL syntax is composed for seven basic elements; constants, predicates, functions, variables, connectives, the equality and quantifiers. Within these elements, atomic and complex sentences can be defined and its veracity or falsity respect to a model can be determined.

2.1.7. Query definition

Queries can be determined as expressions defined by following a semantic specification of a Query Language (QL), which are understandable by a reasoner so it can reply by giving a coherent result with the ontology description. Thereby, a QL is a language used to express queries for reasoning and achieving a result based on the knowledge information contained by the ontology.

2.1.7.1 SPARQL

The most used QL for querying RDF-based ontologies is the Protocol and RDF Query Language (SPARQL) [39]. The queries defined within SPARQL consist in triple patterns permitting the reasoning of the ontologies.

SPARQL-DL can be defined as an extension to SPARQL according to [25] and it offers more expressiveness than SPARQL. In addition, note that SPARQL Update [42] is another QL defined as a companion language to SPARQL. With the objective to show the structure of a simple SPARQL-based query, valid also for SPARQL-DL and SPARQL Update, Figure 5 shows a basic query:

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

Figure 5: SPARQL basic query example

The query presented on Figure 5 shows an example of query which determines all the existing triples in any OWL ontology. This means that, within the presented query, a reasoner is capable to find all the possible triples of the ontology and gives a resulting table composed by three columns; *'subject'*, *'predicate'*, and *'object'* which shows the relations between all the ontology components. Table 2 shows a possible resulting table of the Figure 2 use case with the Figure 5 query. Note that this is an example and the use case could have more defined matches that do not appear on the table.

Table 2: Possible resulting table by querying a use case

subject	predicate	object
WorkStation1	hasCylinder	Pneumatic
WorkStation1	isStoppedBy	Stop
WorkStation1	hasSensor	Proximity
WorkStation2	hasSensor	Pressure
WorkStation2	isActivatedBy	Start
WorkStation2	hasValve	VUVG

Moreover, there are so many possibilities by querying as, for instance, bounding the previous explained results, by using a RDF primitives instead of *'predicate'* variables. Then, the resulting table only is composed by matches that confirm the used primitive as a property between *'subject'* and *'object'*. An example of a RDF primitive use by defining first a *"PREFIX"* is depicted in following Figure 6.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?subject ?object
WHERE
{
?subject rdfs:subClassOf ?object
}
```

Figure 6: Simple query for finding subclasses

Hence, using the query presented on Figure 6, the resulting table is composed by two columns in which each *'object'* is defined as a lower class of a *'subject'*. Note that the functions of the primitives can be also checked on the *'PREFIX'* URL. In this case, as it is explained, *'subClassOf'* expresses that the subject is a subclass of a class. Then, Table 3 shows a possible resulting table of the Figure 2 use case with the Figure 6 query. Note that this is an example and the complete use case could have more defined matches that do not appear on the table.

Table 3: Another possible resulting table by querying a use case

subject	**object**
Workstation	Thing
Cylinder	Workstation
Sensor	Workstation
Button	Workstation
Valve	Workstation

In addition, special patterns as *'FILTER'* used for filtering results, *'DISTINCT'* distinguishing between duplicated results, or *'LIMIT'* for delimiting the number of results, can be utilized and [39] gives a complete list of all the possible uses on SPARQL. The use of these patterns allows developers to do more complex queries for reasoning modelled systems. Note that SPARQL is the QL used for reasoning the ontologies of this thesis work.

2.2. Key Performance Indicators

Effectiveness control of manufacturing systems processes is an essential study for generate better production outcomes. In 1960, Seiichi Nakajima developed a hierarchy of metrics called Overall Equipment Effectiveness (OEE) for evaluating the quality of manufacturing operations. Over the years, the measure of metrics has been truly important and has become an essential part of market strategies for companies.

2.2.1. The two top-level metrics

Total Effective Equipment Performance (TEEP) is a measurement used for reporting the performance of a system. In fact, in [43] it is explained that TEEP and OEE form the two top-level metrics that indicate the efficiently use of facilities, time and material of manufacturing operations. Then, while OEE evaluates the realization of a manufacturing unit, TEEP measures the effectiveness of OEE against time, i.e. hours per day.

2.2.1.1 OEE and TEEP calculations

In addition, four types of measurements compose the four underlying metrics that indicates the gap between OEE and TEEP [43]. These metrics are; loading, availability, performance, and quality.

Firstly, loading is a part of the TEEP metric which represents in percentage the calendar time that, in reality, is scheduled by the operation doing a comparison between the scheduled time and the calendar time, as it is defined by (2). Secondly, availability is a part of the OEE metric which represents the uptime of the operation, comparing its available time with the scheduled time, as it is described by (3). Thirdly, performance is a part of OEE metric that represents the percentage of the working speed, by doing a comparison the actual rate between the standard rate of operation, as it is presented by (4). Fourthly, quality is a part of OEE metric that represents the percentage of well-produced units, comparing the successfully processed units with the started units, as it is shown by (5).

$$Loading = \frac{Scheduled\ Time}{Calendar\ Time} \tag{2}$$

$$Availability = \frac{Available\ Time}{Scheduled\ Time} \tag{3}$$

$$Performance = \frac{Actual\ Rate}{Standard\ Rate} \tag{4}$$

$$Quality = \frac{Good\ Units}{Units\ Started} \tag{5}$$

As it is explained previously, the four underlying metrics are parts of OEE and TEEP because these top-level metrics are calculated with their values as it is demonstrated on the following calculations (6) and (7).

$$OEE = Availability\ \times Performance\ \times Quality \tag{6}$$

$$TEEP = loading\ \times OEE \tag{7}$$

2.2.2. KPI definition and relevance in manufacturing systems

KPI is a variable used to evaluate the success in manufacturing processes. Companies utilize sets of KPIs for analysing, tracking and evaluating different operations. Moreover, KPI is how OEE measurement is commonly used; consequently KPI can be also defined as a success indicator of a system.

The magnitude of controlling KPIs in manufacturing systems lies in the connection that exists between different layers of an Enterprise System (ES). This means that an indicator that is important on the business level of an ES can depend of another which comes from a machine operation in the shop floor. These dependencies between KPIs appear in large amount of situations through systems.

Thereby, organizations invest hugely on controlling KPIs and defining new indicators to improve manufacturing systems and upgrade processes and equipment to achieve a better production, hence better earnings for the companies.

By visual management and with the objective of tracking processes operation for production lines, [44] presents how several KPIs can be displayed reporting critical information of the system processes in real-time. In addition, seven production KPIs as common indicators used by organizations are presented on [44]. These KPIs are; count, reject ratio, rate, target, stroke time, OEE, and downtime. Then, note that there is a wide variety of KPIs that can be used for evaluating and tracking any manufacturing system.

2.2.2.1 Managing KPIs in manufacturing systems

By the previous introduction about KPI relevance and the description of OEE and TEEP calculations, it can be guessed that there is a huge quantity of KPIs that are used for evaluating manufacturing processes. Definitely, lots of indicators can be treated because of the large amount of operations that are performed in production lines processes and, also, for the commented relation that KPIs can have between levels of an ES.

Next sub-sections show how, due to the large amount of KPI that can be defined in manufacturing systems, the use of standards as ISA-95 [6, 7, 8, and 9] is recommended for generalize the process and reduce effort of KPI definition.

2.2.3. ISA-95 Standard

ISA-95 is a standard developed by an ISA Committee of volunteer experts, divided on parts described in [6, 7, 8, and 9], performed to provide guidance to developers for the implementation on ES. Then, as it is explained in [45], ISA-95 describes a complete functional model for enterprise control use.

The ISA-95 parts can be grouped in three main areas [46], which are listed below and depicted on Figure 7.

- Composed by first, second and fifth part, the first area is formed by exchange information models between business logistics systems and manufacturing operations systems.
- Composed by third part, the second area is formed by activity models in manufacturing operation systems
- Finally, composed by fourth and sixth part, the third area is formed by exchange information models with manufacturing operations systems.

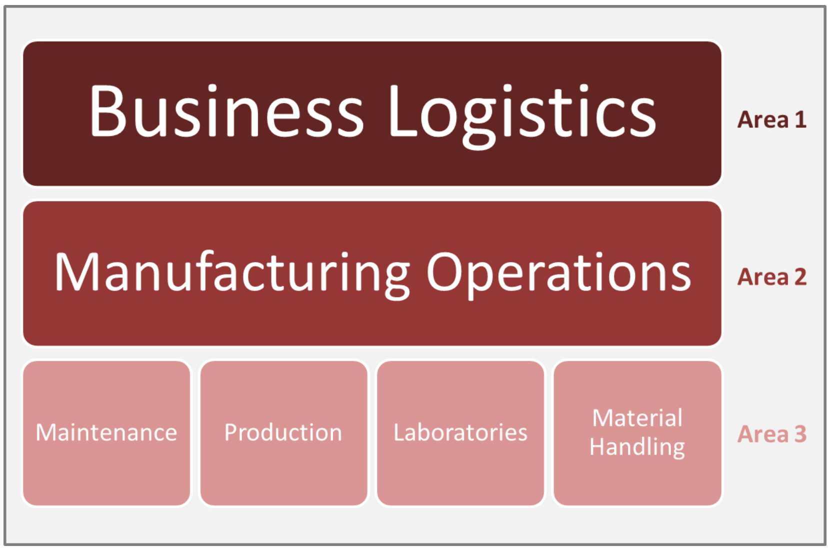


Figure 7: ISA-95 areas

Then, in the first area, ISA-95 is commonly known as a multi-layer stack in which a hierarchy of ES activities is described. In fact, the ISA-95 levels are shown in following Figure 8. Note that this corresponds to the first, second and fifth parts because depicts the integration of business of manufacturing systems [46]. Note that in previous Figure 8 the description of each layer is added at the right side of each level.

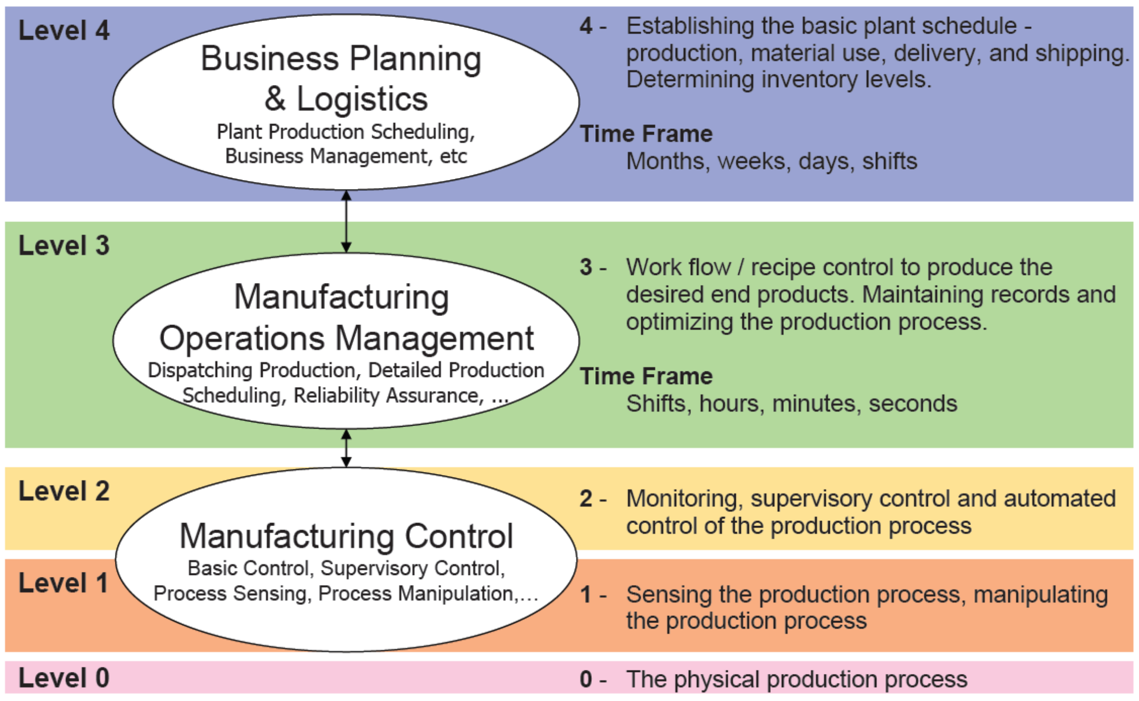

Level 4
Business Planning & Logistics
Plant Production Scheduling, Business Management, etc
4 - Establishing the basic plant schedule - production, material use, delivery, and shipping. Determining inventory levels.
Time Frame
Months, weeks, days, shifts
Level 3
Manufacturing Operations Management
Dispatching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

Figure 8: ISA-95 levels [46]

The second area of ISA-95 defines the activities that occur in Manufacturing Operations Management (MOM) [46]. Figure 9 depicts the production elements of ISA-95.

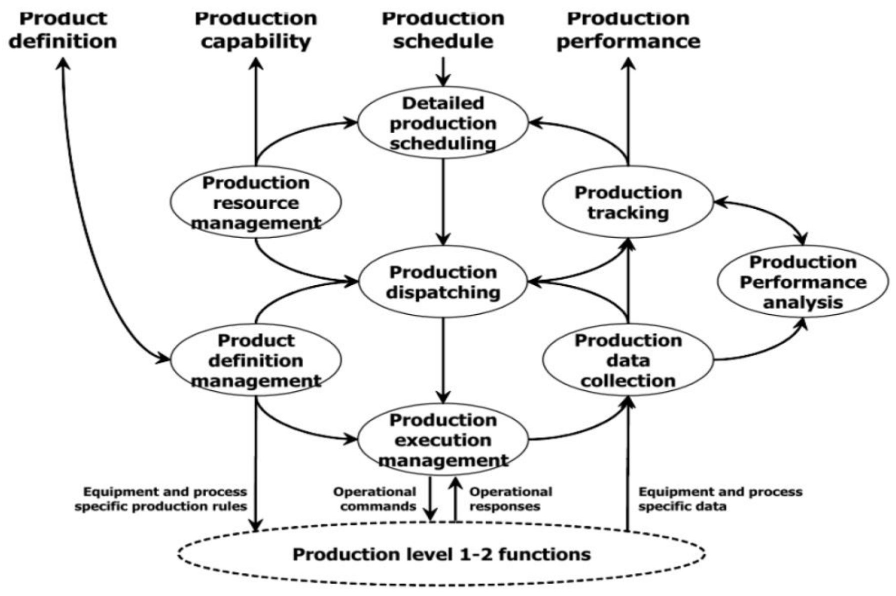

Product definition
Production capability
Production schedule
Production performance
Detailed production scheduling
Production resource management
Production tracking
Production Performance analysis
Production dispatching
Product definition management
Production data collection
Production execution management
Equipment and process specific production rules
Operational commands
Operational responses
Equipment and process specific data
Production level 1-2 functions

Figure 9: ISA-95 production elements [45]

Note that without going out from the scope of this thesis, the KPI management and metric calculation is a work process segment defined in part 4 of ISA-95 which is included in the Production Performance Analysis depicted in Figure 9 and shown in [46].

2.2.3.1 ISA-95-based operations and KPI metrics

As it is explained through this section, KPIs are used for evaluate the production performance of a manufacturing system and the ISA-95 standard is used for provide guidance to developers for the implementation on ES. Then, matching this two statements, [47] explains using ISA-95 standard a KPI description and analysis for its aggregation on service supply chain models.

This thesis work has used the implementation of some ISA-95-based KPIs described on the Table 4, which information can be found in table 3 of [47].

Table 4: Examples of KPIs [47]

Category	KPI
Order Fulfilment	Actual production rate as a percentage of the maximum capable production rate
	Percentage of lots or jobs expedited by bumping other lots or jobs from schedule
	Production and test equipment set-up time
	Production schedules met (percentage of time)
	Actual versus planned volume
Asset Utilization	Average machine availability rate or machine uptime
	Percentage of tools that fail certification
	Hours lost due to equipment downtime
	Cumulative count of machine breakdown
Quality	Major component first-pass yield
	First product, first pass quality yield
	Reject or return rate on finished products
	Reject-rate reduction
	Rework-repair hours compared to direct mfg. hours
	Scrap and rework as a percentage of sales
	Scrap and rework percentage reduction
	Rework and repair labor cost compared to total manufacturing labor cost
	Number of process changes per operation due to errors
	Number of training days
	Yield improvement
Personnel	Percentage increase in output per employee
	Percentage unplanned overtime
	Safety and Security incidents
	Percentage of operators with expired certifications
Productivity	Percentage of assembly steps automated
	Percentage reduction in manufacturing cycle time
	Productivity: units per labor hour
Engineering	HMI data entry count
	Percentage of alarm reduction
Material	Time line is down due to sub-assembly shortage
	Count of supplier shortages per period
	Material consumption variances from standards
Planning	Percentage reduction in component lot sizes
	Manufacturing cycle time for a typical product

As it can be seen in previous Table 4, KPIs are classified by category which allows developers to differentiate between distinct metric types to define them on the right ES level. This means, that not all the KPIs are analysed on same levels of a system. For instance, the *'units per labor hour'* KPI is categorized as *'productivity'* metric and is analysed on the shop floor, meanwhile a *'time required to incorporate engineering changes'* KPI is categorized as a *'planning'* metric and is analysed on the ISA-95 MOM level.

Note that due to the above explanation it could be interesting and highly useful an ontology KPI hierarchy model describing the connections and dependencies of KPIs through different levels of an ES. In fact, this one conclusion of this thesis work and it is purposed and discussed in the *"Further work"* section of this documentation.

2.2.4. B2MML

The Business to Manufacturing Markup Language (B2MML), developed by the World Batch Forum (WFB), is an XML-based implementation of the ISA-95 standards family. B2MML is composed by a set of XML schemas which permits the data models implementation in ISA-95 [48].

Nowadays, B2MML is widely used by industry organizations for the business logics system integration as, for instance, in Enterprise Resource Planning (ERP) integration. The reason of B2MML use by the community has been caused due to be a complete implementation of ISA-95 and, as it is explained in previous sections, because this standard offers a large amount of advantages for system integration on manufacturing systems.

The WFB, in combination with MESA International, supports freely documentation and XML schemas information of B2MML [49]. Some examples of B2MML implementation by using supported descriptions and schemas by [49] are presented in [50, and 46]. In fact, [45] uses a procedure for implementing B2MML with a developed tool consisting, if needed, in applying the required modifications to the cited XML schemas supported by WBF for the use case in which is going to be used, and fill the schema elements and attributes with the values of, in this case, the manufacturing system.

2.2.4.1 B2MML use from ontologies

The implementation presented on this thesis work does not present any B2MML integration because is not the proposed topic and, mainly, because B2MML is implemented from UML models. On the other hand, it has been important to research about B2MML utilizations for further implementations. In fact, a possible manner of working with B2MML from ontologies has been found due to the existence of several transformation tools as the one presented in [51]. These tools allow the transformation of OWL models to UML models.

OWL2XMI is a Java project which facilitates the creation of UML classes from OWL ontologies [51]. However, this project has some limitations because, since OWL is a more descriptive model than UML, the transformation cannot be totally possible.

Beside these limitations, this tool performs the transformation of an OWL file to an XML Metadata Interchange (XMI) file that can be imported by UML tools as, for instance, ArgoUML open source [52].

2.3. IEC-61499

The IEC-61499 is a component-based international standard used for modelling distributed systems and AS [53], since its architecture [54] offers multiple solutions for the actual industrial control systems challenges. Note that this standard is the successor to IEC-61131.

The need of developing the IEC-61499 standard was caused by the evolution of autonomously and intelligence of devices that are integrated on current AS. This means that actual industrial control systems are forced to manage large amount of data flow among system components and within IEC-61499 a model of the information stream between devices is feasible. Then, it can be stated that the main reason for design distributed AS by IEC-61499 is because is a solvent standard for describe the entire system, including its business control logics and the interactions that success on it [55].

This section discusses how IEC-61499 performs a model of AS, describing the main components and functionality of the standard. In addition, an IEC-61499-based implementation is presented since this thesis work development has been done to deliver information and support this application. Note that [53] is a recommended literature for a complete description of IEC-61499 models and concepts.

2.3.1. FB representation

The main design component of IEC-61499 is a FB that encapsulates business logic permitting a modular solution for distributed systems modelling. Following the basic concepts explanation of IEC-61499 described in [1], this standard is event-driven, which means that the FB invocation is based on events. Thereby, the data flow advances through the model depending on the succession of events. The FB structure and characteristics, as the cited event driven functionality, are presented by following Figure 10.

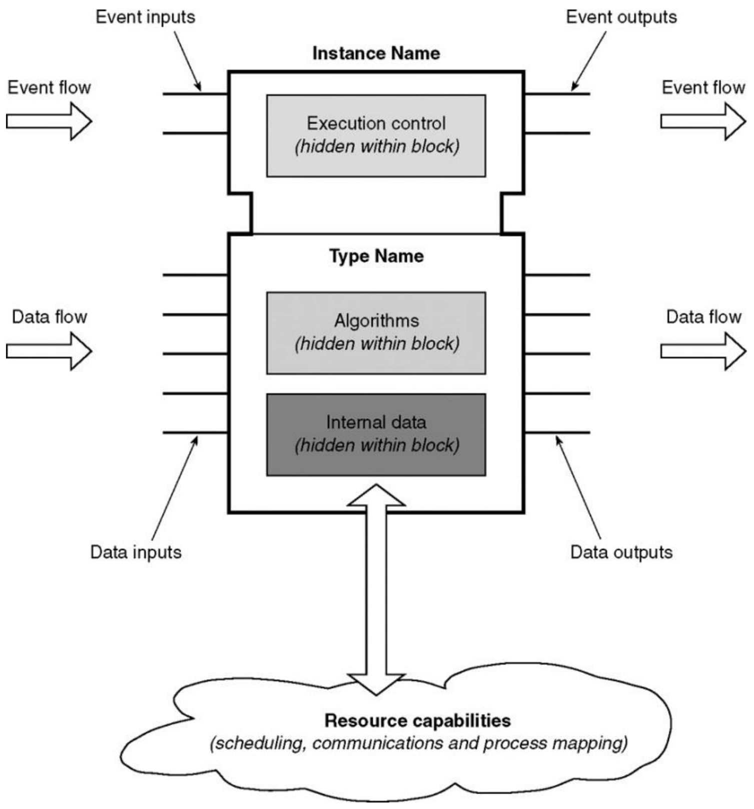


Figure 10: Function block characteristics [53]

Once the characteristics of a basic FB are shown, Figure 11 depicts the execution order by numbered timing phases and Table 5 specifies the task being performed in each time.

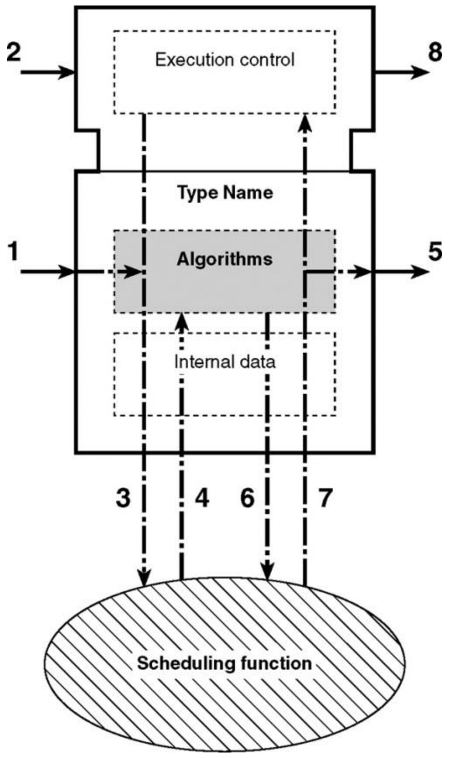


Figure 11: Execution model for basic Function Blocks [53]

Table 5: Timing phases of the execution model for basic Function Blocks [53]

Time	Description
1	Incoming values from other FBs are waits at the FB inputs
2	Input values associated event arrives to event input
3	The execution control of the FB indicates to the Scheduling Function the arrival of input values and that it is prepared for its algorithm execution
4	After loading and performance of resource characteristics, the algorithm of the FB is started by the Scheduling Function
5	Input values and, if needed, internally stored values are processed by the algorithm for the creation of output values written to the FB outputs
6	The execution is completed by the algorithm so the Scheduling Function is informed that waiting output values are prepared
7	The Scheduling Function invokes the FB execution control to generate an output event, which depends on the arrived input events and the execution control state
8	The execution control creates an output event at the FB output event interface. Downstream FB uses the generated event to indicate that the output values generated by this FB can be used

Note that FBs can be implemented as software and/or hardware so that a Function Block Instance (FBI) can be understood also as an industrial control system service. By applying this statement, large amount of data integration projects can also be performed. For exemplify this type of IEC-61499-based integrations, [56] presents an utilization of the standard for semantic description to automation objects for automatic discovery of FBs and composition of applications.

Obviously, a single FB which follows the features described by previous figures is just a part of an integration for accomplish goals of the cited projects. Next section describes how, by the performance of compositions, those solutions can be achieved.

2.3.2. Composition of FBs

IEC-61499 offers the Composite Function Block (CFB) as an upper FB level which contains several FBs that perform a sub-task. This means that CFBs allow the encapsulation of processes or complex functions. In other words, [53] describes that the internal behaviour of a Composition FB (CFB) is performed by a FBN, which is a network formed by FBIs. By using composition of FBs, manufacturing systems applications as a lifter control of a two level conveyor line described in [57] can be modelled. Then, the FBN must define also the data and event connections for the stream of data between FBIs. A possible structure of a FBN composed by FBs is depicted in following Figure 12, which shows a Function Block Diagram (FBD) with the required connections between FBIs for a conveyor test station.

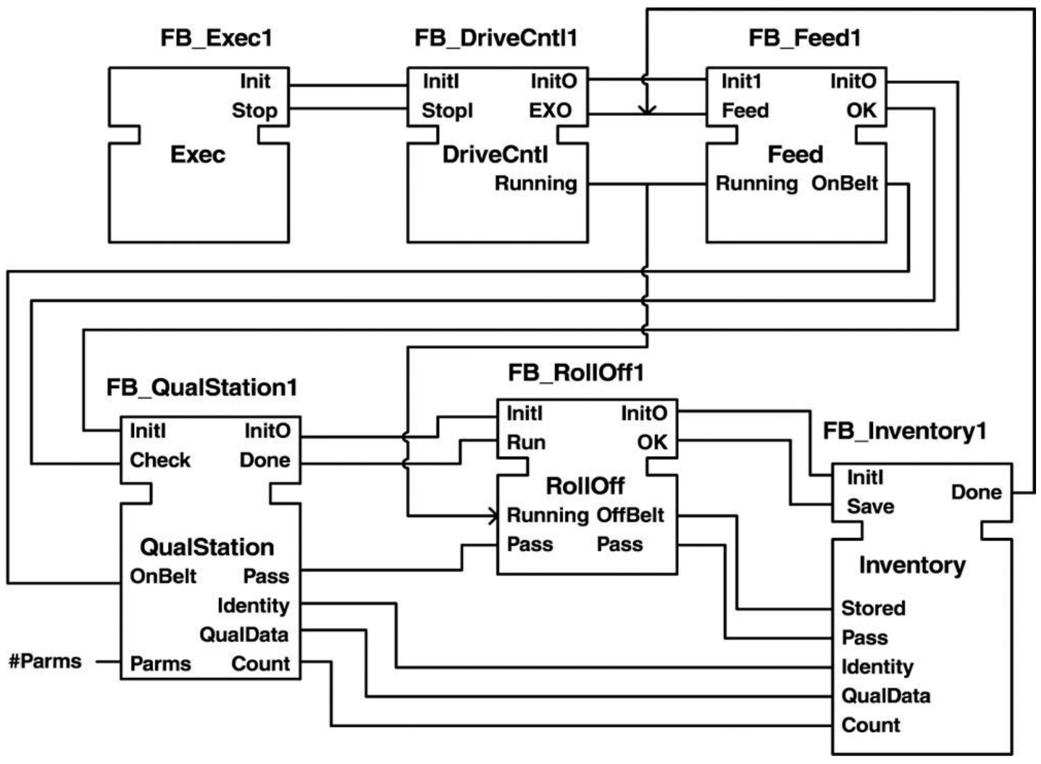


Figure 12: Conveyor test station FBD [53]

Note that this review of IEC-61499 is only describing the general concepts and structures for the implementation of this standard. This means that for understanding the functionality of the previous FBD, shown in Figure 12, it is recommended to consult the referenced literature [53] in where the specifications of the use case are explained.

Concluding, by the use of FBNs and its encapsulation into CFBs, the reusability of components is increased permitting re-configurability of model systems reducing the extension of implementations [58]. However, as is discussed on [58], there is not any semantic description defined on the CFBs. This is a drawback of the standard because it would permit to infer whether FB is compatible by their role or behaviour with other FBs. This type of semantic description is offered by Choreography Description Languages (CDL) as the Web Service CDL (WS-CDL) [59].

2.3.2.1 Choreography on manufacturing systems

In manufacturing systems, the modelling of service compositions is a need because of the large amount of participants for data exchange in a system. In fact, in actual Web Service (WS) integration in manufacturing systems, architectures for describing networks are getting complex and needs control.

The objectives of CDL are to describe the interactions that success between services and do it by performing a decentralized architecture. Firstly, each participant must have a definition of its features as role, type, behaviour. Secondly, the exchange of information is decentralized, meaning that the exchange of information is performed without an orchestrator as a controller, as it happens by using Business Process Execution Language (BPEL), or following a predefined event-driven data flow as happens in IEC-61499 implementations.

In fact, the choreography architecture is possible due to the description of CDL since with it, the participants of a, for instance, WS composition have established their connections with other services of the network, message types, sources, destinations, roles, even descriptions that specifies its behaviours. However, CDL are not execution languages so they need tools support to perform its powerful descriptive capability [58].

3. METHODOLOGY APPROACH

This section describes the selection of technologies and processes for the design of the ontological approach implemented in this thesis work. Therefore, the reasons for the utilization of, for example, concrete tools and languages are explained. In addition, in order to confine the ontological approach scope utilization, this section summarizes and exemplifies the architecture of a system using the ontological approach.

3.1. Manufacturing systems modelling system selection

According to the *"Introduction"* and *"Theoretical background"* sections, KR languages are being used on current manufacturing systems integrations by using ontologies, allowing the implementation of AI for automated reconfigurability, flexibility, and discovering of new system devices, among other benefits.

Then, as it is also introduced by the thesis title, ontologies have been selected for modelling the tested manufacturing system case on this work. After a research in ongoing trends of modelling systems, and the performance review of cited projects along this thesis documentation, the procedures and implementation styles of ontologies have been studied as they are described on *"Ontologies"* section.

RDF, RDFS and OWL are standards used for system representation and exchange of data on, for instance, manufacturing system integrations. For this thesis work, OWL has been the decided language for being used due to its strengths as; support for information integration, separation of syntax from data modelling, support for inference engines as Semantic Web Rules Language (SWRL) and SPARQL, and flexible representation between others presented on [40] and [60].

3.1.1. Support tool for OWL

Once the OWL use has been decided, an ontology editor to support the ontology language implementation must be selected. As it is announced on *"Protégé"* section, the determined tool to be used for supporting the required system modelling of this thesis work has been Protégé.

In fact, the presented implementation has been done with the last release on April 2013, which is Protégé 4.3 available on [37]. Protégé is an open-source platform that is capable to support the modelling system process and knowledge-based ontology applications, allowing managing ontologies and the visualization of its structure.

In fact, Protégé includes an OWL editor which permits the creation of semantic web ontologies, especially OWL, making this platform a right choice for the modelling of manufacturing systems by ontologies.

In addition, the required ontology reasoning has been decided to be performed with SPARQL queries since this is the right QL for query OWL ontologies as it is explained in *"SPARQL"* section and that is a language also supported by Protégé platform.

3.2. ISA-95 based implementation

According to the *"ISA-95 Standard"* section, ISA-95 standard has been performed to provide guidance to developers for the implementation on ES. In fact, manufacturing systems can be modelled within this standard following the UML diagrams that are defined by the standard as is described and presented, for instance, in [45].

Then, it can be seen that a specific terms are used for the description of models. As it is explained in the *"Ontologies"* section, the design of ontologies requires the definition of taxonomies for different domains.

This means that, by linking ontologies and the ISA-95 standard, an efficient option for modelling a production system is to design production ontologies using the ISA-95 taxonomy. This usage reduces the effort of developers to create new terminology for systems description and, also, makes the system understandable for other integration parts of the system and users that utilizes the same standard.

3.3. Required ontologies for thesis work objectives

Once the technologies for modelling manufacturing systems have been selected, the needed ontologies must be decided. According to the thesis work objectives and the problem statement explanation, two different domain ontologies have to be designed.

Meanwhile the Manufacturing System Ontology (MSO) is a generic model for shop floor integrations, which describes the supply chain model of a production system and has a description of KPI calculation following ISA-95 standard; the Configuration FBN Ontology (CFBNO) is a generic model, which describes the required configuration needed KPI calculation performance by the IEC-61499-based interface application presented on *"IEC-61499-based function block network application"* section.

For sketching the information that MSO and CFBNO needs to describe, following Figure 13 shows the ontologies and their information. In addition, note that the ontologies are based on concepts and requirements of PLANTCockpit project as it is described in this section and stated as a duty of this thesis work implementation in *"Assumptions and limitations"* section.

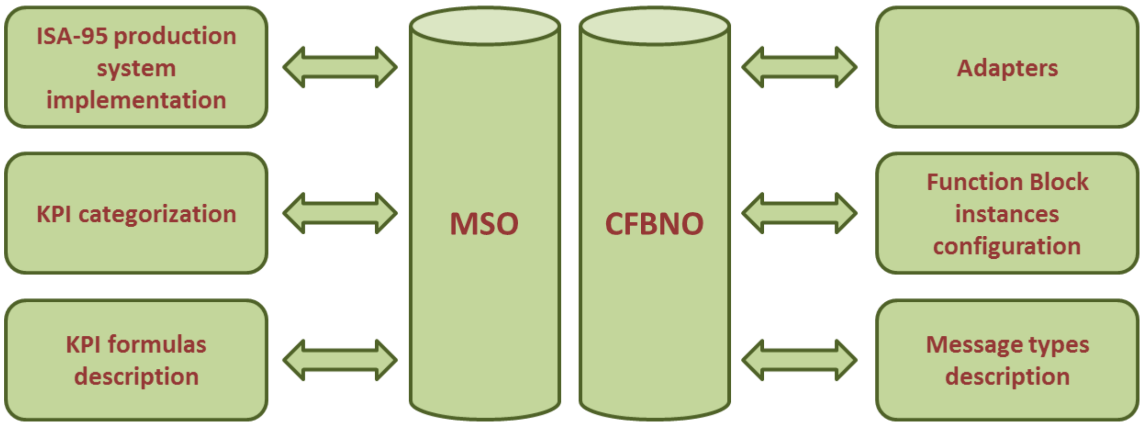

ISA-95 production system implementation
KPI categorization
KPI formulas description
MSO
CFBNO
Adapters
Function Block instances configuration
Message types description

Figure 13: MSO and CFBNO description

On one hand, the MSO describes the manufacturing system and categorizes KPIs following the ISA-95 standard and describes the formulas for KPI calculation. On the other hand, the CFBNO describes the required adapters for retrieving the information model of system components and defines the FBIs configuration and the message types for the configuration of the FBN.

Note that these ontologies are defined as different domain ontologies because they have different application tasks, and usage. Besides this variance of scope, they are used for achieving a common goal which is the FBN configuration for KPI calculation. For this reason, there is the need of interoperability between MSO and CFBNO, being a critical characteristic of the performed implementation in this thesis work.

3.3.1. Interoperability between ontologies

The interoperability between ontologies is understood as the cooperation of both KR systems. This characteristic of the ontological approach is fundamental for communicating the required data to a FBN Interface (FBNI) for configuring the FBN of a manufacturing system for KPI calculation. Therefore, a connection strategy for performing the critical link between ontologies it has to be found.

The wide variety of devices on current manufacturing systems causes the complexity of data exchange due to the diversity of protocols of each component. Adapters are elements of integration systems that allow managing the information model and incoming message types from devices.

Meanwhile the components of production systems represented by MSO produce values that are used by KPI formulas; the adapters represented by CFBNO manage the components information model. This means that the data represented in both ontologies can be used as the link for its interoperability.

Then, it has been concluded that, taking the advantage of the ISA-95 standard use of both ontologies, each ontology data can be reasoned independently but linked with other ontology data because of the use of same taxonomy. This connection is depicted by following Figure 14. Note that this representation does not cover all the ontologies content, since is just an example for explaining the interoperability.

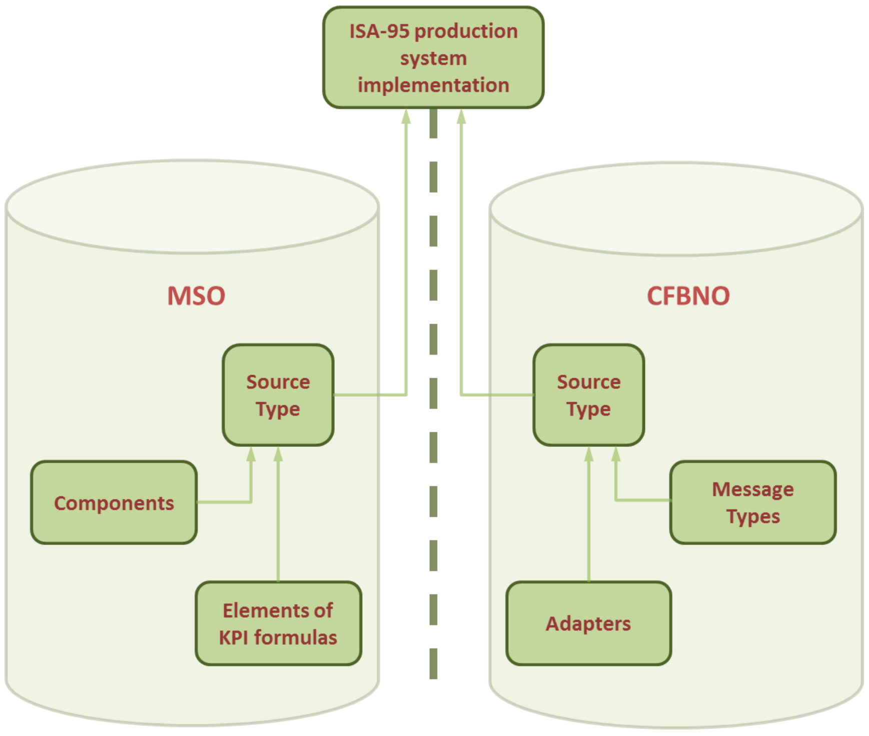


Figure 14: Interoperability of ontologies through ISA-95 taxonomy

The conceptual representation of previous Figure 14 shows that the interoperability between MSO and CFBNO can be performed through ISA-95 standard. This means that, for a MSO example, a value which is retrieved from a work cell that is used by a KPI formula can be defined with a source type that exists in the ISA-95 description. In parallel to this definition, in CFBNO, it can be stated that certain adapter manages that type of source defined in the standard. Hence, the reasoner will understand that the value retrieved from certain component that is needed for a KPI formula, is the same that is manipulated by the adapter.

This interoperability is needed because one of the objectives of this thesis work is to separate the production knowledge and the system data integration knowledge of an ES. By doing this division, a shop floor process expert does not have to be an expert of system data integration and vice versa. Then, the integration of the whole system can be performed by different users working separately, which means that reduces the complexity of integration since each expert supports to the system its knowledge. This division of efforts is depicted by following Figure 15.

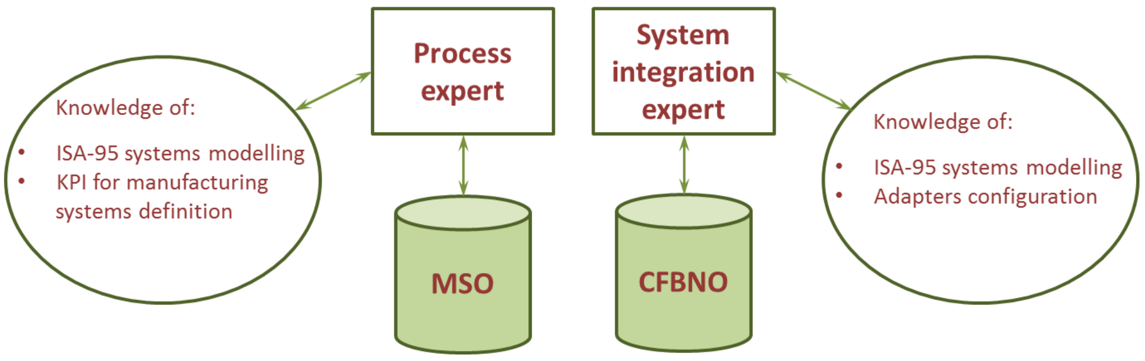


Figure 15: Required domains of experts for defining the ontological approach

Then, it is represented that both experts work on the same integration by doing different tasks for a common goal. Thus, the interoperability depicted on Figure 14 can be achieved, as an example, by the simple situation shown in Figure 15. Note that, obviously, the figure only lists the most important concepts that the experts must know to do this integration possible. In addition, note that both ontology developers have to know about ISA-95 since at the time of designing the ontologies, the standard has to be used to be the link between MSO and CFBNO as it is explained before.

Once the ontologies are prepared, they must be used for developing the FBN. Thereby, the reasoning is presented below as the last stage needed for the configuration and performance of the FBN for KPI calculation.

3.3.2. Reasoning

In this stage of the implementation technology and process description, the architecture of the system is almost entirely presented. In fact, it has been also advanced previously that, for reasoning, this ontological approach has implemented SPARQL queries. This QL can be used by any reasoner supported by Protégé 4.3 for the reasoning of ontologies.

Then, within the reasoning, MSO and CFBNO are queried for retrieving the data required for the FBN performance. In fact, this reasoning is used of an external application, the FBNI, which is described in the following section for developing the FBN for KPI configuration.

3.4. IEC-61499-based function block network application

Once the use of a FBNI for the data integration is described, an IEC-61499-based FBN application is briefly described. In fact, the implementation of this thesis work is used for the configuration of FBNs through this application. Then, the ontology approach must support the data required for the IEC-61499-based application to achieve the FBN for KPI calculation performance, which is the main objective of this thesis work.

As it has been explained in the *"IEC-61499"* section, FBs can be implemented as software and/or hardware and, a composition of them, can perform a model of an industrial control system. One of the objectives of the PLANTCockpit EC project [11] is the support of semantic configuration within ontologies. A Function Block Engine Configurator (FBEC) is presented in [40] as a semantic integration application used for performing required FBNs for configuring manufacturing systems. The implemented ontological approach must be usable by this FBEC.

3.4.1. Function block engine configurator

The following presentation of a FBEC has two main objectives for the methodology of the thesis work. On the one hand, it is demonstrated that the representation of IEC-61499-based implementation for FB compositions and its application in manufacturing systems by using ontological approaches is successfully possible. This means that this implementation can be a powerful approach used for reducing cost and time of data integration efforts. On the other hand, as a presented tool on this methodology approach, it is shown a real application under evaluation that uses the implementation of this thesis work reflecting the value of developed work.

As it has been introduced, a FBEC is developed for configuring and visualizing FBNs for manufacturing systems. A desired result for this thesis work implementation by using the FBEC is depicted by following Figure 16.

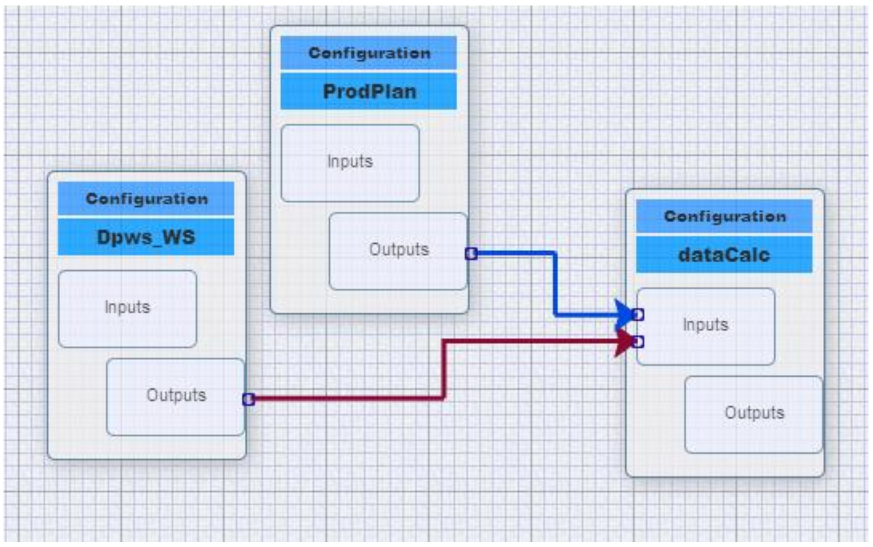


Figure 16: FBN model generated in the FBEC [40]

As it can be seen, Figure 16 shows a basic FBN model correctly generated by the FBEC which performs a scenario consisting in a decentralized interaction between three FBIs named as: *'ProdPlan'*, *'DPWS_WS'*, and *'dataCalc'*, which are, respectively, instances of an excel adapter, Devices Profile for Web Services (DPWS) adapter and data calculator FB for, in this implementation use, KPI calculation. Note that the required data as the names of each FBI, its configuration types, or message types, are given by the ontological approach, which means that the composition depicts real components of a modelled manufacturing system.

Thereby, Figure 16 depicts a possible representation by using the implementation of this thesis work, presented and discussed in following sections. This means that the ontological approach, described in this documentation, is meant to give the required knowledge to the FBEC that, being inferred, is utilized for representing useful FBNs for KPI calculation and modelling of manufacturing systems.

Note that Figure 16 only shows FBI interconnections in a composition performed by the presented FBEC, which means that this visualization does not present the required data management and integration for the FBN performance. One of the main problems for these connections is the variety of protocols that adapters of different manufacturing system devices use. Thus, the generated routes that connect different adapter FB instances with FBs must be manipulated for transforming adapter FB instances outcoming data for doing it understandable for the FB target. As it is explained on [40], the Extensible Stylesheet Language Transformation (XSLT) is done for FB routing proposes. This is another feature that nowadays data integration implementations must face. These transformations are not controlled by the ontological approach since they are done by another mapper tool. However, this tool needs the support of ontologies because it needs the FB instances configuration and message types. For understanding the role of the described FBEC and this transformation tool, the following section depicts the entire architecture in where this thesis work is applicable.

Hence, one of the important values of the presented thesis work relies in the critical assistance that the ontological approach supports to the FBEC to be able to realize configuration compositions, not only modelling the system from which information models of devices are retrieved, but also giving the semantic information required for doing the required tasks for configure the FBN. Thus, in following sections, the implementation for accomplish the required tasks to achieve the goal of this project is described and, then, the results are explained.

3.5. Technologies and architecture review

The objective of this section is to present the structure of the implementation performed in this thesis work. Thus, the concepts and roles of technologies are located and related in following Figure 17.

This schema depicts the realized implementation in green scale colours and, in white colour, the external tools that the ontological approach needs for achieving the objectives. Hence, the expected result and real location of the problem solved in a manufacturing system integration environment is presented.

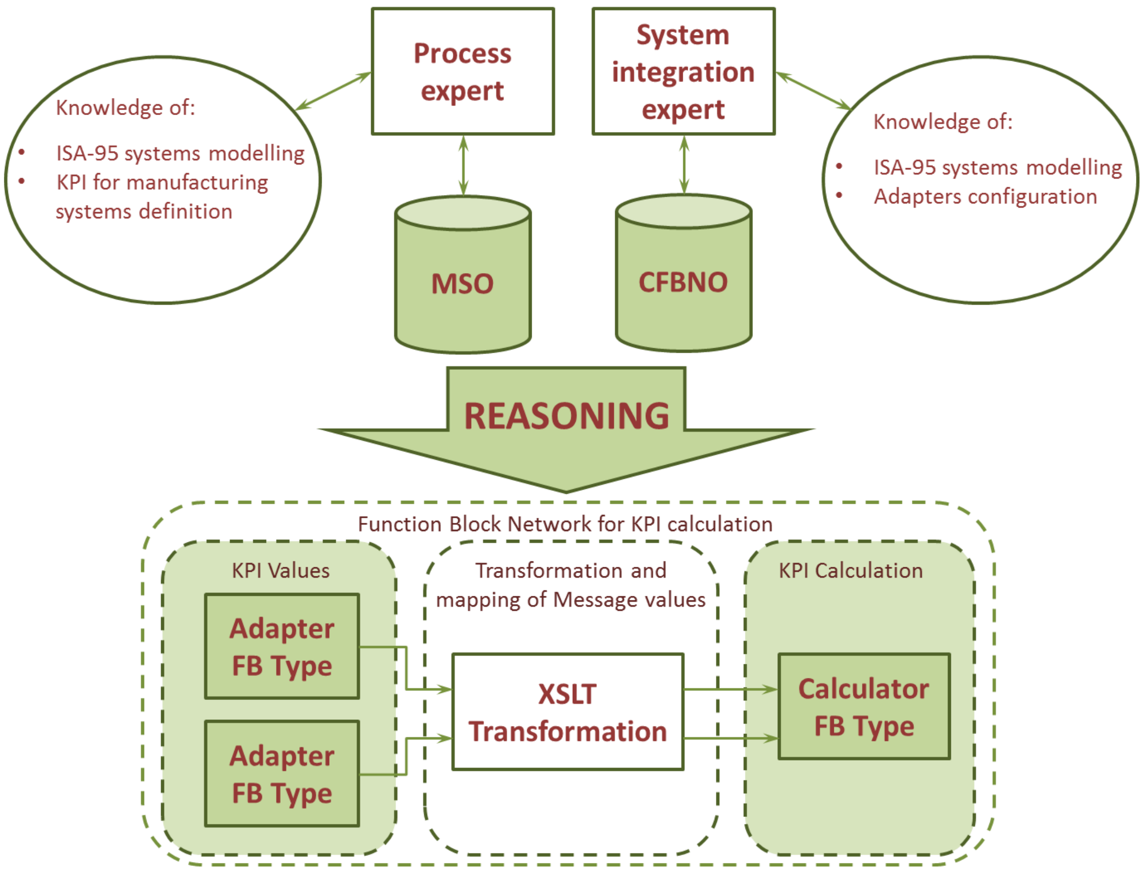


Figure 17: Architecture for data integration by using the thesis ontological approach

In fact, previous Figure 17 can be separated in three phases. The first phase is the start-up of the integration which consists in adding the instances to the ontologies of the scenario in which the ontological approach is being applied. Then, the second phase is the reasoning of the ontologies for retrieving the required data, needed for the engines which perform the FBN for KPI calculation in the last phase. Finally, by the XSLT transformation and data mapping with external tools to this implementation, the scenario can be designed by and visualized by the FBEC. Note that a possible resulting visualization has been already presented in Figure 16.

Then, it has been described through previous explanations the required technologies and processes for the implementation of this thesis work. Now, for finalizing the methodology approach section, the presentation of the real scenario for testing proposes is below defined.

3.6. Use case definition

Obviously, for achieving the goals of this thesis work, the implementation has to be tested. This integration is meant to be applied to manufacturing systems as it has been explained through many sections of this document.

Thereby, a FESTO line, installed in the TUT Factory Automation Systems and Technologies (FAST) Laboratory [61], has been used as the manufacturing system example to be represented by ontologies and, then, evaluating results of the explained methodology tasks. Note that the description of the FESTO line can be found on *“FESTO line”* section.

4. IMPLEMENTATION

This chapter is divided into two main sections. First, a briefly presentation of a FESTO line, which has been the scenario for implementing this ontological approach, is described. Afterwards, the second part of this chapter deals with the implementation of the thesis work itself, exemplifying the performance on the described FESTO line, presenting the design of the defined ontologies. Then, this chapter presents how the implementation is designed to achieve the thesis work objectives by following the descripted methodology approach.

4.1. FESTO line

FESTO supports different versions of MPS for didactic purposes. TUT FAST Laboratory has installed in its facilities a Multi-FMS which is a system composed by two FMS; Micro-FMS and MPS 500-FMS [62]. The combination of MPS 500-FMS and Micro-FMS is a perfect system to teaching students for analysing, controlling and understanding of communication systems. Besides this system utilization, Multi-FMS is being utilized for some research projects testing implementations as the one presented in [63], or for other thesis works implementations as the one described in [45]. Following Figure 18 shows the introduced Multi-FMS of TUT FAST Laboratory.

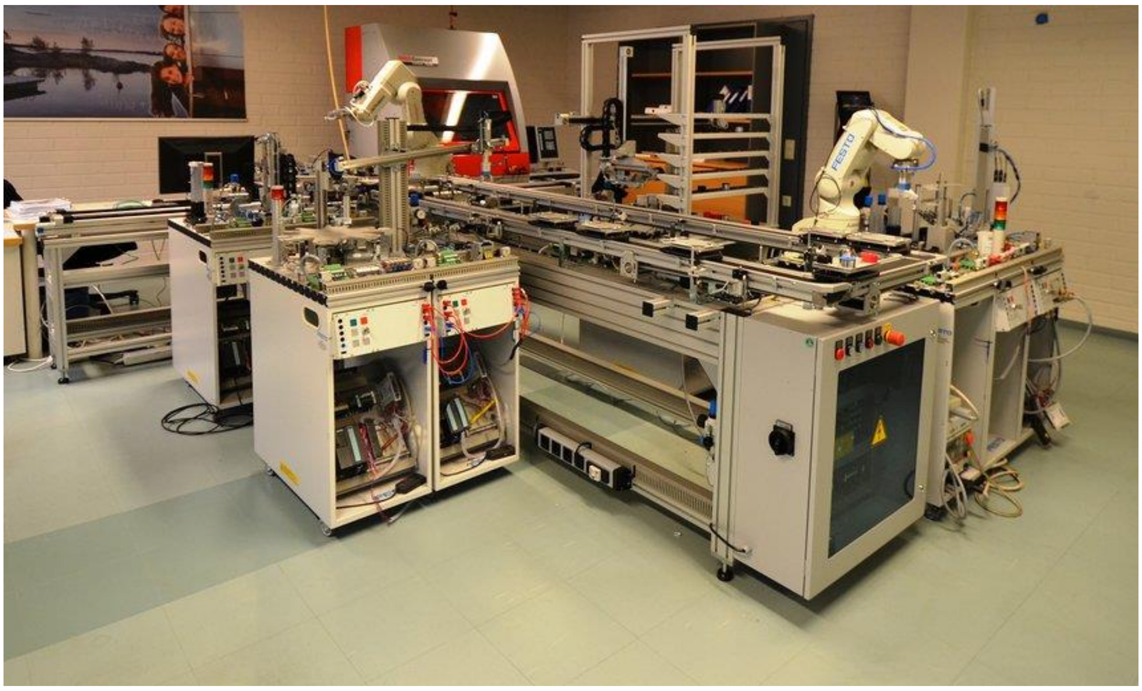


Figure 18: FESTO Multi-FMS on TUT FAST Laboratory facilities [61]

The Multi-FMS is divided into two FMS with a common transport system which is the connection between them for performing a complete assembly process. This main close loop conveyor is the central component of the MPS 500-FMS and has up to six working positions. These are the points in which workstations composed by one or more stations are attached.

Following Figure 19 depicts the configuration of the Multi-FMS on TUT FAST Laboratory facilities. Note that the diagram is not scaled since is just a sketch of the distribution, in which each number on the conveyor determines the corresponding working position.

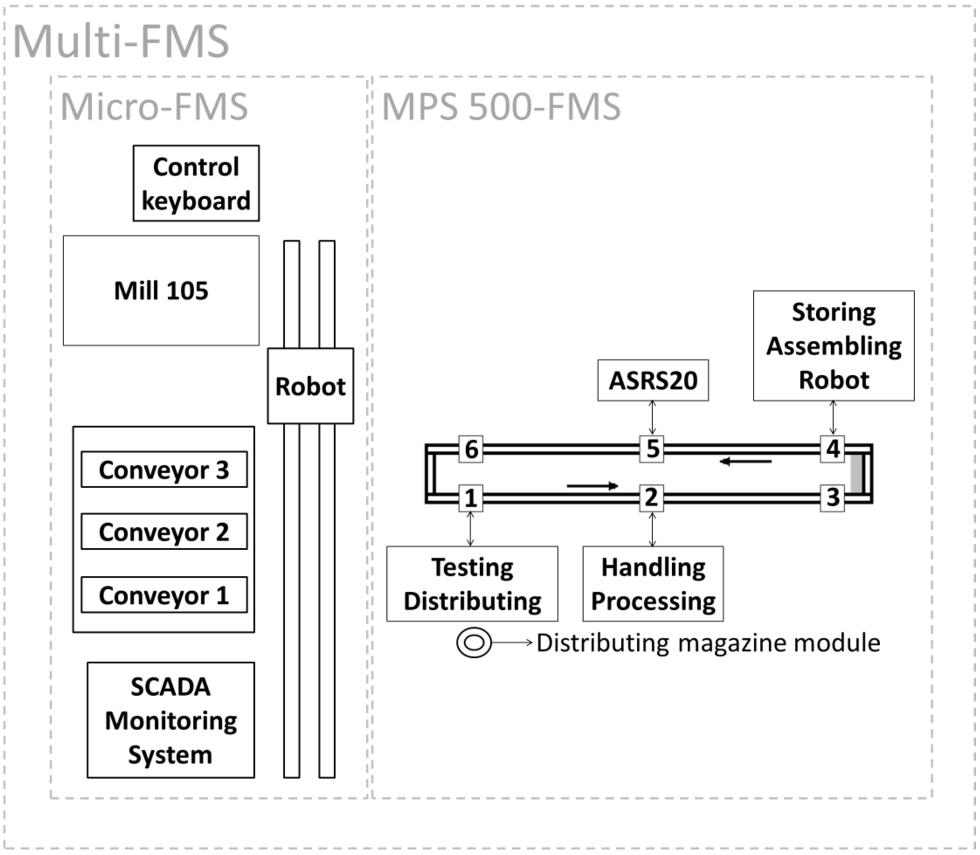


Figure 19: Multi-FMS configuration

The different MPS stations attached to the transport system, represented in previous sketch by one word in its corresponding working position, and the stations which compose the Micro-FMS are briefly described by following Table 6. In addition, note that the control keyboard and Supervisory Control and Data Acquisition (SCADA) monitoring system, shown in previous Figure 19, are software driven components for monitoring and controlling proposes of both FMS.

Note that this description corresponds to the Multi-FMS configuration installed on TUT FAST Laboratory since other configurations are possible as is presented in [62].

Table 6: Multi-FMS stations [62]

FMS	Name	Description
Micro-FMS	Mill 105 station	3-axis milling machine [64]
Micro-FMS	Control keyboard	Control keyboard for define work-pieces drilling programs
Micro-FMS	Robot station (RV-1A)	6-axis RV-1A Mitsubishi industrial robot with pneumatic finger grippers mounted on trolley for cylinder bod-ies transport along Micro-FMS
Micro-FMS	Parts Buffer station	Station with three conveyor belt buffers for supply cylinder bodies and receive assembled workpieces
MPS 500-FMS	AFB pallet transport system	Pallet transport system for material flow to the six working positions
MPS 500-FMS	Distributing station	Start point of workpieces assembly process within a distributing maga-zine module
MPS 500-FMS	Testing station	Tests the colour and height of cylin-der body workpieces
MPS 500-FMS	Handling station	Transports workpieces from pallet to processing station and vice-versa
MPS 500-FMS	Processing station	Processes cylinder body workpieces by the simulation of drilling
MPS 500-FMS	Robot station (RV-2AJ)	5-axis RV-2AJ Mitsubishi industrial robot which picks up workpieces from pallets for assembly them with springs, pistons and convers. Then, it transports assembled workpieces to pallet or to the storing station
MPS 500-FMS	Assembling station	Supplies springs, pistons and covers to robot station
MPS 500-FMS	Storing station	Stores up to eighteen assembled workpieces in three storage levels
MPS 500-FMS	Warehouse station (ASRS20)	Automatic Storage and Retrieval System (ASRS) with twenty loca-tions for workpieces
Multi-FMS	SCADA monitoring system	Computer station, with SCADA sys-tem installed, for supervising pro-cess and requesting predefined cyl-inder body drilling programs to Mill 105 station

4.1.1. Assembly process on Multi-FMS

This section describes a general assembly process performed in Multi-FMS with the objective to understand that a large amount of KPIs can be defined to be managed for efficiency control of the system. This means that the KPIs defined in the production ontology are possible in this general assembly process of the Multi-FMS.

The description of steps for a possible sequence of workpieces assembly by a Multi-FMS is described on following Table 7. Note that, at the beginning, each station is empty of workpieces less than the *"Parts Buffer station"* which has cylinders bodies for supply the system and the *"Assembly station"* that has the required material for assembly workpieces. In addition, all the system is supposed to be active, waiting for start their tasks and the pallets are flowing on the AFB transport system.

Table 7: Example of workpieces assembling sequence in Multi-FMS

Step	Station	Description
1	Parts Buffer	The selected conveyor by the loaded program transports the cylinder body to the end point of the conveyor
2	Robot	Picks-up the cylinder and transports it into Mill 105
3	Mill 105	Processes the cylinder
4	Robot	Picks-up the cylinder and transports it into the distributing magazine module in distributing station
5	Distributing	It delivers a cylinder to testing station when an order is done
6	Testing	Tests height and colour of the cylinder and deliver it on a waiting pallet
7	Handling	The incoming cylinder is transported to the processing station
8	Processing	The incoming cylinder is processed returned to handling pick-up point
9	Handling	It takes the cylinder and transports it to the waiting pallet
10	Robot	It Picks-up the incoming cylinder and assembles it using one spring, piston and cover. Then, the delivery depends on a defined decision. In this case, black workpieces are sent to the storing station and non-black workpieces are transported to the waiting pallet
11	Storing	It stores the incoming black work piece in the three storage level
12	ASRS20	It stores the incoming non-black work piece in a free location

Note that the behaviour of stations for performing sequences as the previously one, are programmable. This means that, for instance, Micro-FMS performs cylinders depending on the program defined by the control keyboard and loaded from the SCADA system computer to the Mill 105 station. On the other hand, MPS 500-FMS, for instance, rejects pieces in testing station depending on the loaded program into the testing station Programmable Logic Controller (PLC), in which can be specified that certain cylinders with some height or colour must be rejected, or even none of them.

Then, note that Multi-FMS offers a large amount of possible output workpieces and the KPIs will reflect the production efficiency of current performing process.

4.2. Production ontology

This section describes the implementation of the production ontology, named previously as MSO. Note that the components of the Multi-FMS which are gathered by the ontology are listed on *"Appendix A"* section. For more detailed information and specifications of components, the Multi-FMS manual supported by FESTO [65] contains full information of the system.

As it has been introduced in the *"Methodology approach"* section, Protégé 4.3 has been selected and used as the supporting tool for modelling the system. The implementation of the production ontology is presented within described screenshots of this software for analysing the modelling process and the resulting design.

4.2.1. Implementation of the production ontology

The design of the production ontology has been developed by following the process described in *"Methodologies for design of ontologies"* section.

Firstly, the domain of the ontology must be defined. For the defined thesis objectives, the scope of the production ontology, named previously as MSO, has to include the description of the Multi-FMS, the KPI definition and, also, the formulas of each KPI. In fact, since this domain is meant to be designed by a new ontological approach, the reuse of other ontologies has been discarded. This means that the second step of the ontology design methodology has been obviated.

Once the domain of the ontology has been described, the terminology needed for representing the manufacturing system and the additional features of the MSO have to be determined. Note that this process consists in defining the taxonomy of the modelled system by the ontology. Thus, the names of classes and relations between them are partially decided by this terminology approach. This means that, with taxonomies, the relevant terms are figured out but in the next step, that is the definition of the hierarchy of classes, new objects can be found to be needed for the ontology description. As it has been explained in Figure 14 of the *"Interoperability between ontologies"* section, the model follows the ISA-95 and, concretely, the source types of the production values must be defined with the standard for the interoperability of ontologies. This part of the implementation is presented in the *"Interoperability between MSO and CFBNO"* section of this chapter.

Then, afterwards of the terminology definition, the hierarchy of classes is designed. Following Figure 20 represents the structuration of the objects as start point before the ontology design.

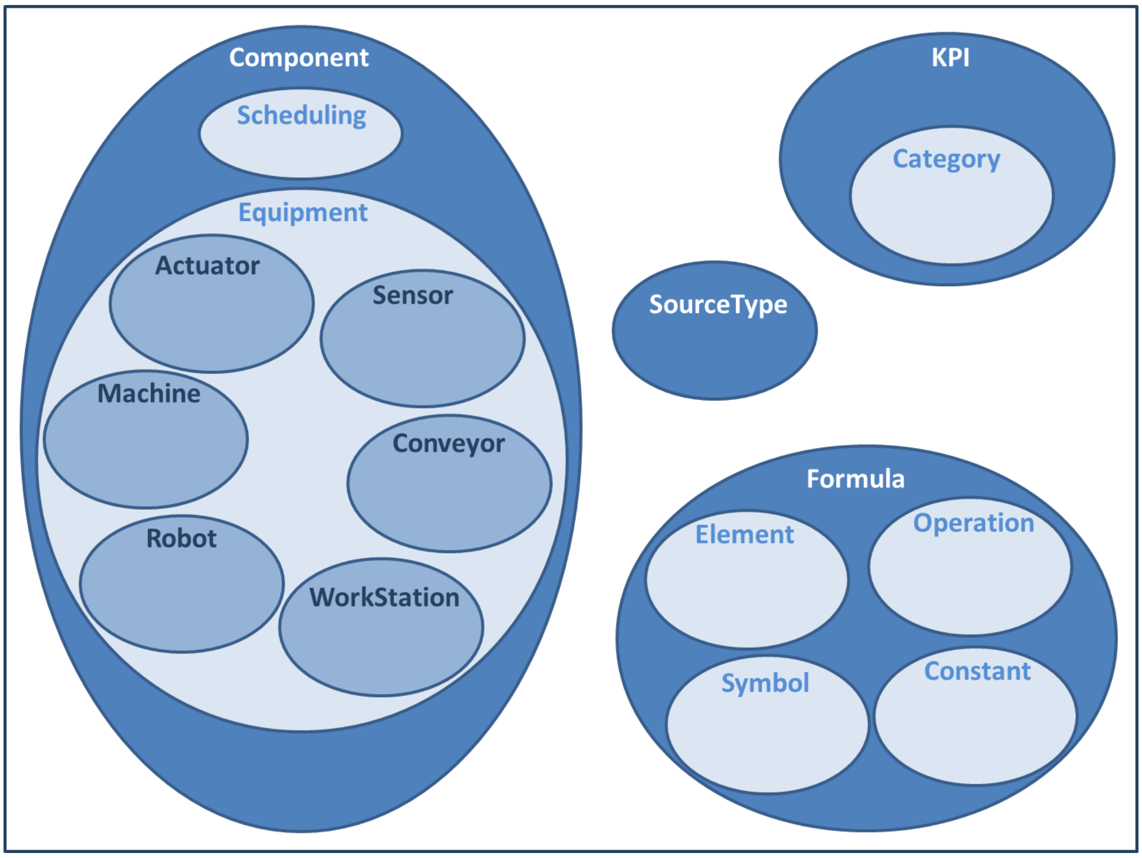
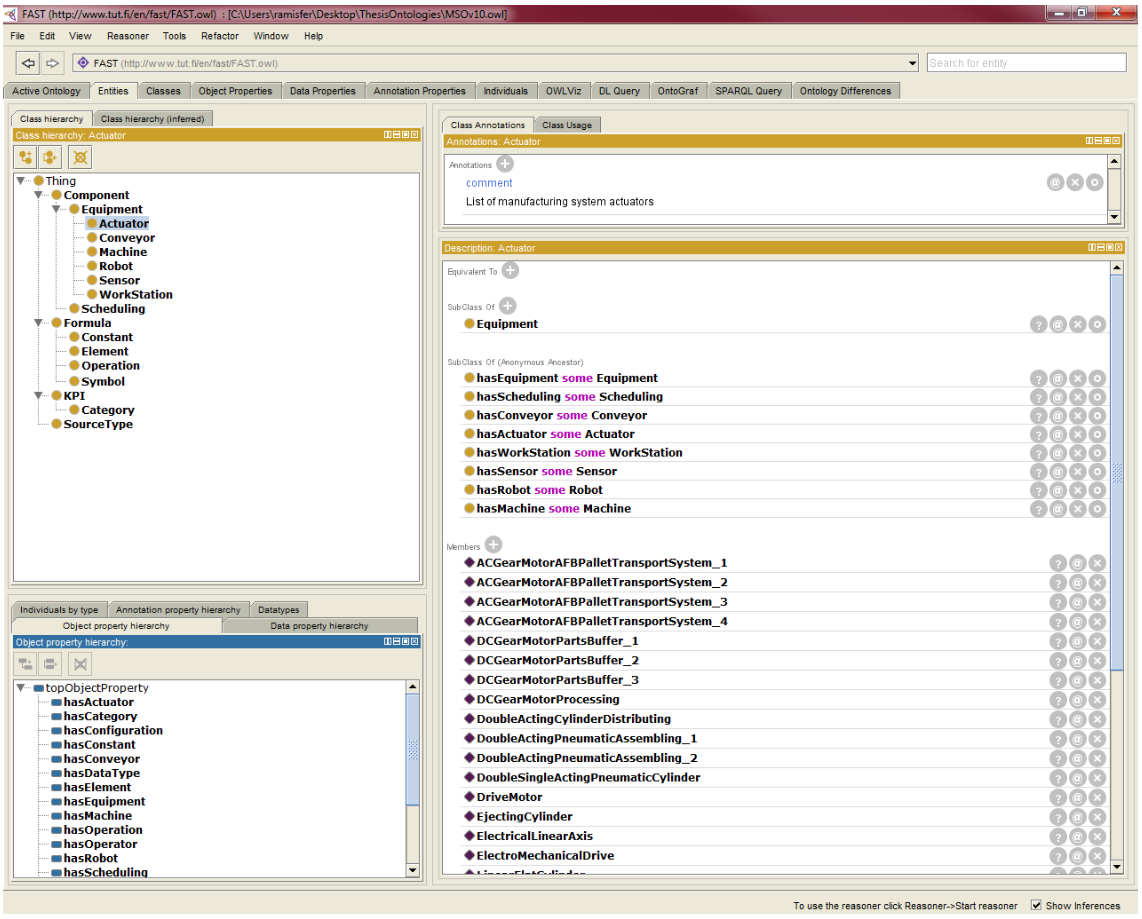


Figure 20: Venn diagram of MSO classes

Thereby, it is shown how the distribution of classes within a Venn diagram can be depicted. In fact, the ontology scope can be understood as a junction of two sub-domains linked with the *'SourceType'* object. Meanwhile the first one, which is composed by the high level *Component'* object, is describing the manufacturing system from where the information model of devices is retrieved; the second one, which is composed by the high level *'Formula'* and *'KPI'* objects, is describing the KPIs and their corresponding formulas. Note that both sub-domains are related to the *'SourceType'* class for its mapping with the ISA-95 description.

Once the whole concept and its structuration are presented, the resulting hierarchy of MSO classes is presented by following Figure 21, which is a screenshot of the OWL design within Protégé 4.3. Note that, by definition, a concept gathers all the classes of ontologies so that, in the previous diagram, the square that includes all the objects can be understood as the concept of the MSO. This concept is normally named as *'Thing'* since all can be classified as a thing.


FAST (http://www.tut.fi/en/fast/FAST.owl) : [C:\Users\ramisfer\Desktop\ThesisOntologies\MSOv10.owl]
File Edit View Reasoner Tools Refactor Window Help
FAST (http://www.tut.fi/en/fast/FAST.owl)
Search for entity
Active Ontology | Entities | Classes | Object Properties | Data Properties | Annotation Properties | Individuals | OWLViz | DL Query | OntoGraf | SPARQL Query | Ontology Differences
Class hierarchy | Class hierarchy (inferred)
Individuals by type | Annotation property hierarchy | Datatypes
Object property hierarchy | Data property hierarchy
Class Annotations | Class Usage
To use the reasoner click Reasoner->Start reasoner
Show Inferences

Class hierarchy: Actuator

```
Thing
  Component
    Equipment
      Actuator
      Conveyor
      Machine
      Robot
      Sensor
      WorkStation
    Scheduling
  Formula
    Constant
    Element
    Operation
    Symbol
  KPI
    Category
  SourceType
```

Object property hierarchy:

```
topObjectProperty
  hasActuator
  hasCategory
  hasConfiguration
  hasConstant
  hasConveyor
  hasDataType
  hasElement
  hasEquipment
  hasMachine
  hasOperation
  hasOperator
  hasRobot
  hasScheduling
```

Annotations: Actuator

Annotations
comment
List of manufacturing system actuators

Description: Actuator

Equivalent To

SubClass Of
Equipment

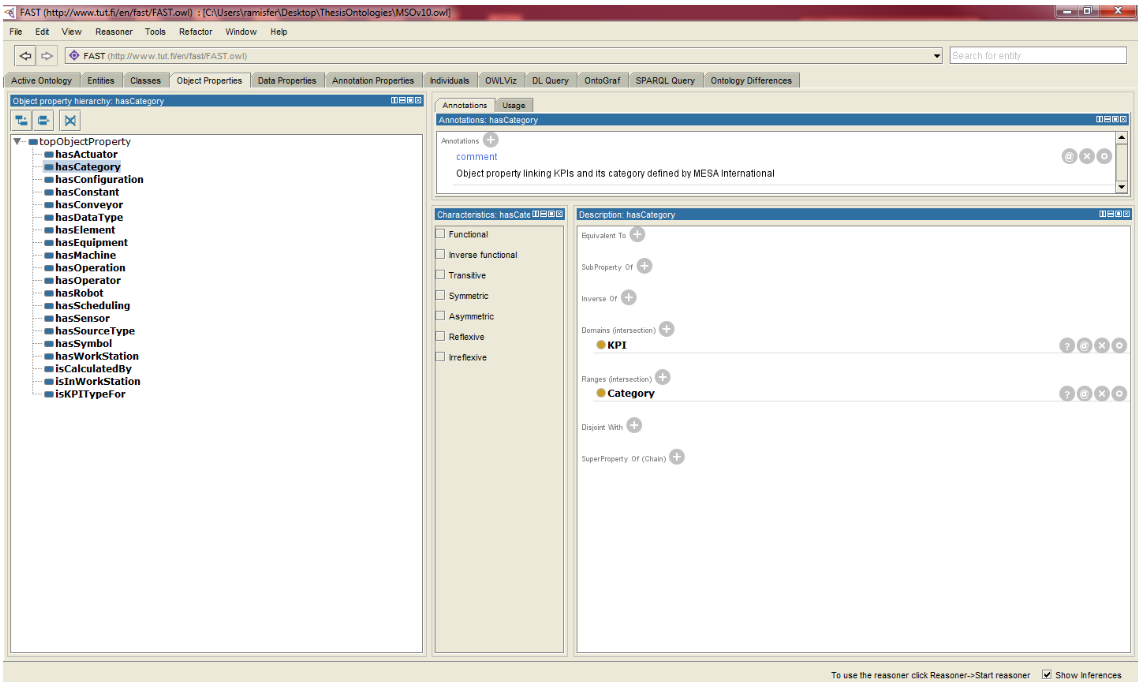
SubClass Of (Anonymous Ancestor)
hasEquipment some Equipment
hasScheduling some Scheduling
hasConveyor some Conveyor
hasActuator some Actuator
hasWorkStation some WorkStation
hasSensor some Sensor
hasRobot some Robot
hasMachine some Machine

Members
ACGearMotorAFBPalletTransportSystem_1
ACGearMotorAFBPalletTransportSystem_2
ACGearMotorAFBPalletTransportSystem_3
ACGearMotorAFBPalletTransportSystem_4
DCGearMotorPartsBuffer_1
DCGearMotorPartsBuffer_2
DCGearMotorPartsBuffer_3
DCGearMotorProcessing
DoubleActingCylinderDistributing
DoubleActingPneumaticAssembling_1
DoubleActingPneumaticAssembling_2
DoubleSingleActingPneumaticCylinder
DriveMotor
EjectingCylinder
ElectricalLinearAxis
ElectroMechanicalDrive

Figure 21: MSO class hierarchy

As it can be seen in previous screenshot, the interface of Protégé 4.3 allows the visualization and graphical design of ontologies. The hierarchy of MSO classes can be seen in the left side window named as *'Class Hierarchy'* of Figure 21. In fact, the level structuration of classes can be compared with the sketch presented on Figure 20 and it can be seen that the hierarchy is respected and successfully designed.

The next MSO implementation step is the definition of object properties between classes. Following Figure 22 shows the defined object properties for this ontology.


FAST (http://www.tut.fi/en/fast/FAST.owl) : [C:\Users\ramisfer\Desktop\ThesisOntologies\MSOv10.owl]
File Edit View Reasoner Tools Refactor Window Help
FAST (http://www.tut.fi/en/fast/FAST.owl)
Search for entity
Active Ontology Entities Classes Object Properties Data Properties Annotation Properties Individuals OWLViz DL Query OntoGraf SPARQL Query Ontology Differences
Annotations Usage
Characteristics: hasCate
Functional
Inverse functional
Transitive
Symmetric
Asymmetric
Reflexive
Irreflexive
Description: hasCategory
Equivalent To
SubProperty Of
Inverse Of
Domains (intersection)
Ranges (intersection)
Disjoint With
SuperProperty Of (Chain)
To use the reasoner click Reasoner->Start reasoner
Show Inferences

Object property hierarchy: hasCategory

- topObjectProperty
  - hasActuator
  - hasCategory
  - hasConfiguration
  - hasConstant
  - hasConveyor
  - hasDataType
  - hasElement
  - hasEquipment
  - hasMachine
  - hasOperation
  - hasOperator
  - hasRobot
  - hasScheduling
  - hasSensor
  - hasSourceType
  - hasSymbol
  - hasWorkStation
  - isCalculatedBy
  - isInWorkStation
  - isKPITypeFor

Annotations: hasCategory

Annotations
comment
Object property linking KPIs and its category defined by MESA International

Domains (intersection): KPI

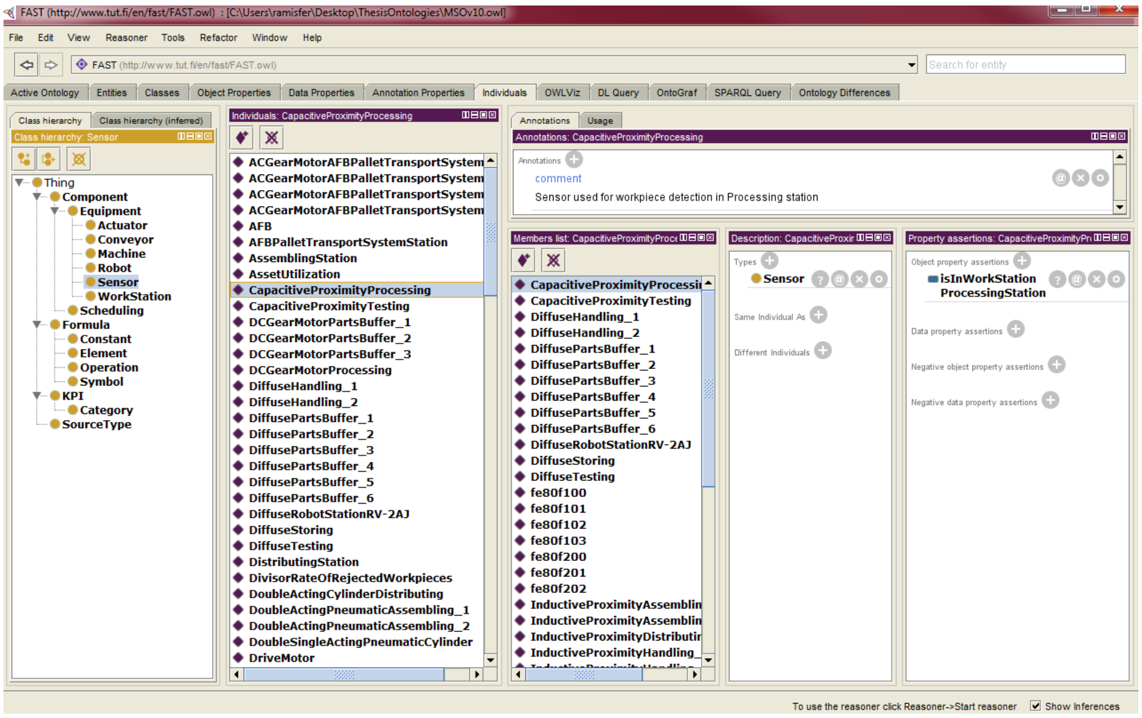
Ranges (intersection): Category

Figure 22: MSO object properties

Then, the defined properties are included in the left side window named as *‘Object property hierarchy’*. As the window name indicates, the properties are also structured by levels, since a property can have associated sub-properties, but there are no cases in this implementation. Note that the utilization of some properties is shown in Figure 21 in the right side window named as *‘Description’*. These definitions are needed to be done for each class for the consistency of the ontology.

The continuous step consists in adding additional properties to the already defined object properties. This can be understood by analysing the previous Figure 22, since these extra characteristics for object properties are defined in the right and central windows named as *‘Description’* and *‘Characteristics’*, respectively. Note that this definition is needed to be done for each object property and the correctly definition of the characteristics of properties can be critical for the functionality of the ontology, as is described in [33] in where the possible design of properties is analysed.

Once the structuration of the ontology is done and presented, the next step is to use it for defining instances which represent the real world for the determined domain. Then, following Figure 23 depicts the definition of MSO instances in Protégé 4.3.


FAST (http://www.tut.fi/en/fast/FAST.owl) : [C:\Users\ramisfer\Desktop\ThesisOntologies\MSOv10.owl]
File Edit View Reasoner Tools Refactor Window Help
FAST (http://www.tut.fi/en/fast/FAST.owl)
Search for entity
Active Ontology Entities Classes Object Properties Data Properties Annotation Properties Individuals OWLViz DL Query OntoGraf SPARQL Query Ontology Differences
Class hierarchy Class hierarchy (inferred)
Class hierarchy: Sensor
Individuals: CapacitiveProximityProcessing
Annotations Usage
Annotations: CapacitiveProximityProcessing
Members list: CapacitiveProximityProce
Description: CapacitiveProxir
Property assertions: CapacitiveProximityPr
To use the reasoner click Reasoner->Start reasoner
Show Inferences

Thing
- Component
  - Equipment
    - Actuator
    - Conveyor
    - Machine
    - Robot
    - Sensor
    - WorkStation
  - Scheduling
- Formula
  - Constant
  - Element
  - Operation
  - Symbol
- KPI
  - Category
- SourceType

ACGearMotorAFBPalletTransportSystem
ACGearMotorAFBPalletTransportSystem
ACGearMotorAFBPalletTransportSystem
ACGearMotorAFBPalletTransportSystem
AFB
AFBPalletTransportSystemStation
AssemblingStation
AssetUtilization
CapacitiveProximityProcessing
CapacitiveProximityTesting
DCGearMotorPartsBuffer_1
DCGearMotorPartsBuffer_2
DCGearMotorPartsBuffer_3
DCGearMotorProcessing
DiffuseHandling_1
DiffuseHandling_2
DiffusePartsBuffer_1
DiffusePartsBuffer_2
DiffusePartsBuffer_3
DiffusePartsBuffer_4
DiffusePartsBuffer_5
DiffusePartsBuffer_6
DiffuseRobotStationRV-2AJ
DiffuseStoring
DiffuseTesting
DistributingStation
DivisorRateOfRejectedWorkpieces
DoubleActingCylinderDistributing
DoubleActingPneumaticAssembling_1
DoubleActingPneumaticAssembling_2
DoubleSingleActingPneumaticCylinder
DriveMotor

Annotations
comment
Sensor used for workpiece detection in Processing station

CapacitiveProximityProcessi
CapacitiveProximityTesting
DiffuseHandling_1
DiffuseHandling_2
DiffusePartsBuffer_1
DiffusePartsBuffer_2
DiffusePartsBuffer_3
DiffusePartsBuffer_4
DiffusePartsBuffer_5
DiffusePartsBuffer_6
DiffuseRobotStationRV-2AJ
DiffuseStoring
DiffuseTesting
fe80f100
fe80f101
fe80f102
fe80f103
fe80f200
fe80f201
fe80f202
InductiveProximityAssemblin
InductiveProximityAssemblin
InductiveProximityDistributir
InductiveProximityHandling_

Types
Sensor
Same Individual As
Different Individuals

Object property assertions
isInWorkStation ProcessingStation
Data property assertions
Negative object property assertions
Negative data property assertions

Figure 23: MSO instances

Then, it can be seen that the MSO instances are defined in the window named as *'Individuals'*. In addition, the adjacent window named as *'Description'*, shows in which class is included the selected individual. Note that, the set of tables included in *"Appendix A"* section contains all the Multi-FMS devices that are going to be defined.

Since there are sensors and actuators that have the same name, the terminology that has been used follows the logic presented on next Figure 24.

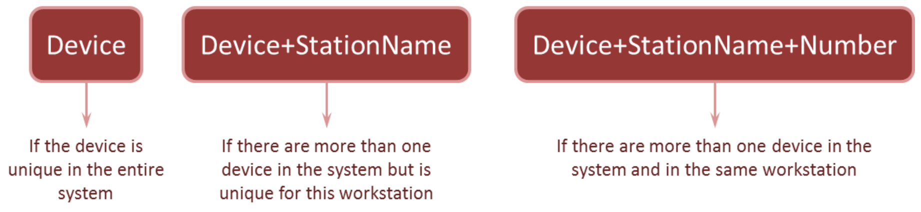
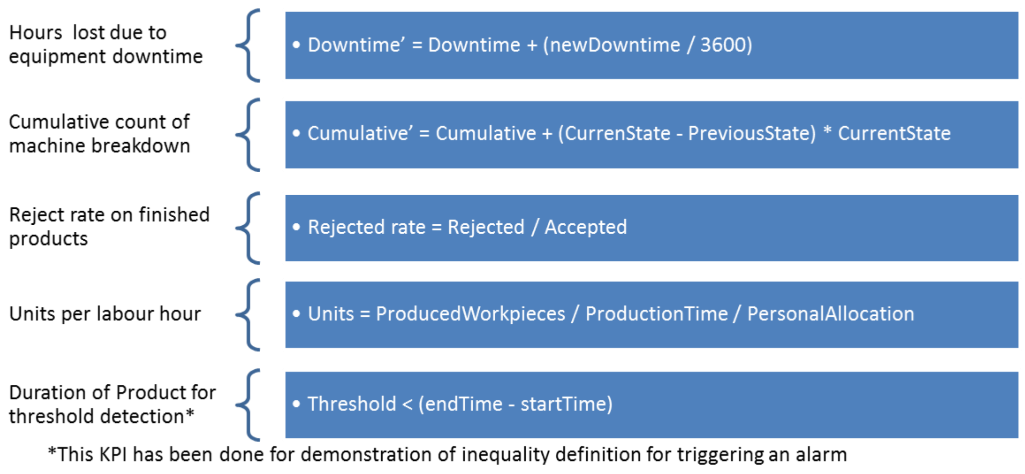

Device
Device+StationName
Device+StationName+Number
If the device is unique in the entire system
If there are more than one device in the system but is unique for this workstation
If there are more than one device in the system and in the same workstation

Figure 24: Logic for naming the MSO instances

As it can be seen, following this logic, while the diffuse sensors of the handling station are defined as *'DiffuseSensorHandling_1'* and *'DiffuseSensorHandling_2'*, the same sensor type of the testing station is defined as *'DiffuseSensorTesting'*, without number because is the only one in the corresponding station. Note that the logic presented by Figure 24 is applied for the definition of Multi-FMS devices to avoid multiple instances with the same name, which is not recommendable because it would cause the inconsistency of the OWL design.

Besides this, a set of KPIs has been selected from the presented Table 4 in *"ISA-95-based operations and KPI metrics"* section. Following Figure 25 presents the chosen KPIs and their formulas.



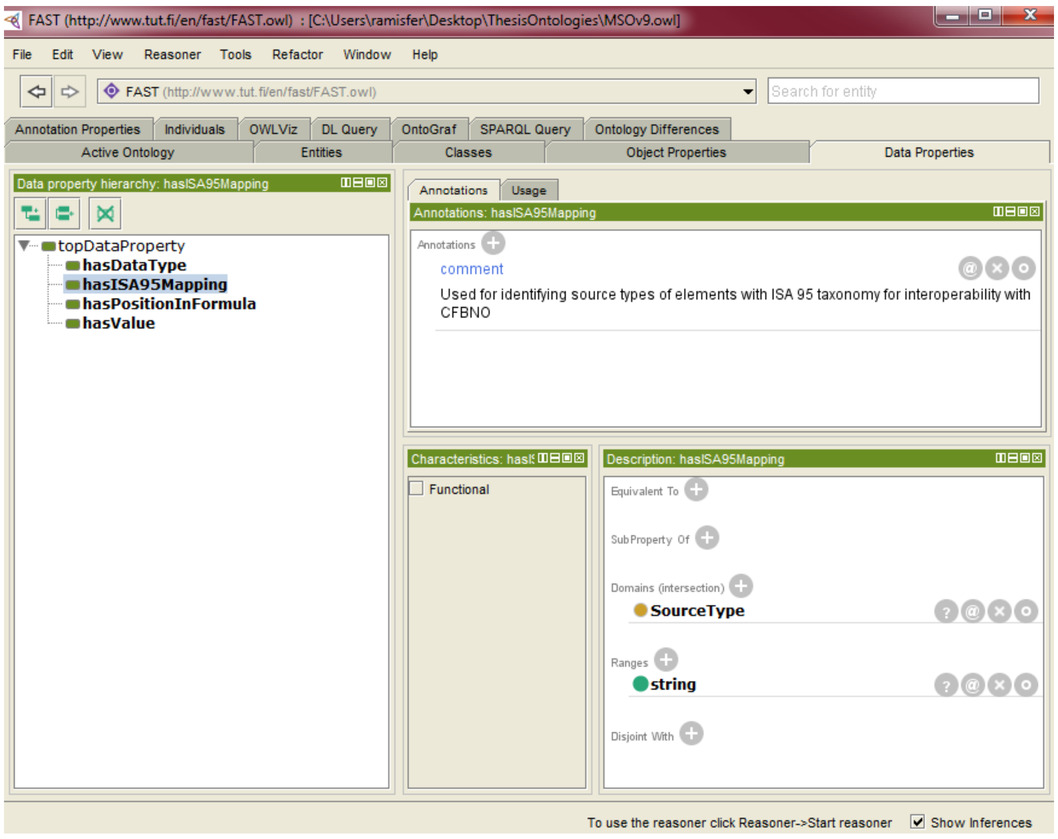
Hours lost due to equipment downtime	• Downtime' = Downtime + (newDowntime / 3600)
Cumulative count of machine breakdown	• Cumulative' = Cumulative + (CurrenState - PreviousState) * CurrentState
Reject rate on finished products	• Rejected rate = Rejected / Accepted
Units per labour hour	• Units = ProducedWorkpieces / ProductionTime / PersonalAllocation
Duration of Product for threshold detection*	• Threshold < (endTime - startTime)

*This KPI has been done for demonstration of inequality definition for triggering an alarm

Figure 25: Selected KPIs and formulas

Then, previous Figure 25 shows the metrics that have been defined in the ontology. Note that each KPI have a different formula that has to be defined by the OWL implementation. To accomplish this objective, equations have been described as a relation of notation elements, which can be seen in the hierarchy of classes presented in Figure 21.

Once the definition of instances is performed, the last step for the production ontology design is to define the *'Data Type properties'*, their characteristics and, after, add them to the generated instances. Following Figure 26 presents the defined *'Data Type properties'*.


FAST (http://www.tut.fi/en/fast/FAST.owl) : [C:\Users\ramisfer\Desktop\ThesisOntologies\MSOv9.owl]
File Edit View Reasoner Tools Refactor Window Help
FAST (http://www.tut.fi/en/fast/FAST.owl)
Search for entity
Annotation Properties Individuals OWLViz DL Query OntoGraf SPARQL Query Ontology Differences
Active Ontology Entities Classes Object Properties Data Properties
Data property hierarchy: hasISA95Mapping
Annotations Usage
Annotations: hasISA95Mapping
Annotations
Characteristics: hasI
Description: hasISA95Mapping
Equivalent To
SubProperty Of
Domains (intersection)
Ranges
Disjoint With
To use the reasoner click Reasoner->Start reasoner
Show Inferences

topDataProperty
- hasDataType
- hasISA95Mapping
- hasPositionInFormula
- hasValue

comment
Used for identifying source types of elements with ISA 95 taxonomy for interoperability with CFBNO

☐ Functional

Domains (intersection): SourceType

Ranges: string

Figure 26: MSO data type properties

It can be seen in previous screenshot that the *'Data Type properties'* definition is similar to the *'Object properties'* presented previously. Thus, its definition is done in the *'Data property hierarchy'* window and the characteristics are defined in *'Characteristics'* and *'Description'* windows.

Finally, as it has been already introduced, the addition of these properties to the instances must be done for ending the production ontology design. Note that not all the instances have both property types because each individual is different. Following Figure 27 presents an example of properties definition for an instance in a window named as *'Property assertions'*.

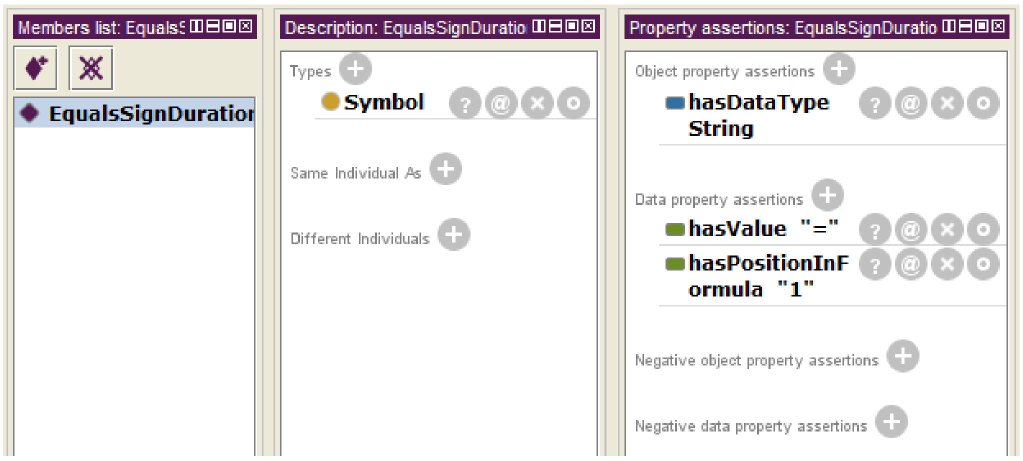

Members list: EqualsS
EqualsSignDuratior
Description: EqualsSignDuratio
Types
Symbol
Same Individual As
Different Individuals
Property assertions: EqualsSignDuratio
Object property assertions
hasDataType String
Data property assertions
hasValue "="
hasPositionInFormula "1"
Negative object property assertions
Negative data property assertions

Figure 27: Example of both properties definition in the same instance

As it can be seen in previous capture, an individual that belongs to the *'Symbol'* class has defined some properties. On the one hand, the object property *'hasDataType'* is used for describing within other individuals the type of data. Then, the data properties *'hasValue'* and *'hasPositionInFormula'* are used for describing its value and position in the formula, respectively.

For concluding this production ontology design presentation, following Figure 28 presents a part of the designed MSO. Protégé allows users to, meanwhile the ontology is being designed, visualize the actual status of the ontology. This is possible with the *'OntoGraf'* window, which permits the expansion of all the components of the ontology. As it can be seen in following Figure 28, this representation is presented by boxes that include the names of the objects or instances and arrows of different colours that join them. In addition, if the link between classes and/or instances is selected, the name of the corresponding property will be shown as it is also depicted by the next capture.

This visualization is useful for the verification of the ontology. In fact, it can help not only the user to understand the ontology but also the developer for checking if, in example, all the defined instances are related to their corresponding classes.

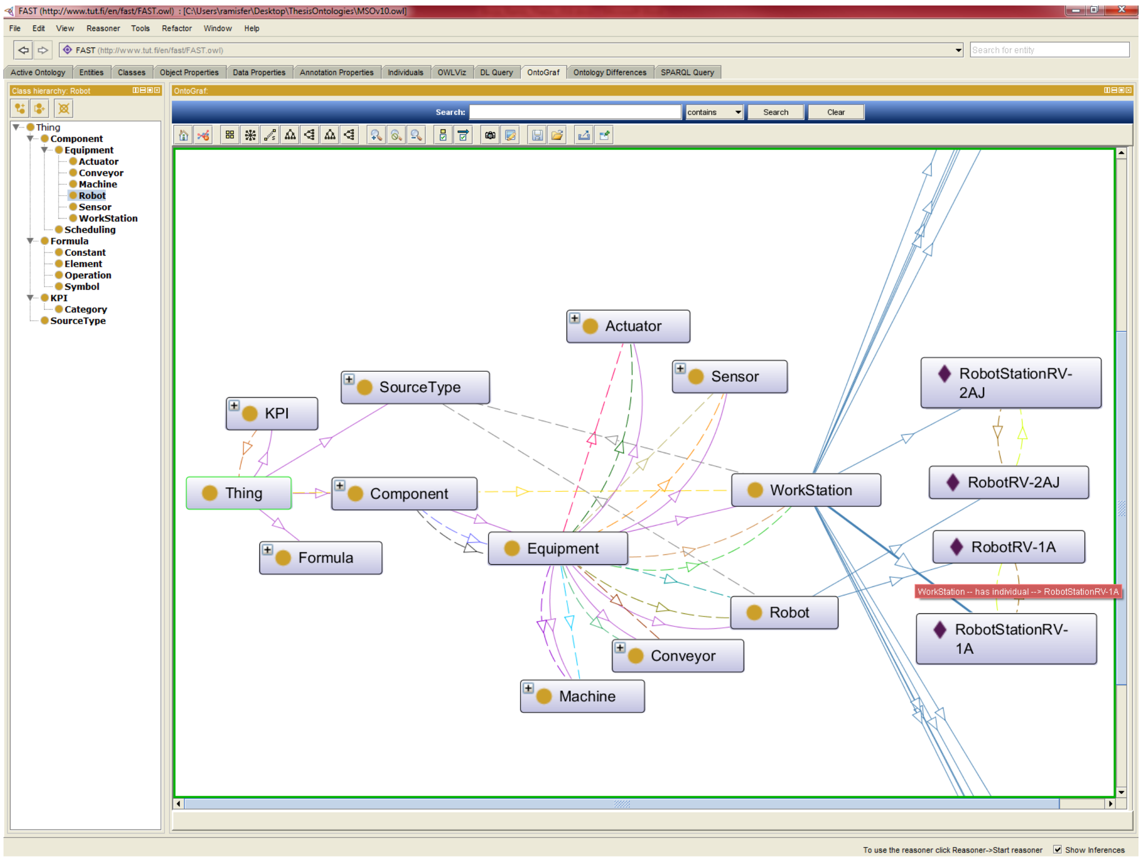


Figure 28: Verifying the MSO design with the *'OntoGraf'* window of Protégé

Note that previous Figure 28 only shows a part of the ontology design because of the extension that the entire ontology would occupy.

4.2. FBN configuration ontology

This section describes the implementation of the FBN configuration ontology, named previously as CFBNO. As in previous production ontology design, the process described in *"Methodologies for design of ontologies"* section has been used to perform this implementation.

As it has been introduced in previous section, Protégé 4.3 has been selected and used as the supporting tool for modelling the system. Then, the implementation of the FBN configuration ontology is presented within described screenshots of this software for analysing the modelling process and the resulting design.

Note that, as it has been explained in *"Function block engine configurator"* section, this ontology has to be functional for been used by a FBEC. Thus, some decisions of this implementation have been concluded for satisfying requirements of the configurator and for allowing the interoperability between technologies of the whole system architecture presented in Figure 17. For this reason, the presentation of the requirements of the FBEC supported by the CFBNO must be presented.

4.2.1. Requirements of the FBEC supported by the CFBNO

As it has been introduced in different sections of this thesis documentation, there are some requirements of the FBEC that must be supported by the CFBNO. Thus, this section presents briefly these needs. Hence, following Table 8 describes a list of descriptions that the CFBNO has to include. Note, once again, that these requirements are given by the ontology to be used for the FBEC as an external tool of the ontological approach of this thesis work.

Table 8: Requirements of the FBEC supported by the CFBNO

Requirement	Functionality
PID	Name of the FBI
JSON schema	Schema of JSON
JSON	Configuration of FBI
Message name	Message type name
Message type schema	Schema of the message types
Message types	Message types of the FBI
Elements	Mapping of elements from each message types

Firstly, the FBI needs to have a proper name for the network so that the PID is required. Then, the FB needs to be properly configured to permit the configurator to start with the performance of the FBN. Then, the configuration of the FB is required. This means that a defined JavaScript Object Notation (JSON), different for each instance, contains the information required to fill the business logic of a FB. Note that the JSON follows the schema of a defined JSON schema.

Once the required description for the FB definition is included, the information of the message types is also needed. For this reason, messages name, its type and schema must be defined. For instance, as it is shown in next section, the message types can be *'In'* or *'Out'*. Note that these possibilities are due to the scenario performance, in where FBIs of adapters are always connected to FBIs of calculators.

Finally, the element description is required for the connection between FBs. Thus, the mapping of elements from each message type can be done. Note that, the connection between FBs is achieved by XSLT transformation of original messages. The resulting new messages are the final ones that are connected with the FB destination. Note that the generation of XSLT scripts define the mapping and transformation from different information models. For depicting the connection result by XSLT transformation, following Figure 29 represents a conceptual resulting scenario in which an adapter type FBI and another FBI are connected.

Note that this mapping and transformation are directly related to the interoperability with ontologies and the utilization of ISA-95 within B2MML schemas. Note that the mapping and transformation used by the FBEC is done by another tool, working for the whole implementation architecture presented in previous Figure 17.

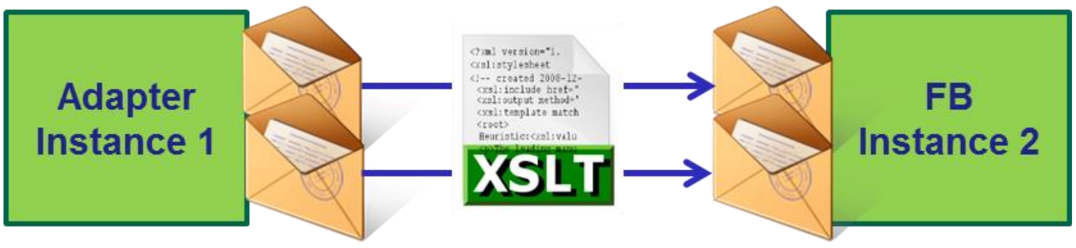


Figure 29: Connection of FBI thought XSL transformation of message types

4.2.2. Implementation of the CFBNO

Once the requirements by external tools of this ontological approach have been described, the implementation of the CFBNO can be understood. Note that this section follows the same structure of *"Implementation of the production ontology"* because it is being presented an ontology implementation using Protégé following the same methodology. Thus, the descriptions about the process and decisions for implementing the CFBNO follow the same concepts described for the MSO.

Firstly, the domain of the CFBNO has to be defined following the stated thesis work objectives for the ontology description. Then, the final scope of the CFBNO is the description of the configuration required for developing a FBN.

Once the domain of the CFBNO has been decided, the terminology required for describing the concept, classes, properties and instances, must be described. Obviously, the taxonomy is composed of the related terms that are coincident with the MSO for the interoperability between ontologies. Besides this restriction, there is more terminology to be defined as the name of FB, adapters, and configuration classes. For this reason, and following the same methodology of the MSO implementation, the following Figure 30 depicts within a Venn diagram the classes of the CFBNO.

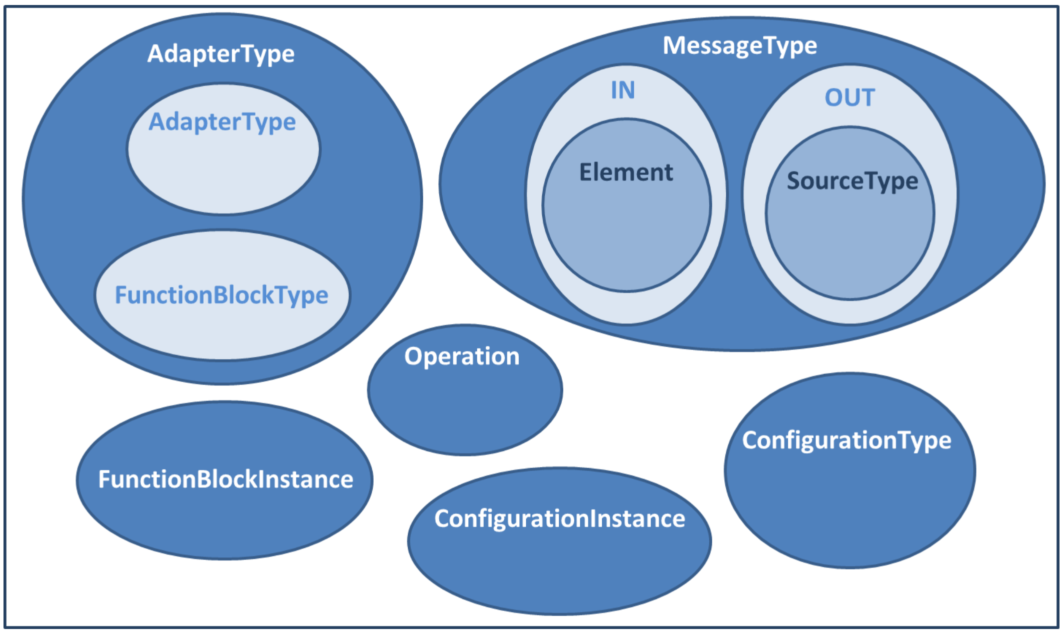
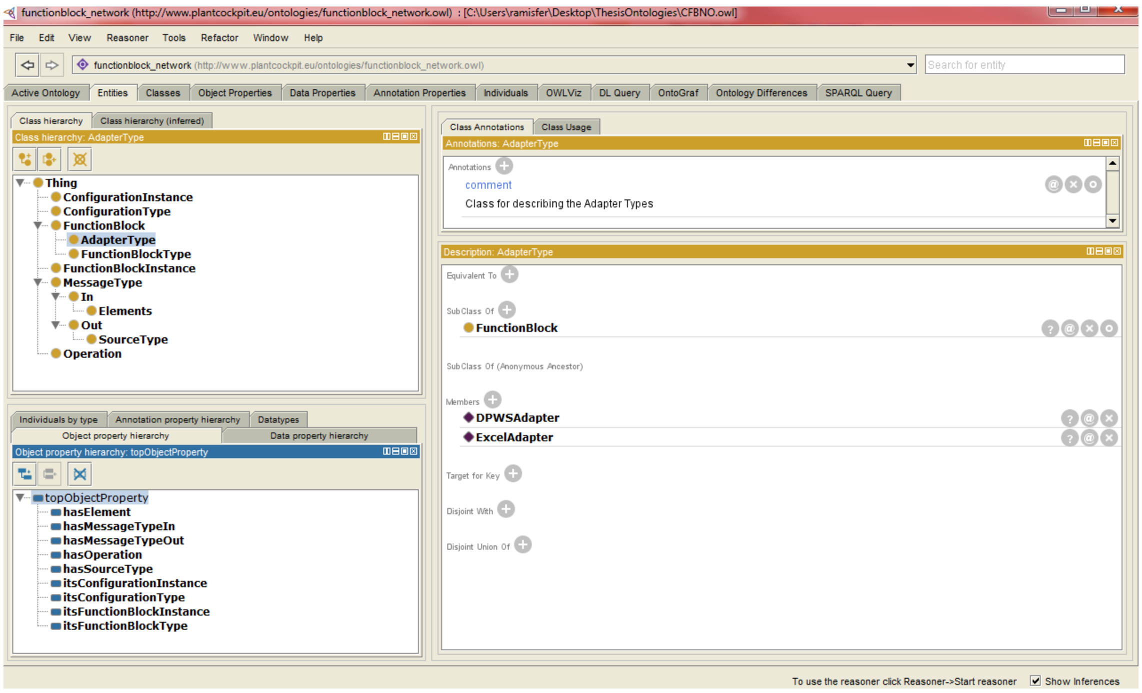


Figure 30: Venn diagram of CFBNO classes

Previous Venn diagram presents the terminology and distribution of CFBNO classes. In fact, the structuration of the classes can be divided in four groups. The first one is composed by the upper level *'FunctionBlockType'* and *'FunctionBlockInstance'* objects in which the names and types of FBs are described. Secondly, the classes *'ConfigurationInstance'* and *'ConfigurationType'* are made for the description of the FB configuration. Thirdly, the 'Operation' class describe the operations that can be managed by certain FB. Finally, the object named as *'MessageType'* describes the message types and their composition. In fact, some of these objects are related with the requirements explained in *"Requirements of the FBEC supported by the CFBNO"* section because, for instance, the classes *'ConfigurationInstance'* and *'ConfigurationType'* are for defining the JSON and JSON schema, respectively. Note that this definitions are part of the implementation specific of PLANTCockpit.

Note that previous Figure 30 shows classes that are equally named. This fact of the design can be easily understood by the explanations in following *"Interoperability between MSO and CFBNO"* section. Basically, the classes that are equally described in both ontologies allow the connection of MSO and CFBNO through these objects.

Afterwards of the terminology definition, the previously presented Venn diagram sketch allows the visualization of the hierarchy of classes that needs to be designed. The final level structuration of classes implemented in Protégé can be seen in following Figure 31.

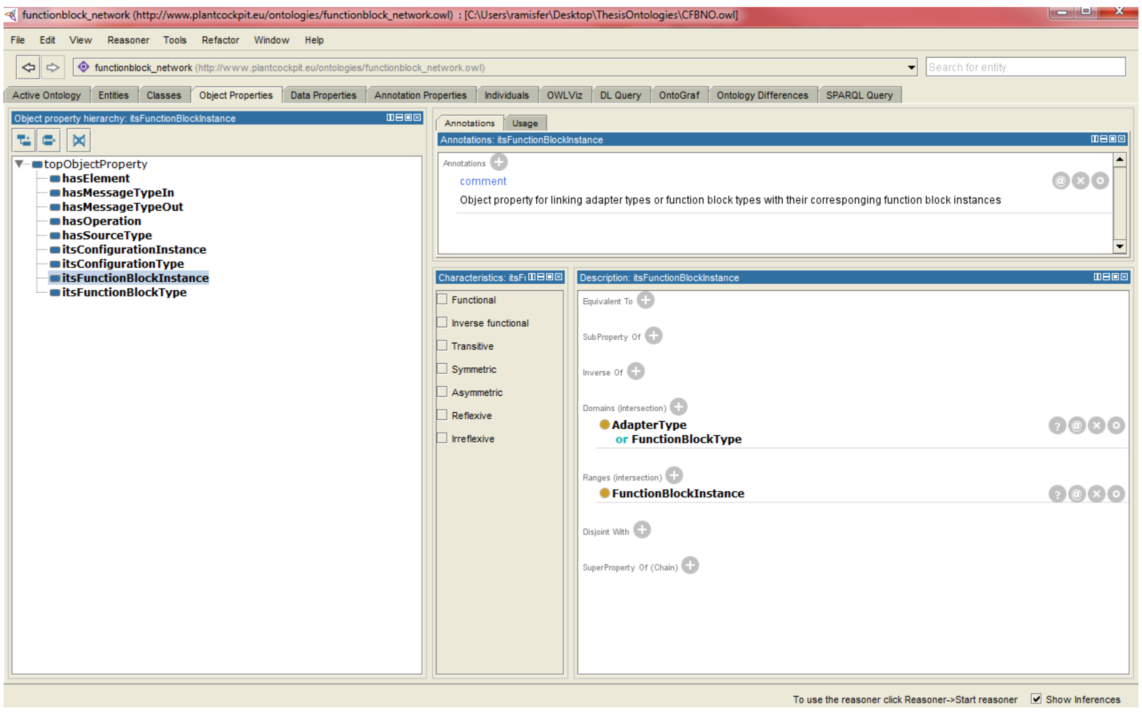

functionblock_network (http://www.plantcockpit.eu/ontologies/functionblock_network.owl) : [C:\Users\ramisfer\Desktop\ThesisOntologies\CFBNO.owl]

```
Class hierarchy: AdapterType
Thing
    ConfigurationInstance
    ConfigurationType
    FunctionBlock
        AdapterType
        FunctionBlockType
    FunctionBlockInstance
    MessageType
        In
            Elements
        Out
            SourceType
    Operation

Object property hierarchy: topObjectProperty
topObjectProperty
    hasElement
    hasMessageTypeIn
    hasMessageTypeOut
    hasOperation
    hasSourceType
    itsConfigurationInstance
    itsConfigurationType
    itsFunctionBlockInstance
    itsFunctionBlockType

Annotations: AdapterType
comment
Class for describing the Adapter Types

Description: AdapterType
Equivalent To
SubClass Of
    FunctionBlock
SubClass Of (Anonymous Ancestor)
Members
    DPWSAdapter
    ExcelAdapter
Target for Key
Disjoint With
Disjoint Union Of

To use the reasoner click Reasoner->Start reasoner   Show Inferences
```

Figure 31: CFBNO class hierarchy

Then, as it can be seen, the class hierarchy of CFBNO has been successfully implemented on Protégé. For instance, Figure 31 shows the description of the *'AdapterType'* class in which it is described its higher class and the instances that are defined of this class type. Note that the design of instances is an advanced step of the methodology so it is explained further in this section.

The following step of the design process is to define the object properties between classes. The resulting object properties of the CFBNO are presented by a capture of Protégé in following Figure 32.


functionblock_network (http://www.plantcockpit.eu/ontologies/functionblock_network.owl) : [C:\Users\ramisfer\Desktop\ThesisOntologies\CFBNO.owl]
File Edit View Reasoner Tools Refactor Window Help
functionblock_network (http://www.plantcockpit.eu/ontologies/functionblock_network.owl)
Search for entity
Active Ontology Entities Classes Object Properties Data Properties Annotation Properties Individuals OWLViz DL Query OntoGraf Ontology Differences SPARQL Query
Object property hierarchy: itsFunctionBlockInstance
Annotations Usage
Annotations: itsFunctionBlockInstance
Characteristics: itsFu
Description: itsFunctionBlockInstance
To use the reasoner click Reasoner->Start reasoner
Show Inferences

topObjectProperty
- hasElement
- hasMessageTypeIn
- hasMessageTypeOut
- hasOperation
- hasSourceType
- itsConfigurationInstance
- itsConfigurationType
- itsFunctionBlockInstance
- itsFunctionBlockType

Annotations

comment

Object property for linking adapter types or function block types with their corresponging function block instances

- [ ] Functional
- [ ] Inverse functional
- [ ] Transitive
- [ ] Symmetric
- [ ] Asymmetric
- [ ] Reflexive
- [ ] Irreflexive

Equivalent To

SubProperty Of

Inverse Of

Domains (intersection)

AdapterType
or FunctionBlockType

Ranges (intersection)

FunctionBlockInstance

Disjoint With

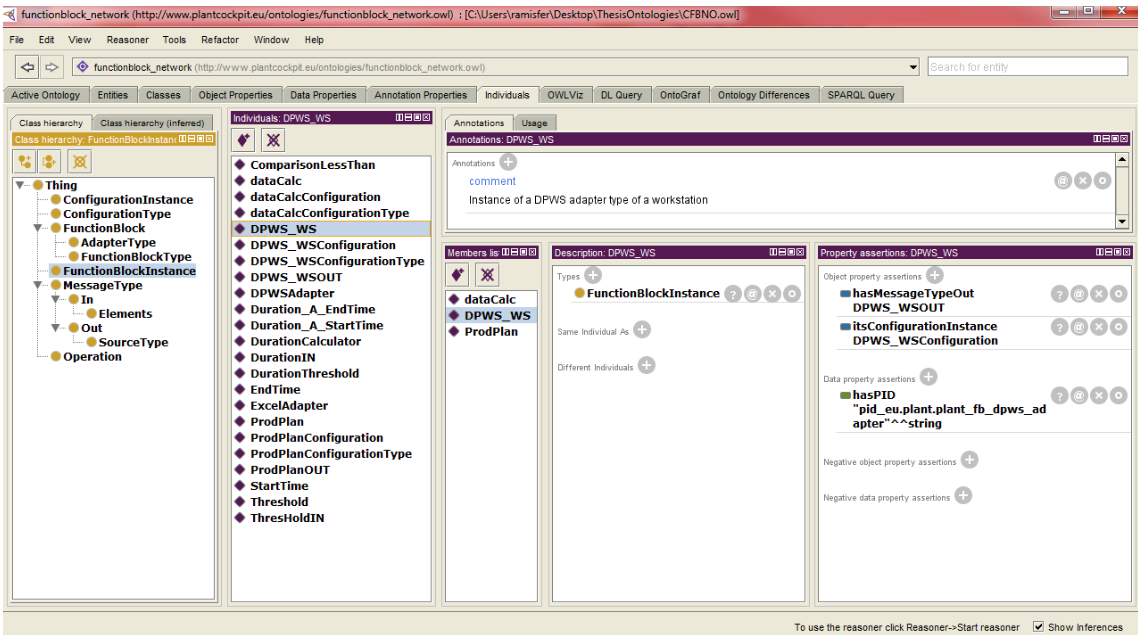
SuperProperty Of (Chain)

Figure 32: CFBNO object properties

Hence, the object properties are shown in the left of the capture. Note that there are not sub-properties of defined properties as it has been also implemented in MSO because there are not required cases for both ontologies. In addition, as an example, it can be seen that the object property *'itsFunctionBlockInstance'* is selected so its description is shown in the right part of the capture. In fact, this description depicts the next step of the methodology which is the addition of additional properties to the already defined object properties.

Then, it can be seen that this object property is a connection between *'AdapterType'* or *'FunctionBlockType'* and *'FunctionBlockInstance'*. The definitions of the *'Domain'* and *'Ranges'* characteristics are needed for define this properties direction because of satisfying the CFBNO consistency.

Now, the ontology is prepared to be filled with instances for representing the real components that are going to be used for the configuration of the FBN. The defined instances for achieving a FB composition are shown in following Figure 33. Note that the presented instances on following capture are needed for performing only one scenario, which consists in the calculation of a KPI through the connection between *'DWPWS_WS'* and *'ProdPlan'* as adapter type instances and a *'dataCalc'* as FBI type.


File Edit View Reasoner Tools Refactor Window Help
Search for entity
Active Ontology | Entities | Classes | Object Properties | Data Properties | Annotation Properties | Individuals | OWLViz | DL Query | OntoGraf | Ontology Differences | SPARQL Query
Class hierarchy | Class hierarchy (inferred)
Annotations | Usage
To use the reasoner click Reasoner->Start reasoner
Show Inferences

functionblock_network (http://www.plantcockpit.eu/ontologies/functionblock_network.owl) : [C:\Users\ramisfer\Desktop\ThesisOntologies\CFBNO.owl]

Class hierarchy: FunctionBlockInstance

- Thing
  - ConfigurationInstance
  - ConfigurationType
  - FunctionBlock
    - AdapterType
    - FunctionBlockType
  - FunctionBlockInstance
  - MessageType
    - In
      - Elements
    - Out
      - SourceType
  - Operation

Individuals: DPWS_WS

ComparisonLessThan, dataCalc, dataCalcConfiguration, dataCalcConfigurationType, DPWS_WS, DPWS_WSConfiguration, DPWS_WSConfigurationType, DPWS_WSOUT, DPWSAdapter, Duration_A_EndTime, Duration_A_StartTime, DurationCalculator, DurationIN, DurationThreshold, EndTime, ExcelAdapter, ProdPlan, ProdPlanConfiguration, ProdPlanConfigurationType, ProdPlanOUT, StartTime, Threshold, ThresHoldIN

Annotations: DPWS_WS

Annotations
comment
Instance of a DPWS adapter type of a workstation

Members lis: dataCalc, DPWS_WS, ProdPlan

Description: DPWS_WS

Types
FunctionBlockInstance
Same Individual As
Different Individuals

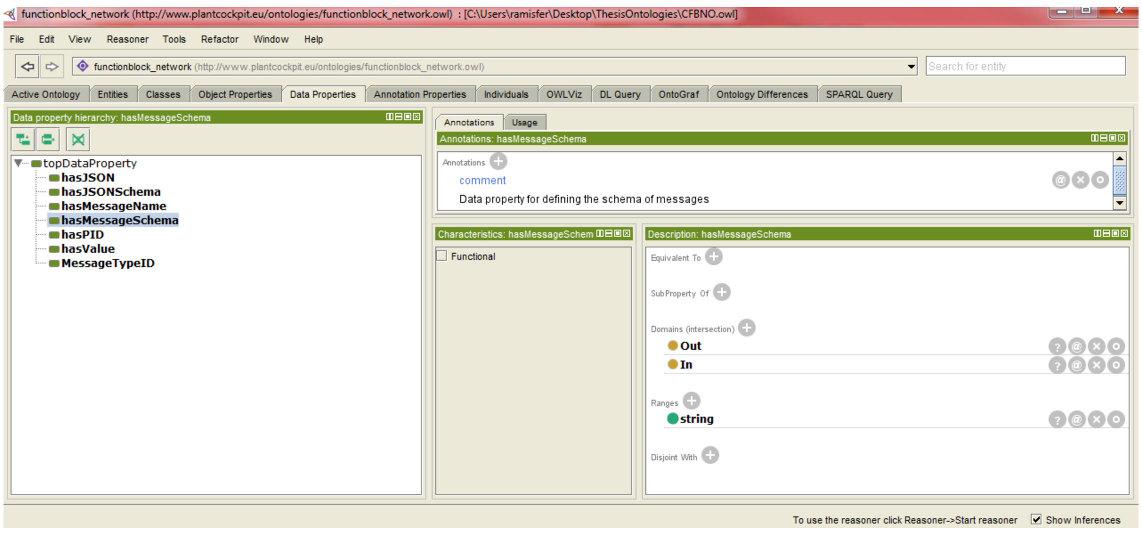
Property assertions: DPWS_WS

Object property assertions
hasMessageTypeOut DPWS_WSOUT
itsConfigurationInstance DPWS_WSConfiguration
Data property assertions
hasPID "pid_eu.plant.plant_fb_dpws_adapter"^^string
Negative object property assertions
Negative data property assertions

Figure 33: CFBNO instances

Then, previous Figure 33 shows the CFBNO instances. As an example, an instance of the class *'FunctionBlockInstance'* is selected. Thus, it is shown how the individual named as *'DPWS_WS'* is an instance of a DPWS adapter type of a workstation. In addition, it can be seen that this instance is related with two instances. Firstly, it is connected with another instance named as *'DPWS_WSOUT'* for describing its message type. Secondly, is connected with an individual named as *'DPWS_WSConfiguration'* for describing the configuration of the FBI.

Also, previous Figure 33 shows the definition of a data property named as *'hasPID'*. The defined CFBNO data properties are presented in following Figure 34. Note that the definition of these properties is developed in the last stage of the followed methodology for ontology design [34].


File Edit View Reasoner Tools Refactor Window Help
Search for entity
Active Ontology | Entities | Classes | Object Properties | Data Properties | Annotation Properties | Individuals | OWLViz | DL Query | OntoGraf | Ontology Differences | SPARQL Query
Annotations | Usage
To use the reasoner click Reasoner->Start reasoner
Show Inferences

functionblock_network (http://www.plantcockpit.eu/ontologies/functionblock_network.owl) : [C:\Users\ramisfer\Desktop\ThesisOntologies\CFBNO.owl]

Data property hierarchy: hasMessageSchema

- topDataProperty
  - hasJSON
  - hasJSONSchema
  - hasMessageName
  - hasMessageSchema
  - hasPID
  - hasValue
  - MessageTypeID

Annotations: hasMessageSchema

Annotations
comment
Data property for defining the schema of messages

Characteristics: hasMessageSchem

Functional

Description: hasMessageSchema

Equivalent To
SubProperty Of
Domains (intersection)
Out
In
Ranges
string
Disjoint With

Figure 34: CFBNO data type properties

It can be seen, as an example, that the data property named as ‘hasMessageSchema’ is defined as a property that can be added to *‘In’* and *‘Out’* message type instances. In addition, note that the data property type is string type.

Finally, for concluding this section, following Figure 35 presents a part of the designed CFBNO. As it has been shown also in MSO implementation, the visualization of the designed ontology can be checked with the *‘OntoGraf’* window, which permits the expansion of all the components of the ontology.

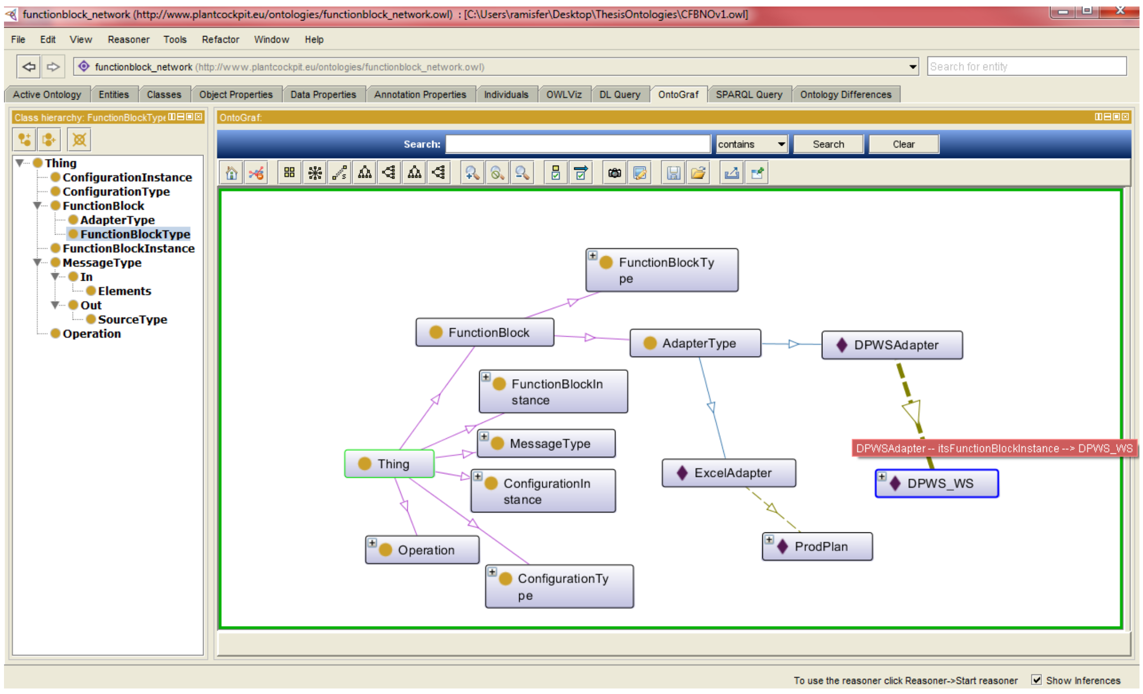

functionblock_network (http://www.plantcockpit.eu/ontologies/functionblock_network.owl) : [C:\Users\ramisfer\Desktop\ThesisOntologies\CFBNOv1.owl]
File Edit View Reasoner Tools Refactor Window Help
functionblock_network (http://www.plantcockpit.eu/ontologies/functionblock_network.owl)
Search for entity
Active Ontology Entities Classes Object Properties Data Properties Annotation Properties Individuals OWLViz DL Query OntoGraf SPARQL Query Ontology Differences
Class hierarchy: FunctionBlockType
Thing
ConfigurationInstance
ConfigurationType
FunctionBlock
AdapterType
FunctionBlockType
FunctionBlockInstance
MessageType
In
Elements
Out
SourceType
Operation
OntoGraf
Search:
contains
Search
Clear
FunctionBlockType
FunctionBlock
AdapterType
DPWSAdapter
FunctionBlockInstance
MessageType
Thing
ConfigurationInstance
ExcelAdapter
DPWSAdapter -- itsFunctionBlockInstance --> DPWS_WS
DPWS_WS
ProdPlan
Operation
ConfigurationType
To use the reasoner click Reasoner->Start reasoner
Show Inferences

Figure 35: Verifying the CFBNO design with the *‘OntoGraf’* window of Protégé

Besides the representation of classes and instances, Figure 35 shows as an example of property between individuals the *‘itsFunctionBlockInstance’*, which is connecting the adapter *‘DPWSAdapter’* instance with the *‘DPWS_WS’* FBI. Note that the capture only shows a piece of the CFBNO design due to the extension that the entire ontology would occupy.

4.3. Interoperability between MSO and CFBNO

Once the implementations of MSO and CFBNO have been presented, the explanation of their interoperability can be understood. Note that the interoperability between ontologies is the key of the implementation that allows MSO and CFBNO to work together to achieve the thesis work common goal. Then, this section describes how the performance of a FBN for KPI calculation trough two different domain ontologies is possible.

For supporting the interoperability explanation of this section, it can be seen how Figure 36 depicts the components that must be highlighted from the ontologies implementation.

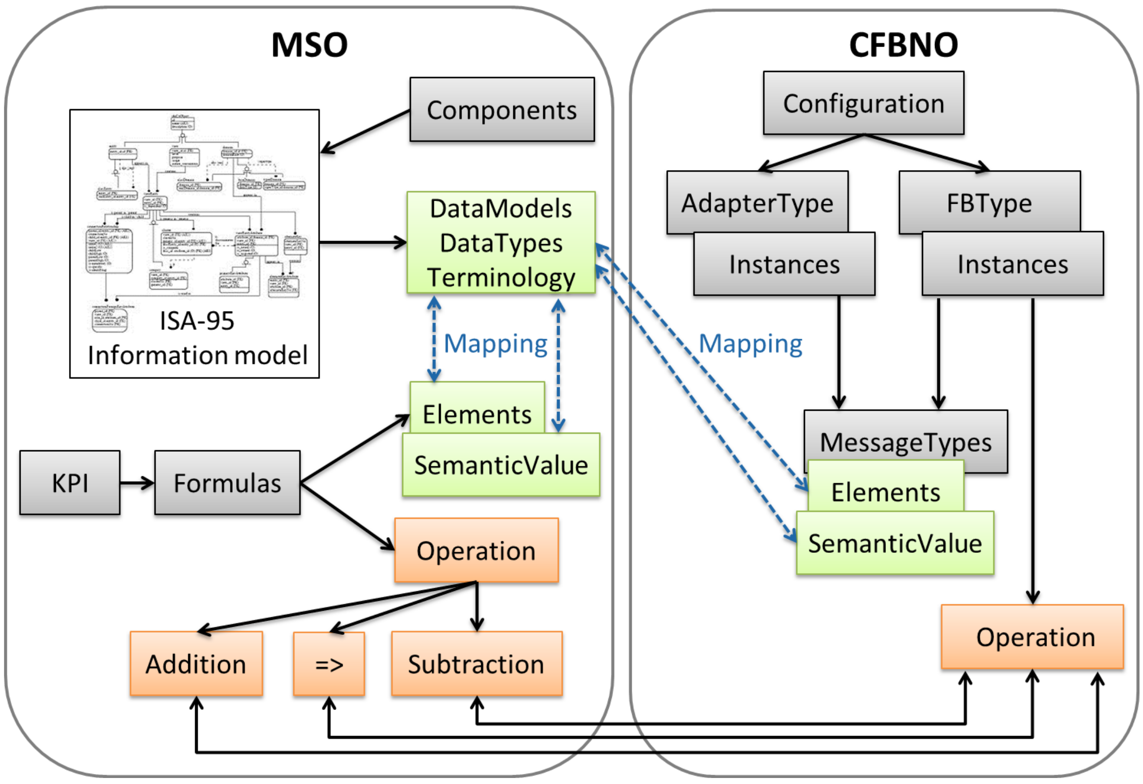


Figure 36: Interoperability between MSO and CFBNO implementation

In fact, the comparison of the presented MSO and CFBNO implementations with the previous picture helps to understand the reason for some definition of classes. Note that not all the blocks, as *"ISA-95 Information model"*, are defined components of ontologies but allows the reader to understand the domain and interoperability of MSO and CFBNO.

The explanation of this section, for the interoperability description, begins with the interpretation of the MSO object roles for linking both ontologies. Afterwards this analysis, the same explanation for CFBNO class roles is done. Then, within these section explanations the accomplished interoperability on this thesis work implementation is finally described.

4.3.1. MSO classes description for interoperability

In the top of the MSO design of Figure 36, it can be seen how the components of the manufacturing system are linked with the ISA-95 information model. This means that the defined components of, for instance, a shop floor line follow the standard. Note that *'Components'* is a class that could include systems described on the ISA-95 levels presented in Figure 8. The reason for using ISA-95 taxonomy is because, as it is also shown in the picture, the data models, data types and terminology will be retrieved from ISA-95 models. Note that, as it has been explained in *"B2MML"* section, the B2MML is composed by a set of XML schemas, allowing the data models implementation in ISA-95. Thus, B2MML can be used for the ISA-95 implementation.

On the other hand, Figure 36 shows that a KPI is associated to a formula of the *'Formulas'* block that, at the same time, is structured in elements and operations. Meanwhile the *'Elements'* block contain all the elements of the formulas, the *'Operation'* block includes all the possible operations that are processed in the KPI formula.

In addition, the elements have a semantic value that is associated with the ISA-95 data types. This means that the elements of the formula have to be defined by following the semantics of the standard as the decided taxonomy. Note that this association is done in the implementation by the explained data type named *'hasISA95Mapping'* in the instances that are defined in the class named *'SourceType'*.

4.3.2. CFBNO classes description for interoperability

In the top of the CFBNO design of Figure 36, it can be seen how the adapter and FB types have a configuration. This configuration block includes the *'ConfigurationInstance'* and *'ConfigurationType'* classes, in which the JSON and JSON schema are defined as it is described in the previous CFBNO implementation explanation. In addition, the picture shows that the Adapter and FB type instances are associated to message types.

Firstly, the adapter type instances are linked with the message types that can be supported by adapters in the FBN scenario. In the same way, FB type instances are also connected to message types. Note that this message types are different depending on the instance type. In fact, as the connections are always from adapters type instances to FB type instances, the message type of adapters is always *'Out'* and the FB message types is always *'In'*, for the KPI calculation FBN scenario. Hence, note that the *'Elements'* are associated to *'In'* message types and *'SourceType'* are associated to *'Out'* message types.

Secondly, as the FB type instance has the role of calculate the KPI from the values of components by using certain operations, they are also linked with the *'Operation'* class which is a replica of the MSO *'Operation'* class.

4.3.3. Ontologies connection through defined classes

Now that all the classes with a role for the interoperability have been described, the ontologies connection can be explained.

Basically, the most important connection is the blue square dot connection which connects the *'Elements'* and *'SemanticValue'* classes with the ISA-95 retrieved information block. Hence, by doing the mapping of the semantic values of elements, the message types of each component can be reasoned. This means that the adapter type that manages this information model can be found through the system. For this reason, the description of elements within a standard is critical for this implementation since is the link of reasoning between MSO and CFBNO.

In addition, for the calculator FB type, the KPIs operations are important for solve the equations. Hence, a last link for the ontologies to working jointly, the instances of FB types, must be related with the operations of the MSO.

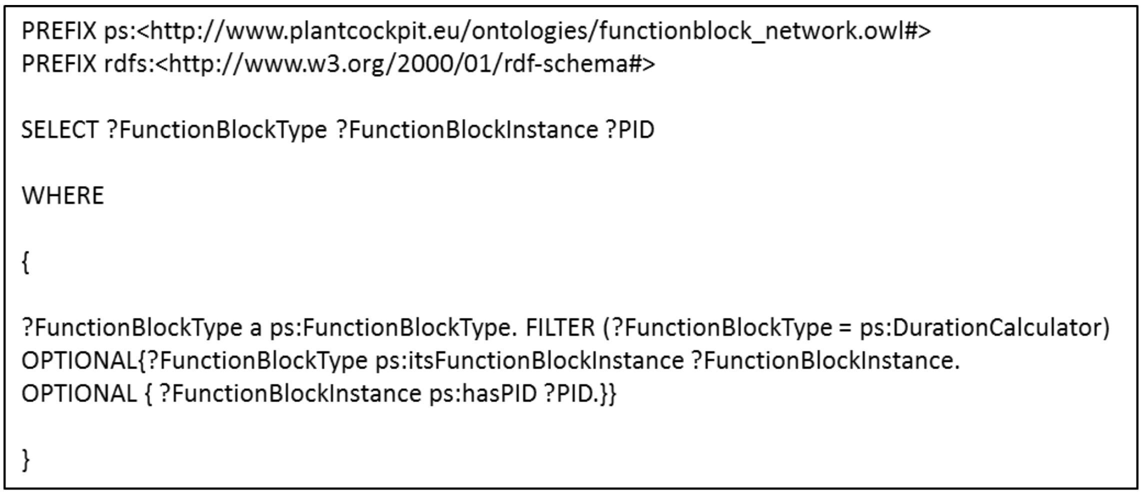
Then, by following this section description, the interoperability between ontologies is successfully implemented. Hence, the common goal of FBN design for KPI calculation can be performed.

4.4. Reasoning ontologies

Once the implementation of the ontologies and its interoperability is presented, the reasoning of MSO and CFBNO can be described. For explaining the implementation of queries in this ontological approach, this section presents a query, which is used for the reasoning, and its result. The presented example uses the patterns and style of all the used queries so that it is adequate for understanding the functionality of queries for the reasoning of MSO and CFBNO.

Note that the reasoning of the ontologies is done by a configuration support external to this ontological approach. However, the testing and query implementation has been made in Protégé through SPARQL queries. Then, when the functionality of queries have been verified, they are included in the configuration support for automating the set of queries that has to be used for retrieving all the required information for the performance of the FBN.

Following Figure 37 shows an example of query used in this implementation. In fact, the example shows a query used in the sixth step of the sequence for CFBNO reasoning.

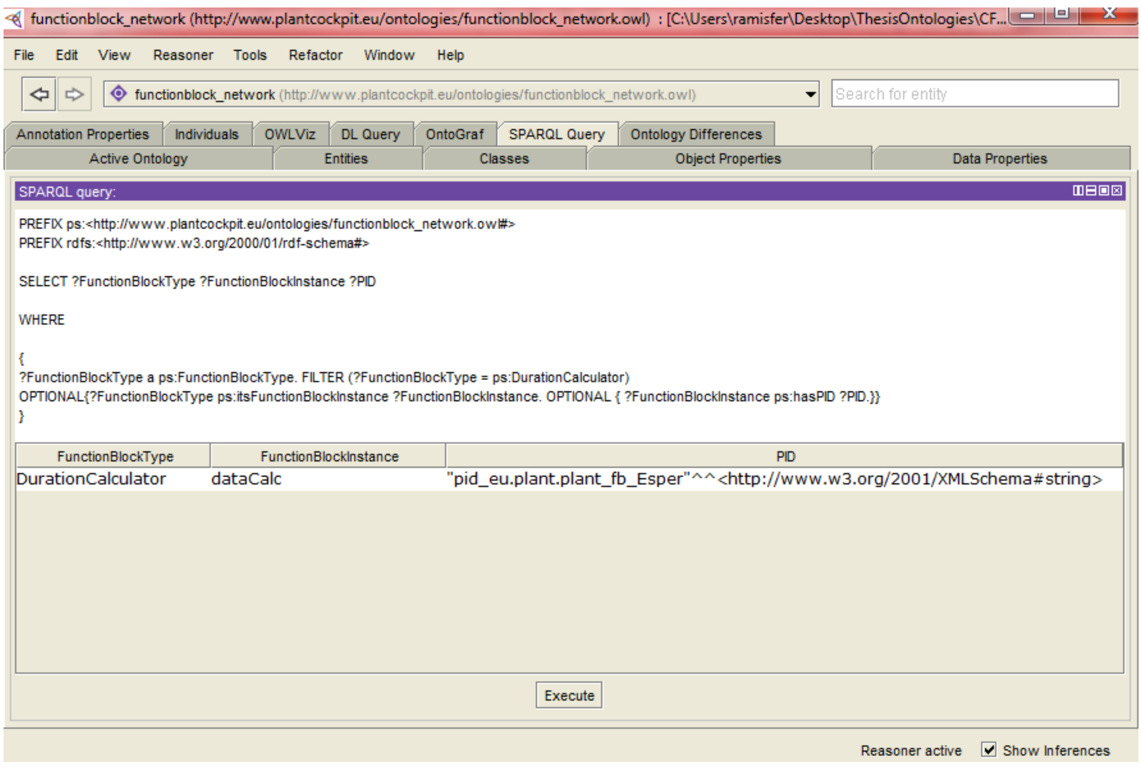


```
PREFIX ps:<http://www.plantcockpit.eu/ontologies/functionblock_network.owl#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>

SELECT ?FunctionBlockType ?FunctionBlockInstance ?PID

WHERE

{

?FunctionBlockType a ps:FunctionBlockType. FILTER (?FunctionBlockType = ps:DurationCalculator)
OPTIONAL{?FunctionBlockType ps:itsFunctionBlockInstance ?FunctionBlockInstance.
OPTIONAL { ?FunctionBlockInstance ps:hasPID ?PID.}}

}
```

Figure 37: Query for retrieving information of a FB type

As it can be seen in previous Figure 37, the query tries to retrieve the information of a FB type. Then, the results that are expected are the PID and the instance name of the FB type. Note that the FB type name has been added as a fixed name because this query is triggered when the configuration support already knows that this FB type is the one from which the information must be retrieved. For understanding the sequence order and processed data in each step, the *"Sequence for FBN performance"* section presents the results of each reasoning step.

Once queries are implemented, Protégé allows its verification through different reasoners. In this case the Pellet reasoner has been used. The results of the previous query for the CFBNO are presented in following Figure 38.



```
PREFIX ps:<http://www.plantcockpit.eu/ontologies/functionblock_network.owl#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>

SELECT ?FunctionBlockType ?FunctionBlockInstance ?PID

WHERE

{
?FunctionBlockType a ps:FunctionBlockType. FILTER (?FunctionBlockType = ps:DurationCalculator)
OPTIONAL{?FunctionBlockType ps:itsFunctionBlockInstance ?FunctionBlockInstance. OPTIONAL { ?FunctionBlockInstance ps:hasPID ?PID.}}
}
```

FunctionBlockType	FunctionBlockInstance	PID
DurationCalculator	dataCalc	"pid_eu.plant.plant_fb_Esper"^^<http://www.w3.org/2001/XMLSchema#string>

Figure 38: Results in Protégé of a query

Then, the result of the query satisfies the expectations, which means that the ontology and the query are well formed. In previous caption, it can be seen that the results are given by tables that shows instances that satisfy the executed queries as it is explained in *"SPARQL"* section. Then, the PID *'pid_eu.plant.plant_fb_Esper'* and instance name *'dataCalc'* of the FB type named *'DurationCalculator'* are found. In the same way, information of the configuration in CFBNO or KPI and manufacturing system components in MSO are retrieved.

For concluding this example of the implemented queries for reasoning MSO and CFBNO, it has to be noted the importance of filters in SPARQL queries. It is demonstrated that use of SPARQL patterns as *'FILTER'*, or *'OPTIONAL'* adds reasoning to the queries since it allows the reasoner to narrow the right choices. However, note that SWRL rules can be also used for increasing the reasoning but for this implementation it has been not needed for retrieving the desired results. In addition, the utilization of SPARQL Update can be useful in this type of implementation for creating, deleting or modifying components of the ontology.

Once the query style and the process for verify its result for the implementation is described, the queries are implemented to a configuration support. Then, all the queries are automatically triggered and reasoned in the described sequence in following section so that the FBN can be finally designed. In addition, the configuration support is in charge of parsing the information of the queries matching cases for extracting each result of the table because the information cannot be treated directly from the tables. Note that this configuration support is needed for connecting this ontological approach with the external tools and reach the desired results.

5. RESULTS

Once the implementation of this thesis work has been shown, this chapter presents the achieved results for the ontological approach testing. Then, this section offers the resulting information within empirical study for demonstrating that the proposed goals have been achieved, determining a satisfactory performance of the realized implementation.

For the presentation of results, it is described the required sequence of queries distributed in steps. The first steps of the sequence are used for retrieving all the information of the ontologies. These queries follow the style and results presented in previous section. In addition, the last steps of the sequence are done by the external tools of this ontological approach for performing the FBN. For concluding this section, a resulting scenario of FBN for KPI calculation by using this ontological approach is shown by a capture of the FBEC visualization.

5.1. Sequence for FBN performance

This section presents a sequence for performing the FBN for KPI calculation. The intention of this description is to demonstrate how the presented implementation is used and to show the resulting FBN.

As it has been explained previously, this sequence is managed by a configuration support which sends queries to the ontologies and receives the response. Then, the information is processed for following queries, if needed, and the sequence continues till the end. Following Figure 39 depicts how this interaction can be understood.

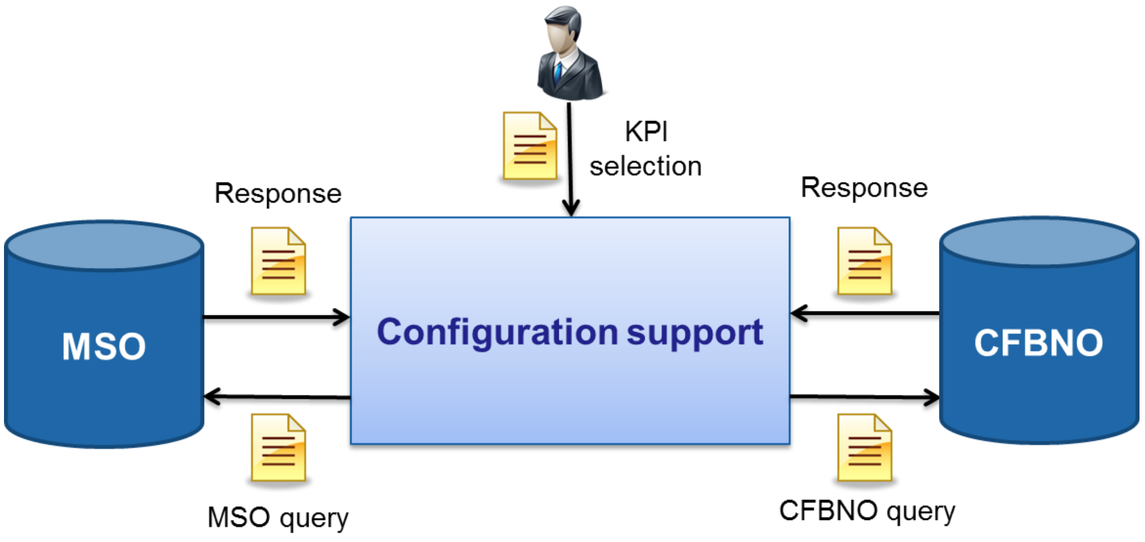


Figure 39: Interactions of the configuration support with the ontologies and user

As it can be seen in previous picture, there are two possible cases of querying. The interaction of the left side is the process for retrieving information from the MSO. On the other hand, the interaction of the right side corresponds to the query sequence for retrieving information from the CFBNO. In fact, MSO is the first ontology queried because the configuration support requires its information for querying the CFBNO. Note that the first action to start the process is a request of a user, which is the selection of the KPI that has to be calculated.

5.1.1. Reasoning of MSO

As it has been introduced for previous section, the sequence starts with the selection of a KPI to be calculated. Once the user introduces to the system the KPI, the sequence of queries can be started by the configuration support. The first ontology to be queried is the MSO.

Following Table 9 lists by order the first three SPARQL queries that the configuration support sends to the MSO. As information, the table presents the requested MSO data asked within the queries. Then, the results are processed by order so that the configuration support can perform further sequence steps.

Table 9: First three queries of the sequence for the MSO reasoning

Query	Requested data
1	Formula and its elements for the selected KPI
2	Source type of the elements according to ISA-95 standard
3	Rest of the formula components: Constants, operations and symbols

Then, it can be seen that after the KPI selection, MSO is asked within three queries for retrieve information. Note that the presented order of queries is always followed by the configuration support.

Firstly, the instance for the KPI formula is requested. Also, in the same query, the formula elements are requested. The objective of this query is to support the instances of elements for the third query.

Secondly, the source types of each element of the formula are requested. In addition, the data type that describes the ISA-95 data type, named as *'hasISA95Mapping'*, is asked in the same query. The objective of this query is that its result supports the required information to be mapped, allowing the interoperability with the CFBNO. Thus, the message types of the CFBNO can be requested by using these elements.

Thirdly, the rest of the formula components are retrieved. This means that the constants, operations and symbols are retrieved. Thus, within the third query, all the required information of the KPI formula is obtained. The objective of this query is to support the operations to the CFBNO for requesting an available calculator FB to perform this operation. In addition, by adding the data type property of *'hasPositionInFormula'* in this query, the position of each component of the formula can be known so the formula can be expressed in the right order.

5.1.2. Reasoning of CFBNO

As it has been introduced previously, the sequence continues for reasoning the CFBNO. Note that as it is below explained, some queries of the CFBNO reasoning uses the results from the data retrieved from the MSO.

Following Table 10 lists by order the rest of the SPARQL queries that the configuration support sends to the CFBNO for conclude the reasoning sequence. As in Table 9, the following Table 10 presents the requested data requested within the queries. Note that the queries are processed by order. In this way, the configuration support can perform further sequence steps till achieving the last query, in which all the required data for the FBN configuration is retrieved.

Table 10: Queries of the sequence for the CFBNO reasoning

Query	**Requested data**
3	Message types and its elements related to ISA-95 mappings
4	Adapter type related to the instance that provides the previous message types
5	Adapter configuration: PID, JSON and JSON schema
6	FB type related to the instance that provides the operator of the formula
7	FB configuration: PID

Then, it can be seen that after the data information retrieving of MSO, the sequence continues by doing seven five queries to the CFBNO. Thus, the ontology is asked for retrieving all the useful data for the configuration of the FBN for the selected KPI calculation.

Firstly, the message types related to the elements of the formula are retrieved. Then, this query uses the results of the first and second queries presented in Table 9. The objective of this query is to find the messages types that contain these elements. This means that the result of this query can be used for finding the adapter types of these messages.

Then, the second query of previous table uses the received message types for requesting the adapter instance and its type. The adapter types are the first FBI that is represented in the visualization of the FBN. As it is explained in previous sections, these FBIs have message types which consist in the elements needed by a calculator for calculating a selected KPI.

Once the adapter type and its instance are retrieved, the next step is to request the configuration. Then, the third query sent to the CFBNO asks for the PID, JSON and JSON schema. Note that within this query all the configuration for required adapter types is retrieved so that the following queries must retrieve the calculator information.

Hence, the following query of previous Table 10 uses the result from the third query of Table 9 for retrieving the FB type and its instance, which is related to the operation of the KPI formula retrieved from MSO.

Finally, the last query requests the configuration of the retrieved FB instance in previous query. Then, the PID is requested. Once this query is successfully answered, the configuration support finalizes the sequence of queries.

Afterwards this process, the required data from the ontologies for performing the FBN is retrieved. Then, as it is introduced by previous sections, within the external tools the mapping and performance of the FBN for KPI calculation is finally accomplished.

5.1.3. Final steps for FBN performance

The last pre-results of the implementation are described in this section. Note that once the information of the ontologies is retrieved, it has still to be treated for perform the desired FBN.

Note that the remaining tasks are performed by external tools, following Table 11 shows the remaining steps for having the final result. In addition, note that this is a thesis that intends to present an ontological approach for manufacturing systems implementation. This means the IT details are not covered by these explanations, since the external applications are not made in this thesis work. However, the results must present the utilized tools because they are used for the FBN performance.

Table 11: Remaining steps for the FBN implementation

Step	Description
1	Generation of data calculator instance
2	Generation of transformation scripts
3	Generation of the connections between FBI
4	Generation of the XML model of the FBN scenario
5	Load the FBN to the FBEC

First, the generation of data calculator instance is done. For this creation, the formula elements are used to be the input of the FB and, then the operators and variables are used for editing its processing business logic template. This means the edition of the JSON. Note that following Figure 40 depicts the business logic of FBIs within the BL indicator. In addition, the following picture is used for following descriptions.

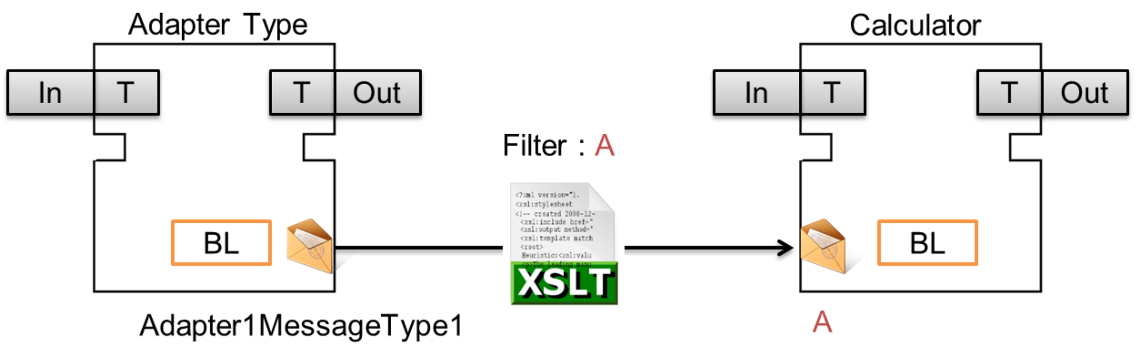


Figure 40: Connections between generated FBIs

Secondly, the generation of transformation scripts has to be done. As it is explained in *"Function block engine configurator"* section, this is the step for performing the mapping of data and the XSLT transformation for the following definition of FBI connections.

Then, the following step is to generate the connections between FBI. This is performed considering that the message flows always are from adapters to FB calculators. In addition, it is stated that the XSLT scripts transforms an original message to a target message so the message filter can be inferred to be the target message type. Note that this filtering is also depicted by previous Figure 40.

Once the connections are generated, the scenario is totally performed. Then, a generation of the FBN in XML is done. Following Figure 41 depicts what is the information that has been generated its transformation into a XML model.

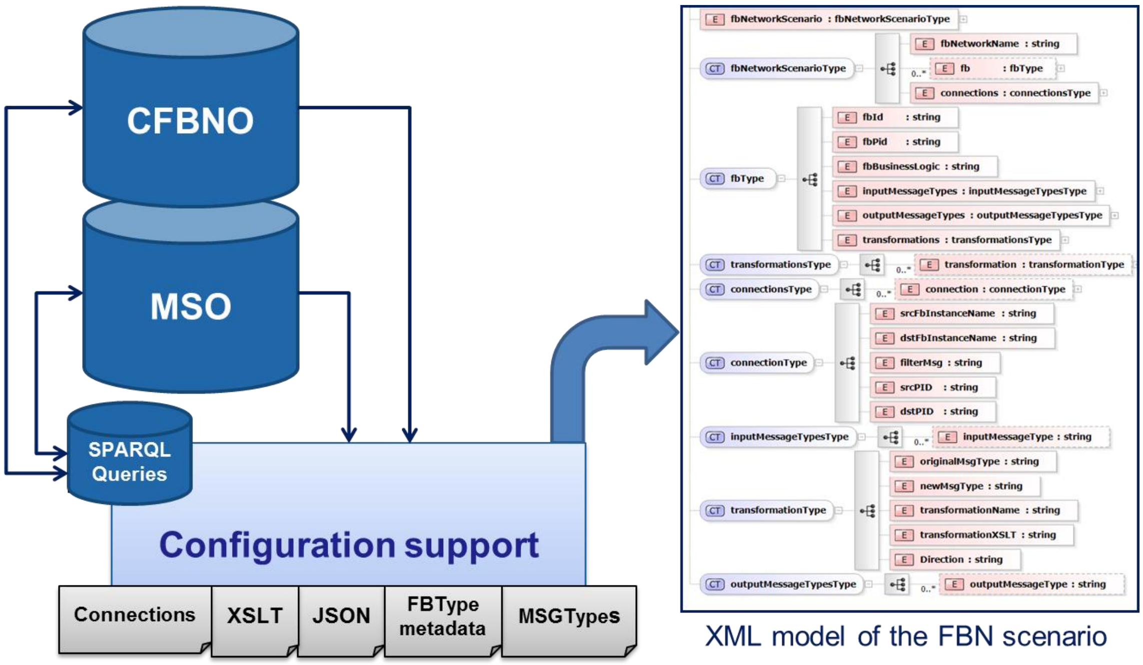
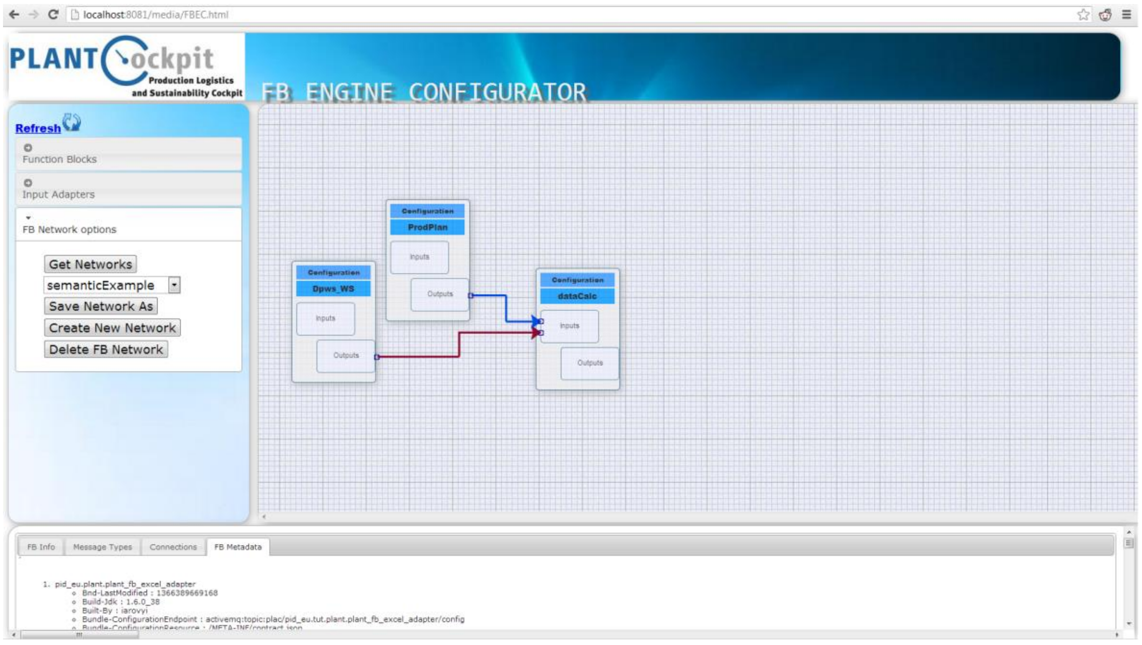

CFBNO
MSO
SPARQL Queries
Configuration support
Connections
XSLT
JSON
FBType metadata
MSGTypes
fbNetworkScenario : fbNetworkScenarioType
fbNetworkName : string
fbNetworkScenarioType
fb : fbType
connections : connectionsType
fbId : string
fbPid : string
fbBusinessLogic : string
fbType
inputMessageTypes : inputMessageTypesType
outputMessageTypes : outputMessageTypesType
transformations : transformationsType
transformationsType
transformation : transformationType
connectionsType
connection : connectionType
srcFbInstanceName : string
dstFbInstanceName : string
connectionType
filterMsg : string
srcPID : string
dstPID : string
inputMessageTypesType
inputMessageType : string
originalMsgType : string
newMsgType : string
transformationType
transformationName : string
transformationXSLT : string
Direction : string
outputMessageTypesType
outputMessageType : string
XML model of the FBN scenario

Figure 41: XML model of the FBN scenario generation

It can be seen in previous Figure 41 how some of the retrieved information gathered by the configuration support. The resulting XML model is the file that can be loaded to the FBEC. Hence, the final step is to load the generated XML model to the FBEC. Then, the FBN for KPI calculation can be visualized in the FBEC interface. Note that the XML model for the FBN scenario is attached on the *"Appendix B"* section.

5.2. Resulting FBN for KPI calculation

Once all the results of the reasoning sequence are processed and the XML model is created as it has been depicted in previous Figure 41, the scenario can be loaded to the FBEC. Then, as the final result of this thesis work, following Figure 42 shows a performed FBN for KPI calculation with this ontological approach.



1. pid_eu.plant.plant_fb_excel_adapter
   - Bnd-LastModified : 1366389669168
   - Build-Jdk : 1.6.0_38
   - Built-By : iarovyi
   - Bundle-ConfigurationEndpoint : activemq:topic:plac/pid_eu.tut.plant.plant_fb_excel_adapter/config
   - Bundle-ConfigurationResource : /META-INF/contract.json

Figure 42: Resulting FBN for KPI calculation using the ontological approach

Then, it is depicted by previous Figure 42 that a FBN for KPI calculation is successfully performed by the configurator. Now, the KPI can be calculated since the DPWS and Excel adapter types instances have the values for the calculation and the calculator FB type instance can solve the equation. Thus, the required values for calculate the duration of product for threshold detection on Figure 25 can be used. Note that meanwhile the *'Dpws_WS'* supports the FBN with *'endTime'* and *'startTime'* retrieved from a workstation; the *'ProdPlan'* supports the threshold value retrieved from an *'Excel'* file.

Again, note that the XML model for the FBN scenario is attached on the *"Appendix B"* section.

6. CONCLUSIONS

This chapter concludes the thesis work documentation, discussing the important of the results and the achieved approaches once the implementation has been finished. In addition, further work on this ontological approach and possible utilities of this thesis work are described and presented.

The implementation of this thesis work allowed to present two conference papers [40 and 58] and a journal is also proposed. Then, this thesis work has been already valued for the community. Note that these publications have been always presented as a research founded by the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 260018 (PLANTCockpit Production Logistics and Sustainability Cockpit).

6.1. Conclusions of implementation and results

Firstly, it can be concluded that this thesis implementation approach the IT to the manufacturing system field with the objective to demonstrate that the reduction of integration efforts is possible with, for instance, ontological approaches. This means that the KR of ontologies and the dynamic data integration of external tools is a powerful methodology for implementing FBN scenarios for KPI calculation.

In fact, this thesis work permits the use of manufacturing system models as source of information during the integration of heterogeneous systems. Then, a process expert can describe, in example, a shop floor to be configured within an ontological approach, which can support its knowledge to other interconnected systems.

Thus, the methodology approach of this thesis work permits shop floor experts to face successfully difficulties in data integration. Obviously, as it is explained through the documentation, the need of a system integration expert is also needed, but this is the reflection of the reduction of efforts that this thesis work proposes for achieving a common goal.

On the other hand, the standardization of the proposed alternative for solving complex data integrations in systems is the most important task of the implementation. This means that the key for the ontological approach functionality is the standardization of systems to let the ontologies to be interoperable. Thus, note that the importance of the well definition of source types, following the standard, by experts is a critical task for the functionality of the thesis work.

In fact, another conclusion of the proposed ontological approach is that the use of other standards would be also possible. This means that, for instance, if the use of ISA-95 for defining source type is not possible due to the knowledge of experts, other standard could be used. In example, BACnet [67] or MTConnect [68] are standards used in manufacturing systems that could be used for this implementation [69 and 70]. Note that this means that the ontological approach is flexible because permits the change of technologies. For implementing this ontological approach with other standards, the experts have just to define the source types in the same protocol and the system can perform the same resulting FBN. In addition, the advantage of this implementation is that queries and interoperability would be possible to work following the same methodology. However, for quality performance, the property which includes the 'ISA95' word should be changed by the name of the used standard.

Moreover, note that an interesting fact discovered in this thesis work is that the devices description is not important for reach the solution. The reason is that the most important data for the KPI calculation is the information model of devices. Then, the ontology knowledge of components is not used for the data integration since this information only permits the description of systems. This is the reason why adapters are not defined in MSO since they are not relevant for the information that is retrieved by the configuration support for performing the FBN.

However, the modeling of the system is an objective of the thesis so that it has been done. In fact, it can be useful, in example, for maintenance or control of the equipment. This means that, once the system is modeled, the MSO can be queried to verify the status of the shop floor.

Finally, it has been discussed the reasoning within queries concept. Implementations that depend on the reasoning of KBs use queries for retrieving the information. However, the use of these queries can confuse the developer enclosing part of the reasoning in the same queries. This fact is problematic since the queries must be just a way of retrieving information. This means that systems that depends on reasoning by rules, or queries forces that the knowledge is stored in this axioms, so that part of the whole KR is not in the integration system.

6.2. Further work

Firstly, note that this thesis work implementation is a start point for the KPI calculation by FBN configured within an ontological approach. This means that the presented solution is not the only one that can be achieved and, obviously, further modifications will be required for finding the solution of new challenges, in manufacturing systems application, for this type of data integrations.

An interesting research for doing this thesis work, even more useful and relevant for the community, would be to investigate the dependences of KPI through different layers of the manufacturing system integrations. This means that it can be found a relation of a KPI which are located on, for instance, the shop floor with another KPI that is on the top level of an ES. The importance of this research relies in the direct relation on production efficiency which, as it is explained in this thesis work document, is the current trend on manufacturing systems. Then, by the control of KPI dependences, the incomes of a company could be better supervised and, so improved.

Then, following this concept, a further option for this implementation would be to separate de knowledge of KPIs from the MSO, designing a third ontology. In this way, KPIs could be better descripted separately and would be still available to be retrieved by querying.

Secondly, note that the presentation of choreography and CDL in *"Choreography on manufacturing systems"* section is important for this thesis work because implementations based on IEC-61499 can be modeled by ontologies. In addition, enriching models with WSCDL based description, results in AI modelled systems which are performed in conjunction with:

- Standards for distributed model systems as IEC-61499
- Description languages as WSCDL
- KR supported by OWL ontologies.

This means that the use of decentralized model systems can combine all the technologies studied by this thesis work.

Note that choreography has been eclipsed by orchestration configurations as it is explained in [58]. On the other hand, [58] describes how choreography configurations allow more expressivity, description and self-performance to modeled systems, without needing the control of an orchestrator. In fact, this thesis work has been done in conjunction with EU PLANTCockpit project allowing the research group of TUT FAST Laboratory publishing papers, as [14, 40, 58 and 66] among others, of current implementation trends on the manufacturing system environment.

On the other hand, the *"Conclusions of implementation and results"* section states that the use of other standards would be possible in this implementation. Then, a possible further work would be to add an extra class for defining the available standards. This means that the source types could be mapped within other standards, depending on the knowledge of experts. Note that this change would also change the configuration support since it should be prepared to distinguish between standard cases.

Finally, note that other researches in parallel to PLANTCockpit project is being developed currently on TUT FAST Laboratory and this thesis work would be useful for them. Specifically, there is an on-going implementation that requires a component for manufacturing systems KPI monitoring [14]. Following Figure 43 depicts a possible architecture of this higher implementation in which this thesis work could play an important role, needing support for recognition and semantic description in heterogeneous SOA, for the discovery and addition of device information to the MSO. Then, a decision support system (DSS) with a control loop analogy can be achieved, using KPI monitoring as the feedback of the system. Then, this thesis work could be a complement for the implementation DSS control loop analogy presented on [14].

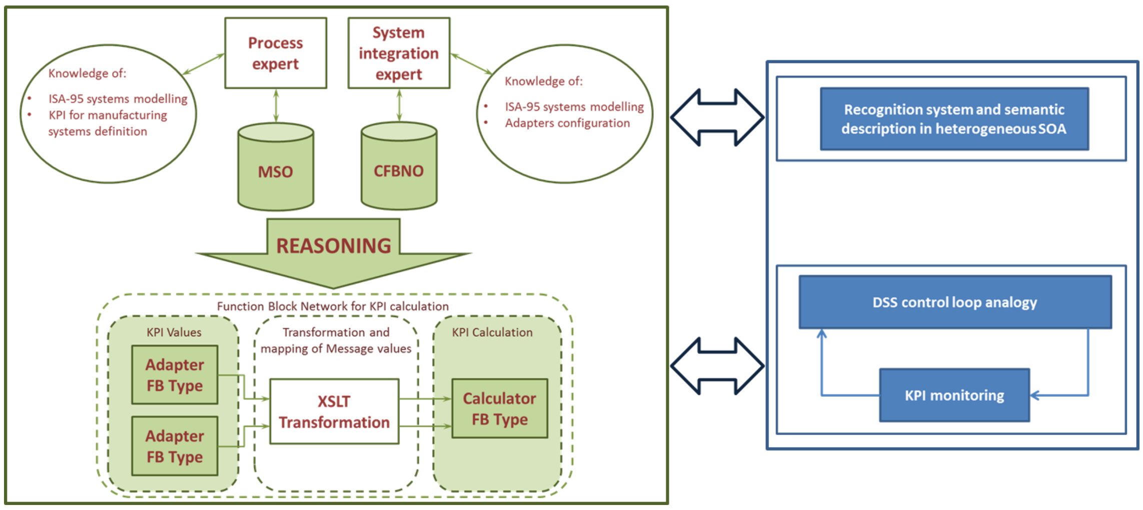


Figure 43: Higher implementation [14] assisted by the ontological approach

REFERENCES

[1] Valeriy Vyatkin, “IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design”; 2006, 271 pages; ISBN – 10: 0-979-1330-1-7

[2] Wang Qifeng, “Semantic Framework Model-Based Intelligent Information System Integration Mode for Manufacturing Enterprises”, Intelligent Information Technology Application, 2008. IITA '08: Second International Symposium, vol.1, pp.223, 227, 20-22 Dec. 2008 doi: 10.1109/IITA.2008.123

[3] Zhao Bo; Hou Kaihu, “Research on SOA-based Cost Information Integration System for Project Manufacturing Industry”, Information Management, Innovation Management and Industrial Engineering (ICIII), 2010 International Conference, vol.2, pp.3,6, 26-28 Nov. 2010. doi: 10.1109/ICIII.2010.165

[4] Gruver, W.A., “Distributed Intelligence Systems: A New Paradigm for Systems Integration”; IEEE IRI 2007 Keynote Speech (III), August 14, 2 pages.

[5] Johannes Minor, Jorge García, Jaacan Martinez, Andrei Lobov, Jose L. Martinez Lastra; “Evaluating Service-Oriented Orchestration Schemes for Controlling Pallet Flow”; ICONS 2012: The Seventh International Conference on Systems; ISBN: 978-1-91208-184-7

[6] The Instrumentation, Systems and Automation Society, “ANSI/ISA-95.00.01-2000: Enterprise Control System Integration” – Part1: Models and Terminology; Released in 2000

[7] The Instrumentation, Systems and Automation Society, “ANSI/ISA-95.00.02-2001: Enterprise Control System Integration” – Part2: Object Attributes; Released in 2001

[8] The Instrumentation, Systems and Automation Society, “ANSI/ISA-95.00.03-2005: Enterprise Control System Integration” – Part3: Models of Manufacturing Operations; Released in 2005

[9] The Instrumentation, Systems and Automation Society, “ANSI/ISA-95.00.05-2007: Enterprise Control System Integration” – Part5: Business to Manufacturing Transactions; Released in 2007

[10] Vyatkin, V., “IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review”, Industrial Informatics, IEEE Transactions, vol.7, no.4, pp.768, 781, Nov. 2011. doi: 10.1109/TII.2011.2166785

[11] “PLANTCockpit” [Online]. Available: http://www.plantcockpit.eu

[12] “Fact sheet”, PLANTCockpit documentation. Version: February 2013. Available: http://www.plantcockpit.eu/fileadmin/PLANTCOCKPIT/user_upload/PLANTCockpit_Fact_Sheet_Feb2013.pdf

[13] Vasyutynskyy, V.; Hengstler, C.; McCarthy, J.; Brennan, K.G.; Nadoveza, D.; Dennert, A., “Layered architecture for production and logistics cockpits”, Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference, pp.1,9, 17-21 Sept. 2012 doi: 10.1109/ETFA.2012.6489548

[14] Jorge Garcia, Jose L.Martinez Lastra; “A Decision support framework for Orchestration of Heterogeneous Service-Oriented manufacturing systems”; paper presented for ICONS 2013, 6 pages. Under revision.

[15] Min Luo; Goldshlager, B.; Liang-Jie Zhang, “Designing and implementing Enterprise Service Bus (ESB) and SOA solutions”, Services Computing, 2005 IEEE International Conference, vol.2, pp. xiv vol.2, 11-15 July 2005. doi: 10.1109/SCC.2005.43

[16] Lobov, A.; Lopez, F.U.; Herrera, V.V.; Puttonen, J.; Lastra, J.L.M., “Semantic Web Services framework for manufacturing industries”, Robotics and Biomimetics, 2008. ROBIO 2008, IEEE International Conference, pp.2104, 2108, 22-25 Feb. 2009. doi: 10.1109/ROBIO.2009.4913327

[17] Frank Manola, Eric Miller, Brian McBride; RDF Primer; 10 February 2004; Avaliable: http://www.w3.org/TR/rdf-primer/

[18] Sean B., Frank van H., Jim H., Ian H., Deborah L., Peter F., Lynn A Andrea S.; RDF Primer; Ed.:Mike D., Guus S.; 10 February 2004; Avaliable: http://www.w3.org/TR/owl-ref/

[19] Uddin, M.K.; Dvoryanchikova, A.; Lobov, A.; Lastra, J.L.M., “An ontology-based semantic foundation for flexible manufacturing systems”, IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society, pp.340, 345, 7-10 Nov. 2011. doi: 10.1109/IECON.2011.6119276

[20] Dufort, Y. C.; “ISA-95-Based Operations and KPI Metrics assessment and Analysis”. White paper 24, A Mesa International, ISA and Ivensys Wonderware co-branded white paper , Nov, 28 2006

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

[22] Gruber T.; “Toward Principles for the Design of Ontologies Used for Knowledge Sharing”; Revision: August 23, 1993. In International Journal Human-Computer Studies 43, pages 907-928. Substantial revision of paper presented at the International Work-shop on Formal Ontology, March, 1993, Padova, Italy; Available as Technical Report KSL 93-04, Knowledge Systems Laboratory, Stanford University

[23] Jose L. Martinez Lastra, Ivan M. Delamer, Fernando Ubis; “Domain Ontologies for Reasoning Machines in Factory Automation”; ISBN: 978-1-936007-01-1, 2010; 138 pages

[24] Fernando Ubis Lopez; “Tools and Methodologies for Knowledge Representation: Generation of an ABAS Domain Ontology”; Master of Science Thesis on Tampere University of Technology; 2006

[25] Ricardo N. S. Cruz Gomes; “An Ontology-Based Representation of an Agent-Based Controlled Robotic Cell”; Master of Science Thesis on Tampere University of Technology; 2009

[26] Simon Tobio Phillips; “A Domain Ontology for Metal Forming and Material Re-moval Processes”; Master of Science Thesis on Tampere University of Technology; 2006.

[27] Zhou Gui-xian; Xie Qing-sheng, “Developing a framework for integration flexible manufacturing systems based on ontology”, Industrial Electronics and Applications, 2008. ICIEA 2008, 3rd IEEE Conference, pp. 1325, 1328, 3-5 June 2008. doi: 10.1109/ICIEA.2008.4582732

[28] Monch, L.; Zimmermann, J., "An Ontology to Support Adaptive Agents for Com-plex Manufacturing Systems," Computer Software and Applications, 2008. COMPSAC '08, 32nd Annual IEEE International, pp. 531, 536, July 28 2008-Aug. 1 2008, doi: 10.1109/COMPSAC.2008.145

[29] Fredrik Arvidsson and Annika Flycht-Eriksson, “Ontologies I”, Retrieved 26, Nov 2008.

[30] Asunción Gómez-Pérez; “Ontological Engineering”; Tutorial on Ontological Engi-neering: IJCAI’99.

[31] Asuncion Gomez-Perez, Oscar Corcho, Mariano Fernandez-Lopez; “Ontological Engineering”: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. Ed. Springer, 420 pages. ISBN-13: 978 – 1852335519, July 22, 2004.

[32] Fan Shuli; Liu Wenling; Zhang Xiankun; Luo Xin, “The Knowledge Description of Teaching Resource and Its Application”, Intelligent Computing and Cognitive Informatics (ICICCI), 2010 International Conference, pp. 156, 159, 22-23 June 2010, doi: 10.1109/ICICCI.2010.36

[33] Matthew Horridge, Simon Jupp, Georgina Moulton, Alan Rector, Robert Stevens, Chris WroeA; “Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools”, Edition 1.1, October 16, 2007

[34] Natalya F. Noy, Deborah L. McGuinness, “Ontology Development 101: A Guide to Creating Your First Ontology”. Available online on: http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness-abstract.html

[35] V. Maniraj, Dr. R. Sivakumar; “Ontology Languages – A Review; International Journal of Computer Theory and Engineering”, Vol.2, No.6, December, 2010, 1795-8201, 5 pages

[36] Object Management Group (OMG), “OMG Unified Modeling Language (OMG UML), Infrastructure”; Version 2.4.1, August 2011. Available versions in: http://www.omg.org/spec/UML/

[37] “Protégé project” website: http://protege.stanford.edu/

[38] Singh, S.; Karwayun, R., "A Comparative Study of Inference Engines," Information Technology: New Generations (ITNG), 2010 Seventh International Conference, pp. 53, 57, 12-14 April 2010. doi: 10.1109/ITNG.2010.198

[39] W3C, “SPARQL Query Language for RDF”. Ed. Eric Prud’hommeaux, W3C, Andy Seaborne, Hewlett-Packard Laboratories, Bristol. W3C Recommendation, 15 January 2008. Available: http://www.w3.org/TR/rdf-sparql-query/

[40] Jorge Garcia, Borja Ramis, Jose Luis Martinez Lastra, Alexander Dennert, Martin Wollschlaeger, Volodymyr Vasyutynskyy; “An ontological approach for modeling configuration of Factory-Wide data integration systems based on IEC-61499”, paper presented for IECON 2013, 6 pages.

[41] John Harrison, “Handbook of Practical Logic and Automated Reasoning”; Cambridge University Press, 2009, (702 pages), ISBN: 9780521899574

[42] Andy S., Geetha M., Chris B., John B., Souripriya D., Ian D., Steve H., Kingsley I., Olivier C., Kjetil K., Benjamin N.; “SPARQL Update”, W3C Member Submission 15 July 2008. Available: http://www.w3.org/Submission/SPARQL-Update/#sec_intro

[43] CAPSTONE METRICS, “Overall Equipment Effectiveness (OEE) A General Discussion with Calculations Methods”; White paper, 9 pages.

[44] Red Lion, “Seven Common KPIs for Production Monitoring, Using Virtual Management to Drive Productivity”; White paper, 7 pages.

[45] Dazhuang He, “An approach for ISA-95 application to industrial systems”; Master of Science Thesis on Tampere University of Technology, 2012

[46] Dennis Brandl, “BR&L Consulting IEC65E/JWG5 Convener ISA 95 Editor”; Presentation, 2008-05-19, 32 slides

[47] Dufort, Y. C. ; “ISA-95-Based Operations and KPI Metrics assessment and Analysis”; White paper 24, A Mesa International, ISA and Ivensys Wonderware co-branded white paper , Nov, 28 2006

[48] MESA International, “B2MML V0500”, Withe paper, Sep, 2012. Available: https://services.mesa.org/ResourceLibrary/ShowResource/03181327-8375-44b7-b42a-07ac3dba2119

[49] WBF, “Business To Manufacturing Markup Language, B2MML”, Version 0500 March 2011. Document Type: Set of documents revised on September 2012, available: https://services.mesa.org/ResourceLibrary/ShowResource/03181327-8375-44b7-b42a-07ac3dba2119

[50] Greg Johnson, “Production to Business Interoperability: ISA 95 and B2MML”. Document type: White paper, September 2005.

[51] “OWL2XMI” Project webpage: http://www.owl2xmi.sourceforge.net/

[52] “ArgoUML” webpage: http://argouml.tigris.org/

[53] Robert Lewis, “Modelling Control Systems Using IEC 61499: Applying Function Blocks to Distributed Systems”, 2001, 192 pages, ISBN: 9780852967966

[54] IEC 61499, “Function Blocks for Industrial-Process Measurement and Control Systems—Part 1: Architecture”, IEC 61499, 2005.

[55] Vyatkin, V., “The IEC 61499 standard and its semantics”, Industrial Electronics Magazine, IEEE, vol.3, no.4, pp. 40, 48, Dec. 2009. doi: 10.1109/MIE.2009.934796

[56] Lopez Orozco, O.J.; Lastra, J.L.M., “Adding Function Blocks of IEC 61499 Semantic Description to Automation Objects”. Emerging Technologies and Factory Automation, 2006, ETFA '06. IEEE Conference, pp. 537, 544, 20-22 Sept. 2006. doi: 10.1109/ETFA.2006.355455

[57] Lastra, J.L.M.; Godinho, L.; Lobov, A.; Tuokko, R., “An IEC 61499 application generator for scan-based industrial controllers”, Industrial Informatics, 2005, INDIN '05. 2005 3rd IEEE International Conference, pp. 80, 85, 10-12 Aug. 2005. doi: 10.1109/INDIN.2005.1560356

[58] Borja Ramis, Jorge Garcia, Jose L. Martinez Lastra; “Assessment of IEC-61499 and CDL for Function Block composition in factory-wide system integration”, IEEE IES INDIN13, 6 pages, July 2013

[59] N. Kavantzas et al. “Web Service Choreography Description Language (WSCDL) 1.0”; http://www.w3.org/TR/ws-cdl-10/

[60] Dave Reynolds, Carol Thompson, Jishnu Mukerji, Derek Coleman; “An assessment of RDF/OWL modeling”, HPL Technical Report 2005-189, October 28, 2005, 24 pages.

[61] “TUT FAST Laboratory” website: http://www.tut.fi/en/fast/

[62] FESTO; “Festo Didactic 2008”, 83 084 EN, 2008 Festo Didactic GmbH & Co. KG

[63] TUT FAST Laboratory, “FESTO line monitoring”. Document type: Slideshare presentation available in: http://www.slideshare.net/fastlaboratory/system-events-concept-presentation, February 2013

[64] EMCO Maier Ges.m.b.H., “Emco Concept Mill 105 PC-controlled milling machine for training. Machine Description Emco Concep Mill 105”. Ref.-No: EN 2105. Edition A2003-02

[65] Schober / Espenberger, Festo Didactic GmbH & Co., “Manual MULTI FMS”, D-73770 Denkendorf, May 2011

[66] Sergii Iarovyi, J. I. Garcia, and J. L. M. Lastra, “An approach for OSGi and DPWS interoperability: Bridging enterprise application with shop-floor”, IEEE IES INDIN13, 6 pages, July 2013

[67] “BACnet” official website: http://www.bacnet.org/

[68] “MTConnect” official website: http://mtconnect.org/

[69] Taejin Park; Seung-Ho Hong, “A new proposal of network management system for BACnet and its reference model”, Industrial Informatics (INDIN), 2010 8th IEEE International Conference, pp. 28, 33, 13-16 July 2010, doi: 10.1109/INDIN.2010.5549470

[70] Skoogh, A.; Michaloski, J.; Bengtsson, N., “Towards continuously updated simulation models: combining automated raw data collection and automated data processing”, Simulation Conference (WSC), Proceedings of the 2010 Winter, pp. 1678, 1689, 5-8 Dec. 2010 doi: 10.1109/WSC.2010.5678901

APPENDIX A – MODELLED MULTI-FMS STATION COMPONENTS

This section presents the Multi-FMS devices what have been contemplated and included in the MSO for this thesis work.

Table 12: Mill 105 station components

Component	Type	Description
Mill 105	Machine	To mill and drill workpieces

Table 13: Robot station (RV-1A) components

Component	Type	Description
Robot RV-1A	Robot	To transport the workpieces from the parts buffer station to different points of the Micro FMS
Drive motor	Actuator	For the linear axis motion of the robot

Table 14: Parts Buffer station components

Component	Type	Description
Inductive proximity sensor	Sensor	Used for end position sensing of the cylinder
Diffuse sensor	Sensor	Used for detection of the workpieces
Short stroke cylinder	Actuator	Used to actuate the separator
DC gear motor	Actuator	Used to drive the conveyor

Table 15: AFB pallet transport system components

Component	Type	Description
Inductive proximity switch	Sensor	Used for detecting the pallets
Optical proximity switch	Sensor	Used for detecting the workpieces
AC gear motor	Actuator	Used to drive the conveyor system
Double/single acting pneumatic cylinder	Actuator	Used to stop pallets

Table 16: Distributing station components

Component	Type	Description
Inductive proximity sensor	Sensor	Used for end position sensing of the cylinder
Through-beam sensor	Sensor	Used for monitoring the stack module level
Micro switch	Sensor	Used for end stop sensing of the swivel drive
Vacuum switch	Sensor	Used for detecting partial vacuum suction cup
Double-acting cylinder	Actuator	Used for pushing workpieces
Pneumatic handling device	Actuator	Used for transporting workpieces

Table 17: Testing station components

Component	Type	Description
Capacitive proximity sensor	Sensor	Used for detection of workpieces
Diffuse sensor	Sensor	Used for colour detection of workpieces
Retro-reflective sensor	Sensor	Used for monitoring the working space
Inductive proximity sensor	Sensor	Used for end position sensing of the cylinder
Linear displacement sensor	Sensor	Used for measuring the height of workpieces
Rod-less lifting cylinder	Actuator	Used for raising or lowering workpieces
Ejecting cylinder	Actuator	Used for pushing workpieces

Table 18: Handling station components

Component	Type	Description
Diffuse sensor	Sensor	Used for detection of workpieces
Diffuse sensor	Sensor	Used for detecting colour of workpieces
Inductive proximity sensor	Sensor	Used for end position sensing of the linear axis
Inductive proximity sensor	Sensor	Used for end position sensing of the cylinder
Pneumatic linear axis	Actuator	Used to rapid positioning
Linear flat cylinder	Actuator	Used as a lifting cylinder for the Z-axis

Table 19: Processing station components

Component	Type	Description
Capacitive proximity sensor	Sensor	Used for detection of workpieces
Inductive proximity sensor	Sensor	Used for the rotary indexing table positioning
Inductive proximity sensor	Sensor	Used for testing the orientation of workpieces
Micro switch	Sensor	Used for end stop sensing of the linear axis
DC gear motor	Actuator	Used to drive the rotary indexing table module
Solenoid actuator	Actuator	Used for checking position of workpieces
Solenoid actuator	Actuator	Used for clamping workpieces during drilling

Table 20: Robot station (RV-2AJ) components

Component	Type	Description
Diffuse sensor	Sensor	Used for checking the orientation of workpieces
Robot RV-2AJ	Robot	Used to determine characteristics of workpieces, assembling, and pass them to subsequent stations

Table 21: Assembling station components

Component	Type	Description
Inductive proximity sensor	Sensor	Used for end position sensing of cylinder A
Inductive proximity sensor	Sensor	Used for end position sensing of cylinder B
Micro switch	Sensor	Used for detection of the spring
Through-beam sensor	Sensor	Used for monitoring the cap magazine level
Through-beam sensor	Sensor	Used for monitoring the ejected cap
Double acting pneumatic cylinder	Actuator	Used for moving springs to the transfer point
Double acting pneumatic cylinder	Actuator	Used for pushing out the bottom cap from the magazine

Table 22: Storing station components

Component	Type	Description
Inductive proximity sensor	Sensor	Used for end position sensing of the cylinder
Inductive proximity sensor	Sensor	Used to sample the rotary drive reference point
Diffuse sensor	Sensor	Used for identifying the colour of workpieces
Pneumatic sliding unit STL	Actuator	Used for extend and retract the gripper
Electrical linear axis	Actuator	Used for driving the linear drive
Servo motor	Actuator	Used for driving the storage module

Table 23: Warehouse (ASRS20) station components

Component	Type	Description
Reed-switch	Sensor	Used to send information when Z axis is down
Reed-switch	Sensor	Used to send information when gripper is open
Pneumatic linear drive	Actuator	Used to extend and retract the gripper
Electro-mechanical drive	Actuator	Used to drive along the X axis

APPENDIX B – POSSIBLE SEMANTIC SCENARIO

This section presents a possible resulting semantic scenario in XML, consisting in a generated FBN by the FBEC. Note that this FB composition is achieved by using the ontological approach data integration of this thesis work.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <fbNetworkScenario>
        <fbNetworkName>semanticExample</fbNetworkName>
        <fb>
            <fbId>ProdPlan</fbId>
            <fbPid>pid_eu.plant.plant_fb_excel_adapter</fbPid>
            <fbBusinessLogic>{ "Con-
fig":{"extractions":{"eu.plantcockpit.plant_fbs.plant_fb_excel_adapter.impl.ExtractionConf":[{"attributes":{"attributes":[{"eu.plant
cock-
pit.plant_fbs.plant_fb_excel_adapter.impl.Attribute":[{"name":"ProductPlantDuration","value":""}]}]},"conditions":[{"eu.plantcock
pit.plant_fbs.plant_fb_excel_adapter.impl.Condition":[{"offset":{"columnEnd":1,"columnStart":1,"rowEnd":1,"rowStart":1},"sheet
":"Sheet1","type":0}]}],"formats":[{"eu.plantcockpit.plant_fbs.plant_fb_excel_adapter.impl.ExcelCellFormater":{"format":"","id":0
,"postString":"","preString":"","type":3}}],"sheet":"Sheet1","type":0}]},"inputFileName":"C:/Users/fast/Desktop/MyTestExcel.xls"
}}</fbBusinessLogic>
            <fbBusinessLogicSchema>{"type":"object","$schema": "http://json-schema.org/draft-03/schema","id":
"#","required":false,"properties":{ "Config": { "type":"object", "id": "Config","title":"Config" , "required":false, "properties":{
"extractions": { "type":"object", "id": "extractions", "required":false, "properties":{
"eu.plantcockpit.plant_fbs.plant_fb_excel_adapter.impl.ExtractionConf": { "type":"array", "id":
"eu.plantcockpit.plant_fbs.plant_fb_excel_adapter.impl.ExtractionConf", "title":"ExtractionConf", "_inputex": { "description":
"Extraction configuration "} , "required":false, "items": { "type":"object", "id": "0", "required":false, "properties":{ "attributes": {
"type":"object", "id": "attributes", "title":"ElementNames" , "required":false, "properties":{ "attributes": { "type":"array", "ti-
tle":"ElementName" ,"id": "attributes", "required":false, "items": { "type":"object", "id": "0", "required":false, "properties":{
"eu.plantcockpit.plant_fbs.plant_fb_excel_adapter.impl.Attribute": { "type":"array", "id":
"eu.plantcockpit.plant_fbs.plant_fb_excel_adapter.impl.Attribute", "title":"ElementName" , "_inputex": { "description": "Defines
XML element name containing extraction. NOTE! Only &amp;quot;name&amp;quot; have to be filled, the
&amp;quot;value&amp;quot; can be left empty"} , "required":false, "items": { "type":"object", "id": "0", "required":false, "proper-
ties":{ "name": { "type":"string", "id": "name", "required":false }, "value": { "type":"string", "id": "value", "required":false } } } } }
} } } }, "conditions": { "type":"array", "id": "conditions", "required":false, "items": { "type":"object", "id": "0", "required":false,
"properties":{ "eu.plantcockpit.plant_fbs.plant_fb_excel_adapter.impl.Condition": { "type":"array", "id":
"eu.plantcockpit.plant_fbs.plant_fb_excel_adapter.impl.Condition", "title":"Condition" , "required":false, "items": { "type":"object",
"id": "0", "required":false, "properties":{ "offset": { "type":"object", "id": "offset", "required":false, "properties":{ "columnEnd": {
"type":"number", "id": "columnEnd", "required":false }, "columnStart": { "type":"number", "id": "columnStart", "required":false },
"rowEnd": { "type":"number", "id": "rowEnd", "required":false }, "rowStart": { "type":"number", "id": "rowStart", "required":false,
"_inputex": { "description": "The conditions specify location of the cell(s). NOTE! &amp;quot;0&amp;quot; in all fields will select
the cell &amp;quot;A0&amp;quot; in excel, all &amp;quot;1&amp;quot; will select &amp;quot;B1&amp;quot; and so on."} } } },
"sheet": { "type":"string", "id": "sheet", "required":false }, "type": { "type":"string","choices" : [{ "value": 0, "label" : "Just add
offset" },{ "value": 1, "label":"Position merged field and add offset" },{ "value": 2, "label":"Hard set" },{ "value": 3, "label":"Below
and offset" },{ "value": 4, "label":"Change to height and offset" },{ "value": 5, "label":"Change to width and offset" }], "id": "type",
"required":false } } } } } } }, "formats": { "type":"array", "id": "formats", "required":false, "items": { "type":"object", "id": "0",
"required":false, "properties":{ "eu.plantcockpit.plant_fbs.plant_fb_excel_adapter.impl.ExcelCellFormater": { "type":"object", "id":
"eu.plantcockpit.plant_fbs.plant_fb_excel_adapter.impl.ExcelCellFormater", "title":"ExcelCellFormater", "required":false, "proper-
ties":{ "format": { "type":"string", "id": "format","_inputex": { "description": "The format which can be used for date such as
&amp;quot;&amp;quot;"} , "required":false }, "id": { "type":"number","id": "id", "required":false }, "postString": { "type":"string",
"id": "postString", "required":false }, "preString": { "type":"string",  "id": "preString", "required":false }, "type": { "type":"string",
"choices" : [{ "value": 0, "label" : "String" },{ "value": 1, "label" : "Date" },{ "value": 2, "label" : "Double" },{ "value": 3, "label" :
"Integer" }],"id": "type", "required":false } } } } } }, "sheet": { "type":"string", "id": "sheet", "required":false }, "type": {
"type":"string", "choices" : [{ "value": 0, "label" : "Single cell" },{ "value": 1, "label" : "Table" }], "id": "type", "required":false } }
} } } }, "inputFileName": { "type":"string", "id": "inputFileName", "_inputex": { "description": "Enter the location of the file.
Example: C:/folder/file.xls"} , "required":false } } } }}</fbBusinessLogicSchema>
```

```
            <inputMessageTypes>
                <inputMessageType>plant_fb_excel_adapter_MyInput1</inputMessageType>
            </inputMessageTypes>
            <outputMessageTypes>
                <outputMessageType>plant_fb_excel_adapter_MyOutput1</outputMessageType>
            </outputMessageTypes>
            <transformations>
                <transformation>
                    <originalMsgType>plant_fb_excel_adapter_MyOutput1</originalMsgType>
                    <newMsgType>ProductPlantDuration</newMsgType>
                    <transformationName>plant_fb_excel_adapter_MyOutput12ProductPlantDuration</transformationName>
                    <transformationXSLT>&lt;?xml version="1.0"?&gt; &lt;xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"&gt;&lt;xsl:template match="/"&gt;&lt;ProductPlantDuration&gt;
  &lt;value&gt;&lt;xsl:value-of select="/ExcelAdapterData/Data/ProductPlantDuration" /&gt;&lt;/value&gt;
&lt;/ProductPlantDuration&gt;&lt;/xsl:template&gt;&lt;/xsl:stylesheet&gt;</transformationXSLT>
                    <direction>OUTPUT</direction>
                </transformation>
            </transformations>
        </fb>
        <fb>
            <fbId>dataCalc</fbId>
            <fbPid>pid_eu.plant.plant_fb_Esper</fbPid>
            <fbBusinessLogic>{ "EsperConfigura-
tion":{"CEPServiceName":"dataCalc","EventProcessingQuery":{"queryStatement":"Select prodTimeEvent.endTime - prod-
TimeEvent.startTime as duration, ProductPlantDurationEvent.value as threshold  from pattern [every (prodTimeEvent = ProdEvent
-&gt; ProductPlantDurationEvent = ProductPlantDuration)].win:time_batch(3 sec) where prodTimeEvent.endTime - prod-
TimeEvent.startTime &gt;= ProductPlantDuration-
Event.value\n"},"InputMessageDefinitions":{"InputMessage":[{"elements":{"Element":[{"Name":"startTime","SampleValue":"555
5"},{"Name":"endTime","SampleValue":"7000"}]},"rootElementName":"ProdEvent"},{"elements":{"Element":[{"Name":"value",
"SampleValue":"1000"}]},"rootElementName":"ProductPlantDuration"}]}}}</fbBusinessLogic>
            <fbBusinessLogicSchema>{"type":"object","$schema": "http://json-schema.org/draft-03/schema","id":
"#","required":false,"properties":{ "EsperConfiguration": { "type":"object", "id": "EsperConfiguration", "required":false, "proper-
ties":{ "CEPServiceName": { "type":"string", "id": "CEPServiceName", "required":false }, "EventProcessingQuery": {
"type":"object", "id": "EventProcessingQuery", "required":false, "properties":{ "queryStatement": { "type":"string", "id":
"queryStatement", "format":"text", "_inputex": { "rows":3, "cols":70, "description": "Event Processing Language Query expected "
} , "required":false } } }, "InputMessageDefinitions": { "type":"object", "id": "InputMessageDefinitions", "required":false, "proper-
ties":{ "InputMessage": { "type":"array", "id": "InputMessage", "required":false, "_inputex": {"description": "Input message decla-
ration " } , "items": { "type":"object", "id": "0", "required":false, "properties":{ "elements": { "type":"object", "id": "elements",
"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>ProdEvent</inputMessageType>
                <inputMessageType>ProductPlantDuration</inputMessageType>
            </inputMessageTypes>
            <outputMessageTypes>
                <outputMessageType>dataCalc_OUT</outputMessageType>
            </outputMessageTypes>
            <transformations>
                <transformation>
                    <originalMsgType>ProdEvent</originalMsgType>
                    <newMsgType></newMsgType>
                    <transformationName></transformationName>
                    <transformationXSLT></transformationXSLT>
                    <direction>INPUT</direction>
                </transformation>
                <transformation>
                    <originalMsgType>ProductPlantDuration</originalMsgType>
                    <newMsgType></newMsgType>
                    <transformationName></transformationName>
                    <transformationXSLT></transformationXSLT>
                    <direction>INPUT</direction>
                </transformation>
                <transformation>
                    <originalMsgType>dataCalc_OUT</originalMsgType>
                    <newMsgType></newMsgType>
                    <transformationName></transformationName>
                    <transformationXSLT></transformationXSLT>
                    <direction>OUTPUT</direction>
                </transformation>
            </transformations>
        </fb>
        <fb>
            <fbId>Dpws_WS</fbId>
            <fbPid>pid_eu.plant.plant_fb_dpws_adapter</fbPid>
            <fbBusinessLogic>{ "DPWSCon-
fig":{"subscribe":{"string":["192.168.2.62;http://www.plantcockpit.eu/fast/festo/FestoStationServicePortType/ProductionTimeEven
t"]}}}</fbBusinessLogic>
            <fbBusinessLogicSchema>{"type":"object","$schema":"http://json-schema.org/draft-
03/schema","id":"#","required":false,"properties":{"DPWSConfig":{"type":"object","id":"DPWSConfig","title":"DPWSConfig","re
quired":false,"properties":{"subscribe":{"type":"object","id":"subscribe","required":false,"properties":{"string":{"type":"array","titl
e":"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 adapt-
er"},"choices":[{"value":"192.168.2.62;http://www.plantcockpit.eu/fast/festo/FestoStationServicePortType/EndTimeEvent"},{"valu
e":"192.168.2.62;http://www.plantcockpit.eu/fast/festo/FestoStationServicePortType/WorkstationAlarm"},{"value":"192.168.2.62;h
ttp://www.plantcockpit.eu/fast/festo/FestoStationServicePortType/WorkstationEvent"},{"value":"192.168.2.62;http://www.plantcoc
kpit.eu/fast/festo/FestoStationServicePortType/OperatorInput"},{"value":"192.168.2.62;http://www.plantcockpit.eu/fast/festo/Festo
StationServicePort-
Type/WorkstationStatus"},{"value":"192.168.2.62;http://www.plantcockpit.eu/fast/festo/FestoStationServicePortType/WorkpieceSt
atus"},{"value":"192.168.2.62;http://www.plantcockpit.eu/fast/festo/FestoStationServicePortType/WorkpieceProperties"},{"value":
"192.168.2.62;http://www.plantcockpit.eu/fast/festo/FestoStationServicePortType/TransferIn"},{"value":"192.168.2.62;http://www.
plantcock-
pit.eu/fast/festo/FestoStationServicePortType/ProductionTimeEvent"}],"id":"string","required":false}}}}}}}}</fbBusinessLogicSc
hema>
            <inputMessageTypes/>
            <outputMessageTypes>
                <outputMessageType>plant_fb_dpws_adapter_MyOutput1</outputMessageType>
            </outputMessageTypes>
            <transformations>
                <transformation>
                    <originalMsgType>plant_fb_dpws_adapter_MyOutput1</originalMsgType>
                    <newMsgType>ProdEvent</newMsgType>
                    <transformationName>plant_fb_dpws_adapter_MyOutput12ProdEvent</transformationName>
                    <transformationXSLT>&lt;?xml version="1.0"?&gt; &lt;xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"&gt;&lt;xsl:template match="/"&gt;&lt;ProdEvent&gt;
  &lt;startTime&gt;&lt;xsl:value-of select="/ProductionTime/StartTime" /&gt;&lt;/startTime&gt;
  &lt;endTime&gt;&lt;xsl:value-of select="/ProductionTime/EndTime" /&gt;&lt;/endTime&gt;
&lt;/ProdEvent&gt;&lt;/xsl:template&gt;&lt;/xsl:stylesheet&gt;</transformationXSLT>
                    <direction>OUTPUT</direction>
                </transformation>
            </transformations>
        </fb>
        <connections>
            <connection>
                <srcPID>pid_eu.plant.plant_fb_excel_adapter</srcPID>
                <srcFbInstanceName>ProdPlan</srcFbInstanceName>
                <dstPID>pid_eu.plant.plant_fb_Esper</dstPID>
                <dstFbInstanceName>dataCalc</dstFbInstanceName>
                <filterMsg>ProductPlantDuration</filterMsg>
            </connection>
```

```
          <connection>
               <srcPID>pid_eu.plant.plant_fb_dpws_adapter</srcPID>
               <srcFbInstanceName>Dpws_WS</srcFbInstanceName>
               <dstPID>pid_eu.plant.plant_fb_Esper</dstPID>
               <dstFbInstanceName>dataCalc</dstFbInstanceName>
               <filterMsg>ProdEvent</filterMsg>
          </connection>
     </connections>
</fbNetworkScenario>
```