



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Development of ORMA+ Ontology toward Zero Defect Manufacturing in the Digital Twin Framework

TESI DI LAUREA MAGISTRALE IN
MANAGEMENT ENGINEERING
INGEGNERIA GESTIONALE

Author: Lorenzo Ghedini

Student ID: 10586137

Advisor: Marco Macchi

Co-advisor: Adalberto Polenghi

Academic Year: 2021-22

Abstract

In the context of Industry 4.0 there has been a change in the manufacturing system which has been modified following the development of some important technologies. In this context the ability of the production systems to meet customer expectations has increased while aiming to obtain an higher product quality and greater customization. To control product quality one of the most promising tool is the so-called Zero Defect Manufacturing. The use of ZDM can lead to an improvement in the quality of the products, an increase of the efficiency of the production system, a reduction of the waste and many other benefits. Although this methodological “tool” shows considerable potential, it needs some auxiliary technologies. The creation of a virtual copy of a physical resource toward the use of a Digital Twin appears to be of particular interest, enabling real-time monitoring within the industrial production. Considering the Digital Twin framework in the long-term perspective, to obtain a solution applicable to a greater number of different industrial contexts, in this thesis it is proposed a model, the ORMA+ ontology. This is capable of applying Detect and Repair strategies to go with the creation of a Digital Twin. The ORMA+ ontology can be used to support decision making in an industrial context: it provides, to any operator in the production process, an indication about the quality of the product, also advising some corrective actions if needed, like the repair or the disassembly of the product and the subsequent recycle of the good quality components. To obtain this research outcome, first of all an analysis of the literature was performed to determine the gaps present in the literature, then an ontology editor allowed the creation of the ORMA+ ontology and, finally, the ontology was validated in the context of Industry 4.0 Laboratory at the Politecnico di Milano. In the Industry 4.0 Laboratory the proposed solution was populated with the data coming from the servers, determining the quality of the product in function of the state of some product components and the condition of one of the assets installed in the production line.

Key-words: Zero Defect Manufacturing, Ontology, Digital Twin, Quality Monitoring.

Abstract in italiano

Nell'ambito dell'Industria 4.0 c'è stato un cambiamento nel sistema produttivo, che è stato modificato in seguito allo sviluppo di alcune rilevanti tecnologie. In questo contesto è aumentata la capacità dei sistemi produttivi di soddisfare le aspettative dei clienti puntando ad ottenere una maggiore qualità del prodotto e una maggiore personalizzazione. Per controllare la qualità del prodotto, uno degli strumenti più promettenti è la cosiddetta Zero Defect Manufacturing. L'utilizzo della ZDM può portare un miglioramento della qualità, un aumento dell'efficienza del sistema produttivo, una riduzione degli scarti e altri vantaggi. Sebbene questo "strumento" metodologico mostri un potenziale considerevole, necessita di alcune tecnologie ausiliarie. Di particolare interesse è la creazione di una copia virtuale di una risorsa fisica per l'utilizzo di un Digital Twin, che consente il monitoraggio in tempo reale all'interno della produzione industriale. Considerando il framework DT in una prospettiva a lungo termine, per ottenere una soluzione applicabile ad un maggior numero di differenti contesti industriali, in questa tesi viene proposto un modello, l'ontologia ORMA+. Questo è in grado di applicare le strategie di Detect e Repair per accompagnare la creazione di un DT. L'ontologia ORMA+ può essere utilizzata a supporto del processo decisionale in un contesto industriale: fornisce, a qualsiasi operatore del processo produttivo, un'indicazione sulla qualità del prodotto, consigliando anche alcune azioni correttive se necessarie, come la riparazione o lo smontaggio di il prodotto e il successivo riciclo dei componenti di buona qualità. Per ottenere questo risultato, prima di tutto è stata eseguita un'analisi della letteratura per determinare le lacune presenti, poi un editore di ontologie ha permesso la creazione dell'ontologia ORMA+ e, infine, l'ontologia è stata validata nel contesto del Laboratorio di Industria 4.0 al Politecnico di Milano. Nel Lab la soluzione proposta è stata popolata con i dati provenienti dai server, determinando la qualità del prodotto in funzione dello stato di alcuni componenti e lo stato di uno degli asset installati in linea di produzione.

Parole chiave: Zero Defect Manufacturing, Ontologia, Monitoraggio della Qualità, Digital Twin.

Contents

Abstract	i
Abstract in italiano	iii
Contents	v
1 Introduction	1
1.1. Quality Evolution in Manufacturing	1
1.2. Zero Defect Manufacturing	4
1.3. Digital Twin	7
1.3.1. Cognitive Digital Twin	8
1.4. Ontology	9
2 Literature review on Zero Defect Manufacturing	12
2.1. The Literature Review Methodology	12
2.2. Research protocol on Zero Defect Manufacturing	13
2.3. Descriptive statistics	15
2.4. Synthesis of literature review findings	21
2.4.1. Articles retrieved	21
3 Research Design	25
3.1. Main results of the Systematic Literature Review	25
3.2. Research Gaps.....	25
3.3. Research Objectives.....	27
4 Ontology	31
4.1. Ontology Background	31
4.1.1. Knowledge Reuse: BFO ontology	34
4.1.2. Knowledge reuse: IAO ontology.....	39
4.1.3. Knowledge reuse: CCO ontology	41
4.2. Methontology applied for ZDM	43
4.2.1. Specification	44
4.2.2. Knowledge acquisition	45
4.2.3. Conceptualization and Integration	45
4.2.4. Implementation and Evaluation.....	46
4.3. Proposed ontology for ZDM	47

4.3.1.	Product	47
4.3.2.	Asset	51
4.3.3.	Quality	57
4.3.4.	Process	65
5	Application of the ontology	72
5.1.	Validation of the ontology logic.....	72
5.1.1.	Case study description.....	72
5.2.	Validation of the ORMA+ ontology.....	74
5.2.1.	Regular_Assembly_Quality case	76
5.2.2.	Repairable_Assembly_Quality case.....	89
5.2.3.	Defective_Assembly_Quality case	97
5.2.4.	Competency questions.....	104
6	Ontology Implementation	112
6.1.	Ontology Application	114
6.2.	Ontology population: Repair and Disassembly Stations.....	118
6.2.1.	Repairable Cellphone case:.....	118
6.2.2.	Defective Cellphone case	123
7	Result and discussion	129
7.1.	Results: Ontology Validation.....	129
7.2.	Results: Population of the ontology.....	130
7.3.	Potentiality of the ORMA+.....	130
8	Conclusions	132
8.1.	Research Contribution	132
8.2.	Practice Contribution	135
8.3.	Limitations of the current work	135
8.4.	Future Researches.....	136
9	Bibliography.....	137
A)	APPENDIX: Data Collection	144
	OPC-UA.....	144
	Work Performed.....	146
	Linux Virtual Machine.....	147
	Docker	147
	Golang.....	148
	InfluxDB.....	149
	Python	149
B)	Appendix: Python Code	151

B.1. Python code for the data extracted by the FML.....	151
B.2. Python code for an hypothetical enhanced FML.....	161
List of Figures	167
Acknowledgement	176

1 Introduction

The aim of this chapter is to provide an overall introduction to this thesis work. The chapter is developed starting from the presentation of a general overview of quality evolution in manufacturing; thereafter, it moves on presenting the background in three topical areas of major interest: Zero Defect Manufacturing, Digital Twins / Cognitive Digital Twins, Ontologies. Therefore, the reader finds in this chapter just the point this investigation starts from, and can intend it as a general introduction.

1.1. Quality Evolution in Manufacturing

Over the years, the manufacturing industry has seen a constant growth and change. As a matter of fact manufacturing practices are rapidly evolving with continuous advancement in material sciences, production and information technology. In particular with the advent of the fourth industrial revolution (Industry 4.0) and thanks the development and adoption of relevant advanced technologies, the modern manufacturing systems have been reshaped. The term Industry 4.0 was coined in 2011 in Germany [1] in the Hannover Fair and it referred to the initiative of the government to prepare the German manufacturing to the digital era.

As a matter of fact if we look at the history of production, besides this last change, there were other 3 important changes in the production paradigm [2] which are showed in the image 1-1.

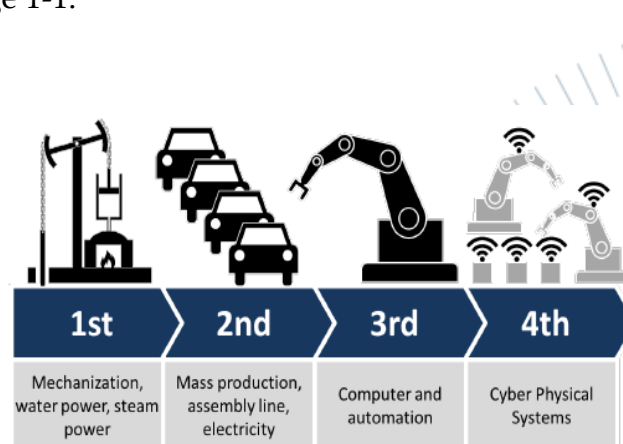


Figure 1-1: Evolution in time of the Industrial Revolutions

The first industrial revolution was known as Craft Production [2] and the biggest change in the industry was the introduction of water and steam power to help the mechanical production [1]. Due to the change from the manual to the machine production it was possible to satisfy costumers requirements, characterized by a high cost and a limited number of products. The necessity to satisfy costumers needs brought, almost a century later, the need for low-cost products realized with a large scale production, leading to the second industrial revolution or Mass Production [2]. In this industrial revolution the variety of the products was still small and limited but the number of product realized increased. In order to reach a higher variety of products, Industry 3.0 or Mass Customized Production made possible to achieve a production system characterized by a higher flexibility respect to the conventional ones due to the Advent of the Digital Revolution leading to a greater productivity, lower costs and an increased variety [1][2]. In the current period of time a new driving force to produce emerged, costumers desire [2]. As a matter of fact the fourth industrial revolution is also known as Mass Personalization Production (MPP) since during the current years has been enhanced the ability to meet the costumers' expectation, such as for greater quality, better features, shorter development times, better performance and higher customization at an affordable price [2][3]. In the current period, thus, giving customers products whenever they want it and the way they want them is the key source of production but with the emergence of this trend the costumers' requests have increased while the product development cycle has reduced significantly [4]. So in the current environment new challenges for manufacturers have been introduced, since now it is necessary to perform in an environment characterized by high variability. In fact in the current context due to the different customers demand and the higher complexity introduced with the increase of the production steps needed to produce a product, the variability of the production system is increased [2].

Due to these motivations, in order to deal with the growing customer expectations of customized products realized at a higher rate, manufacturers had to take some countermeasures, like the reduction of batch sizes and the imposition of shorter production time, to cope with the higher rate of products failing to keep an adequate level of quality [2][4]. It may seem trivial, but it is important to mention that in the current environment poor product quality can have a huge impact on many levels. For example it can have an impact on the economic aspect like the financial losses that a company may experience, or on an environmental aspect in relationship to the waste or an higher consumption of resources [5]. It is necessary to consider also what can occur at the company social level, since in the current high connected environment the

image of the company could be heavily damaged or improved by customer opinion about the quality of its products [6]. Therefore, in the current industrial environment it is crucial for manufacturing companies to improve the quality of the products and to limit the number of defective items as much as possible. This could be achieved by introducing an optimal quality management in order to maintain or increase companies' financial and operational performance [7]. To achieve high-quality parts with minimum performance loss, several different aspects must be taken into consideration. Accordingly to [6] the first one to consider is the raw material used for production, as a matter of fact cheap material are not able to meet high quality standards and consequently meet costumers expectation. An additional aspect is workforce experience which need to be specialized enough or supported by experts when needed. Further, company alignment is also essential, there needs to be a common and established understanding within the company of how the products is realized and if it is able to fulfill costumer request. Lastly manufacturers must implement at least one quality improvement (QI) method [8].

The introduction of a quality improvement method is necessary since those companies that focus on quality can dominate the market due to the reduction of production cost and the better use of machines and materials acquiring a competitive advantage respect to those which don't [5]. An example of this can be found in the case of the Japanese industries of the 1970s which managed to obtain a great competitive advantage over rival companies by pandering to the wishes and expectations of customers, making several US companies declare bankrupt [6]. The Japanese experience shows how focusing on quality improvement can be a valid tool on how to improve the company competitive advantage, but the application of these techniques is not a simple process and today there are still challenges on how to manage quality optimization [6]. So even if traditional QI methods, such as six sigma (SS), lean manufacturing (LM), theory of constraints (TOC), and total quality management (TQM), have been widely used by manufacturers for more than three decades, there is still much ground to cover.

As a matter of fact even if these techniques are still utilized and provide good results, these methodologies were created keeping into consideration the manufacturing standards of that period of time, but in the actual context of Industry 4.0 it is possible to leverage technologies that make possible to realize things that in the past would have been impossible to realize [8]. In fact thanks the introduction of Industry 4.0 related technologies quality control has been boosted by increase in the computational capacity of computer systems, the advanced capabilities of data management systems, and the significant drop in the price of sensors [9]–[12]. As further evidence of what

has been said a recent paper presented traditional product QI policies and it showed their characteristics, enabling factors, and barriers, and it concluded that these methodologies, even if they are still usable and are used by many companies, could be substituted or at least improved by others innovative technologies which can overcome their defects and achieve better performance and efficiency [13]. An example of technologies that could be used for this purpose are for example machine learning, deep learning, artificial Intelligence (AI) and the Digital Twin (DT) [8].

Based on what was previously written with the advent of these new technologies it is possible to create new business opportunities by exploiting, for example, the large amount of data that is generated by industries. For this reason, the development of intelligent systems capable of extracting useful and meaningful knowledge from data has led to the development of better decision-making systems and new optimization opportunities [14]. This shift has been called Quality 4.0 and it permits to use several quality management approaches combining several practices from different fields like logistics, maintenance, process control and many others [3]. In this context a quality management approach we have to discuss is the Zero Defect Manufacturing (ZDM) strategy. The ZDM strategic goal is improving the manufacturing by guaranteeing that no defective product leave the production site and reach the customer. Furthermore, the standard of ZDM is the way to go companies that aim to be more environmentally friendly and to maintain or also improve their assembly system while realizing products with zero defects [15].

1.2. Zero Defect Manufacturing

The Zero Defect Manufacturing strategy is one of the most promising strategy of today and its objective is to reduce the number of defective products by “doing things right the first time” reducing the number of defect to zero [2], [4], [5], [13], [15]. Even if this manufacturing strategy is considered very promising at the current time, this is not new in the manufacturing field since its origin date back to 1960 when the American army used it during the Cold War [8], [5], [6], [15]. The mindset of zero defects is mostly attributed to Philip B. Crosby whom was the quality control department manager on the Pershing Missile program at the Martin Company [5]. As a matter of fact Crosby published a book in 1985 called “Absolutes of Quality Management” where he stated that the personal standard of workers and managers for manufacturing should be zero defect and so they should not accept anything less [5]. Influenced by Crosby’s philosophy and Dr. W. Edwards Deming’s teachings in 1960 the Toyota company launched a Zero Defect Campaign which was further adopted in

1968 when in the Toyota's corporate policy Zero Defects was introduced [5]. During these years of time these events were followed by the development of others methodologies known as traditional methodologies like Lean Production, Total Quality Management, Six Sigma and others as a way to reach the objective of zero defects in a company [3]–[5]. Although Zero Defect Manufacturing has shown important results for the cases mentioned above, its implementation has nevertheless been limited over time. The reason for this goes back to the technological limitations of the time that did not allow to fully exploit this methodology [15]. However, in the current context of Industry 4.0, a plethora of new technologies has showed a new range of possibilities for ZDM and made it possible to obtain those results that once would have been considered impossible to obtain, thus making this methodology reusable and very promising [15]. As a matter of fact due to the greater presence of techniques like Industrial internet of things, Artificial Intelligence, Digital Twin and many others which can be considered as the core of this methodology which in order to attain its goal of zero defect needs to extract patterns and knowledge within the data [5], [14].

This is one of the most important aspects to take into consideration since there are several quality philosophies which aims to improve the whole manufacturing system, like the methodologies cited before, but the huge difference between them and the ZDM is that these “classic” methodologies focus on defect reduction by analyzing just the historical data of the company. This is a huge limitation since these traditional methodologies want to improve the future by analyzing data coming from the past and not taking into consideration the actual data, losing, in this way, a huge amount of valuable knowledge [2]–[5]. So ZDM is beyond traditional quality approaches because it does not only aim to eliminate defects by detecting and correcting eventual defective products and process parameters but there is also a learning mechanism involved in order to predict and prevent the formation of defects [5].

In order to implement a ZDM based approach it is necessary to leverage the Industry 4.0 tools and by utilizing the following functional requirements according to [2], [14], [16], [17]:

- I. a data acquisition system (as, data gathering, storage and analytics system) in order to exploit the intelligent sensor network
- II. an automatic signal processing, filtering and feature extraction
- III. data mining and knowledge discovery system for prognosis and diagnosis
- IV. a monitoring system of process parameters
- V. an online predictive maintenance
- VI. the re-configuration and re-organization of production

Concerning the implementation of the ZDM strategy we can consider 2 different approaches: a product oriented ZDM or process oriented ZDM [2], [8], [15]. The difference between the 2 approaches is that the product oriented one studies the defect that are present in the part trying to find a solution while for the process oriented approach the manufacturing equipment is studied and in function of its condition we can evaluate if the manufacturing products are in good conditions or not. Even if these approaches have a different starting point the steps which are followed are essentially the same and so this is seen like a unique framework [2]. Now that has been discussed how to implement this methodology we have to discuss the strategies that characterize the ZDM in its operations which are 4: Detection, Repair, Prediction, Prevention [2]–[5], [8], [15]. These four strategies can be further divided in two different categories which are the triggering factors and triggering actions, the first ones are the strategy of detection and prediction which are responsible to identify possible defective items during the production process [4], [8]. The second ones, repair and prevention, are the actions that may be taken in response to a triggering factor since defects can be easily amplified and diffused in the following steps causing more damages [4], [8]. This may seem a contradiction since we stated before that the ZDM is an emerging paradigm which aims to completely eliminate defects by doing the right thing at the first time [2], [4], [5], [13], [15], but in the manufacturing environment this task cannot be achieved because of the intrinsic characteristics of manufacturing systems which are characterized by huge variability and uncertainty [6]. Since defects are unavoidable, with the term “Zero Defect” it is usually considered the number of failures happening during operation or the defective products which arrive at the end of the manufacturing process and which are delivered to the costumers [6], [14], [15], so it is necessary to apply the triggering ZDM strategy for each product realized. We have now to see more in detail the characteristics of the four strategies mentioned before. In order to do so we are going to consider the definitions reported in [2]:

- **Detection:** This strategy focuses on the early discovery and identification of defects. In order to detect possible defects it is necessary to keep track of some relevant parameters during production. Once these parameters exceed a certain value or present a strange behavior, it may indicate that a defect has happened causing an undesirable effect. Besides the inspection when we talk about “Detection” we have also to take in consideration the different actions that address the generation and the propagation of the defects along the next stages of the production.

- Repair: This strategy consists on re-working or re-manufacturing a product, when it is possible to do so, when it is found defective. As a matter of fact, when a defect is discovered it may be repairable or not, if the piece is not in a good condition to be repaired then it is rejected, otherwise it is possible to repair it. In order to repair an object it is necessary to have a good trade-off between the time and the effort needed to do so, and the performance of production flow.
- Prediction: This strategy revolves mainly on defect, anomaly and fault prediction, since it aims to determine the quality of the product before its realization by analyzing historical and current data. With this strategy usually also the state of health of the machines or equipment is evaluated in order to prevent unexpected failures by organizing preventive maintenance operations.
- Prevention: This strategy concerns quality control and inspections tools in order to monitor the quality of the machines and consequently the quality of the manufactured piece. Once a defective product is found, the affected parameters are highlighted in order to adapt them on the next runs in order to avoid possible downtimes and losses.

Since the ZDM is an emerging concept according to the papers [4], [16] at present most research on this topic is focused on developing methodologies for specific uses, rather than studying a general implementation of this methodology. Furthermore, usually the approaches dealing with the ZDM are applied in a static and sequential way and, therefore, when a process is analyzed and solved, usually the company moves on to the next stage thus preventing the adaptation of the operations to the changing production target [16]. For these reasons, in this thesis work two additional tools will be used, an ontology and a digital twin, to adopt a more dynamic capability.

1.3. Digital Twin

Digital twins (DT) have gained a lot of attention by the industrial communities and the academia in the recent years [18], [19]. As a matter of fact the numbers of articles dealing with DT have increased and now have reached a discrete amount, but despite DT has gained so much attention there is still missing a common universal definition of what a DT is [20], [21]. In its essence a DT is composed by three elements: the physical entity of the real world, the virtual entity in the virtual space and the connection between the real world physical entity and the virtual one [18]. In its first applications DT was utilized for prognostic and health management applications but thanks the advancement of Industry 4.0 and its relevant technologies, DT has started

to be used in several industrial sectors, especially the ones dealing with productions [18].

So, respect to its original function, the DT is not intended anymore as a mirror of the physical entity, but the concept has evolved and the DT can be utilized in different phases of the production system or different life-phases of a product such as the design phase of a system/product or the operation phase, etc. [21], [22]. At the current time one of the central aspect of a DT is the ability of the tool to provide information, which are not just mere data, but they can also be coupled with semantic modelling. In this way it is possible to get more information about the system as the physical status of a system or how it interacts with the others components of the system [18], [19]. So in the Industry 4.0 context a DT can have a different role respect to the past, it can be a cognitive digital twin so a: “DTs with augmented semantic capabilities for identifying the dynamics of virtual model evolution, promoting the understanding of interrelationships between virtual models and enhancing the decision-making” [18].

1.3.1. Cognitive Digital Twin

According to what just said in the context of Industry 4.0, a Digital Twin can be enhanced with cognitive capabilities using semantic technologies becoming a Cognitive Digital Twin (CDT) [18]. In the literature there are several articles and papers dealing with CDT and, as for the DT, there are several definition of what a CDT is [18], [19]. Although there is not a common definition of what a CDT is, there are some common features and elements which are widespread across the different researches according to [18]:

- 1) DT-based: CDT is an extended or augmented version of DT. It contains at least the three basic elements of DT described before but the main difference is that the CDT usually contains multiple DT models with unified semantics topology definitions. If we are dealing with a complex industrial system the CDT should include also digital models of its subsystems and components, where for each of them there is a description of the status across the entire lifecycle.
- 2) Cognition Ability: As the name suggests a CDT must be characterized by the ability to have certain cognitive capabilities, so the ability to perform such activities as perception, attention, comprehension and so on. So a CDT function is to recognize strange and unpredicted behaviors in a dynamic way. This target has been made possible thanks the development of Industry 4.0 and some

relevant technologies like artificial intelligence or industrial internet of things which made possible to realize cognition capabilities at a certain level.

- 3) Full lifecycle management: The CDT of a complex system is composed by the digital models of the components of the system and so covers the different phases of the system across its entire lifecycle, including the start, the middle and the end of the life of the system. So, in order to support the afore mentioned cognitive abilities, it is necessary to process all the data coming from the different life-phases of the system.
- 4) Autonomy capability: The activities performed by a CDT must be executed in an autonomous manner without the help of a human operator. So a CDT can take decisions based on the analyzed data and react in a certain way without the intervention of a human operator.
- 5) Continuous evolving: Along the entire lifecycle of the system, the CDT must evolve together with the system. The evolution of the system can be according to the change of the system, can be due to the interaction among the different models contained and finally due to the feedback from the other lifecycle phases.

According to the characteristics expressed before the definition we are going to consider for a Cognitive Digital Twin will be the one reported in the [18]: “Cognitive Digital Twin (CDT) is a digital representation of a physical system that is augmented with certain cognitive capabilities and support to execute autonomous activities; comprises a set of semantically interlinked digital models related to different lifecycle phases of the physical system including its subsystems and components; and evolves continuously with the physical system across the entire lifecycle”. Accordingly to this definition by enhancing a DT with semantic technologies, like semantic modelling or ontologies, it is possible to capture the characteristics and status of the system and its components, as well as how a component interacts with the other components of a complex system [18]. In this thesis work the semantic technology which is going to be considered is the ontology: the ontology serves as a mean to guarantee technical and semantic interoperability which is essential to exploit the complexity of an industrial system [23].

1.4. Ontology

The word “ontology” comes from the Greek world, in particular it comes from the philosophy and works of Plato, but in computer science the concept of ontology is expressed as: “a formal description of all the entities in a domain and the relations

existing between these entities” [68]. Ontologies are elements of the so called “symbolic” artificial intelligence, which is a way to organize knowledge to make it comprehensible to a computer, as opposed to machine learning [68]. In order to have a better understanding of what is an ontology we can compare it to some artefacts like relational databases and conceptual data models [24]. By comparing conceptual data models and ontologies the first difference is that data models provide an application-specific representation of the data while ontologies provide an application-independent representation of a specific subject domain, for this reason an ontology can be re-used in several applications [24]. By comparing relational databases with ontologies as knowledge based entities the main difference between the two concepts is that ontologies have an explicit representation of the knowledge and it has some rules that combined with an automated reasoning (beyond plain queries) permits to infer implicit knowledge and detect inconsistencies of the knowledge base [24].

By making this comparisons is a bit easier to understand what an ontology is, but a definition of this concept is needed. For this purpose we are going to use the definition given by the World Wide Web Consortium [24]:

Definition: An ontology being equivalent to a Description Logic knowledge base.

This is a very restrictive definition but in the context of the most frequent application area of ontologies, the semantic web, the tendency is to make the ontology equivalent to a logical theory, in particular to a description logic knowledge.

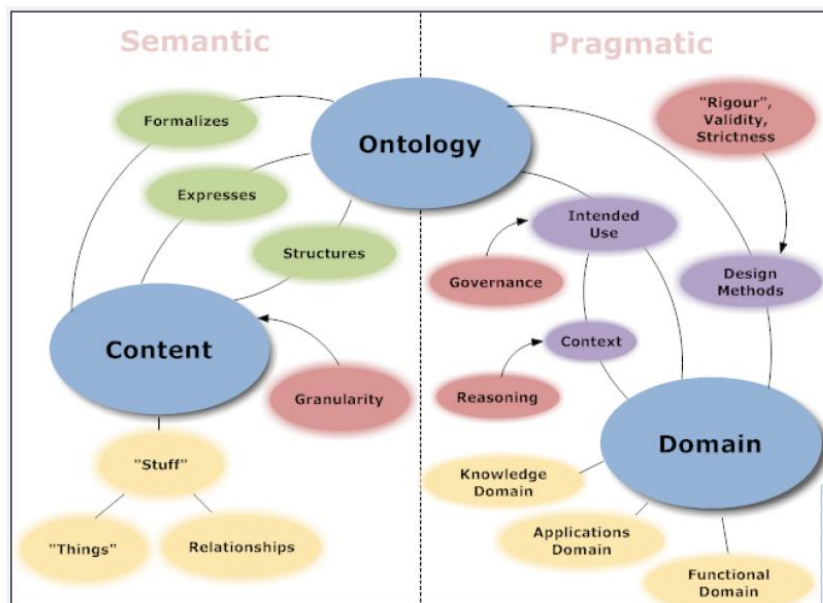


Figure 1-2: Representation of the pragmatic and semantic side of an ontology

In the image 1-2 it is possible to see a representation of the 2 sides of an ontology. Where the “semantic” side has to do with the meaning represented in the ontology and the “pragmatic” side has to do with the practicalities of using ontologies [68], [24].

So an ontology can be defined as the conceptualization of a specification. In particular, given an area of interest, the ontology provides means to describe it by listing related concepts and relating them in the appropriate way. The models developed in this way are, generally speaking, general enough to permit knowledge reuse, so the models generated can be applied in several contexts [23]–[27]. In the ontology engineering field there are several ontologies that can be exploited but the ones this thesis work is going to focus on is the computational ontology. The computational ontologies are ontologies which describe the systems and that can be used for computational purposes, which found their application in the Semantic Web [23], [26]. The Semantic Web is an extension of the Web and these ontologies were used here since [23]:

- They permits to present the information in a different way respect to the information itself;
- The meaning of the information is clearly defined;
- The information presented in this way can be processed by the machines.

The computational capability, the scaling up of data quantity and the interoperability characteristics of the ontology have made ontologies appealing for industry as well. As such, ontologies for industrial purposes are arising, as the literature demonstrates [23]–[28]. Nevertheless, it is necessary to explore if ontologies can be used for ZDM purposes, by checking if there are some gaps that emerge from the literature in order to unlock research opportunities and motivate this research.

2 Literature review on Zero Defect Manufacturing

The aim of this chapter is to describe the procedures adopted to perform the literature review used in this thesis work. This process has been performed to give an exhausting explanation of the Zero Defect Manufacturing concept and its application in the Industry 4.0. Subsequently the results obtained from the performed inquiries are going to be investigated.

2.1. The Literature Review Methodology

The Literature review has been performed in order to contextualize the thesis work within the research environment. Accordingly to what has been said in the *Introduction* about ZDM, this is a concept characterized by a particular set of strategies that are usually carried out when we deal with the objective to reach zero defect in an industrial environment. To reach such an objective a *systematic literature review* methodology has been adopted in order to collect as much information as possible about this concept. The methodology adopted is inspired by [29], as a matter of fact the research has been performed by selecting a certain number of papers that later has been analyzed. This systematic research has been performed on the main abstract and citation databases of the web. This process consists in narrowing the research results by filling up the research prompts of the websites one at a time. The result is going to be a “funnel-like” research process where relevant papers are retrieved and extract extant scientific literature dealing with Zero Defect Manufacturing is obtained. As a matter of fact this research was not done just by reading papers in a random way, but the main abstract and citation databases of the web were investigated. This process has been performed in order to narrow the number of articles found by visiting the most significative websites one at a time. The result of this methodology will be a “funnel-like” research process in which just a limited number of papers that are considered as more significative will be kept.

The methodology adopted can be divided in two main phases:

1. The research phase
2. The analysis phase

The research phase started by looking at the main abstract and citation databases of the web and the number of inquiries performed in each database was 1 and only the articles following these criteria were kept:

- Only engineering-related documents are considered;
- Only peer-reviewed journal papers;
- Only papers including something applicable for the work case are kept, excluding those that are too far from the case considered.

The first two criteria are usually applied leveraging on the database searching engine while the third option is executed at a later time.

2.2. Research protocol on Zero Defect Manufacturing

The starting point of this work of literature was to define the key-words to find specific articles among several fields of science and technology. The *key-word* selected for this purpose was:

- **Zero_Defect_Manufacturing**

After selecting this *key-word* only those articles coming from specific fields of knowledge were kept. As a matter of fact the inquiry was limited to the following fields:

- Industry;
- Manufacturing;
- Production;

This procedure was adopted to limit the amount of possible articles into those fields which are relevant for the research plan. The research phase started by looking at the following databases, Scopus, Web of Science (WoS) and IEEE Xplore where the selection of the keywords reported above led to this result:

- IEEE= 45 articles;
- Web of Science= 140 articles;
- Scopus= 192 articles;

The goal of the work is the ZDM and its implementation, therefore the research carried out had to focus on this topic, but the number of articles reported above is not representative of the total number of articles retrieved since a skimming of the articles

was performed. Here is a list of the reasons why the discarded articles were not feasible for the research:

- Some articles could not be downloaded because of permission issue and so the full text of the article was not obtainable;
- Some article were present twice and so if the article was present in one database it had to be discarded in the other one.

For the reasons reported above the final number of articles obtained from the databases is 150 where the composition of the articles is represented below in the image 2-1:

- IEEE= 43 articles;
- Web of Science= 74 articles;
- Scopus= 33 articles;

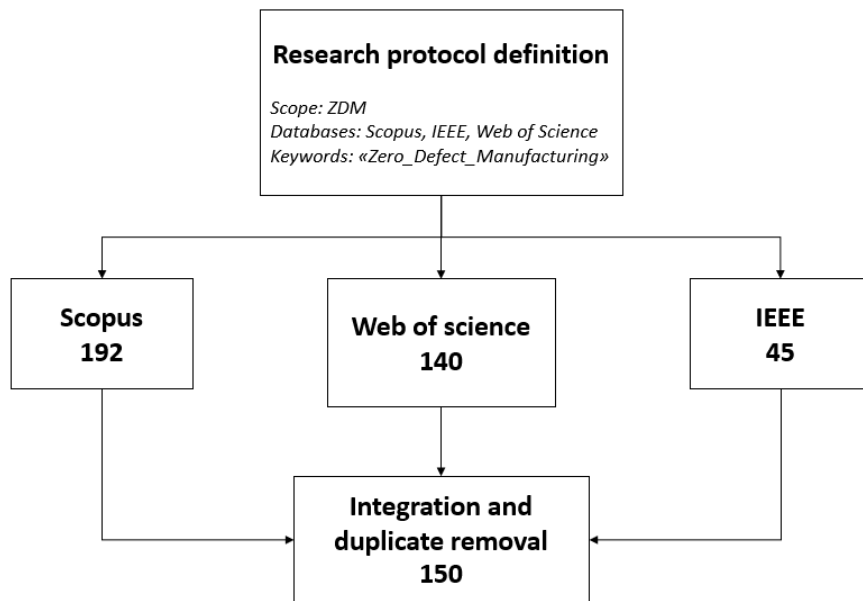


Figure 2-1: Retrieval of articles from the databases

The first two criteria of the research process are summarized in Figure 2-1, but now we have to focus on the third phase of the research protocol in which the screening phase oversees the removal of documents not aligned with the goal of this work.

The analysis phase started with the study of the titles of the articles in order to evaluate whether the articles were useful or not to the research aims. This screening phase took place in two separate times, at first those articles the articles dealing with Zero Defect Manufacturing, quality inspection in industry 4.0 and data management were kept. In

fact in the initial phase of the work, in order to carry out the data collection, several articles were retrieved to have more knowledge on how to perform this activity.

To select the pertinent articles the following procedure has been executed:

1. Title screening;
2. Abstract screening;
3. Content screening;

The title screening consists in the analysis of the title of the paper and if the article is considered aligned with the scope of the thesis work is kept otherwise it is eliminated. After this phase 44 articles are eliminated. Those articles which goes through the title-skimming are then evaluated in function of their abstract in order to see if they are still considered pertinent or if they need to be eliminated. After this second phase 12 articles were eliminated, so 94 articles were kept. The last phase consists in the analysis of the content of the article following the same criteria of the phases before. After this last screening process the number of articles discarded are 6 and so the number of articles kept are 88. This choice was done in order to keep a discrete number of articles to avoid the risk of losing potential important information. As a matter of fact, these 88 articles has been used as a base for having a general knowledge of the ZDM and its possible applications in quality detection and monitoring process together with other papers required to have a greater knowledge on how to treat and analyze the data collected by the machineries.

2.3. Descriptive statistics

In light of the considerations made previously the number of articles kept from the literature review is 88. This choice was made to have a more general knowledge about the Zero Defect Manufacturing, but as soon as the laboratory activities started it was possible to determine with greater precision if an article was useful or not. For this reason a second screening was carried out following this schema:

- 1) is the article closely related to Zero Defect Manufacturing?
- 2) if the article deals with the ZDM which strategies of the ZDM are applied?
- 3) in which sector of application is the ZDM strategy applied? which ZDM strategy is applied?
- 4) is the article useful for the activities to be carried out in this thesis work?

After this second screening process the number of articles kept went from 88 to 26 eligible papers, which were considered the best papers to carry out this thesis work.

The first phase of the second screening process was the following question:

1) is the article closely related to Zero Defect Manufacturing?

The papers held can be divided as follows: 57 papers regarding the ZDM and 31 regarding the data management and data analysis as showed in Figure 2. To distinguish between the two, the following criteria were adopted: if an article specifically mentions zero-defect manufacturing, even though the article deals with data management and data analysis, it is included in the ZDM category since it is dealing with an application of ZDM; otherwise, if the article only deals with data management and data analysis without mentioning ZDM, it belongs to the other category.

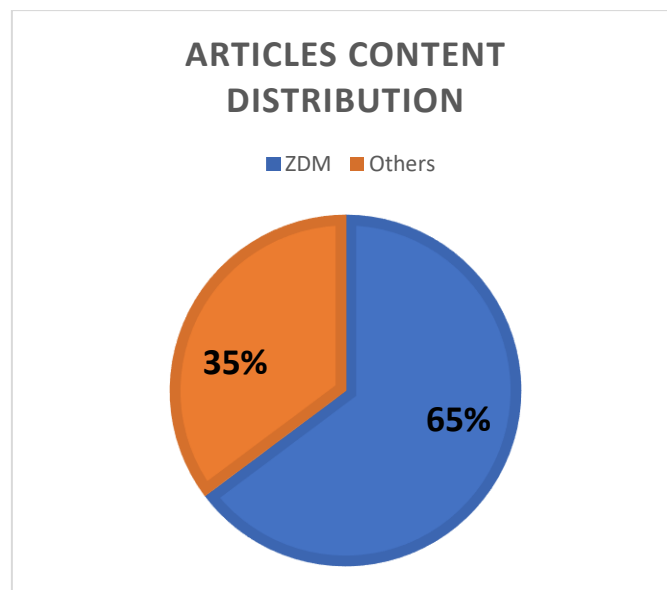


Figure 2-2: Distribution of the collected articles

However, in the image 2-2 for articles related to ZDM we take into account both articles dealing with a specific application of one or more strategy of the ZDM methodology and articles about ZDM in general, therefore articles dealing with what are the potentials of this methodology or a state of art of ZDM. For this reason it is necessary to move on the second phase, so it is necessary to consider the following question:

2) If the article deals with the ZDM which strategies of the ZDM are applied?

In this phase it is necessary to make a distinction between articles that deal with an application of this methodology and articles that talk about it in general terms, so we

have 47 articles which are dealing with applications of the ZDM and 10 articles dealing with a general description of ZDM as represented in Figure 2-3

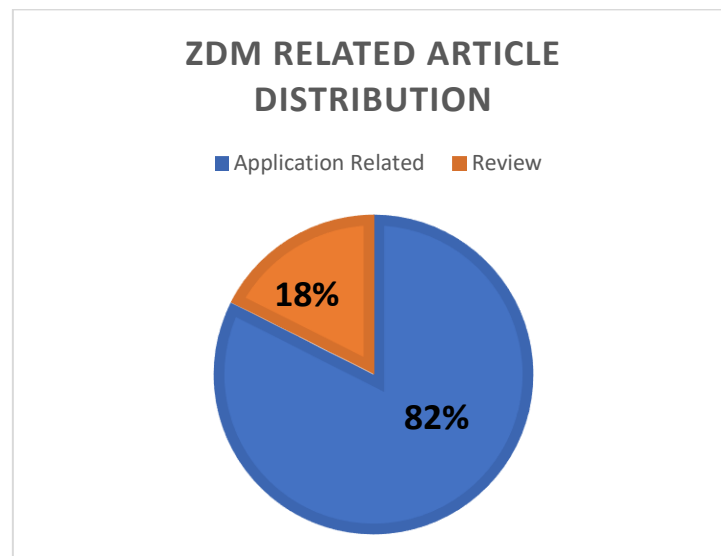


Figure 2-3: Percentages of ZDM articles which deals with ZDM application

So given these 47 articles we can now determine some new metrics which are the sectors of application and the ZDM strategy applied in each article. So it is now possible to answer the following question:

3) In which sector of application is the ZDM strategy applied? Which ZDM strategy is applied?

The sectors will be presented in the figure 2-4 while the strategy will be presented in the figure 2-5

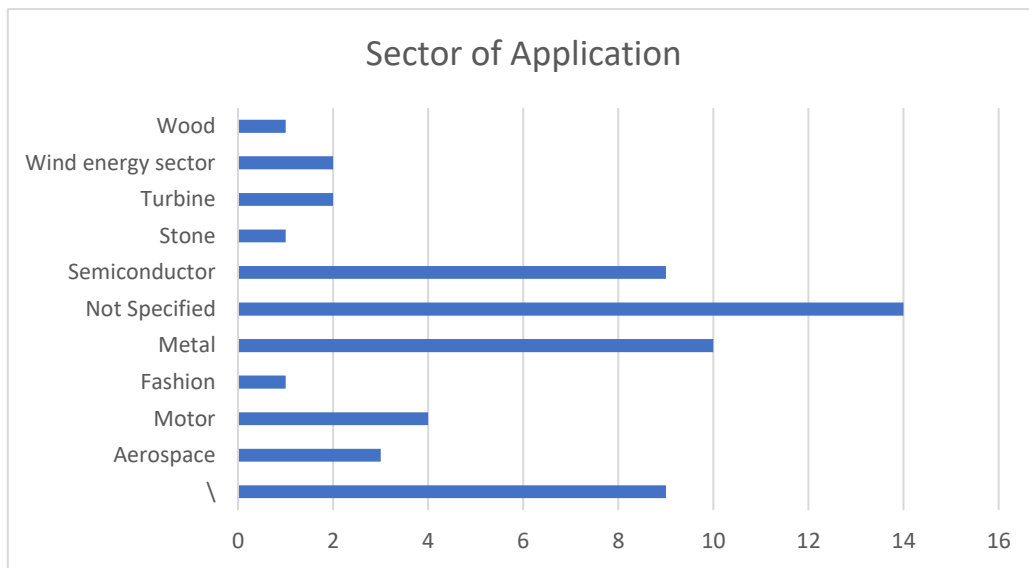


Figure 2-4: Representation of the sectors in which ZDM is applied

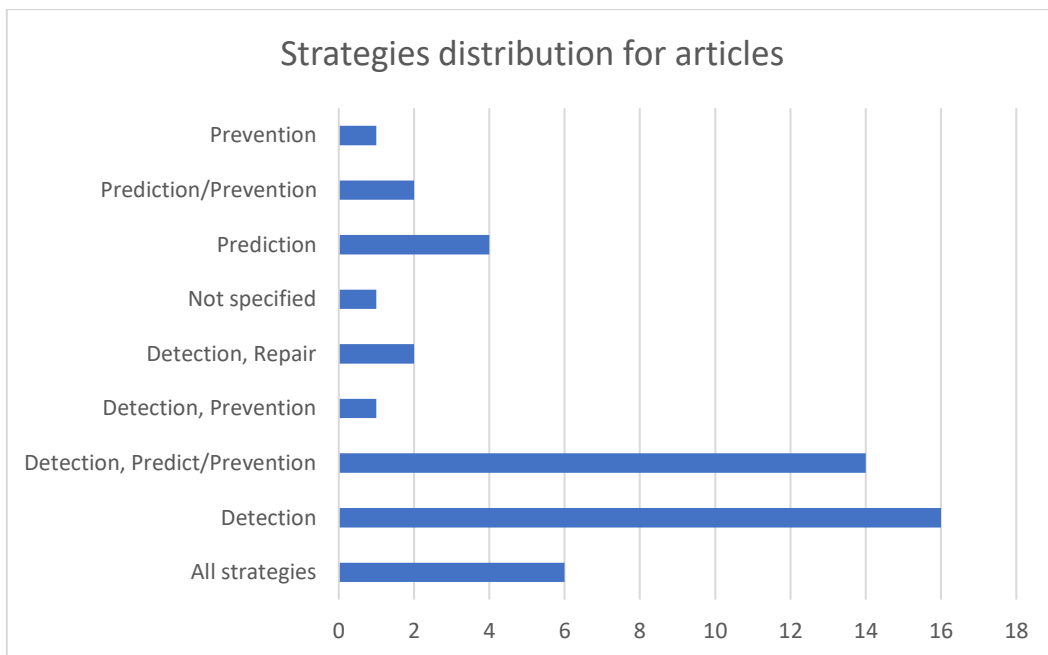


Figure 2-5: Representation of which ZDM strategy is applied in each article

In Figure 2-4 the sector of application of the papers is treated and it is necessary to make a distinction between the terms "Not Specified" and "\". As a matter of fact when the label "Not Specified" is used it means that the article deals with an application of the ZDM but it is not specified which sector of application is considered. If the "\" indicator is present instead, this means the article does not cover the application of ZDM. Figure 2-5 shows the statistics for all strategies used, and we can see that the majority of articles dealt with detection strategies or combinations of strategies using

detection. After this phase was carried out the number of articles kept were 47, but not all these articles were found useful to carry out the objectives of this thesis work. For this reason a last phase was carried out:

4) Is the article useful for the activities to be carried out in this thesis work?

This last phase was performed once the practical laboratory activity was started as it was possible to determine with greater precision the activities that had to be carried out in order to realize the thesis work. So after this skimming phase the number of papers kept went from 47 to 26 eligible papers. The reasons why some of the articles have been discarded are reported here:

- One of the main reason was the inconsistency in terms of field of application with respect to the manufacturing and industrial environment. For example, if an article was relevant in terms of key words, but the field of application was too different from the field we are going to apply it, it was discarded. Here some examples of fields that have not been considered relevant: natural stone industry, aerospace industry, metallic industry and so on.
- Some articles were too specific in the description of the implemented algorithms to perform data analysis and in the management of the data. For these reasons those articles in which it was explained the computer science point of view of the issue or those articles in which there was an in depth description of the physical layer necessary to deal with data analysis and data management were not considered pertinent and so they were discarded since the analysis to be performed does not require this level of detail.
- Sometimes the focus of the articles was too broad and not focused on the operative industrial environment .

Therefore, following this second screening, the results obtained are showed in figure 2-7 while the result of the first screening are present in the figure 2-6.

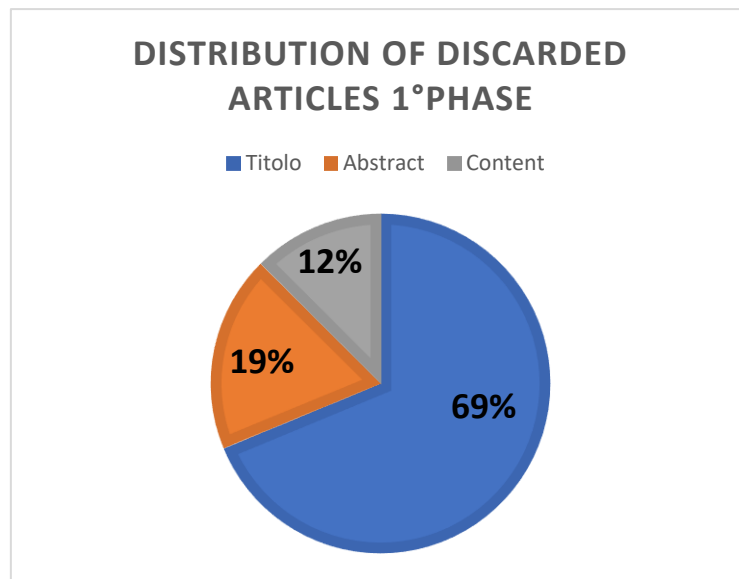


Figure 2-6: Distribution of the eliminated articles after the 1° screening

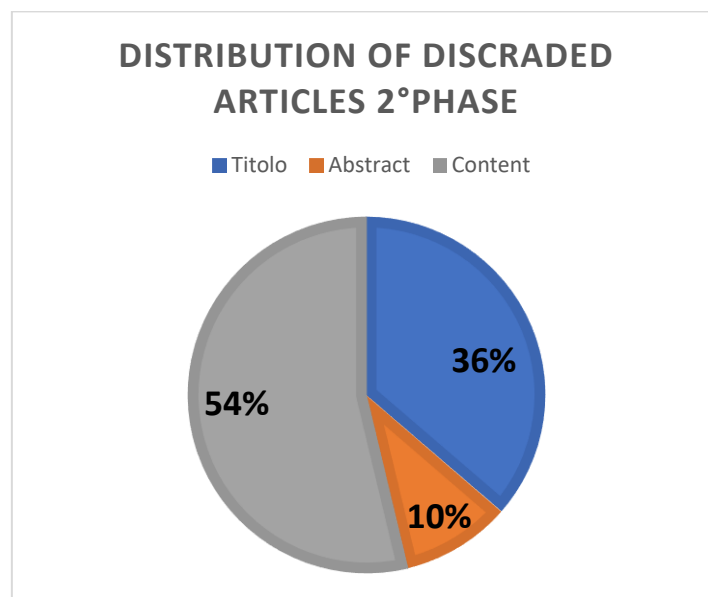


Figure 2-7: Distribution of the articles after the 2° screening

As showed in Figure 2-6 and Figure 2-7 it is possible to see that after the second skimming process, the majority of articles were discarded in function of their content since, as stated previously, an higher number of articles were kept in order to avoid potential information loss, but as soon the laboratory activity started it was easier to detect when an article was useful or not.

2.4. Synthesis of literature review findings

2.4.1. Articles retrieved

The 26 eligible documents are summarized in figure 3-8 and some considerations have been made accordingly. Table 1 has to be read in this way:

- Each row stands for an article;
- There are 2 columns dedicated to the year of publication of the article and one dedicated to the author of the article;
- One column is dedicated to which ZDM strategy is been utilized in the article, if there is an “\” in that row it means that it is not clear which strategy of the ZDM is adopted;
- The last column specify the sector of application of the ZDM strategy and it behave as the one before, so if there is a “\” in the row it means the article does not specify the sector in which the ZDM strategy is adopted;

Year	Reference	Strategy	Sector
2018	Peres et al.	Detection	Motor
2018	Eger et al.	Detection, Predict/Prevention	Not Specified
2021	Fraile et al.	Not Specified	Not Specified
2019	Eger et al.	Detection	Turbine industry
2021	Papageorgiou et al.	\	\
\	Caccamo et al.	Detection	Not Specified
2021	Psarommatis et al.	All strategies	Semiconductor Industry
2015	Teti et al.	Detection, Predict/Prevention	Metallic industry
2018	Eger et al.	All strategies	Not Specified
2017	Vafeiadis et al.	Detection, Predict/Prevention	Metallic industry
2019	Lindstrom et al.	All strategies	Metallic industry
2015	Lu et al.	\	\
2021	Zheng et al.	\	\
2021	Armendia et al.	Detection	Wind energy sector
2019	Eger et al.	\	\
2018	Kharlamov et al.	\	\
2013	Wang et al.	Detection, Predict/Prevention	Not Specified
2021	Psarommatis et al.	Detect, Repair	Semiconductor Industry
2021	Schmidt et al.	All strategies	Not Specified
2021	Mourtzis et al.	Prediction	Metallic industry
2020	Magnanini et al.	Prevention	Not Specified
2021	Psarommatis et al.	Prediction	Not Specified
2021	Lodgaard et al.	\	\
2020	Nikolaidis et al.	Prediction	Not Specified
2021	Azamfirei et al.	Detection	Not Specified

Figure 2-8: Representation of the eligible documents

The 26 papers showed in the figure 2-8 focus on the period 2015-2021 and should be analyzed to identify potential gaps and research opportunities. The first article [30] deals with the deployment of a GOODMAN architecture in the DC motor field and in

particular the deployment of the GOODMAN data model focusing on a shared data representation to enable both generic and application/domain specific semantic and syntactic descriptions with the use of classes to describe the entire system and the relations between its elements. The second article [31] deals with a Knowledge Capturing Platform where the knowledge is extracted from the data collected in the central database of the system. The data collected represent the main source of knowledge about the causes of defect generation and their propagation mechanisms along the production line. This knowledge must be extracted and structured in the KCP so that it can be used for all upcoming developments. The article from Fraile et al. [32] all deals instead with the creation of a ZDM platform to reduce latency and data throughput between the edge tier and the cloud platform while providing to companies the possibility to collaborate using multi-tenant applications. The fourth article [33] is based on the production of a knowledge capturing platform which architecture can be divided in 4 layers: the multi-stage production system, a database, the Knowledge Capturing Platform and the application of the ZDM strategy. The paper from Papageorgiou [34] instead is a survey on the AI technologies which can be used in a production system to move toward ZDM focusing in particular to AI enhanced computer vision and IoT and AI for quality assessment. The work of Caccamo et al [7] focus on the realization of a conceptual model of the Data Quality Management architecture. In particular the focus is on an architecture to cover all the passages from the generation of the data, to the visualization and finally the utilization of the data to execute rapid line qualification and reconfiguration. The paper of Psarommatis et al [4] consists in the development of a digital twin to predicting the results of the developed scheduling tool. Using the DT, multiple ZDM parameter combination sets can be created and plugged into the model. This process generates ZDM performance maps that show the effect of each ZDM strategy at each manufacturing stage under different control parameters. The eighth paper [35] deals with a set of activities done in order to determine machining process conditions, in particular the following activities are analyzed: sensorial perception, data processing and feature extraction and the development of a cognitive decision making paradigm including the adoption of corrective actions. In the work of Eger et al. [16] the focus is on a ZDM architecture where the first pillar consists in a comprehensive Data Acquisition System, which collect and synchronize all the data gathered from different, heterogeneous, multi-resolution and multi-scale data sources distributed in the production line. This data-acquisition system will feed a Data Management Platform that will store and update the acquired data in a structured and formalized

way. This platform is enriched with data management, extraction and aggregation features in order to support the knowledge-based analysis of the relevant inter-stage correlations. In the work of Vafeiadis [36] the focus is on the development of an Early Stage Decision Support System (ES-DSS) with the scope to facilitate real-time inspection, condition monitoring and control - diagnosis at the shop-floor utilizing multiple data streams and run the suitable models to monitor operations and quality performance. The paper of Lindstrom et al. [17] is instead focused on a ZDM approach characterized by the following pillars: i) Monitoring of process parameters ii) Collaborative manufacturing iii) Continuous quality control iv) On-line predictive maintenance v) Data storage and analytics vi) Re-configuration and re-organization of production vii) Re-scheduling of production applied in the metallic industry. The following paper instead, the work of Lu et al [37], is not dealing with ZDM but it deals with ontology design applied to Model-based systems engineering (MBSE) which provides an important capability for managing the complexities of system development and so it has been considered useful for the creation of an ontology. The work of Zheng et al. [18] is focused on digital twin, in particular cognitive digital twin and the paper describe how these methodologies can achieve the vision of the Industry 4.0 by reviewing existing studies relevant to the CDT concept, and further explores its definitions and key features. In the paper is also included a reference architecture based on the RAMI4.0 and some other existing architectures to facilitate CDT development. The work of Armendia et al. [38] deals with the creation of an automated machine tool characterization procedure, called Fingerprint, which permits to determine some Key Performance Indicators of the status of the machine tools based on IoT technologies, like the total power consumption, number of machined references, etc. The work of Kharlamov et al [19] speaks about how semantic technologies and semantic models in particular can be a promising modelling paradigm for Digital Twins, since they allow to capture the knowledge of complex systems and come with a system to design, maintain, query and navigate such models. The next paper is the work of Wang et al. [14] which deals with a Data Mining approach associated with Zero Defect Manufacturing. The paper says that one of the main challenges of ZDM is to deal with massive raw datasets and so how Data Mining can be applied to extract interesting knowledge within huge datasets. The paper of Psarommatis et al. [39] deals with detection and repair-based ZDM strategies to avoid the present of defect in production. In particular this paper focuses on the use of the MASON ontology to describe the production domain and enrich the available data with contextual information, so real time data are used to analyze defects, identify

them and suggest alternative repair plans. The work of Christou [3] instead deals with the presentation of an end to end platform to collect, manage and route data streams from different cyber physical systems in configurable and interoperable ways. The architecture proposed supports data analytics of a machine learning framework to leverage quantitative rule mining. The next paper is the work of Schmidt et al [40] which deals with the introduction of an architecture for intelligent deburring. In particular the article deals with anthropomorphic manipulators and vision systems together with metrological sensors to allow the identification of the burrs and the overall quality and pose of the workpiece. The paper of Mourtzis [41] et al proposes the conceptualization, design and initial development of a platform to utilize the data derived from the industrial environment to optimize equipment design using a combination of data-processing, data-acquisition and simulation. The paper of Magnanini et al. [42] deals with a method to reach Zero Defects by measuring products and comparing them with the required specification and compensate the presence of defects in the downstream production steps in the case there are some deviations. The article of Psarommatis [43] deals with a sub-branch of the Zero Defect Manufacturing strategy: the Prediction strategy. In particular the paper deals with the key control parameters to effectively execute a predictive maintenance by introducing these parameters in a dynamic scheduling tool to execute simulations. The next paper of Lodgaard et al. [44] deals with how the role of the shop-floor operators has changed with time and how they will still be relevant in the industrial system. The article of Nikolaidis et al. [45] focus on how Zero Defect Manufacturing can be used in industrial manufacturing using a prediction of the Remaining Useful Life (RUL) of grinding machines providing useful informations about the deterioration rate of assets, considering as manufacturing environment DANOBAT. The last paper [46] focus on the challenges that quality inspection has to face to be suitable for Industry 4.0, in particular a multi-layer quality inspection framework is developed consisting of (a) the work-piece to be inspected, (b) the measurement instrument, (c) the actuator that manipulates the measurement instrument and possibly the work-piece, (d) an intelligent control system, and (e) a cloud-connected database to the previous resources.

3 Research Design

3.1. Main results of the Systematic Literature Review

During the SLR 150 articles has been retrieved by the following databases: Scopus, Web of Science and IEEE. These articles have gone through 2 screening processes in order to obtain first 88 and then 26 articles. These 26 articles have been considered as the most appropriate articles to deal with the objectives of this thesis work. Once these articles have been selected, the analysis phase proceeded with a full-length reading of these papers looking for useful information to be used to build adequate knowledge background to finally motivate the objective of the thesis work (in accordance with the methodology expressed in [47]). As a matter of fact these documents have gone through a content analysis and some gaps and opportunities present in the literature have been determined in order to motivate this thesis work. The following gaps and opportunities will be presented in the next chapter.

3.2. Research Gaps

Building on the literature the following general features have been found, hereafter classified in three aspects:

- *Methodology*: The documents analyzed show how important is to gather and manage data obtained from the physical assets installed in the shopfloor or from other information sources in order to develop an application for ZDM. As a matter of fact ZDM, respect to traditional Quality Analysis methods, is not based just on analyzing historical data, but also on the analysis of current data in order to reason over defect and learn from them by identifying patterns and extracting knowledge to finally avoid the presence of the same defect over time selecting the most efficient actions for different defect scenarios.
- *Applicability*: Independently from the sector ZDM can be used in a multitude of sectors, even very different from each other, although there are still sectors in which this methodology is more used than in others such as the semiconductor industry or the metalworking industries.
- *Integration of multiple data*: The implementation of the strategies regarding the ZDM is based on the collection and union of data coming from different systems and machines that, for this reason, can use different data schemes or even a

different semantics, so it is necessary to find a method to integrate these data in order to adopt one or more ZDM's strategies

The detailed analysis of the documents allows to highlight and further discuss these features, seen as criticalities in the extant scientific literature. These are the main gaps discovered during the literature review, which can unlock research opportunities and motivate this research work.

- *Repair strategy*: In relation to the articles considered in the literature findings, it is interesting to observe how the Repair strategy (unlike the other strategies) is not adopted very much. The major reason appears to dependent on the fact that, to apply this strategy, there must be a convenient trade-off between the cost and time dedicated to repairing a defective part (rather than discarding it). In particular, the repairing operation must be performed without affecting the overall production of the system, but sometimes this is not possible and for this reason it is easier to discard the part (rather than repairing it).
- *Lack of standardization*: As previously observed different sectors are experiencing the application of the ZDM, however this implies that, for each sector in which this methodology is applied, there is typically a different perspective on what the ZDM is, or on the techniques of the ZDM. This means that standardization is needed to reach wider and general consensus independent sectors, also to avoid possible confusions caused by the numerous tools required for applying quality inspection (QI) methods. In addition to a terminology standardization, the introduction of standards are required to clearly define and standardize the methods related to ZDM. Doing so will make ZDM generally attractive to the manufacturers and simultaneously ensure the results of the ZDM approach.
- *Unified approach*: in the eligible articles, no unified procedure for data collection, management and elaboration is provided in a unified framework. As a matter of fact in the selected papers one or more of these aspects are treated but not all three in the same approach.
- *Product oriented approach*: In the selected articles a product based approach is more frequently adopted to implement the ZDM, nevertheless the procedure usually used in these papers consist in determining the possible presence of defects in the products to check if they comply with the specifications. A broader sense may be given to the ZDM methodology.

According to these gaps and characteristics a solution has been proposed, in particular an ontology model based on a previous ontology, named ORMA, was created and used as base for the realization of a Digital Twin.

3.3. Research Objectives

In function of the gaps found in literature, this thesis work proposes to fill them by introducing an ontology, the ORMA+ ontology. This ontology will be utilized in order to realize a digital twin of the Product obtained by a production line (the FML of the Industry 4.0 Laboratory). Provided the literature gaps, the main research objective for this thesis work is firstly defined, then an illustration is provided on the way through which this solution is expected to fill the gaps found in the literature.

- **Main Objective:** Creation of a Digital Twin using as base an Ontology in order to enable a cognitive capability to apply a Detection and Repair strategy to finally pursue the objective of a ZDM

The Digital Twin realized in this way will be a Cognitive Digital Twin, thus a digital model able to gather data coming from the physical assets and able to determine the quality of the product obtained by the line in function of the data gathered during the production process.

Based on the definition of the main objective the proposed solution fills the gaps found in the Literature in the following way:

- *Secondary Objective 1:* In the Literature it is shown how the Repair strategy of the ZDM is not frequently utilized as, in order to use that strategy, is necessary to find a good trade-off between the utility of the strategy and the cost and time that the introduction of that strategy may determine. The proposed solution consists in the combined use of the Detect and the Repair strategies. In fact, the Detect strategy is the most documented and used strategy in the literature, therefore the use of this strategy makes the ontology conform to what was found in the literature. At the same time, given the versatile nature of the ontology, the Repair strategy was also introduced in the proposed solution in order to make the ontology more extended, innovative and complete, exploiting a combination of triggering factor, the detection strategy, and triggering action, the Repair strategy. This combination may be a way to integrate the Repair strategy in the approach, even if it is not directly solving the trade-off analysis.
- *Secondary Objective 2:* It has been observed that in the literature there are several papers in which an application of a ZDM strategy was presented in different sectors, this may make it difficult to understand which tools / techniques can be used and the requirements necessary to obtain a ZDM in a more general sense. The use of an ontology can prove to be fundamental since the ontology in itself involves the introduction of a standardization of the terms and the introduction

of a series of definitions, which can make this proposed solution also applicable to other contexts. In fact, one of the key points of the ontology is the Knowledge Reuse; for this reason it is possible to use the knowledge present in the ontology, about ZDM, to construct a new specific solution for any new context.

- *Secondary Objective 3:* In the eligible articles, no unified procedure for data collection, management and elaboration is provided in a unified framework. In order to cope with this aspect, the solution proposed can be applied in a real industrial environment where the population of the ontology is realized with the data obtained by the production line (the FML of the Lab) under analysis and, accordingly to the data collected by the ontology, it is possible to determine the quality of the realized product. This may be a demonstrative test-bench for the unified procedure.
- *Secondary Objective 4:* In this thesis work a process approach has been introduced, in particular the status of an asset (the drilling machine of the Laboratory) following the data collected by the line (the FML of the Laboratory). This approach has been introduced since according to [48], even if both the product and process based approach have the same final objective, in function of the application and selection of the equipment one of the two methods can perform better than the other. Accordingly in this thesis work both the two approaches can be implemented to make the user choose which one of the two performs better in the system.

In order to realize an ontological model able to fulfill the following gaps this thesis work has been conducted in the following way. First of all, as announced before, a Systematic Literature Review has been performed in order to gain as much knowledge as possible about ontology and ZDM and to determine which are the gaps in the literature in order to determine a model able to fill these gaps. Then the ontology model has been created using as base some common ontologies like the BFO [48], IAO and CCO ontology in order to use as much common and established knowledge as possible. Then, once these ontologies have been exploited, the IOF and the ORMA ontology [23] have been analyzed since these are specific ontologies dealing with the manufacturing field. Accordingly, the ORMA+ ontology has been defined using these ontologies as base and whenever necessary some elements have been introduced by the author of this thesis work in order to make the ontological model compliant to the main objective of this thesis work. Once the ontological model has been defined it has been verified and, once the verification phase has been completed, the population of

the ontology has been executed in order to check the possible application of the ontology in a simulated context (the Laboratory). This has been done in order to explore all the potentialities of the model and to see in a real context, as the FML in the Industry 4.0 Lab, which are the limitations of the ontological model. Once the population of the ontology has been conducted the results obtained have been discussed in order to explain which take aways have been obtained and what about this ontological model could be of support for the industry.

In order to verify the ontology realized in this thesis work 2 activities have been carried out. The first one was a verification of the logic inside the ontology in order to check if there are any logical contradictions inside the ontological model. The second one was to determine if the ontological model realized was able to answer the following competency questions certifying that the ontology is able to represent the current knowledge about the system.

CQ1 What is the quality of the pieces that the system realizes?

This competency question has been introduced since the ontological model must be able to determine the quality of the realized product in order to determine if the Product is in good state and so it can be sold to the costumer or instead if it needs to be repaired or discarded.

CQ2 Which components realize the product?

This question has been introduced for some reasons. First of all, depending on the quality of the components that make up the product or depending on the operations performed on the components, a product does not necessarily have to have all the components: if the quality of the product is already perceived during the monitoring as insufficient, it is not recommendable to finish the entire production process, then inserting all the components. This question is also asked in order to be able to determine, in the event of a defective product, which components can be recycled for the production of other products .

CQ3 Which are the processes required to realize product x?

The following question has been introduced as the ontological model to be created must be able to determine which production process has been performed accordingly to the quality of the product obtained. In fact, if the quality of the product is insufficient then it is not necessary to carry out a complete production process and the disassembly of the product has to be considered into the process. Otherwise if the product needs to be repaired it is necessary to take into account the repair of the product during the production process

CQ4 Which product/s is/are not feasible considering the current component/asset state?

This competency question has been introduced since, in order to determine the quality of the Product realized by a line (the FML of the Lab), 2 elements have to be considered: the state of the components of the Product and the state of the Asset that process the Product. In function of these 2 elements is therefore possible to determine the quality of the Product and so to understand which products can be sold to the costumers and which one have to be repaired in order to be sold or discarded.

4 Ontology

The aim of this chapter is to provide a description of the ontology developed during this thesis work. The chapter is developed starting from the presentation of a general background about the ontology concept and some relevant ontologies, as, the BFO, IAO and CCO ontology; thereafter, it moves on presenting the methodology used in order to build the ontology and its main steps. Therefore, the reader finds in this chapter the general background of the ontology concept and the structure of the ORMA+ ontology realized.

4.1. Ontology Background

As already mentioned in the introduction the definition of “ontology” can be expressed as: “a formal description of all the entities in a domain and the relations existing between these entities” [68]. To better understand the theory associated with ontologies it is possible to look at an example of ontology as the African Wildlife Ontology (AWO), a basic tutorial ontology based on the examples in the “A Semantic Web Primer” book. This is an ontology containing knowledge about wildlife, such as that giraffes eat leaves and twigs, that they are herbivores, that herbivores are animals, and so on. This is just one example between all the possible results that we can get when we deal with ontologies, as a matter of fact ontologies, in computing and intelligent software development, are defined as an artifact one can play with and manipulate [68]. According to this vision the actual artefact can appear in multiple formats that depends to the aim of the user, but, at the core of it, there is a logic-based representation that the computer can process [24]. To better understand this concept the following element of the AWO ontology will be considered:

“all lions eat herbivores, and they also eat some impalas”

This is an information which may be contained in the AWO ontology, but first it must be converted in a machine-processable format that faithfully adheres to the ontology logic. Nevertheless there are some serializations of the ontology into a text file that are easily computer-processable and the most widely-used is the Web Ontology language (OWL) format which is the RDF/XML format [24]. For what concerns instead the authoring of the ontology there are ontology development environments which render the ontology graphically, textually, or with a logic view, like Protégé.

Now that is more clear what an ontology is it is possible to see what are the possible uses it is possible to have for an ontology. Ontologies, in the information system filed, were used initially to contribute solving the issue of data integration, since an ontology

can be used as a vocabulary for the application [24]. Accordingly over the years ontologies have been used also for different purposes including manufacturing, construction and process industries, because of the intense digitalization that the manufacturing companies are going through [26], but in the history of industry application ontologies have experienced repeated failures. According to [27] one of the main reason for this is that each industrial environment, in which ontologies are applied, believes that the best way to deal with ontologies is to use an ad hoc ontology for that specific case, creating the same problem of data siloes but with ontologies. This occurs because the best practices in ontology modelling, like the usage of a foundational ontology and knowledge reuse, and structured ontology modelling methodologies are often not considered. Especially there is a loss of opportunities in regard to knowledge reuse, which could be promoted building on the modular reuse of ontological knowledge reducing workload and increasing quality [26]. Furthermore there have been several attempts to connect siloed ontologies, but ontology mapping have not been very successful, in fact mappings can be very brickly and became invalid when the ontology evolves [27]. One way to solve this issue is to focus on knowledge reuse and interoperability, as, in the current industrial scenario, intra- and inter-enterprise interoperability of systems is mandatory to reach operational excellence since it guarantees proper data sharing and valorization [26]. “Interoperability”, according to [27], can be defined as the ability of two or more heterogeneous systems to communicate, correctly interpret and act on information meaningfully and accurately with minimal effort. Therefore ontology engineering is central to foster interoperability at a semantic and technical level. At semantic level, ontologies allow creating shared concepts and related meanings between different stakeholders, while, at technical level, ontologies guarantee consistent data formats amongst systems for advanced data management [26]. Reasoning and inference can make ontologies even more powerful, since, they are able to augment the data stored in traditional relational database and deduce conclusions based on the available dispersed informations [26]

In order to reach the objectives previously defined, it is necessary to build an ontology by using an ontological model development and verification, the core steps of the ontology building activity. These steps are generally present in several methodologies, like the ones reported in the papers [50]–[55], cited in the scientific literature and some of them do not only cover the ontology building phase, the ontology specification and implementation, but might include also the entire ontology lifecycle. In this work the methodology adopted is known as METHONTOLOGY since it has the most basic structure of the more advanced methodologies [55].

In figure 4-1 is possible to see the main phases that characterize the Methontology method.

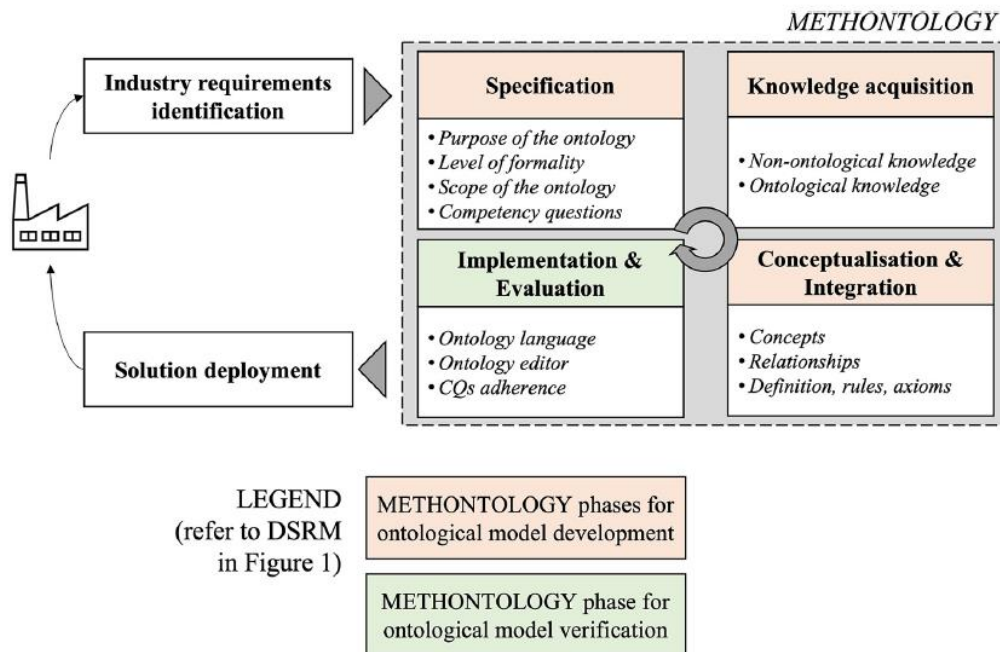


Figure 4-1: Representation of the methontology method

The flow presented above is the one of the paper [23] to build the ORMA ontology, thereafter this flow was used also to develop the ORMA+ ontology. The first activity performed was a planification phase, in which the main tasks to be performed have been evaluated as its purpose and scope. In this part a preliminary phase was introduced to evaluate the necessary requirement that the ontology might need in function of the operations that are performed by the line. Once enough knowledge has been collected, the next phase was the conceptualization phase, in which the knowledge has been transformed in a conceptual model able to describe the problem and its solutions. Once the conceptual model has been realized, it was formalized using a description logic representation system.

Once formalized, in order to build the ontology, an integration phase has been executed where previous existing ontologies have been inserted inside the ontology to reuse the existing knowledge present in these ontologies. The ontologies considered are the BFO [49], the CCO [71], the IAO [70], the IOF [72] and the ORMA ontology [23] which was developed in the industry 4.0 Lab. In order to make the ontology computable two steps were executed: an implementation and an evaluation phase. In the implementation phase the ontology was implemented in a formal language, the OWL language, while for the evaluation phase an Hermit reasoner was utilized to evaluate if the logic of the ontology was correct. Once the ontology is implemented

and verified, it is necessary to insert the ontology in a wider solution in order to support a fictitious industry decision-making using as test bench the Laboratory of Industry 4.0 at Politecnico di Milano. Since the ontology presented in this thesis work is based on the already developed ORMA ontology [23], other two activities need to be considered. The first one is an inner loop in the implementation and evaluation phase that represent the iterative process necessary in order to build the ontology according to the desired performance that you want to achieve. The second one represents the continuous exchange and adjustment of the ontology with the industry in order to satisfy the company requirements.

4.1.1. Knowledge Reuse: BFO ontology

Dealing with knowledge reuse it is necessary to analyze the BFO ontology [49], which according to the definition reported on [69] it can be defined as: “BFO grows out of a philosophical orientation which overlaps with that of DOLCE and SUMO. Unlike these, however, it is narrowly focused on the task of providing a genuine upper ontology which can be used in support of domain ontologies developed for scientific research, as for example in biomedicine within the framework of the OBO Foundry. Thus BFO does not contain physical, chemical, biological or other terms which would properly fall within the special sciences domains.”

At the highest granularity level there is entity which definition is reported hereafter

- entity according to the BFO 2 Reference [001-001] is defined as “An entity is anything that exists or has existed or will exist” [69]

So inside the entity term are contained all the terms of the ontology, so in order to describe all the elements of the ORMA+ ontology it is necessary to take a step further into entity and go to a lower granularity level. Inside entity two terms are present: continuant and occurrent, represent in image 4-2.

- continuant according to the BFO 2 Reference [008-002] is defined as “A continuant is an entity that persists, endures, or continues to exist through time while maintaining its identity” [69]
- occurrent according to the BFO 2 Reference [077-002] is defined as “An occurrent is an entity that unfolds itself in time or it is the instantaneous boundary of such an entity (for example a beginning or an ending) or it is a temporal or spatiotemporal region which such an entity occupies_temporal_region or occupies_spatiotemporal_region” [69]

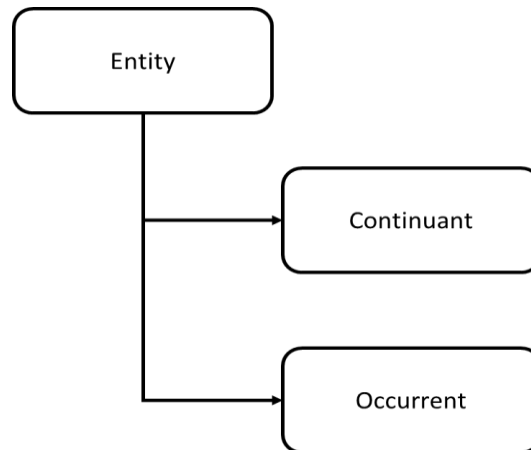


Figure 4-2: Entity class of the BFO ontology

These are the two elements that compose the entity term and, at this layer, there is a first classification of the elements that compose the ontology, as a matter of fact there is a distinction between the elements which continue to exist in time maintaining their identity and the elements that exist for a certain amount of time.

Let's focus at the moment on the continuant class, represented in the figure 4-3, it can be further divided in 3 terms: Generically Dependent Continuant, Specifically Dependent Continuant and Independent Continuant.

- Generically Dependent Continuant according to the BFO 2 Reference [074-001] can be defined as: "b is a generically dependent continuant = Def. b is a continuant that g-depends_on one or more other entities" [69]
- Specifically Dependent Continuant according to the BFO 2 Reference [050-003] can be defined as: "b is a specifically dependent continuant = Def. b is a continuant & there is some independent continuant c which is not a spatial region and which is such that b s-depends_on c at every time t during the course of b's existence" [69]
- Independent Continuant according to the BFO 2 Reference [017-002] can be defined as: "b is an independent continuant = Def. b is a continuant which is such that there is no c and no t such that b s-depends_on c at t" [69]

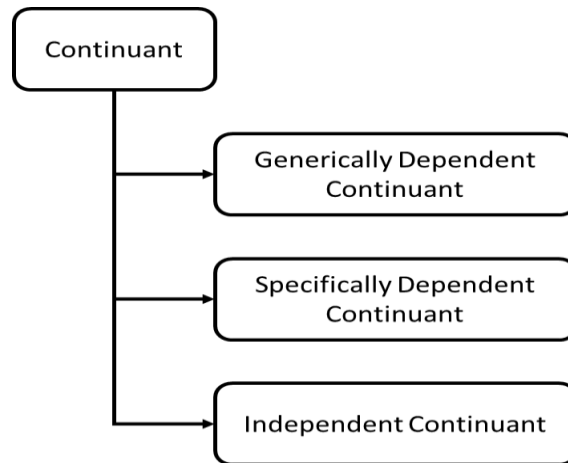


Figure 4-3: Continuant class of the BFO ontology

For what concerns the occurrent term, figure 4-4, it can be divided in 2 terms: Process and Process Boundary.

- Process according to the BFO 2 Reference [083-003] can be defined as “p is a process = Def. p is an occurrent that has temporal proper parts and for some time t, p s-depends_on some material entity at t.” [69]
- Process Boundary according to the BFO 2 Reference [084-001] can be defined as “p is a process boundary =Def. p is a temporal part of a process & p has no proper temporal parts” [69]

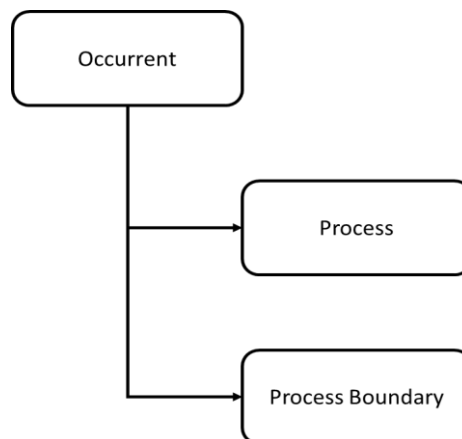


Figure 4-4: Occurrent class of the BFO ontology

Concerning instead the continuant part, which is represented in figure 4-5, it is necessary to deal with the relationships among the different elements of the ontology, in particular the correlations between the elements that compose the ontology.

At this point of the analysis a further decomposition is required, in fact the elements that has been described until now are not specific enough to describe the elements that compose the ontology. Therefore the following classes will be seen in greater detail: Independent Continuant and the Specifically Dependent Continuant. The first

one can be further divided as material entity and immaterial entity. These elements do not have a definition given by the BFO, but they can be defined in a broad sense following what is written in [56] in this way:

- Material Entity defines a continuant which has a portion of matter as part
- Immaterial Entity defined as a continuant which does not have a portion of matter as part

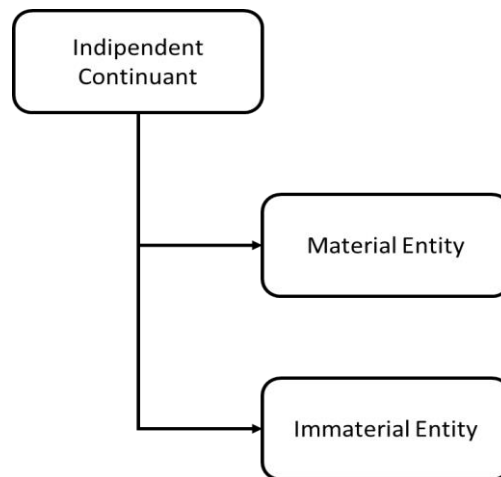


Figure 4-5: Independent Continuant class of the BFO ontology

Whereas for portion of matter the definition reported in the [56] is going to be adopted, so a portion of matter has to be intended as anything that includes elementary particles within it. Concerning instead specifically dependent continuant, the elements to deal with at a lower granularity level are Quality and Realizable Entity, which definition is reported below and their representation is in figure 4-6

- Quality according to the BFO 2 Reference [055-001] can be defined as “a quality is a specifically dependent continuant that, in contrast to roles and dispositions, does not require any further process in order to be realized” [69]
- Realizable Entity according to the BFO 2 Reference [058-002] can be defined as “To say that b is a realizable entity is to say that b is a specifically dependent continuant that inheres in some independent continuant which is not a spatial region and is of a type instances of which are realized in processes of a correlated type” [69]

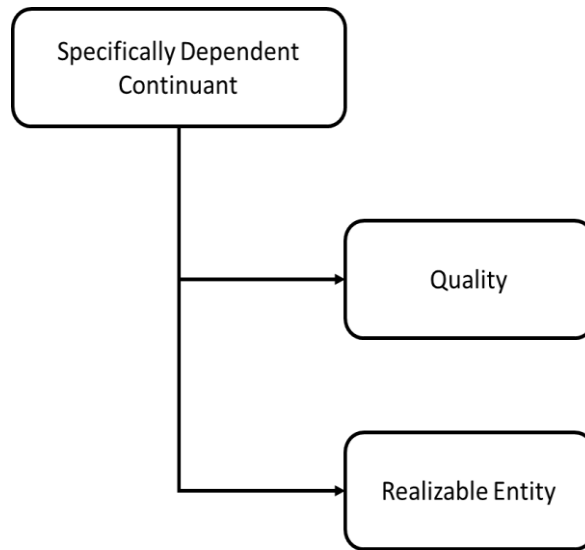


Figure 4-6: Specifically Dependent Continuant class
of the BFO ontology

Going deeper in detail inside Material Entity the classes object and object aggregate are present, figure 4-7. These classes can be defined as:

- Object according to the BFO 2 Reference [024-001] can be defined as “b is an object means: b is a material entity which manifests causal unity of one or other of the types listed above & is of a type (a material universal) instances of which are maximal relative to this criterion of causal unity” [69]
- Object Aggregate according to the BFO 2 Reference [025-004] can be defined as “b is an object aggregate means: b is a material entity consisting exactly of a plurality of objects as member_parts at all times at which b exists” [69]

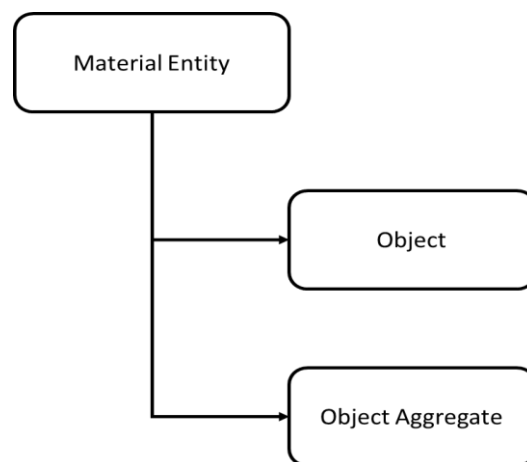


Figure 4-7: Material Entity class of the BFO ontology

So the BFO structure presented in the ontology can be represented as showed in the image 4-8

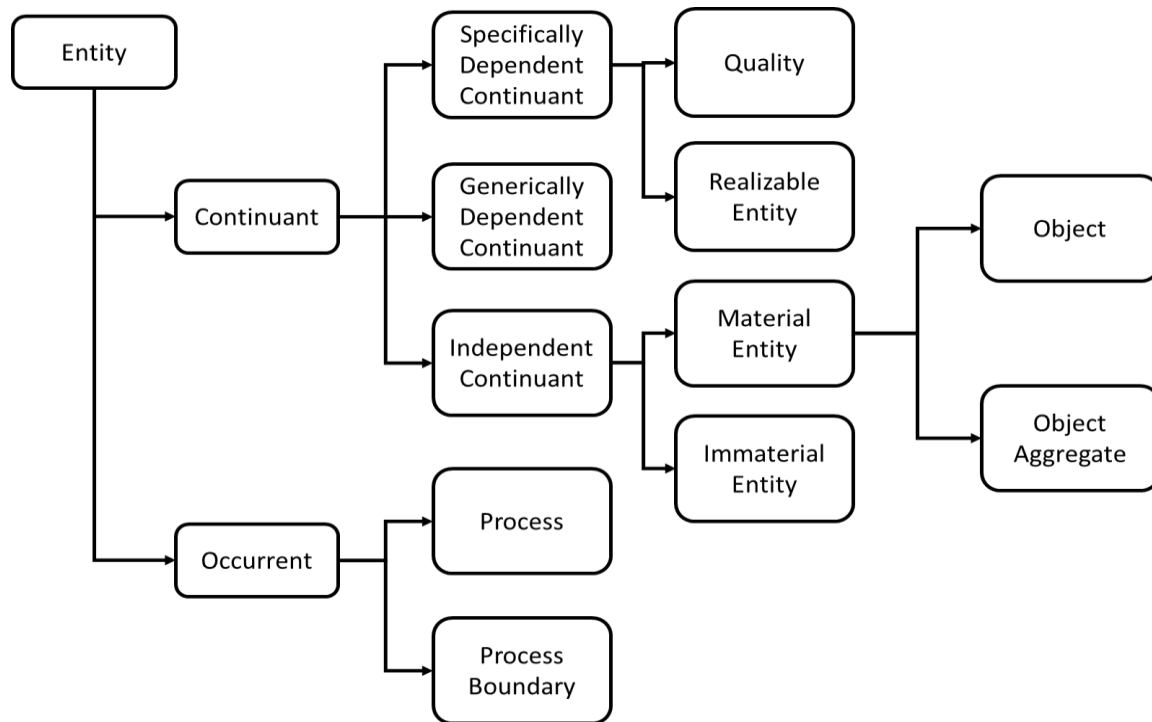


Figure 4-8: Complete representation of the BFO ontology

The definitions provided above make easier to understand what are the elements that may be present in the ORMA+ ontology, but, nevertheless, the definitions provided are still not sufficient to clearly and thoroughly describe what are the four key elements of the ontology to deal with ZDM, namely Asset, Product, Quality and Process. In fact, BFO constitutes only the backbone of the ontology, but, to be able to fulfill the main objective of this thesis work, it is necessary to talk about what are the terms added specifically into the ontology to deal with the ZDM strategy.

4.1.2. Knowledge reuse: IAO ontology

IAO ontology is one of the key ontologies that make up the ORMA+ ontology realization. However, not all elements present in this ontology has been used to create ORMA+, so only those elements present in ORMA+ are analyzed. In particular, the elements that have been taken from this ontology are reported below:

- Identifier according to the IAO definition can be described as: “An identifier is an information content entity that is the outcome of a dubbing process and is used to refer to one instance of entity shared by a group of people to refer to that individual entity”. [70]

The identifier class is presented below in figure 4-9

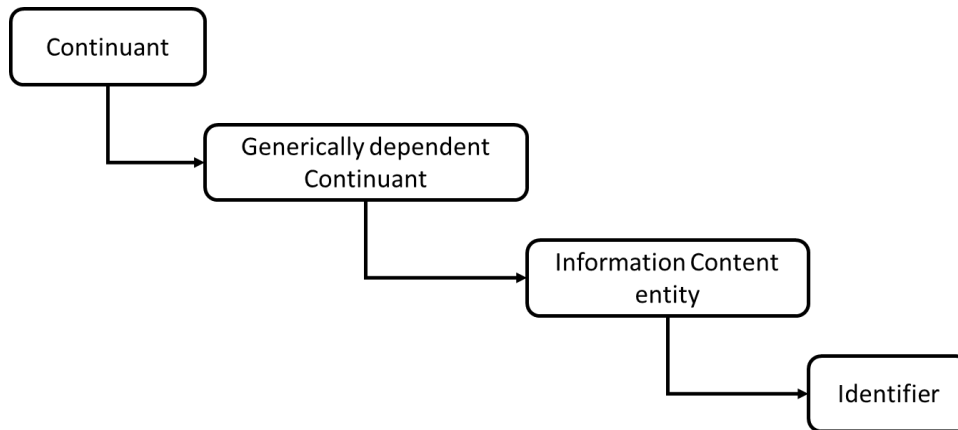


Figure 4-9: Identifier class of the IAO ontology

About the Specifically Dependent Continuant class, showed in figure 4-10, the following elements have been introduced:

- Physical Object Quality according to the IAO definition can be described as: “A quality which inheres in a continuant” [70]
- Physical Quality according to the IAO definition can be described as: “A quality of a physical entity that exists through action of continuants at the physical level of organization in relation to other entities” [70]
- Disposition according to the BFO2 Reference [062-002] can be described as: “b is a disposition means: b is a realizable entity & b’s bearer is some material entity & b is such that if it ceases to exist, then its bearer is physically changed, & b’s realization occurs when and because this bearer is in some special physical circumstances, & this realization occurs in virtue of the bearer’s physical make-up” [70]
- Function according to the BFO2 Reference [064-001] can be described as: “A function is a disposition that exists in virtue of the bearer’s physical make-up and this physical make-up is something the bearer possesses because it came into being, either through evolution (in the case of natural biological entities) or through intentional design (in the case of artifacts), in order to realize processes of a certain sort” [70]
- Role according to the IAO definition can be described as: “A realizable entity the manifestation of which brings about some result or end that is not essential to a continuant in virtue of the kind of thing that it is but that can be served or participated in by that kind of continuant in some kinds of natural, social or institutional contexts” [70]

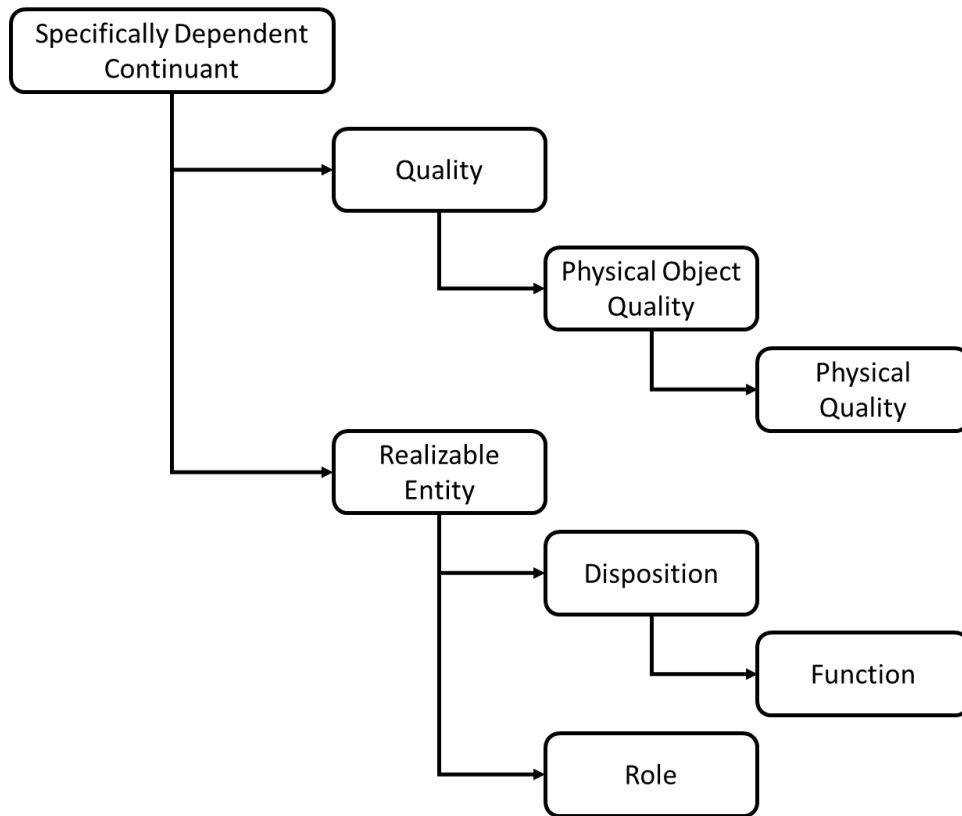


Figure 4-10: Specifically Dependent Continuant class of the IAO ontology

- Planned Process according to the IAO definition can be described as: “A process that realizes a plan which is the concretization of a plan specification”. [70]

In the figure 4-11 the Planned Process class is represented

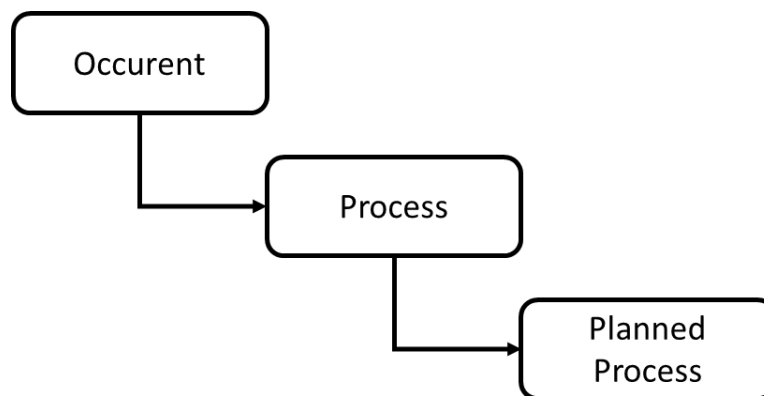


Figure 4-11: Planned Process class of the IAO ontology

4.1.3. Knowledge reuse: CCO ontology

According to the reference [71] Common Core Ontologies (CCO) comprise twelve ontologies that are designed to represent and integrate the taxonomies of generic classes and relations across all domains of interest. As a matter of fact the CCO is a

mid-level extension of Basic Formal Ontology (BFO), since BFO aims to represent the most generic categories of entity and the most generic types of relations that hold between them. CCO extends from BFO so every class in CCO is asserted to be a subclass of some class in BFO and CCO adopts the generic relations defined in BFO. In particular between the 20 ontologies that constitutes CCO, the Artifact Ontology is the one being considered in this thesis work. Hereafter some elements have been taken from this ontology and inserted in the ORMA+, these elements are reported below in the figure 4-12 and their definition are reported here:

- Artifact according to the Artifact Ontology can be defined as: “An Object that was designed by some Agent to realize a certain function” [71]
- Transducer according to the Artifact Ontology can be defined as: “An Artifact that is designed to convert one form of energy to another” [71]

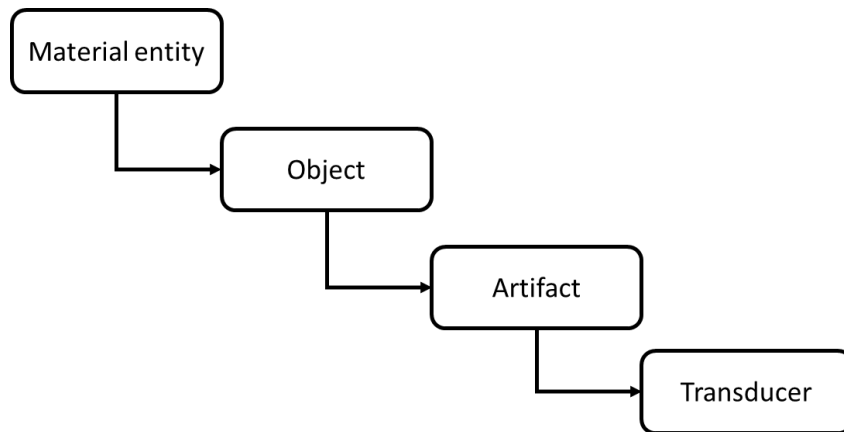


Figure 4-12: Artifact and Transducer class of the CCO ontology

- Artifact Function is represented below in the image 4-13 and according to the Artifact Ontology can be described as: “A function that inheres in some Artifact in virtue of that Artifact being designed to be used in processes that require that function to be realized” [71]

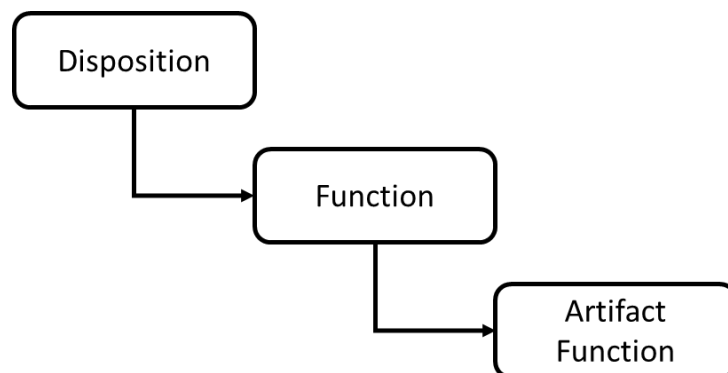


Figure 4-13: Artifact Function class of the CCO ontology

- Artifact Identifier, figure 4-14, according to the Artifact Ontology can be defined as: “A Designative Information Content Entity which designates some Artifact” [71]

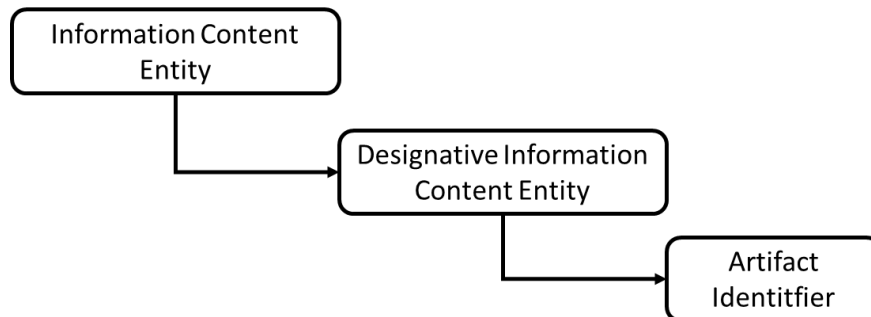


Figure 4-14: Artifact Identifier class of the CCO ontology

- Artifact Function Specification according to the Artifact Ontology can be defined as: “A Directive Information Content Entity that prescribes some Artifact Function and which is part of some Artifact Model” [71]
- Quality specification according to the Artifact Ontology can be defined as: “A Directive Information Content Entity that prescribes some Quality” [71]

These 2 classes are represented in the following image, figure 4-15.

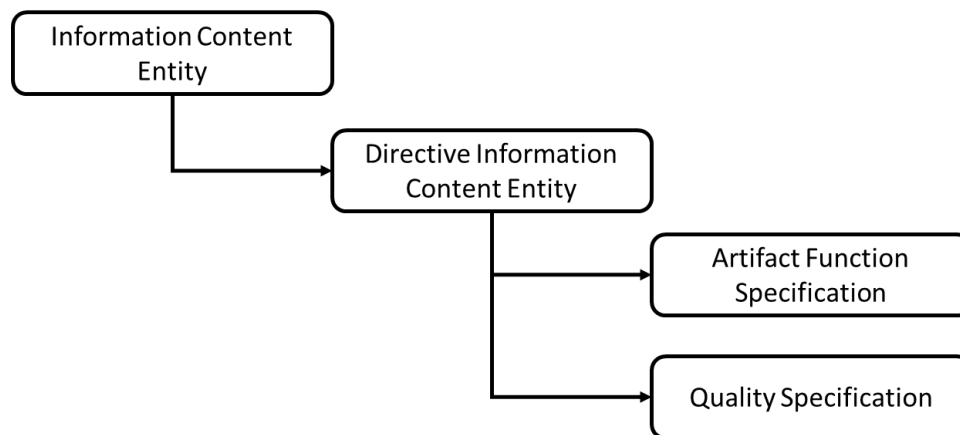


Figure 4-15: Artifact Function Specification and Quality Specification class of the CCO ontology

4.2. Methontology applied for ZDM

The development of the ontology starts with the identification of the necessary requirement that the ontology must satisfy. One of the key aspect to fulfill is the integration of data and information to get a coordinated and joint decision-making between the organizational functions, giving particular attention to production. As a matter of fact the greater effort has been put in the identification of what the

potentialities of the ontology could be in the identification of the state of the products, as healthy, to be repaired and defective, and the asset health state in order to develop a tool able to support ZDM strategies in an industrial environment. In order to reach such an objective the procedure shown in figure 4-1 is followed.

4.2.1. Specification

The domain of the ORMA+ is the industrial or production field. In particular, being ORMA + based on the ORMA ontology, the interest of this ontology is the same of the reference one. So, as reported by the paper [23], the interest of the ORMA ontology is directed mainly toward those companies in which the production management is characterized by high flexibility/reconfigurability. In fact the focus of the ORMA ontology are those automatic production systems having flexible routings, even though, it can also be used for configurations having discrete part production and not so flexible routings [23]. Therefore, the basic idea around ORMA+ ontology is to support companies utilizing the data coming from the production process to apply one or more ZDM strategies to improve decision making.

For this reason the ontology, as the ORMA ontology before, can be classified as a subdomain ontology, so an ontology which is enough specific for a certain industry, but not so specific to not be utilized for multiple contexts [23]. For these reasons inside the ontology there are terms like Asset and Asset_Component, which are related mainly to the maintenance field and there are also terms like product and process which instead refer to the production field. The foundation ontologies for the ORMA+ are the ones present in the chapter 4.1 (Ontology Background), which provides a concise high-level conceptualization.

The next step to define accordingly to [55] is to determine the competency questions that the ontology wants to answer. Therefore the following questions have been determined:

CQ1 What is the quality of the pieces that the system realize?

CQ2 Which components realize the product?

CQ3 Which are the processes required to realize product x?

CQ4 Which product/s is/are not feasible considering the current component/asset state?

By answering these questions is possible to verify the ontology while certifying that the ontology is able to represent the current knowledge of the system.

4.2.2. Knowledge acquisition

The ORMA+ ontology relies on knowledge reuse and so, to build the ontology, it is necessary to retrieve knowledge from the maintenance field and the production system as well. For this reason, the knowledge base for the ORMA+ ontology is the same of the ORMA one and so, accordingly to the paper [23], the knowledge base of the ontology is:

- On the side of knowledge for maintenance:
 - PHM knowledge: two works by Nuñez and Borsato [57], [58] and the ISO 13374;
 - Physical decomposition: the work by Zhou et al. [59], the ISO 14224 and the FMECA-related IEC 60812;
 - General knowledge: the work by Karray et al. [25], and the ontology developed by the IOF (Industrial Ontologies Foundry, link);
- On the side of knowledge for product and process:
 - Ontological modular structure and main concepts: the work by Colledani et al. [60];
 - Manufacturing process formalisation: the ontologies MSDL [61] and MRO [62];
- Scientific knowledge of the research group is also elicited;
- Relevant Ontological sources:
 - Information artifact ontology (IAO);
 - Common Core Ontologies (CCO).

The retrieval of this knowledge, added with some consideration made on the basis of the functioning of the line of the Industry 4.0 Lab, permits to balance the terminology utilized as well as selecting the most appropriate terms for the ontology.

4.2.3. Conceptualization and Integration

Once the ontological and non-ontological knowledge has been collected, it is necessary to provide a definition of the terms included in the ontology and the relations that holds among them. Following the structure of the METHONTOLOGY methodology [55], in this phase of integration, the reuse of knowledge can advance the ontological knowledge, so, following this practice, the ORMA+ ontology is grounded on BFO [49][69], IAO [70], CCO [71] and the ORMA ontology [23].

Given the use of the ORMA+ ontology there are four main concepts modules that can be recognized as relevant for the production system: the asset, the product, the process and the quality. For what concern the first three concepts the definition reported for

the ORMA ontology in the paper [23] are going to be used, while quality was introduced specifically for the ORMA+:

- The Asset is the physical asset that characterize the production process and so the asset module describes the physical asset and provides to explain the function of the asset;
- The Product contains all the information regarding the production process and all the necessary working steps in order to obtain the product;
- The Process is the conjunction element between the product and the asset since it links the function of the asset with the needed operations that have to be performed on the process in order to obtain it;
- The Quality includes the description of the physical quality of the product obtained, as, it bonds the product with the process by relating the state of the product components with the operations performed to obtain the product.

4.2.4. Implementation and Evaluation

The ontological model is implemented in OWL (Web Ontology Language) since this language supports reasoning. The used ontology editor is Protégé, because it is open-source and allows the installation of several plug-ins to better investigate the ontology characteristics. To interrogate the ontology, SPARQL queries and Snap SPARQL queries are used. The CQs tested are the ones derived from the specification phase of the methodology, that are:

CQ1 What is the quality of the pieces that the system realizes?

CQ2 Which components realize the product?

CQ3 Which are the processes required to realize product x?

CQ4 Which product/s is/are not feasible considering the current component/asset state?

These questions are used for the verification of ORMA+ considering the three products that the production system realizes: `regular_product` (Dummy Cellphone) which represents the good quality product that requires all the operations of the line; `product_to_be_repaired` (Repairable Cellphone) which represents an intermediate quality product that require all the operations of the line, but not until it is repaired and `defective_product` (Defective Cellphone) which represent a product to be discarded and so it does not need all the operations of the line.

4.3. Proposed ontology for ZDM

Once the backbone of the ontology has been analyzed, it is necessary to define the other concepts which are present in the ontology and the relationships that hold between them. To discuss the missing elements, it is necessary to follow the order seen during the description of the BFO [49][69], IAO [70] and CCO [71]. In particular, a cascade approach will be followed whereby starting from the main concepts of the ORMA+ ontology, namely product, asset, quality and process, all the elements of the ontology will be analyzed.

4.3.1. Product

4.3.1.1. Product Class

The *Product* class in the ORMA+ ontology is a subclass of the *Object_Aggregate* class named *Assembly*, as it is possible to see from the image 4-16. The reason of this choice lies in the fact that the ontology was meant to treat production systems characterized by a combination of assembly and transforming operations, like the FML of the Laboratory in which it will be tested. For this reason, an *Assembly* is an aggregation of objects, specifically an aggregation of components. Accordingly to what has been said the *Assembly* class has been defined as:

- *Assembly*: “an *Object Aggregate* which consists in the fitting together of manufactured parts into a complete machine, structure, or unit of a machine”.

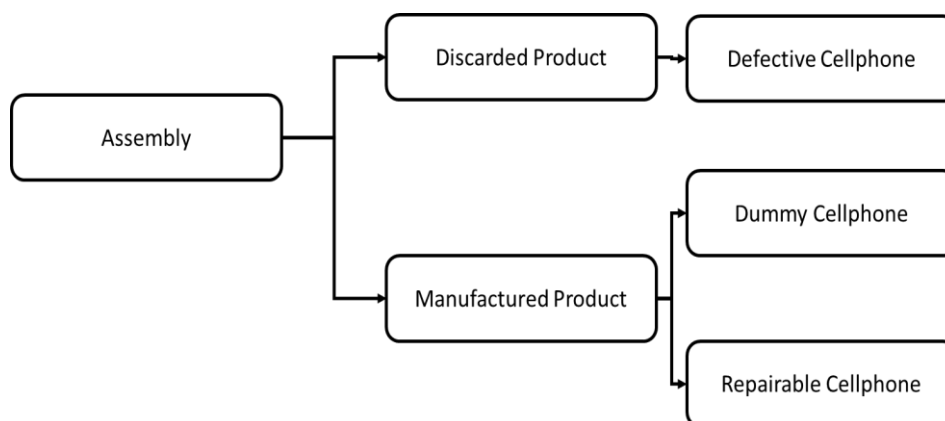


Figure 4-16: Assembly class of the ORMA+ ontology

As it is possible to see from the image 4-16, this class can be further divided into subclasses to describe the different kinds of products which can be produced by the assembly line. In the context of this thesis work, the assembly class has two different subclasses, the *Manufactured Product* class and the *Discarded Product* class. Where the

Manufactured Product class has the sub classes *Dummy Cellphone* and *Repairable Cellphone* while the *Discarded Product* class has the *Defective Cellphone* sub class. The logic behind these classes is that a product obtained from an assembly process can have a sufficient quality to be sold to costumers (directly at the end of the process in the line, or after a repair activity) or it has to be discarded accordingly to the ZDM logic. Product quality was, therefore, defined as the feature that distinguishes manufactured and discarded products. For this reason, it is necessary to define those elements that permits to determine the product quality. There are two decisive factors in this paper that determine the quality of the product: the state of the product components and the state of the manufacturing assets through which the process is performed. However, these aspects will be treated with more care and detail when dealing with the corresponding chapter of this thesis work. Accordingly to what has been described until now it is possible to give the following definitions:

- Manufactured Product can be described as: “An assembly whose quality is sufficient to be sold to a costumer”;
- Discarded Product can be described as: “An assembly whose quality is not sufficient to be sold to a costumer”;
- Dummy Cellphone can be described as “A Manufactured product obtained through a regular assembly process”;
- Repairable Cellphone can be described as “A manufactured product obtained through a repairable assembly process”;
- Defective Cellphone can be described as “A discarded product obtained through a defective assembly process”.

4.3.1.2. Elements correlated to the Product class

Now that the *Assembly* class has been analyzed it is necessary to see the other classes which are strictly correlated to the *Assembly* class, namely *Component* and *Assembly_Position* class.

4.3.1.2.1 Assembly_Position class

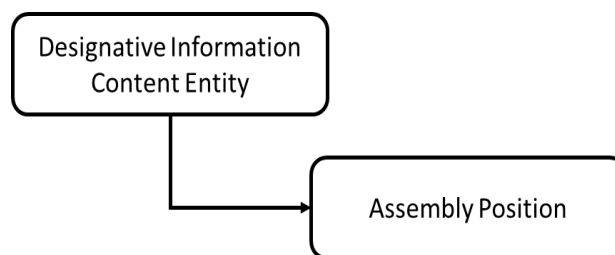


Figure 4-17: Assembly Position class of the ORMA+ ontology

According to the definition reported above a *Designative Information Content Entity* is: "An *Information Content Entity* that consists of a set of symbols that denote some *Entity*", so, in the ORMA+, an entity can be accurately indicated by its position, as it is possible to see in the figure 4-17. This element is introduced into the ontology by the author of this thesis work since in the literature there was found no article dealing with this class or a similar one. This class was made to create a very simple tracking mechanism whereby each time a product passes in front of a station, its position is updated. This is done to check each stage of the process, where the process is, so that in the event of a mistake it is possible to restore the product's location and check the last station where the product was handled. According to this, the definition adopted for this concept was:

- Assembly Position: "A *Designative Information Content Entity* that describe the location of a product".

4.3.1.2.2 Component class

Concerning the *Component* subclass some clarifications have to be made. This subclass is taken from the ORMA ontology, but in the ORMA+ ontology has a different meaning with respect to the ORMA ontology [23]. In fact in the latter the class *Component* was defined in this way: "Artifact that is part of the functional unit and does not perform specific function per se, but concur, with other components to the realization of a simple function" [23]. This definition of *Component* is very asset centric, while in the ORMA+ ontology the assumption is that, in the *Component* class, there could also be entities which compose the product. Accordingly *Component* can be defined in this way:

- Component can be defined as: "artifact that is part of the Asset or the Assembly and does not perform specific function per se, but concur, with other components to the realization of a simple function".

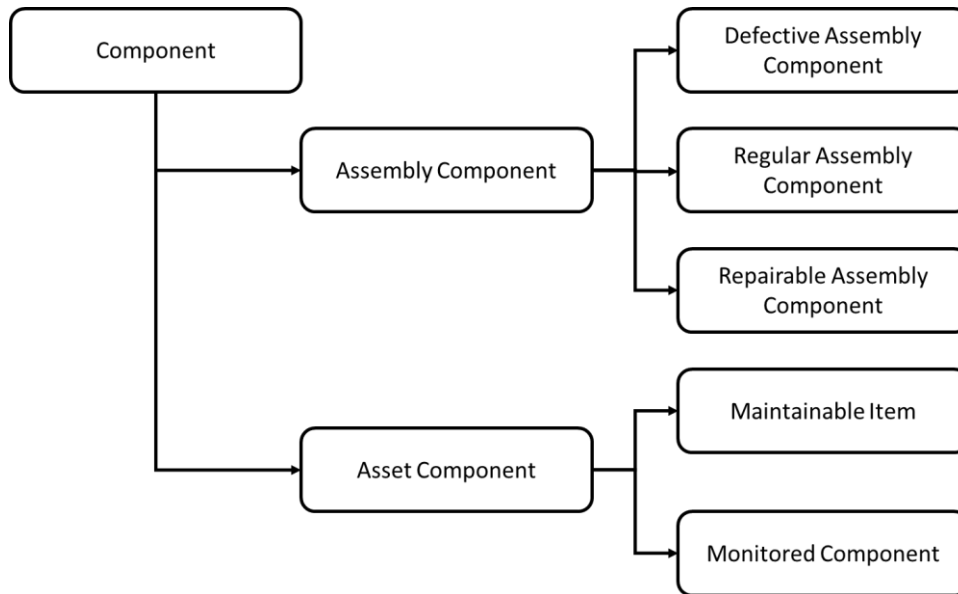


Figure 4-18: Component class of the ORMA+ ontology

In this section of the thesis work only the *Assembly Component* class will be taken into consideration. From the image 4-18 it can be seen that there are 3 possible component types: *Regular Assembly Component*, *Repairable Assembly Component* and *Defective Assembly Component*. The logic behind these classes is that *Assembly Component* represents the elements that compose the *Product*, but since there are 3 different kind of products it is therefore possible to distinguish 3 kinds of components. In particular what is different is not the component, but the information that the component has, as a matter of fact a back cover is still a back cover if the product is repairable or in perfect condition but the information contained is different. In fact in one case the back cover is a component of a *Dummy Cellphone* while in the other case it is a component of a *Repairable Cellphone*. So accordingly the following definitions can be given:

- Defective Assembly Component can be defined as: “A component which is part of a Defective Product”;
- Regular Assembly Component can be defined as: “A component which is part of a Regular Product”;
- Repairable Assembly Component can be defined as: “A component which is part of a Repairable Product”.

4.3.1.3. Object Properties of the Product class

The Object properties which are correlated to the Product class are the following ones in figure 4-19 and 4-20.

- has_component (Inverse of is_component_of)

Domain: Material_Entity

Ranges: Component

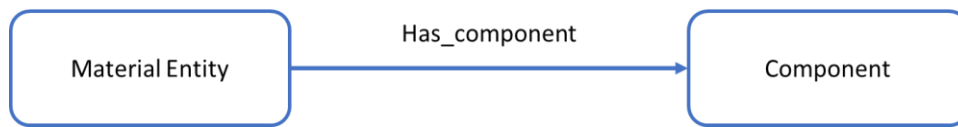


Figure 4-19: Has_component property of the ORMA+ ontology

- has_position

Domain: Assembly

Ranges: Position

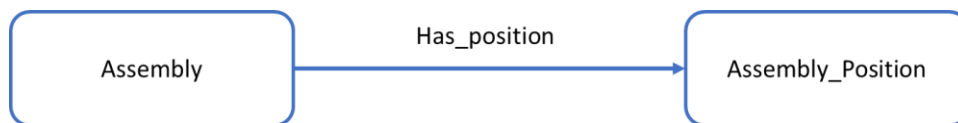


Figure 4-20: : Has_position property of the ORMA+ ontology

4.3.2. Asset

4.3.2.1. Asset class

The *Asset* class is a sub class of the *Artifact* class together with the *Component* and *Transducer* class. This class was taken from the ORMA ontology [23] and so its definition has been kept the same of the ORMA ontology. So the *Asset* class can be defined as:

- Asset can be defined as: “the artifact performing space and species processes on products, tools and pallets, i.e., their movements as well as the changing of their shape and dimension, respectively” [23]

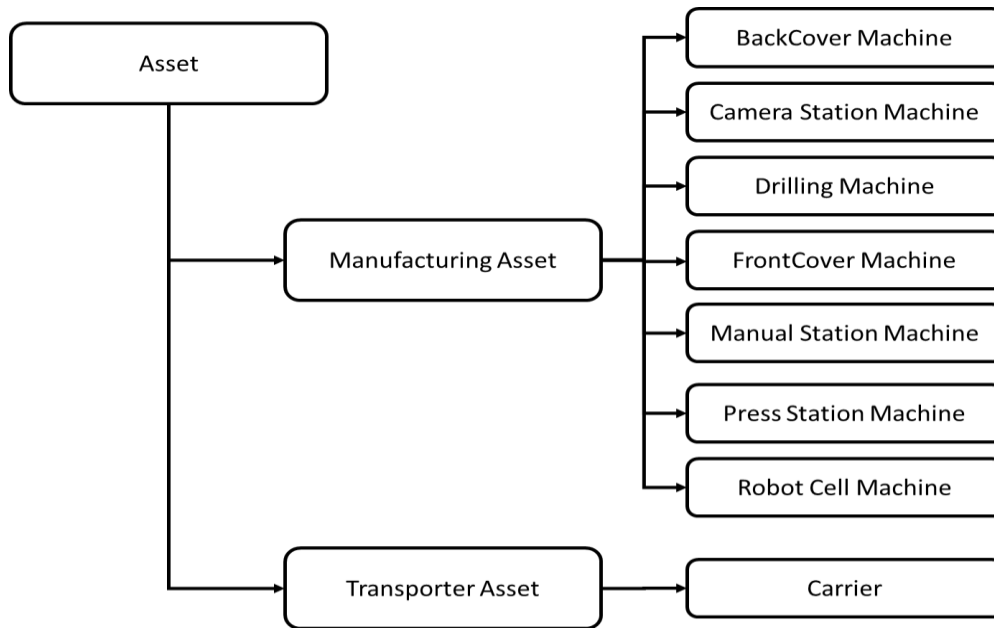


Figure 4-21: Asset class of the ORMA+ ontology

By looking at the image 4-21 it is possible to see that the *Asset* subclass includes the description of the physical assets of the Flexible Manufacturing Line considering all the machines which are present in the line and their related functions. For this reason the author of this thesis work introduced the distinction between the *Manufacturing Asset* and the *Transporter Asset*. The first one is an asset dedicated to perform manufacturing activities, like the insertion of a component or a transformation process like a drilling operation, while the second one is an activity dedicated to the movement of the entities. Accordingly these two elements can be defined like:

- Manufacturing Asset can be described as: “An Asset dedicated to perform manufacturing operations”;
- Transporter Asset can be defined as: “ An Asset dedicated to perform transport operations”.

By looking more in detail at these subclasses, it is possible to see all the assets which are present in the line analyzed during this thesis work. Before giving a definition to these subclasses, an elucidation has to be done. In fact between all of these classes just the *Drilling Machine* class will be taken into account. The logic behind this decision is due to the structure of the Flexible Manufacturing Line where the ontology will be tested. In fact only the drilling machine of the FML can simulate a malfunction while the others machine are considered to operate perfectly.

- FrontCover_Machine can be defined as: “The machine dedicated to the insertion of the front cover”;

- Drill_Station Machine can be defined as: “The machine dedicated to the drilling of the front cover”;
- Robot_Cell Machine can be defined as: “The machine dedicated to the insertion of the PCB and the fuses”;
- Camera_Station Machine can be defined as: “The machine dedicated to the check of the Product quality”;
- BackCover_Machine can be defined as: “The machine dedicated to the insertion of the back cover”;
- Press_Station Machine can be defined as: “The machine dedicated to the pressing of the product”;
- Manual_Station Machine can be defined as: “The machine dedicated to the extraction of the product from the line”.

4.3.2.2. Elements correlated to the Asset class

4.3.2.2.1 Descriptive Information Content Entity class

Accordingly to the CCO ontology this class can be defined as: “An Information Content Entity that consists of a set of propositions that describe some Entity” [71]. In particular this class is characterized by the following sub-classes: *Failure Mode and Effect Analysis* and *Failure Mode Code*.

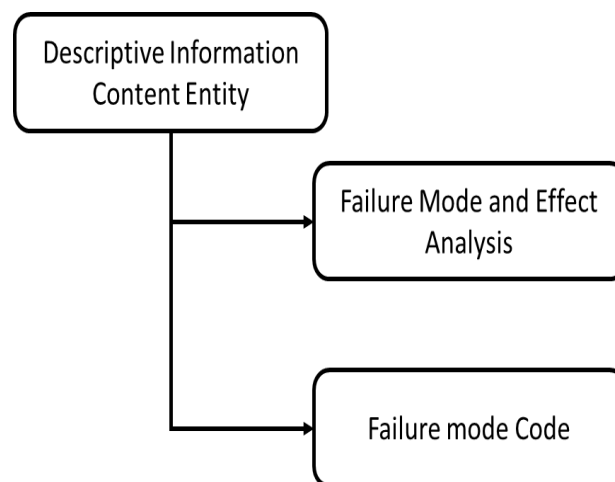


Figure 4-22: Failure Mode and Effect Analysis and Failure Mode Code class of the ORMA+ ontology

According to the IOF the classes showed in figure 4-22 can be defined as:

- Failure Mode Code according to the IOF ontology can be defined as: “a Descriptive Information Content Entity describing a failure mode” [72];

- Failure Mode and Effect Analysis according to the IOF ontology can be defined as: “a descriptive Information Content Entity that describes the output of a failure mode and effects activity” [72].

These elements are IOF classes that were introduced because ORMA+ is an ontology that has to deal with industrial environments and the application of ZDM strategies. For these reasons, it is necessary to introduce some elements that allow to describe the effect that a failure can have on the system under control and elements that associate a code with the failure mode, to facilitate the identification of the fault. This is done to better describe the reality of an industrial system, where a mistake can always occur and cause consequences in the production process. These factors are also useful for the implementation of ZDM strategies, as it is true that the goal of ZDM is to reduce defects to zero but it is impossible to completely avoid defects. For this reason it is important to have some elements that can facilitate fault identification and describe the effect that the faults may have on the system enabling to learn from the errors.

4.3.2.2.2 Asset Component class

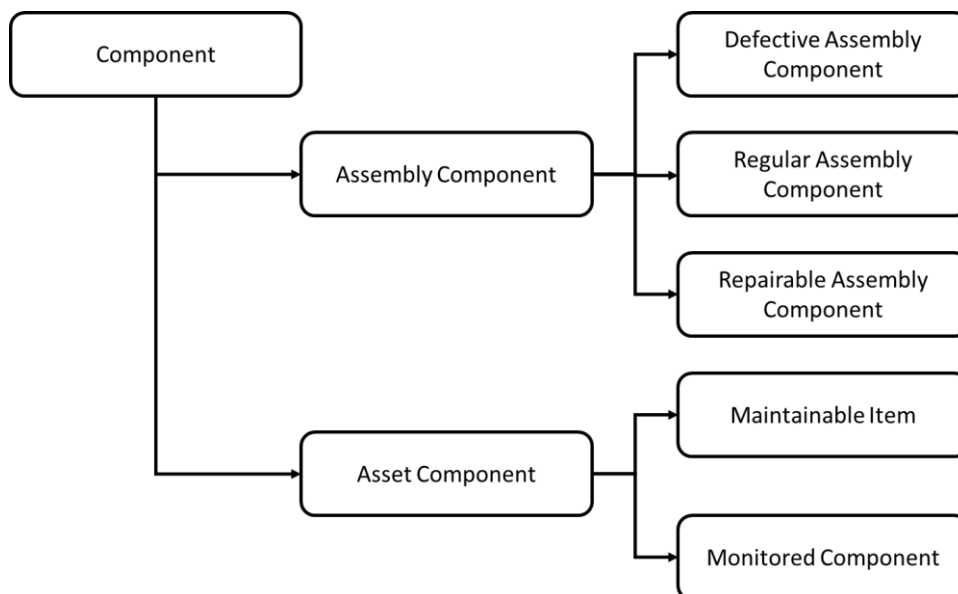


Figure 4-23: Asset Component Class of the ORMA+ ontology

The component class was treated in the *Product* class, figure 4-18, but now it is necessary to deal with the *Asset Component* class shown in figure 4-23. This class was taken from the ORMA Ontology [23], so the following sub classes have already been defined in the ORMA ontology as:

- Maintainable Item can be defined as: “a CCO artifact that bears a maintainable_item_role in the context of maintenance strategy” [23];

- Monitored Component can be defined as: “A ORMA Component that participates in an ORMA Monitoring process” [23].

The logic behind these classes is that an *Asset* may fail and so there are some elements of the *Asset* which have to be monitored and maintained in order to prevent the failure.

4.3.2.2.3 Asset Function class

The Asset Function class is a subclass of the IAO *Function* class [70], as represented in figure 4-24, which was introduced in the ontology by the author of this thesis work. This element was introduced in the ontology to express what the asset function may be in the Flexible Manufacturing Line. So the definition of this class is:

- Asset Function can be defined as: “A BFO:function which an AO:artifact have to deliver value to an AO: organization” [70].

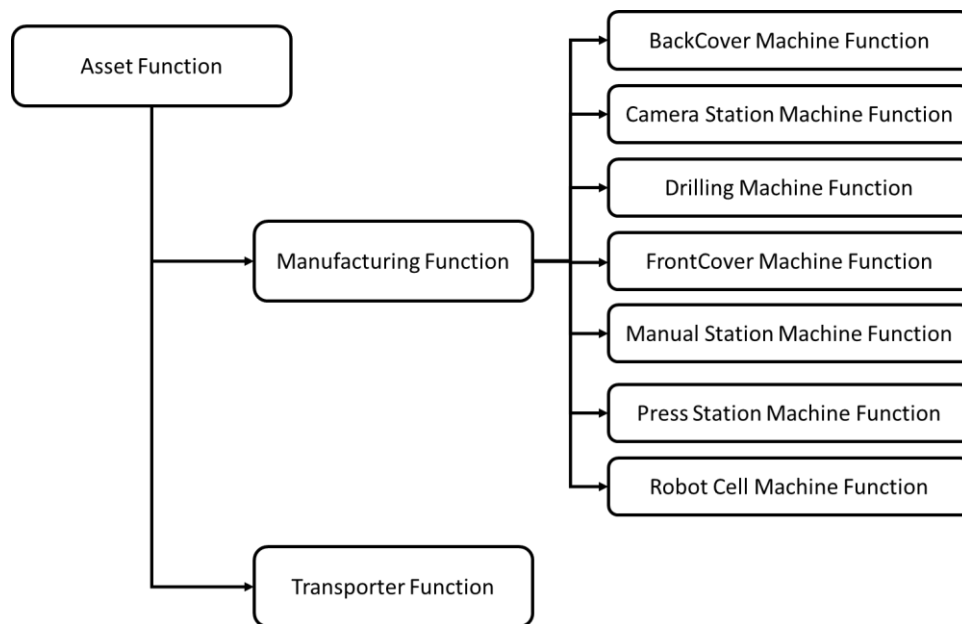


Figure 4-24: Asset Function class of the ORMA+ ontology

The following class, as shown in the image 4-24, can be divided into two subclasses: the *Manufacturing Function* subclass and the *Transporter Function* subclass. These two classes were introduced into the ontology to distinguish between the functions that the assets in the line can have. In particular, when it comes to *Manufacturing Functions*, it is possible to see a list of all the manufacturing assets that were introduced before and their respective functions.

4.3.2.2.4 Failure Cause class

The *Failure Cause* class is an IOF class which is a subclass of the *Occurrent* class.

- Failure Cause can be defined in several ways: "Circumstances associated with design, manufacture, installation, use and maintenance that have led to a failure (ISO 14226) OR The circumstances during design, manufacture or use which have led to a failure (iec 60300.3.11)" [72].

This class has one subclass, the *Asset Failure Cause*, shown in figure 4-25. The logic behind the introduction of this class has partially been stated before, this is done to better describe the reality of an industrial system, like for the *Failure Mode Code* and *Failure Mode and Effect analysis* class. This class can be defined in the following way:

- Asset Failure Cause can be defined as: "the Failure Cause that an Asset has" [72].

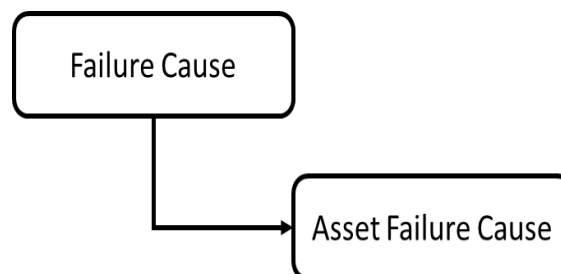


Figure 4-25: Failure Cause class of the ORMA+ ontology

4.3.2.3. Object Properties of the Asset class

The Object properties which are correlated to the Asset class are the following ones in figure 4-26 and 4-30.

- has_failure_cause

Domain: Artifact

Ranges: Failure_Cause

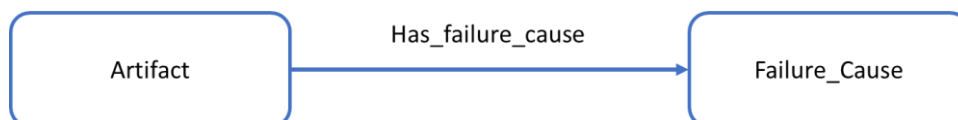


Figure 4-26: Has_failure_cause property of the ORMA+ ontology

- has_failure_event

Domain: Artifact

Ranges: Failure Event

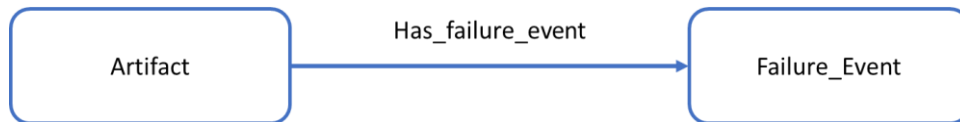


Figure 4-27: Has_failure_event property of the ORMA+ ontology

- has_failure_mode_and_effect

Domain: Artifact

Ranges: Failure Mode and Effect Analysis

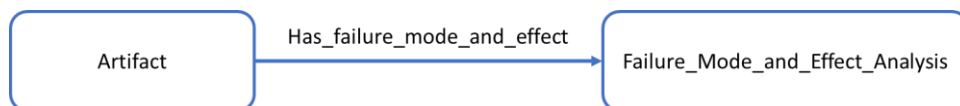


Figure 4-28: Has_failure_mode_and_effect property of the ORMA+ ontology

- has_failure_mode_code

Domain: Artifact

Ranges: Failure Mode Code

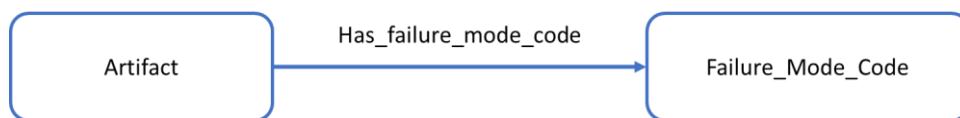


Figure 4-29: Has_failure_mode_code property of the ORMA+ ontology

- has_function (inverse of is_function_of)

Domain: Artifact

Ranges: Function

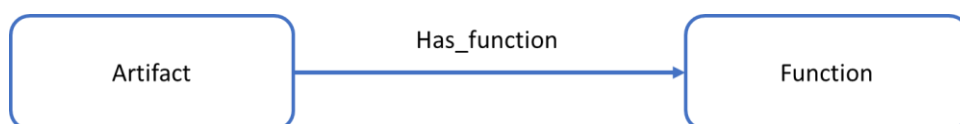


Figure 4-30: Has_function property of the ORMA+ ontology

4.3.3. Quality

4.3.3.1. Quality class

The quality class is the third pillar of the ORMA+ Ontology and, as was already mentioned in the chapter “Ontology Background”, the Quality class is a sub class of the Specifically Dependent Continuant class, as it is possible to see from the figure 4-31.

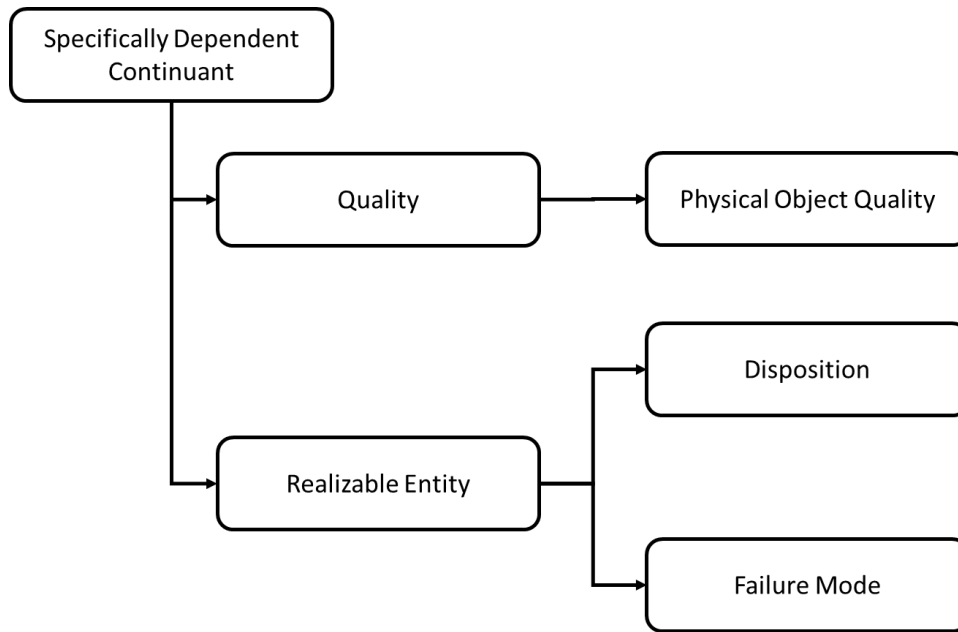


Figure 4-31: Quality class of the ORMA+ ontology

The Quality class has been defined as:

- Quality according to the BFO Ontology Reference [055-001] can be defined as: “a quality is a specifically dependent continuant that, in contrast to roles and dispositions, does not require any further process in order to be realized” [69].

This definition alone is not sufficient to make the ontology suitable to apply the ZDM strategies, so other elements have been introduced in the ontology in order to cope with this aspect. In particular the sub class Physical Object Quality, in figure 4-32, was introduced and all the sub classes which are correlated to this sub class.

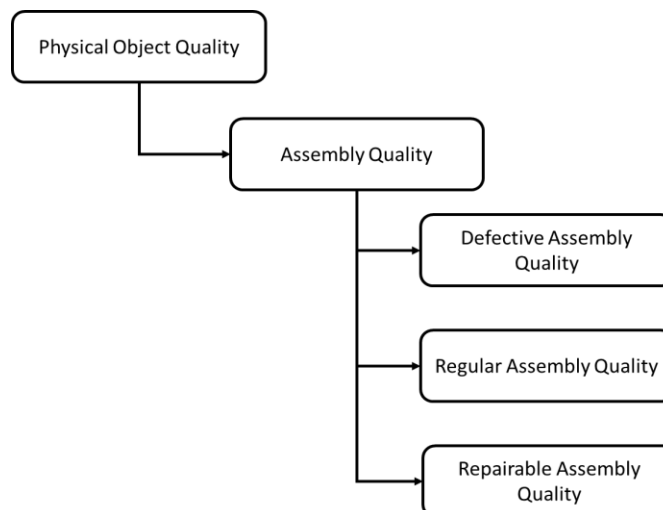


Figure 4-32: Physical Object Quality class of the ORMA+ ontology

- Physical Object Quality can be defined as: “The physical object quality is the quality that a physical object has”;

- Assembly Quality can be defined as: "The Assembly quality is the quality that an Assembly has";
- Defective Assembly Quality can be defined as: "The defective assembly quality is the quality that a defective product has";
- Regular Assembly Quality can be defined as: "The regular assembly quality is the quality that a regular product has";
- Repairable Assembly Quality can be defined as: "The repairable assembly quality is the quality that a repairable product has".

The logic behind the introduction of these classes stems from the fact that the goal of the ORMA+ ontology is to handle the application of ZDM in an industrial context and *Quality* class is the enabling factor for this. In fact in the ORMA+ ontology the application of detection and repair strategies are associated with the *Quality* class, which is the necessary elements for applying these strategies. In fact, *Quality* class is correlated with *Feature* and *Specification* class, which are core elements of the detection strategy used in the ontology. Nevertheless, *Feature* and *Specification* classes are necessary elements for recognizing product quality. As a matter of fact *Feature* class must correspond to *Specification* class, otherwise, product quality is not good enough to sell the product to customers. For this reason, a detection phase must be performed to ensure that the specifications are met and, depending on the results of the detection phase, the quality of the product can be established. Products can belong to three different quality categories: *Dummy Cellphone Quality*, *Repairable Cellphone Quality* and *Defective Cellphone Quality*. If the *Product* quality belongs to the *Repairable Cellphone Quality* then it is possible to apply the Repair strategy, while if the *Product* quality belong to *Defective Cellphone quality* a Disassembly strategy can be actuated. In the specific case of *Repairable Cellphone quality*, the Repair strategy consists of repairing the necessary components in order to meet specification and making possible to sell the *Product* to the costumers. Instead, in the case of *Defective Cellphone Quality* a Disassembly strategy is actuated where the *Product* is disassembled and the good state components are recycled.

4.3.3.2. Elements correlated to the Quality class

4.3.3.2.1 Feature Class

The *Feature* class is taken from the IOF ontology and its definition is:

- Feature according to the IOF Definition: " "feature" (like "characteristic") is an umbrella term including in its coverage domain: qualities, parts of a material product (for example, chromium plating), a hole within a material product (for

example, a button hole), as well as information entities, as well as metalevel characteristics such as availability, reliability, average dimensions, as well as characteristics of processes such as rate, continuity, and so forth” [72]

In the ORMA+ ontology this concept is utilized to describe the characteristics that an artifact, like an asset or a component, and an assembly, may have, as showed in figure 4-33.

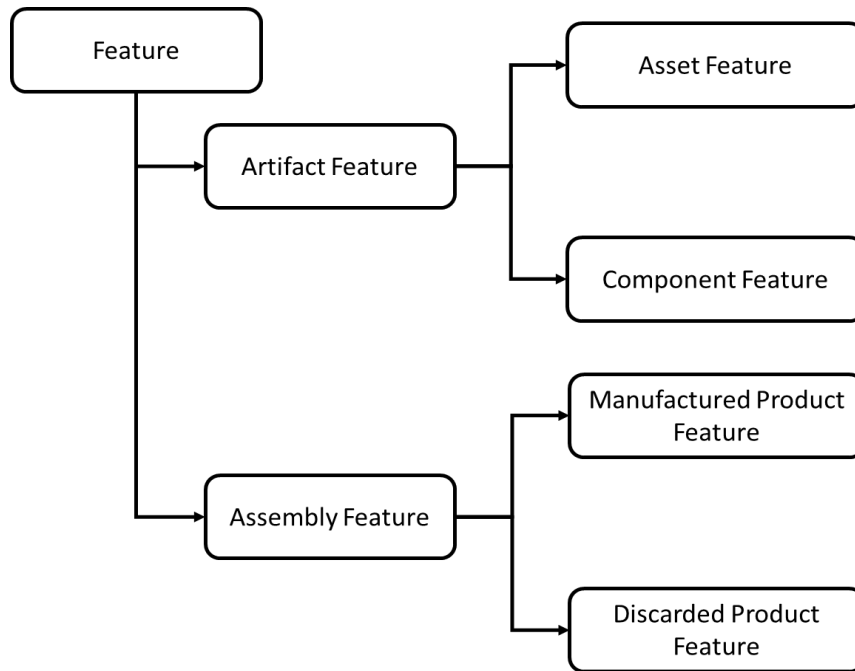


Figure 4-33: Feature class of the ORMA+ ontology

The basic concept behind this class is that an *Artifact* or an *Assembly* may have some characteristics worth of notice, which may reduce the *Quality* of an *Assembly* – it is the case of the *Asset Feature* and the *Component Feature* – or, otherwise, better describe the characteristics of the *Product* – it is the case of the *Assembly Feature*.

In the *Artifact Feature* case a *Component* may have some characteristics, thus *Component Feature*, which may reduce its quality or which may make the *Component* not conform at all to specification, making a *Product* not good enough to be sold to the costumer. For what concern the *Asset Feature*, the working condition of the *Asset* may not be good enough to produce a sufficiently good product, and so the *Product* cannot be sold to the costumer.

In the case of *Assembly Feature* the features that the *Product* may have are correlated to the characteristics that the *Product* obtained may have.

Summarizing, accordingly to what has been said, the following definitions can be given:

- Artifact Feature can be defined as: “The Feature that an Artifact has”;
- Assembly Feature “The Feature that an Assembly has”;
- Asset Feature “The Feature that an Asset has”;
- Component Feature “The Feature that a Component has”;
- Manufactured Product Feature “The Feature that a Manufactured Product has”;
- Discarded Product Feature “The Feature that a Discarded Product has”.

4.3.3.2.2 Specification class

To better understand how the ontology assign the *Quality* of a *Product* it is necessary to deal with the *Specification* class. The *Specification* class is a sub class of the *Directive Information Content Entity* and, as shown in the image 4-34, it has two subclasses.

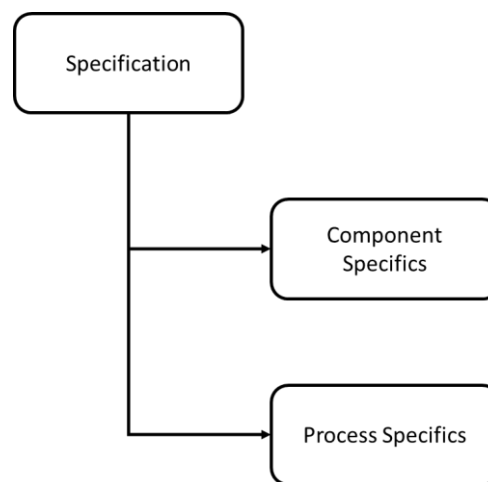


Figure 4-34: Specification class of the ORMA+ ontology

This class was already introduced when the *Feature* class was analyzed, but it is necessary to look more closely at the relationship between the two classes. In particular, it is necessary to emphasize the difference between the two. A specification is a set of requirements that an entity must meet, while a feature has been defined in the IOF [72] as a quality or a part of the physical entity. According to this, a physical entity can have one or more features, but some of them need to verify one or more specifics, otherwise the product needs to be discarded since not compliant. From what has just been said, the definition of specification can be given:

- Specification according to the IOF Ontology can be defined as: “A directive information content entity that prescribes some part or feature or some outcome of a planned process”.

Now it is necessary to go into more detail inside the *Specification* class, namely the *Component Specifics* subclass and the *Process Specifics* subclass, which are represented in the figure 4-35, are going to be examined.

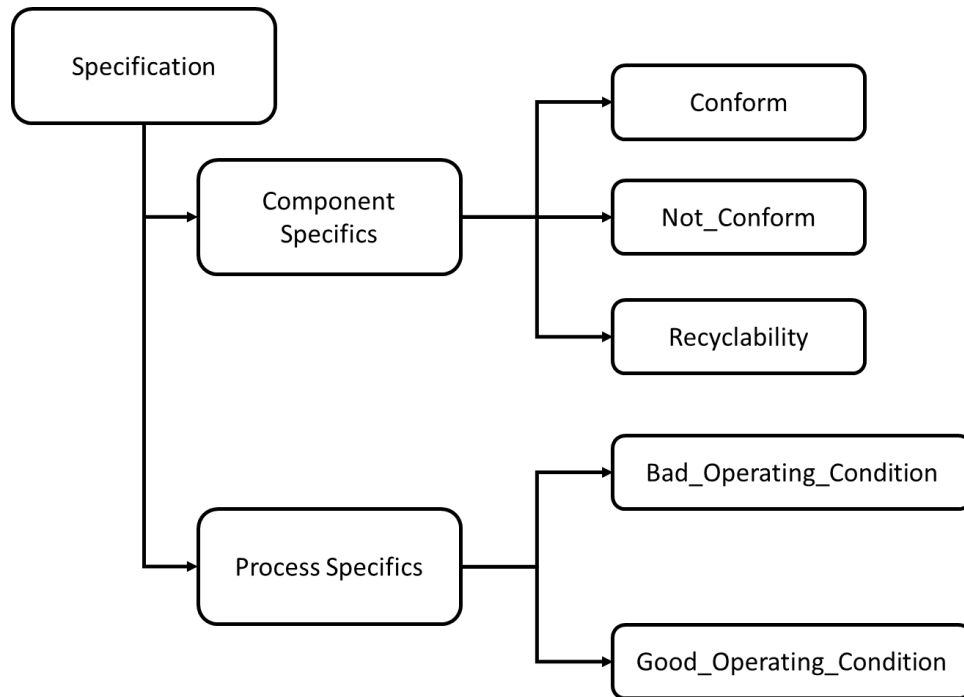


Figure 4-35: Component Specifics and Process Specifics class of the ORMA+ ontology

The *Component Specifics* subclass is one of the aspects which were added in the ontology by the author of this thesis work to contribute to the definition of *Product Quality*. In particular, this subclass refers to the specific characteristics that a Component must follow to manufacture a product with sufficient quality to be sold. In particular, this subclass can be subdivided into 3 elements: *Conform*, *Non-Conform* and *Recyclable*. The *Conform* subclass is used to verify that the component's feature is aligned with the component specification, otherwise, the component is *Non-Conform*. However, a *Non-Conform* component can be made compliant as long as it is not completely damaged, therefore the following subclasses have been defined: *Repairable* and *Nor Repairable*. For what concern the *Recyclable* subclass instead, this class was added in the ontology inspired by [63] since, as was said before, a *Defective Cellphone* is a *Product* which quality is insufficient to be sold, but not necessarily all the components of the cellphone have an insufficient quality to be reused. For this reason, this class represents the specifics to verify for the re-utilization of a component. All these specifics are presented in the figure 4-36.

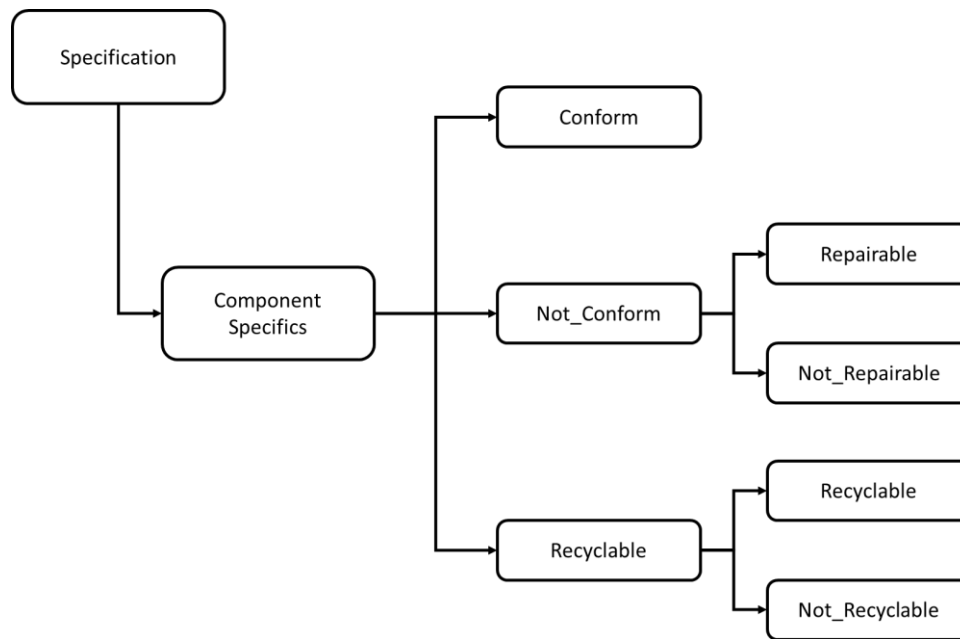


Figure 4-36: Component_Specifics class of the ORMA+ ontology

Accordingly, the following definitions can be assigned:

- Component Specifics can be defined as: “A directive information content entity that prescribes some component”;
- Conform can be defined as: “A component which fulfills all the specifics it needs to check”;
- Not Conform can be defined as: “A component which do not fulfills all the specifics it needs to check”;
- Repairable can be defined as: “A component which do not fulfills all the specifics it needs to check but can be repaired in order to meet the specifics”;
- Not Repairable can be defined as a “A component which do not fulfills all the specifics it needs to check and can not be repaired to meet the specifics”;
- Recyclability can be defined as “A component which have the possibility to be reused for other products”;
- Recyclable can be defined as: “A component which can be reused for the production of other products”;
- Not Recyclable can be defined as: “A component which can not be reused for the production of other products”.

The *Process Specifics* is the second element that characterize the *Specification* class, which has been added in the ontology to contribute to defining *Product Quality*. In particular, this subclass refers to the specific characteristics that the assembly line process must follow in order to produce a product of sufficient quality. Specifically, this subclass can be subdivided into the following sub classes:

Good_Operating_Condition and *Bad_Operating_Condition*. In particular, these factors depend on the status of the asset and the operations which are executed by the asset. For example, for a smooth assembly line process, all assets must have a good status, meaning that the machines are working as usual. Accordingly it is possible to give a definition to these elements:

- *Good_Operating_Condition* can be defined as: “An assembly line process which fulfills all the specifics it needs to check”;
- *Bad_Operating_Condition* can be defined as: “An assembly line process which do not fulfills all the specifics it needs to check”.

4.3.3.3. Object Properties of the Quality class

The Object properties which are correlated to the Asset class are the following ones in figures from 4-37 to 4-39.

- *has_feature* (inverse of *is_feature_of*)

Domain: Artifact

Ranges: Feature

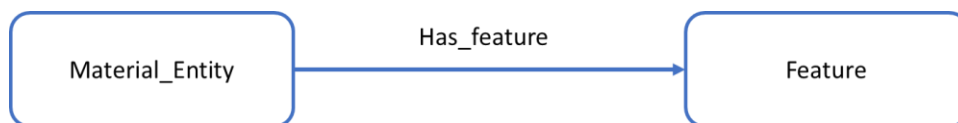


Figure 4-37: *Has_feature* property of the ORMA+ ontology

- *has_quality* (inverse of *is_quality_of*)

Domain: Material Entity

Ranges: Quality

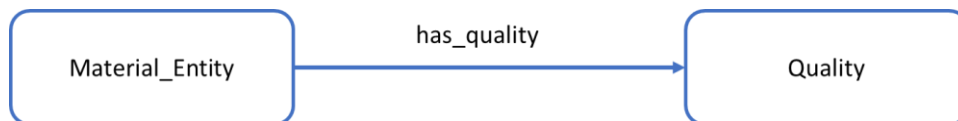


Figure 4-38: *Has_quality* property of the ORMA+ ontology

- *has_specifics* (inverse of *is_specific_of*)

Domain: Material Entity

Ranges: Specifics

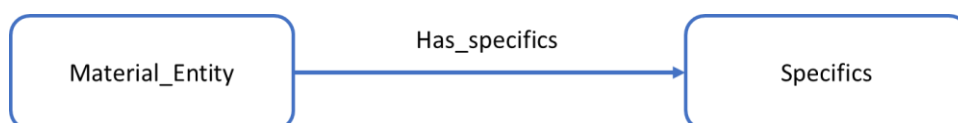


Figure 4-39: *Has_specifics* property of the ORMA+ ontology

4.3.3.4. Data Properties of the Quality class

- Malfunction

Domain: Feature

Range: boolean

This Data Property is used in order to determine if the *Asset* is working the way it should or if it has a malfunction. This element is correlated to the *Quality* of the *Product* since the functioning of the *Asset* influences the *Product Quality*, accordingly to what has been said before.

4.3.4. Process

4.3.4.1. Process Class

The *Process* class is the last pillar of the Ontology and it is an IOF class [72] which can be defined in the following way:

- Process can be defined as: “p is a process, p is an occurrent that has some temporal proper part and for some time t, p has some material entity as participant” [72].

The *Process* class is the last pillar of the ORMA+ ontology and it is shown in figure 4-40.

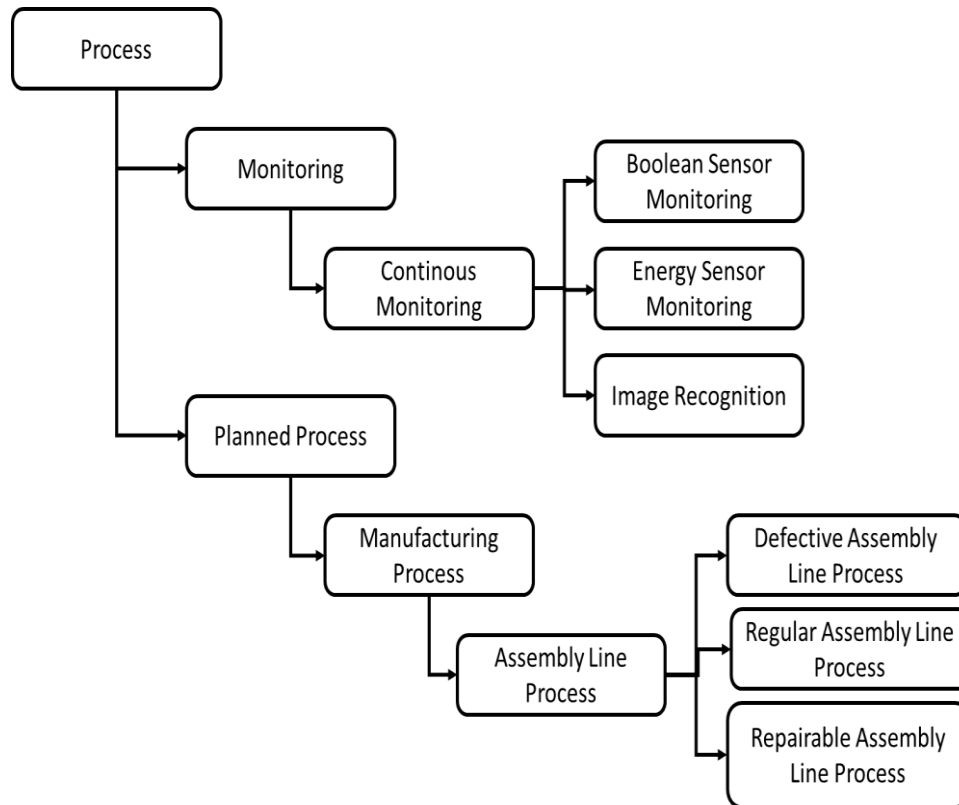


Figure 4-40: Process class of the ORMA+ ontology

The *Monitoring* class was a class inherited from the ORMA ontology [23], which was added to the ORMA+ ontology by the author of the thesis to introduce one element which could be utilized to implement the Zero Defect Manufacturing strategy of Detection. As a matter of fact the logic behind this class is to introduce the necessary elements to interpret the signals coming from the sensors present in the line. Accordingly to what has been said the classes of *Monitoring* can be defined as:

- Monitoring can be defined as: “BFO:Process to monitor an AO:Artifact by measuring a specific phenomenon” [23];
- Continuous Monitoring can be defined as: “A monitoring process which is continuous in time” [23];
- Boolean Sensor Monitoring can be defined as: “A continuous monitoring process which monitors boolean values” [23];
- Energy Sensor Monitoring can be defined as: “A continuous monitoring process which monitors energy values” [23];
- Image Recognition can be defined as: “A continuous monitoring process which monitors images of the components”.

It is necessary to say a few words about *Image Recognition* class. This element is one of the key elements in implementing the ontology of the ZDM detection strategy. In fact,

in order to carry out the detection strategy on the inspected line, the piece must arrive at the camera station of the line, where the piece is checked by the camera. In the specific case of the FML, each PCB is characterized by an image: a blank image, a Square and a Triangle, which are shown in the figure 4-41. Depending on what you observe, you can determine if a component meets specifications, so you can determine the *State* of the component.

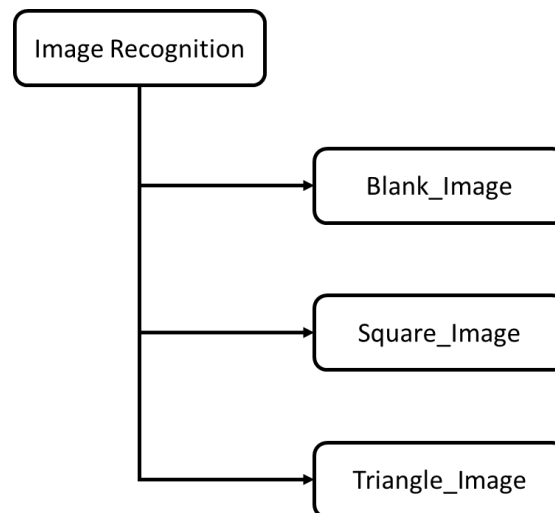


Figure 4-41: Image Recognition class of the ORMA+ ontology

Concerning the *Planned Process* class this is an IOF class which can be defined in the following way:

- Planned Process can be defined as: “A processual entity that realizes a plan which is the concretization of a plan specification” [72].

According to the definition of the *Planned Process* class it is possible to define the subclass *Manufacturing Process*, which indicates the process it is necessary to go through in order to obtain a *Product*. As a matter of fact the *Manufacturing Process* class has the following subclass: *Assembly Line Process*. This class is characterized by the following subclasses: *Defective Assembly Line Process*, *Regular Assembly Line Process* and *Repairable Assembly Line Process*. These classes define the processes that must be passed through to obtain a product.

- Manufacturing Process can be defined as: “A planned process that realizes a product which is the concretization of a plan specification”;
- Assembly Line Process can be defined as: “A manufacturing process which realizes an assembled product”;
- Defective Assembly Line Process can be defined as: “An assembly line process which realizes a defective product”;

- Regular Assembly Line Process can be defined as: “An assembly line process which realizes a regular product”;
- Repairable Assembly Line Process can be defined as: “An assembly line process which realizes a repairable product”.

4.3.4.2. Elements correlated with the Process class

4.3.4.2.1 Process Boundary

This is a class of the IOF ontology which can be defined in the following way:

- Process Boundary according to the BFO ontology Reference 2 [084-001] can be defined as: “p is a process boundary =Def. p is a temporal part of a process & p has no proper temporal parts” [72].

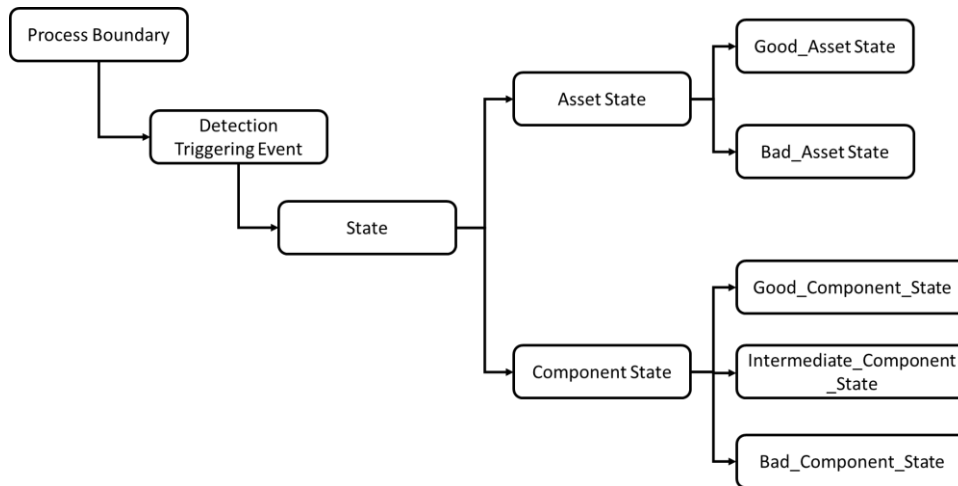


Figure 4-42: Process Boundary class of the ORMA+ ontology

As it is possible to see from the figure 4-42 there are several subclasses which are connected to this one, the first one is the *Detection Triggering Event* subclass. The *Detection Triggering Event* class was introduced in the ontology, by the author of this thesis work, to implement the ZDM detection strategy. Nevertheless this element is used in order to assign a *State* to the components of the *Product* each time the camera inspection process present in the line is carried out, and the *State* of the *Asset* in function of the data collected from the line.

Concerning the *Component State* it has been stated before in the *Image Recognition Class* of the *Quality* pillar that in the FML there are 3 type of images that can be found: Blank Image, Square Image and Triangle Image. In function of which image is recognized it is possible to assign a *State* to the *Component*.

For what concern the *Asset State*, in the FML the drilling station – the asset on which it is focused the demonstration –O is characterized by the presence of a shaker, which

simulate if the process is going smoothly or if the process is not going according to plan.

In the end, in function of both the *Component State* and the *Asset State* it is possible to attribute a *Quality* to the *Product*. In fact, if an *Asset* has a sub-optimal state, then the operations conducted by the *Asset* are going to be not optimal and the *Quality* of the *Product* may be influenced and the same goes for the *Component State*; besides, if a *Product* is made of sub-optimal component, the *Quality* of the product will be sub-optimal too. Accordingly to what has been said:

- Detection Triggering Event can be defined as: “a process boundary where some detection process is carried out on an Assembly”;
- State can be defined as: “the condition that an artifact has”;
- Component State can be defined as: “The condition that a Component has”;
- Asset State can be defined as: “The condition that an Asset has”.

4.3.4.2.2 Sensor class

The *Sensor* class is a sub class of the *Transducer* class as showed in figure 4-43. This classes are taken from the ORMA ontology [23] and they are the classes dealing with the sensors that are present in the line, which can be defined in the following way:

- Transducer can be defined as: “An Artifact that is designed to convert one form of energy to another” [23];
- Sensor can be defined as: “A Transducer that is designed to detect events or changes in its environment, and then provide a corresponding output” [23].

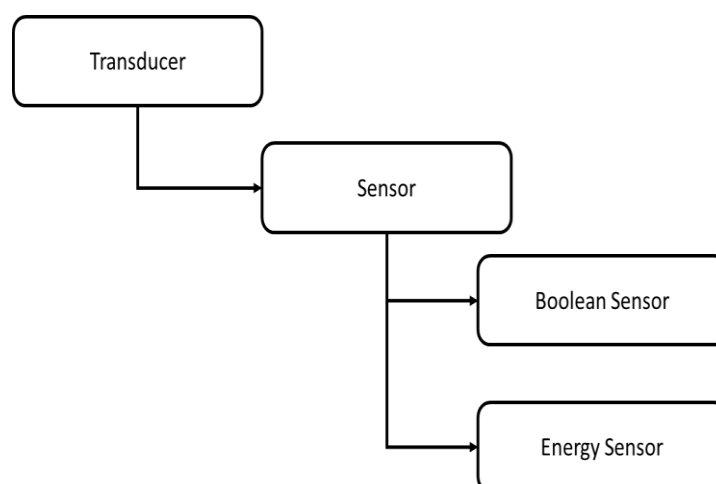


Figure 4-43: Transducer class of the ORMA+ ontology

As shown in the image 4-43 there are two sub classes of *Sensor* class: the *Energy Sensor* class and the *Boolean Sensor* class. These classes have been introduced since only these

two categories of sensor are considered for this thesis work, as a matter of fact the *Energy Sensor* class represent those sensors which monitors the energy of the *Asset* while the *Boolean Sensor* class represent those sensor which monitors the passage of the *Product* in the stations of the FML. Accordingly the following definitions can be given:

- Boolean Sensor can be defined as a: “Sensor designed to detect changes in the position of the Product, to which a Boolean value is associated” [23];
- Energy Sensor can be designed as: “Sensor designed to detect changes in the energy value of the Asset” [23].

4.3.4.3. Object Properties of the Process class

The Object properties which are correlated to the Process class are the following ones in figure 4-44 and 4-50.

- has_assembly_process (inverse of is_assembly_process_of)

Domain: Assembly

Ranges: Assembly Line Process

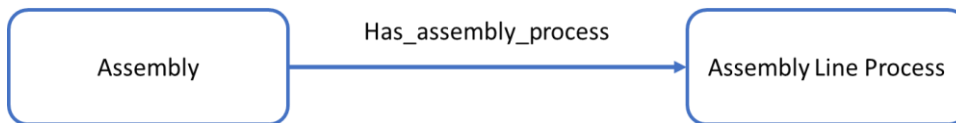


Figure 4-44: Has_assembly_process property of the ORMA+ ontology

- has_monitoring

Domain: Sensor

Ranges: Monitoring

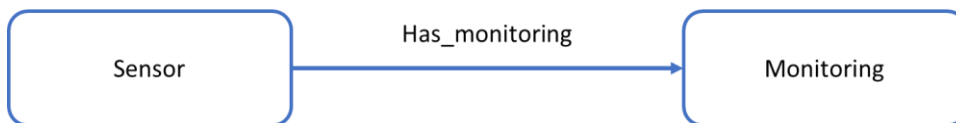


Figure 4-45: Has_monitoring property of the ORMA+ ontology

- has_process_specifics (inverse of is_a_process_specific_of)

Domain: Manufacturing_Process

Ranges: Specification

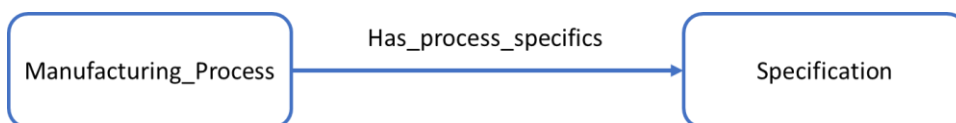


Figure 4-46: Has_process_specifics property of the ORMA+ ontology

- has_state (inverse of is_state_of)

Domain: Artifact

Range: State

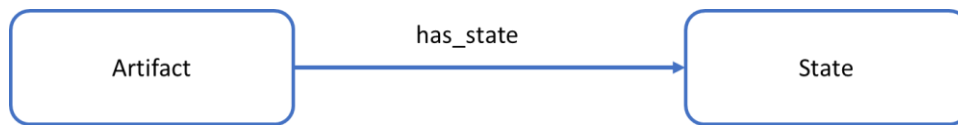


Figure 4-47: Has_state property of the ORMA+ ontology

- determine

Domain: Image_Recognition

Range: Component_Feature

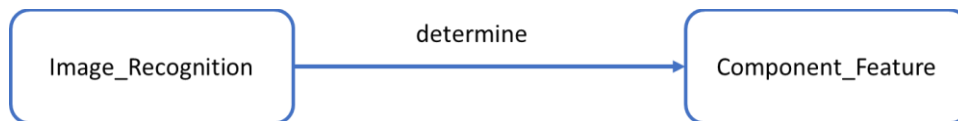


Figure 4-48: determine property of the ORMA+ ontology

- results_in

Domain: Asset_State

Range: Quality

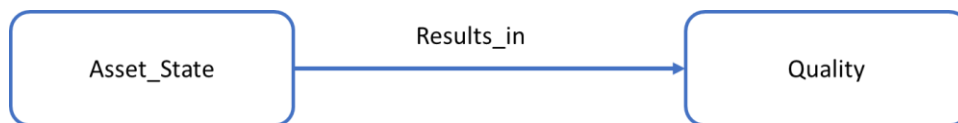


Figure 4-49: Results_in property of the ORMA+ ontology

- brings_to

Domain: Component_State

Range: Quality

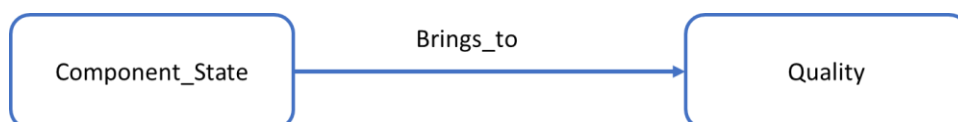


Figure 4-50: Brings_to property of the ORMA+ ontology

5 Application of the ontology

In this section, the last phase of the research methodology is faced, that is the demonstration and evaluation. According to METHONTOLOGY [55], it includes the implementation of ORMA+ and its evaluation, based on the validation of the logic of the ontology and the answer to the CQs.

5.1. Validation of the ontology logic

The first step for the application of the ontology is to validate the logic of the ORMA+ ontology in order to see if there are any discrepancies in the structure of the ontology. In order to validate the logic of the ontology, it is required the application in the FML at laboratory-scale, allowing the deployment of an integrated solution in a controlled environment. Therefore, after providing insights on the FML, it is necessary to describe the implementation and validation phase where the CQs are answered.

5.1.1. Case study description

The flexible manufacturing line is composed of seven stations and two branches which are utilized for transportation only. The objective of the line is to obtain a *Dummy Cellphone* composed of four parts: one PCB (Programmable Computer Board), one front cover, two fuses and one back cover. Each of the seven station that compose the FML accomplishes a specific operation on the semi-finished product.

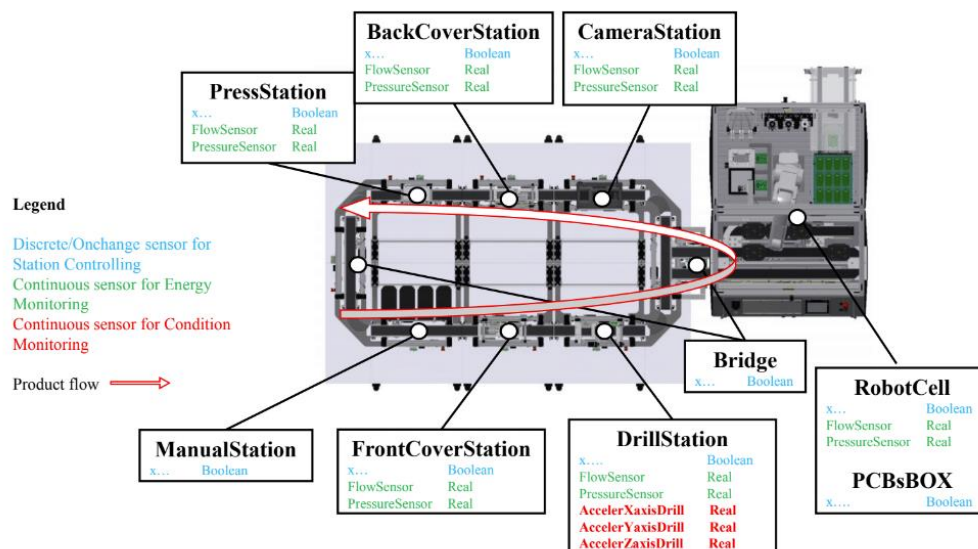


Figure 5-1: Representation of the Flexible Manufacturing Line in the Industry 4.0 Lab [23]

As a matter of fact in order to obtain the fuzzy mobile phone it is necessary for the product to follow a complete production cycle composed by the following operations which are presented in figure 5-1:

- 1) When a product needs to be realized, an empty carrier starts upstream of the front cover station. When arriving in the front cover unit, the station verifies if some conditions are satisfied, like correct identification number of the carrier (through RFID, Radio Frequency IDentification) and if front cover release is needed for that product. After completing the operation, the carrier leaves the front cover station towards the drilling station.
- 2) At the drilling station the process is similar to the front cover one, but the drilling holes are realized. In particular in this station it is also possible to simulate the malfunctioning of the machine by activating the shaker present in the station. After completing the drilling operation, the carrier leaves the front cover station towards the Robot Cell.
- 3) Once the piece is at the Robot Cell the manufacturing process performed is the insertion of the PCB and the insertion of the fuses. So in this station the Robot checks the product and positions it in the assembly zone in which the additional elements are inserted in the product. After these components are inserted the piece is positioned back in the branch and it goes to the Camera station.
- 4) At the Camera Station the piece is controlled with a camera in order to check the condition of the product. In this phase are two the elements which are kept under consideration to determine the condition of the product: if the shaker was activated in the drilling station and the PCB. As a matter of fact not all the PCB are the same since on the PCB there can be inserted some simple images like a square or a triangle that the camera station is able to detect in order to determine the condition of the PCB. So after the conditions of the products are checked the piece is moved to the next station, the back cover station.
- 5) The back cover station behaves like the front cover station, the only difference is the manufacturing operation since in this station the back cover station is inserted. After the last component is inserted the product moves to the Press station.
- 6) In the press station the product is pressed in order to obtain the final product of the line. After this last step is performed the fuzzy mobile phone goes to the next station: the manual station.

7) This is the last station of the line and in this station the piece is taken and in function of the condition detected at the camera station the piece may be kept or it may be discarded.

Considering how FML works, each station has a total of two main functions:

- manufacturing of goods;
- transportation of goods.

The first is optional and depends on the particular product cycle selected by the MES, but the second is mandatory on almost all stations except robot cell where the product passage is not guaranteed. As a matter of fact the MES (Manufacturing Execution System) is used to select the product to be launched in production and to control the system during the operational phase with updates on product completion state.

5.2. Validation of the ORMA+ ontology

In order to validate the ontology created it is necessary to identify all the possible scenarios that the ORMA+ considers. These cases have been selected in order to show the potential of ontology and how it can fill the gaps that have been mentioned in the chapter dedicated to the Literature Review. These scenarios can be divided into 3 categories according to the quality of the final product, namely *Dummy Cellphone*, *Repairable Cellphone* and *Defective Cellphone*. The *Dummy Cellphone* represents the case of an *Assembly* with *Good Quality* or *Regular_Assembly_Quality*, so an *Assembly* which has *Good_State_Components* and is processed by an *Asset* with *Good_Operating_Condition*. The *Repairable Cellphone* represent the case of an *Assembly* with *Intermediate Quality* or *Repairable_Assembly_Quality*, so an *Assembly* which may present one of the following characteristics:

- 1) an *Assembly* which has *Good_State_Components* and is processed by an *Asset* with *Bad_Operating_Condition*;
- 2) an *Assembly* which has at least one *Intermediate_State_Component* and is processed by an *Asset* with *Good_Operating_Condition*;
- 3) an *Assembly* which has at least one *Intermediate_State_Component* and is processed by an *Asset* with *Bad_Operating_Condition*;
- 4) an *Assembly* which has at least one *Bad_State_Component* and is processed by an *Asset* with *Good_Operating_Condition*.

The *Defective Cellphone* represents the case of an *Assembly* with *Bad Quality* or *Defective_Assembly_Quality*, so an *Assembly* which has at least one *Bad_State_Component* and is processed by an *Asset* with *Bad_Operating_Condition*.

Accordingly to what has been said, the determining factors for establishing the quality of the products are the operating condition of the asset under examination, therefore the state of the drilling machine of the FML, and the state of the components of the product, especially the PCB, a component of the product manufactured by the FML. This choice was made in order to reconcile the vision of the product with that of the process / machine accordingly to the reasons expressed in chapter 2. Therefore, the quality of the final product is obtained from the combination of these two factors, thus defining the 6 possible scenarios which have been described before and presented in figure 5-2:

	Good State Component	Intermediate State Component	Bad State Component
Bad Operating Condition	Repairable Cellphone → Repairable Assembly Quality	Repairable Cellphone → Repairable Assembly Quality	Defective Cellphone → Defective Assembly Quality
Good Operating Condition	Dummy Cellphone → Regular Assembly Quality	Repairable Cellphone → Repairable Assembly Quality	Repairable Cellphone → Repairable Assembly Quality

Figure 5-2: Representation of the cases deal with the ORMA+ ontology

In these scenarios, a combination of Detection and Repair strategy are used. In the *Dummy Cellphone* case, for example, the strategy adopted is the one of detection since there is no need to repair the Product, while in the *Repairable Cellphone* case, as the name suggests, one or more components are not conform but can be repaired in order to make them conform and so a combination of Detection and Repair strategy are used. In the *Defective Cellphone* case instead the quality of the product is insufficient and the cost to repair it is too big to be convenient and so the Product is discarded. In this last case a disassembly operation is performed in order to retrieve those components whose quality is sufficiently good to be used for other products. This disassembly strategy was inspired by the following papers [64]–[67] but it was implemented in a simplified way since this strategy is not applicable in the line at the moment and so this has been just theorized.

Now these 3 cases, *Regular_Assembly_Quality*, *Repairable_Assembly_Quality* and *Defective_Assembly_Quality*, will be presented more in detail in the following section.

5.2.1. Regular_Assembly_Quality case

The first case to be analyzed is the first scenario, so the case in which there are *Good_State_Components* and the Asset is characterized by a *Good_Operating_Condition*.

Accordingly to the case study description in chapter 5.2.1 the process to create a *Dummy Cellphone* will follow this cycle:

- 1) The starting point of the process is the arrival of the carrier to the *FrontCover_Station*. In this station the only operation to be performed is the mounting of the front cover on the carrier. For this reason, when the carrier arrives, a new individual is created in the *Assembly* class, that is going to be called *Product*. The Individual *Product* will have a new component, an individual named *FrontCover* and its position is going to be updated, as a matter of fact the position of *Product* is now *FrontCover_Station*. So by looking at the relations below and the figures 5-3 and 5-4 it is possible to see all the elements which has been described until now:

Product has_component FrontCover

Product has_position FrontCover_Station

FrontCover has_feature FrontCover_Feature

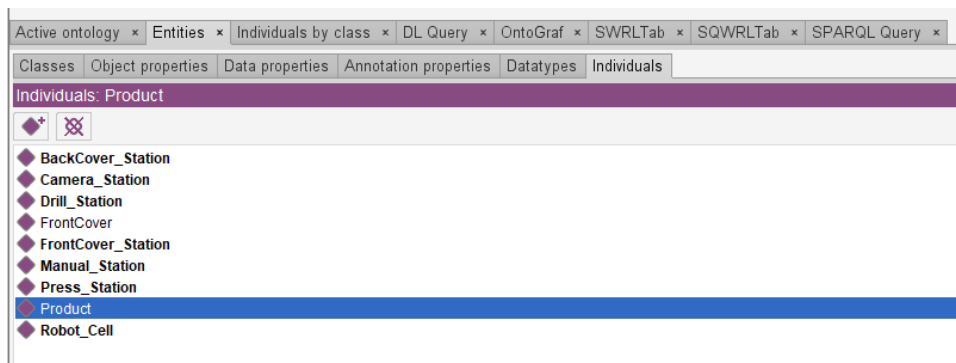


Figure 5-3: Representation of the individual Product

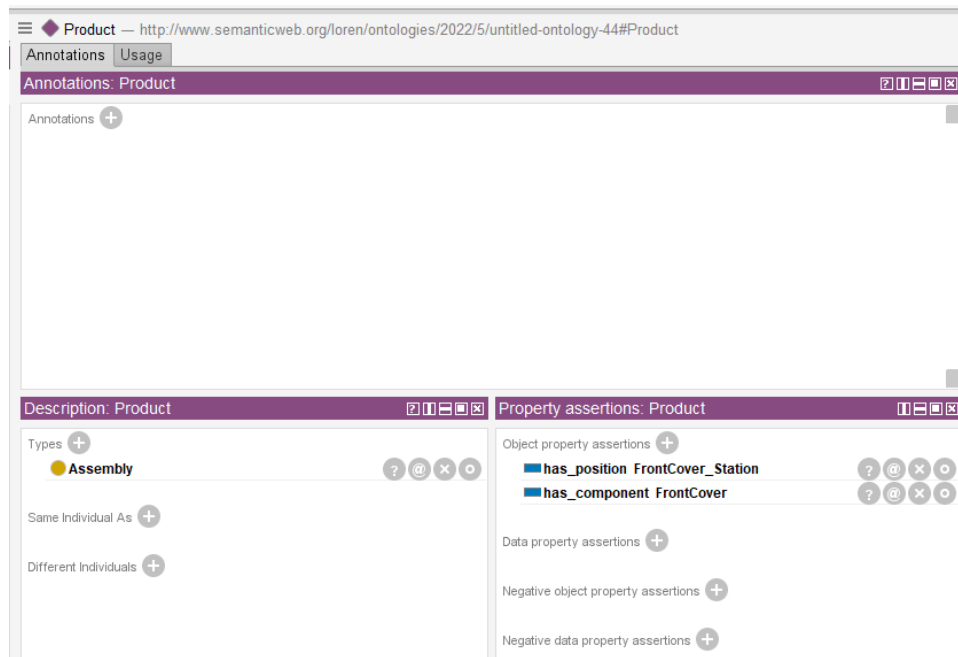


Figure 5-4: Property assertion of product

- 2) The next step of the manufacturing process is the arrival of the product at the Drilling Station. In this station a drilling operation will be performed on the front cover, so at this station will be performed not an adding operation but a transforming operation. Accordingly to the holes drilled by the drilling machine, the state of the front cover is going to change, infact if the holes are conform to the specifics, then the front cover is being processed correctly otherwise the front cover needs to be repaired. Therefore the drilling machines determines the condition of the front cover. This kind of information is obtainable from an element present on the drilling machine, the shacker. This element simulates the malfunctioning of the drilling machine by making the machine shake. To determine if the shacker is active or not it is necessary to process the data coming from the servers in the line (the FML of the Laboratory). Since the case of a *Regular_Assembly_Quality* is being considered then the shacker is not active. So by looking at the figures from 5-5 to 5-10 the following relations can be determined:

Drilling_Machine has_feature Drilling_Machine_Feature

Product has_position Drill_Station

Product has_component FrontCover

Drilling_Machine_Feature → Malfunction value false

Drilling_Machine_Feature settles FrontCover_Feature

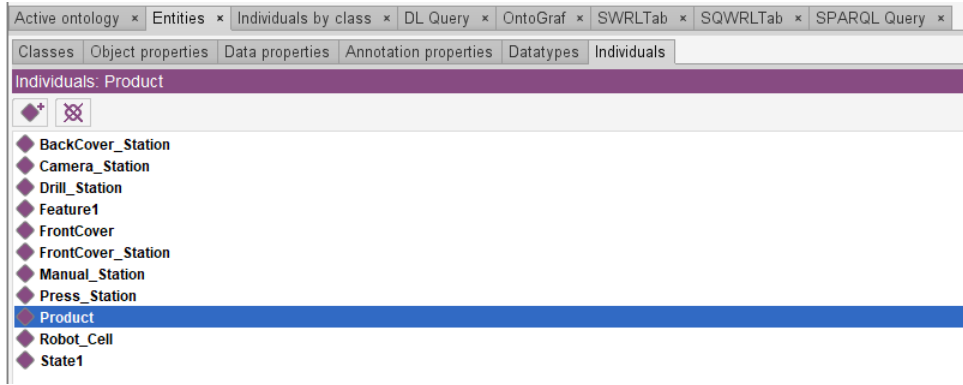


Figure 5-5: Product at the Drill_Station

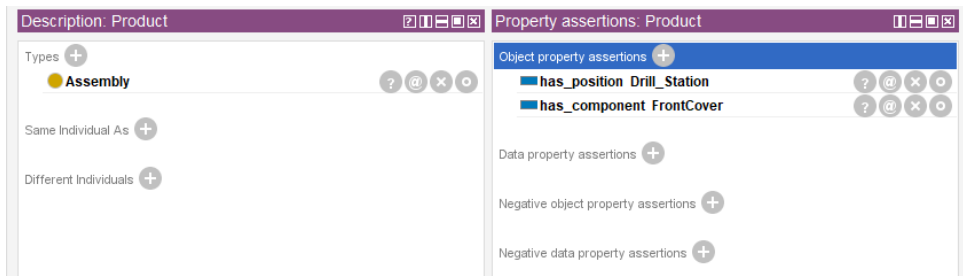


Figure 5-6: Property Assertion of Product

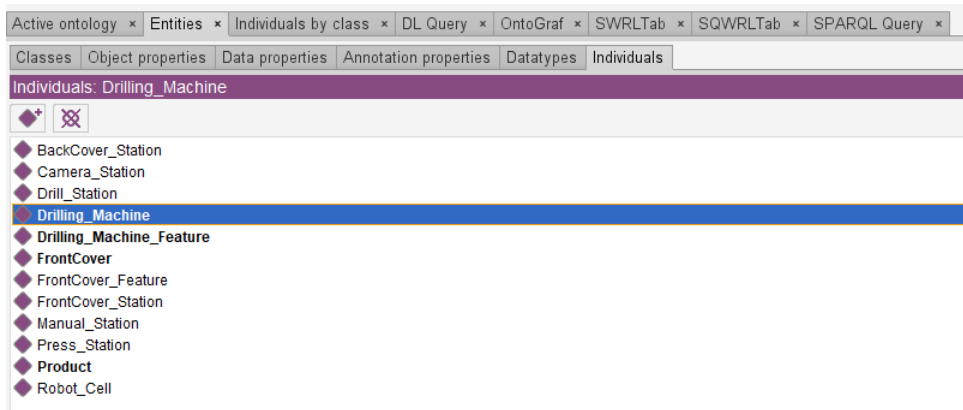


Figure 5-7: Individual drilling machine

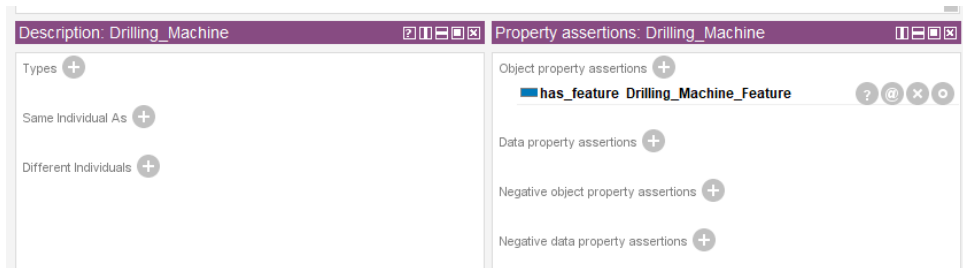


Figure 5-8: Property assertion of drilling machine

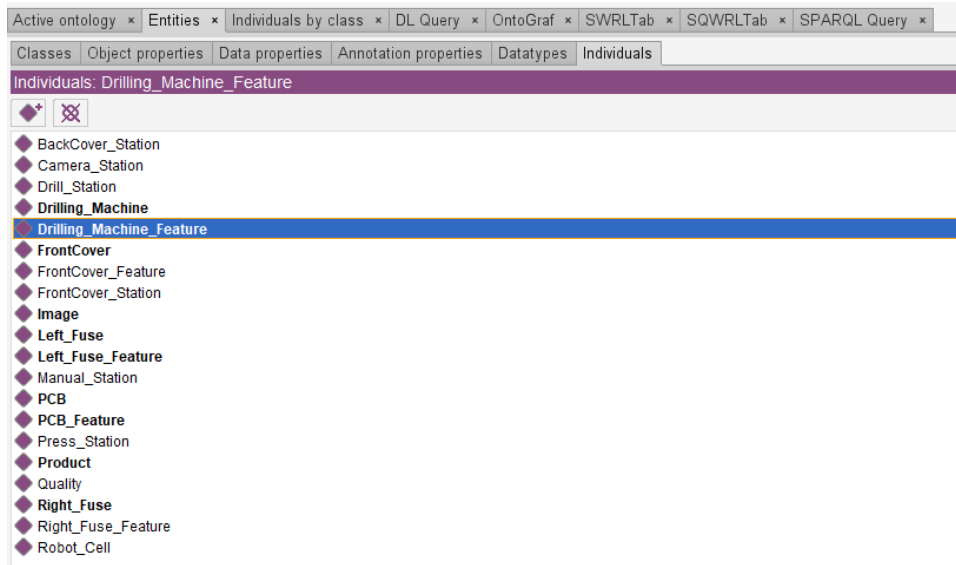


Figure 5-9: Representation of the individual Drilling Machine Feature

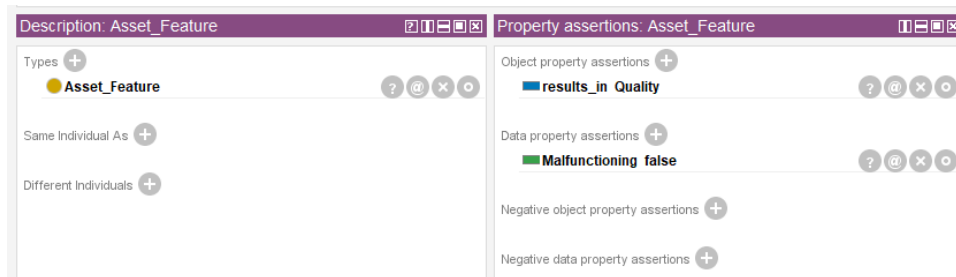


Figure 5-10: Representation of the property assertions of Drilling Machine Feature

- 3) Once the drilling operations have been performed the *Product* individual goes to the Robot Cell Station. In this station an adding operation will be performed since the fuses and the PCB will be inserted on the front cover, figures 5-11 and 5-12. As expressed before on the PCB an image is present to determine if the PCB is conform to specifics or not, but this information cannot be obtained now since in the Robot Cell there is no element to visually inspect the *Assembly*. For the fuses, since in the scope of this study no operations have to be executed on them, it is considered by default that these components are *Conform* as supplied, indeed the ontology is not able to determine any other possible status for them. So the following relations can be determined:

Product has_position Robot_Cell

Product has_component Left_Fuse

Product has_component Right_Fuse

Product has_component PCB

PCB has_feature PCB_Feature

Left_Fuse has_feature Left_Fuse_Feature
Left_Fuse_Feature → Conform
Right_Fuse has_feature Right_Fuse_Feature
Right_Fuse_Feature → Conform

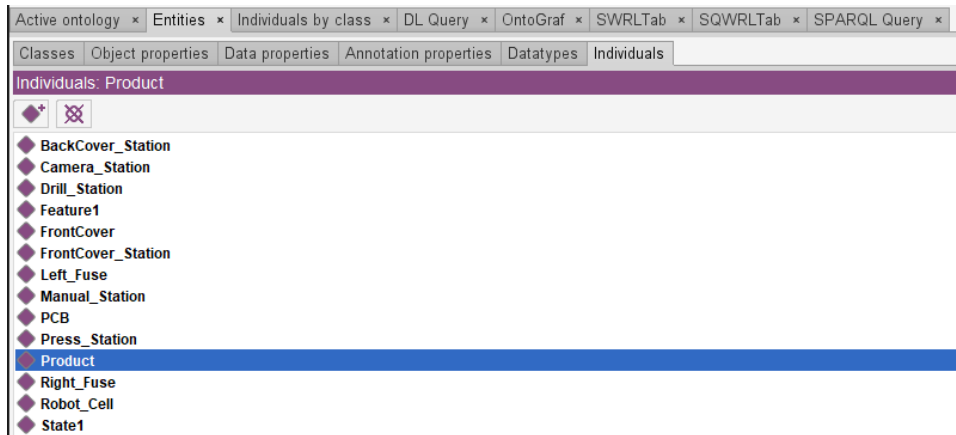


Figure 5-11: Product individual at the Robot Cell

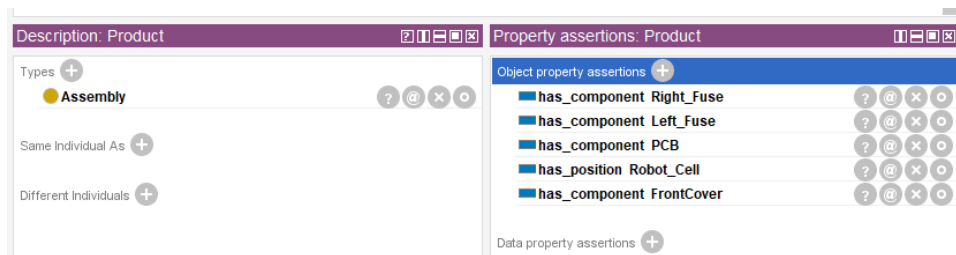


Figure 5-12: Property assertion of Product

- 4) The next step of the production process is the arrival of the Product at the Camera Station. At this station the Assembly is visually inspected by a camera to determine if some irregularities are present on the product. In particular in this station it is possible to check the condition of the PCB by looking at the image present on it. In fact the PCB may have a square image, a triangle image or no image at all, for this reason the following images have been established: *Blank_Image*, *Square_Image* and *Triangle_Image*. These images represent respectively the condition of the PCB as *Conform*, *Not Conform but Repairable* and *Not Conform*. Since in this case the *Regular_Assembly_Case* has been considered the PCB is *Conform* to specifics and so a *Blank_Image* is present on the PCB. So it is possible to determine the following relationships which are shown in figure 5-13 and 5-14:

PCB has_image Image

Image determine PCB_Feature

Image → Blank_Image

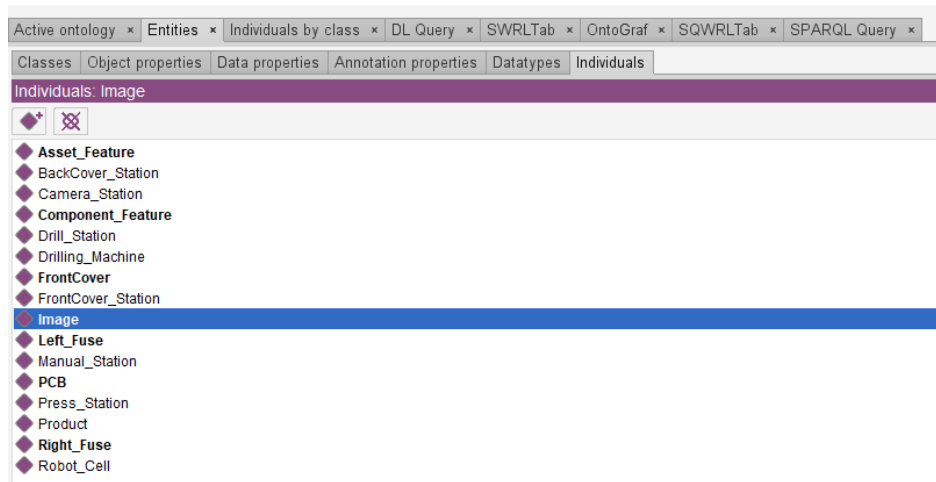


Figure 5-13: Representation of the Image individual

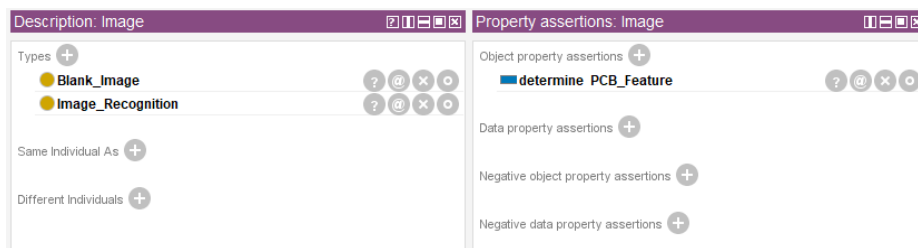


Figure 5-14: Representation of the Property Assertion of Image

In function of these relations it is possible to determine the quality of the *Product* individual. As a matter of fact it has been expressed before as the *Quality* of the *Assembly* is determined by the combination of two factors: the working condition of the *Asset* and the state of the components, in particular the *PCB*. For this reason now that it is possible to establish the condition of the *PCB* the following relations can be determined, which are presented in the figures from 5-15 to 5-18:

PCB_Feature brings_to Quality

Drilling_Machine_Feature results_in Quality

Product has_quality Quality

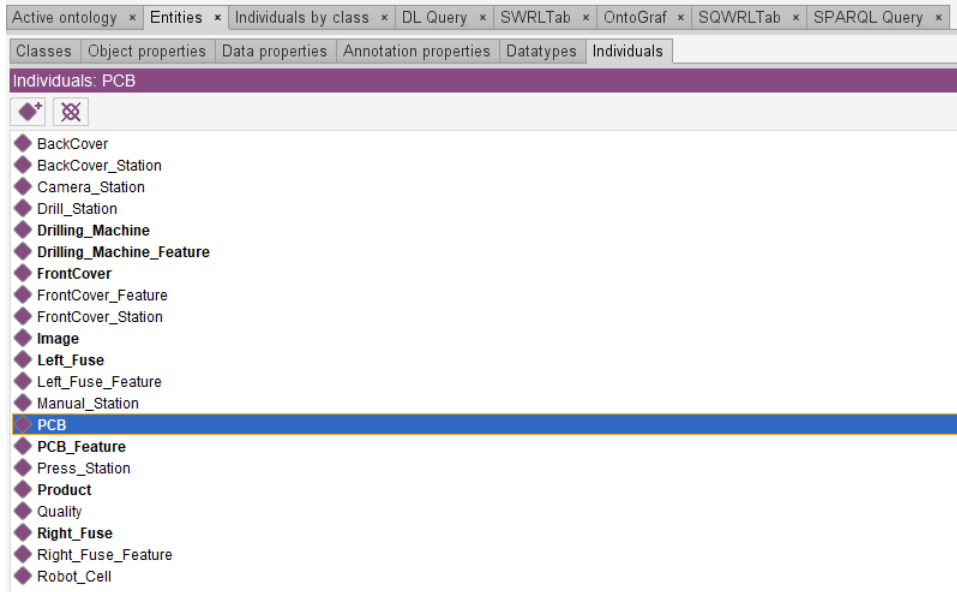


Figure 5-15: Representation of the individual PCB

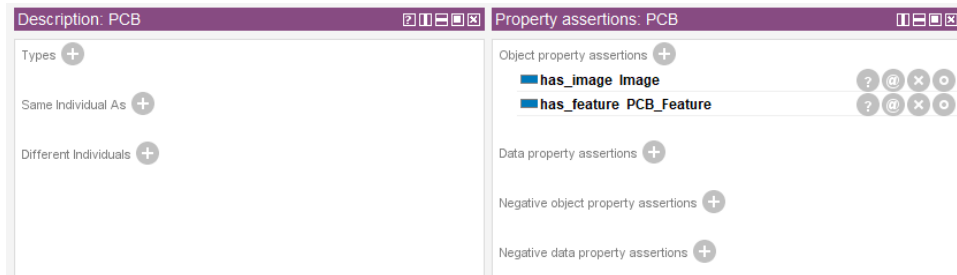


Figure 5-16: Representation of the Property Assertion of PCB

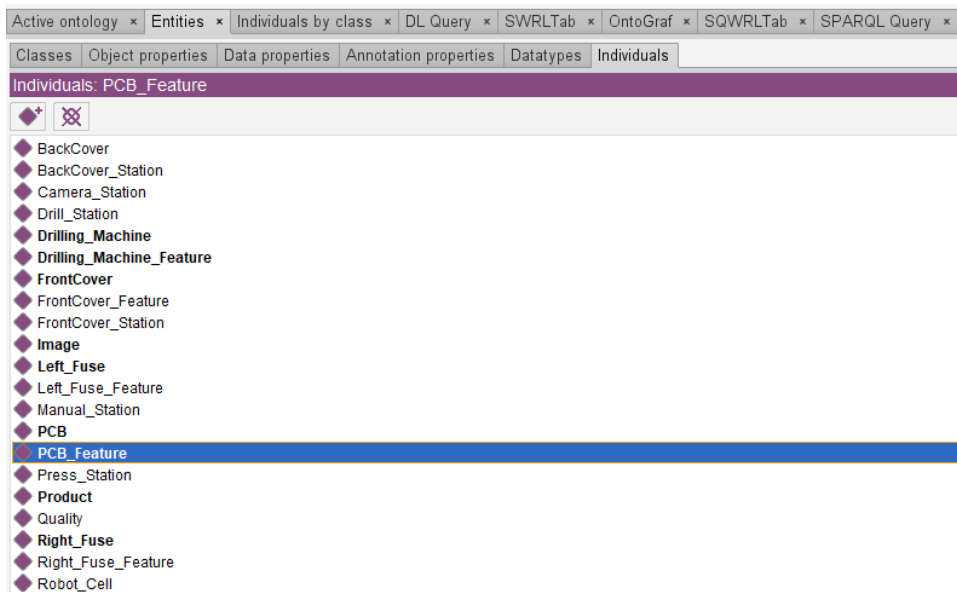


Figure 5-17: Representation of the individual PCB_Feature

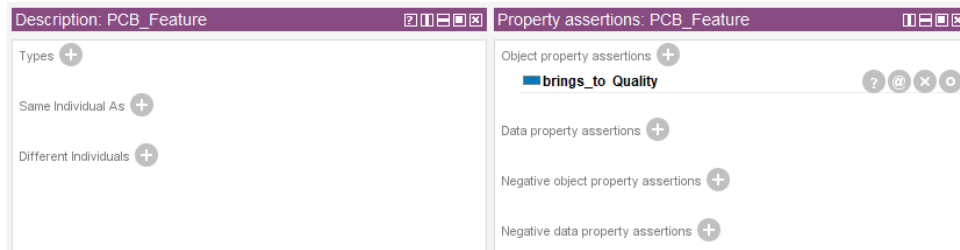


Figure 5-18: Representation of the Property assertion of PCB_Feature

5) After the visual inspection of *Product* is concluded, the *Assembly* goes to the *BackCover_Station*, where an adding operation is executed: the *BackCover* is mounted on the *Product*, figures 5-19 and 5-20. Therefore the following relations can be determined:

Product has_position BackCover_Station

Product has_component BackCover

BackCover has_feature BackCover_Feature

BackCover_Feature → Conform

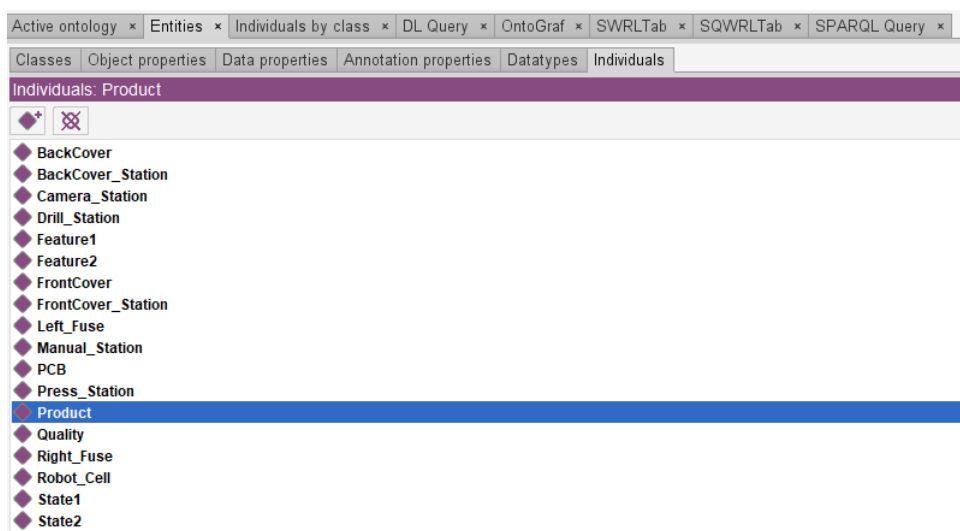


Figure 5-19: Representation of the individual Product at BackCover_Station

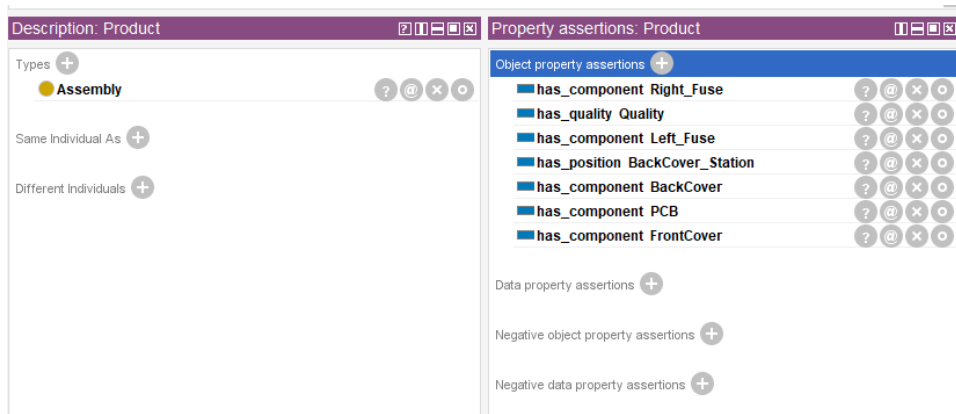


Figure 5-20: Representation of the Property Assertion of Product

- 6) After the back cover has been inserted on *Product*, the individual moves to the next station: the Press Station. At this station a transforming operation is being executed since the *Product* has to be pressed in order to attach all the components of the product together. This is the last operative station of the line so after this operation is executed it is possible to assign the individual *Process* to *Product*, figures 5-21 and 5-22. Where the individual *Process* varies accordingly to the quality of the product obtained. Therefore the following relations can be determined:

Product has_position Press_Station

Product has_process Process

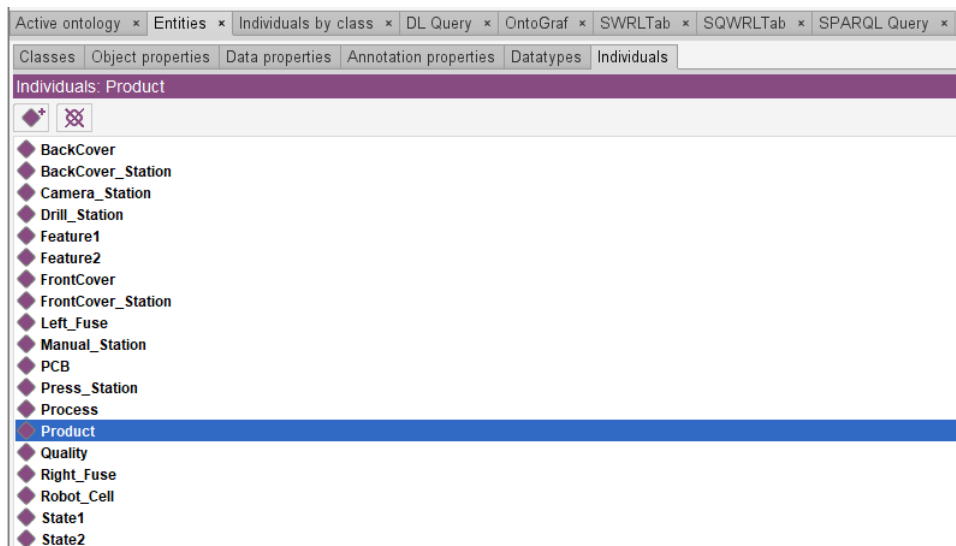


Figure 5-21: Representation of the individual Product at Press_Station

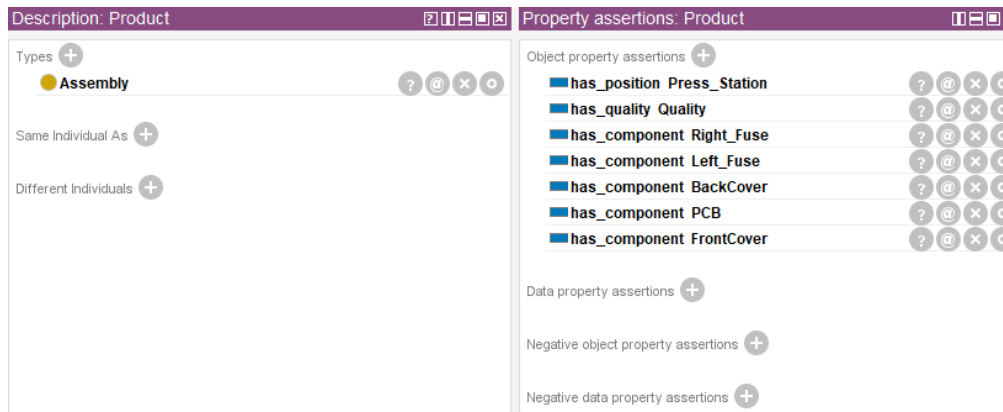


Figure 5-22: Representation of the Property Assertion of Product

7) The last step of the production process is the arrival of the product on the Manual Station. At this station no further operations are going to be performed, but it is necessary to determine if *Product* has to be kept, repaired or discarded. This decision can be taken accordingly to the results obtained from the ontology, which, in function of all the operations executed until now, it is able to determine the quality of *Product*. Since in this case the *Regular_Assembly_Quality* case has been dealt with, the *Product* is a Dummy Cellphone and so it does not need any reparations or disassembly, therefore *Product* can be kept. To verify if all the operations described until now have brought to obtain a Dummy Cellphone, it is necessary to run the reasoner in the ontology. The reasoner adopted is an Hermit 1.4.3.456 and the following relations have been inducted:

Product → *Dummy Cellphone*

PCB_Feature → *Conform*

PCB_Feature → *Good_State_Component*

Drilling_Machine_Feature → *Good_Operating_Condition*

Drilling_Machine_Feature → *Good_State_Asset*

FrontCover_Feature → *Conform*

Process → *Regular_Assembly_Process*

Quality → *Regular_Assembly_Quality*

PCB → *Regular_Assembly_Component*

Left_Fuse → *Regular_Assembly_Component*

Right_Fuse → *Regular_Assembly_Component*

FrontCover → *Regular_Assembly_Component*

BackCover → *Regular_Assembly_Component*

These relations can be seen in the images which goes from 5-23 to 5-32. In the images 5-23 and 5-24 it is possible to see the reasoning executed on *Product* showing that all the relations that compose the *Assembly* determine a *Dummy Cellphone*. In the images 5-25 and 5-26 the *PCB_Feature* is reported, in these images it is therefore possible to see how the PCB is *Conform* according to all the relations described until now. In the images 5-27 and 5-28 the *Drilling_Machine_Feature* is reported and it is possible to verify that, as previously said, since there is no *Malfunction*, the drilling machine has *Good_Operating_State*. In the images from 5-29 to 5-32 the *Process* and the *Quality* of the production process are reported and it is possible to verify that the *Process* is a *Regular_Assembly_Process* and the *Quality* is a *Regular_Assembly_Quality*.

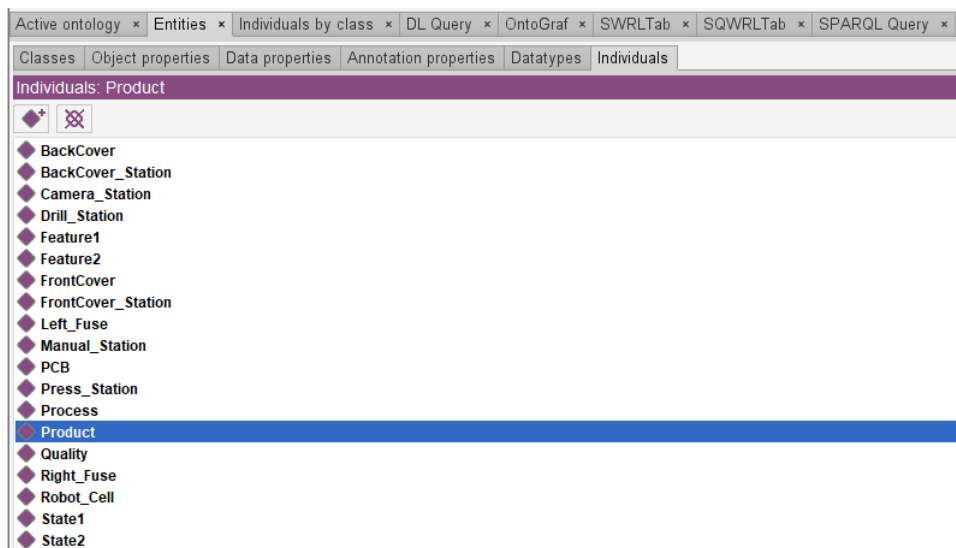


Figure 5-23: Representation of the individual *Product* at *Manual_Station*

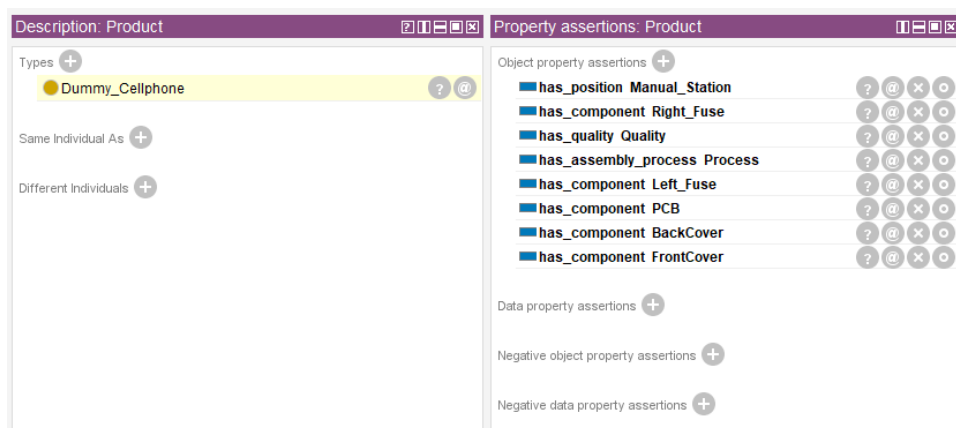


Figure 5-24: Representation of the Property Assertion of *Product* with the reasoner active

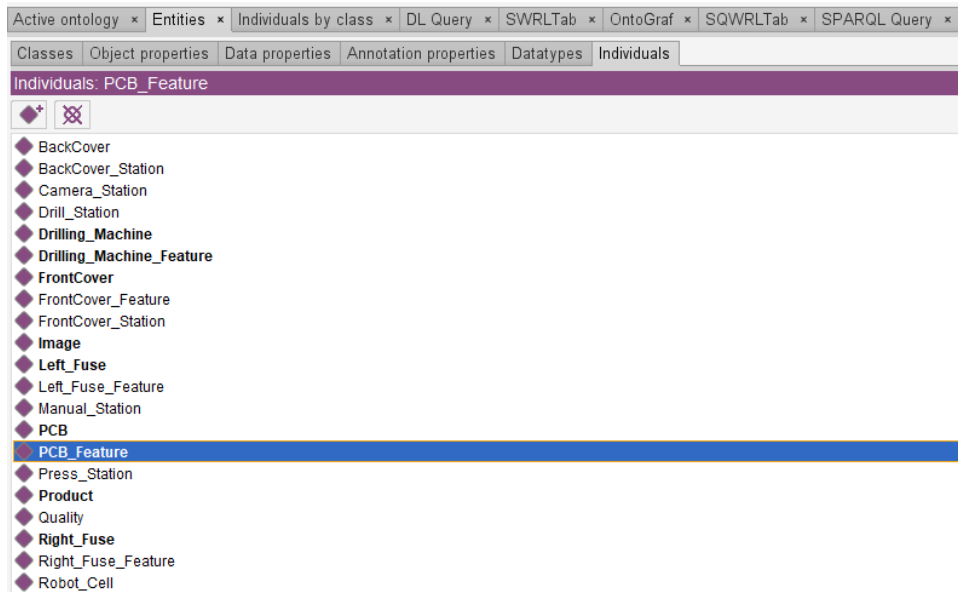


Figure 5-25: Representation of the individual PCB_Feature

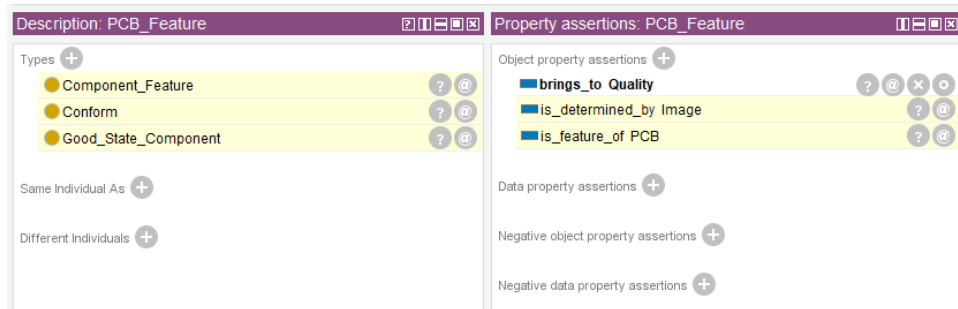


Figure 5-26: Representation of the Property Assertion of PCB_Feature with the reasoner active

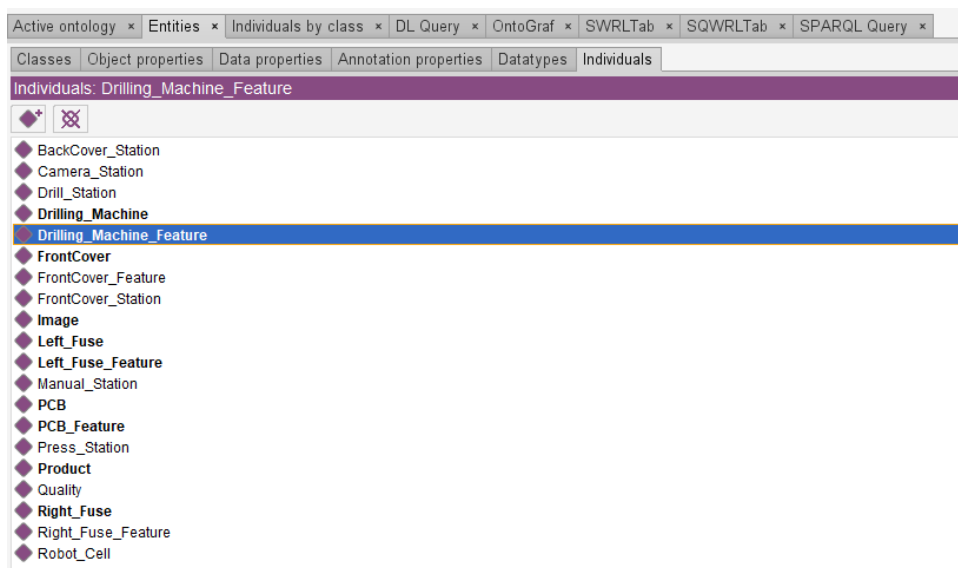


Figure 5-27: Representation of the individual Drilling_Machine_Feature

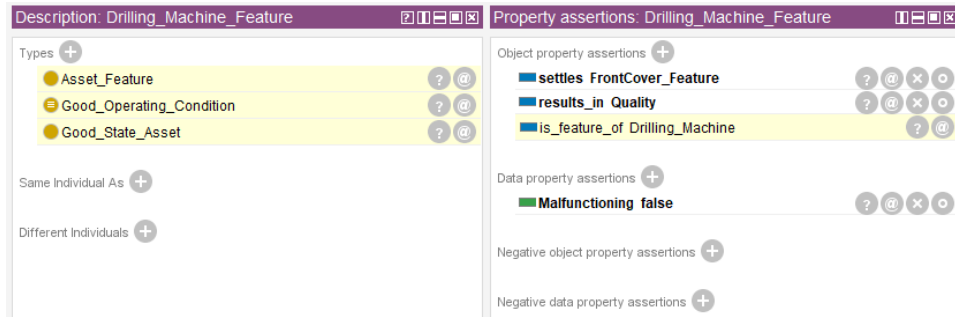


Figure 5-28: Representation of the Property Assertion of Drilling_Machine_Feature with the reasoner active

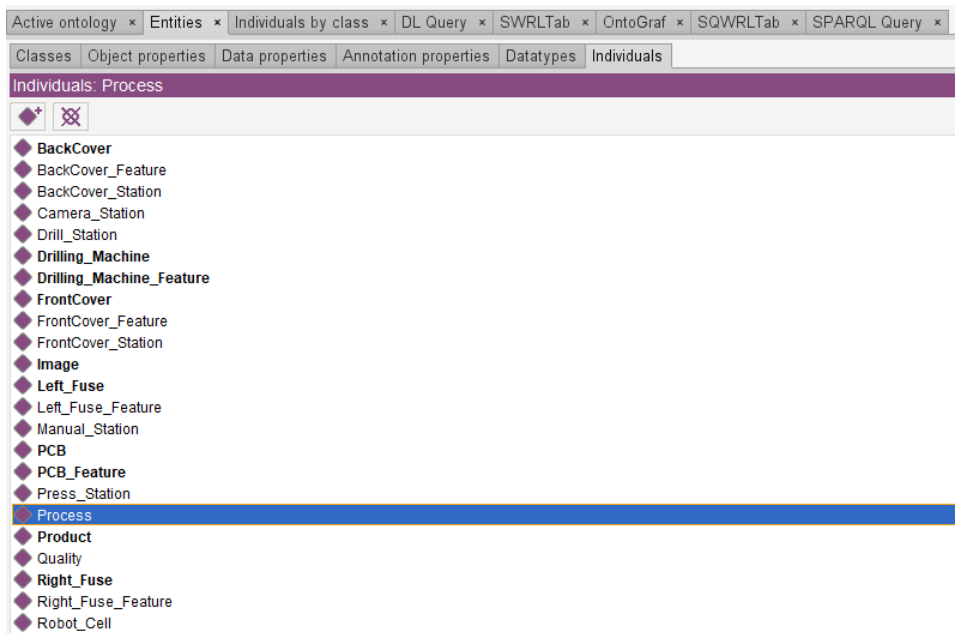


Figure 5-29: Representation of the individual Process

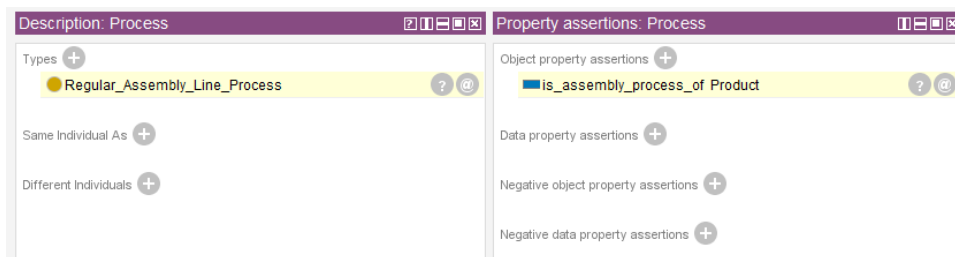


Figure 5-30: Representation of the Property Assertion of Process with the reasoner active



Figure 5-31: Representation of the individual Quality

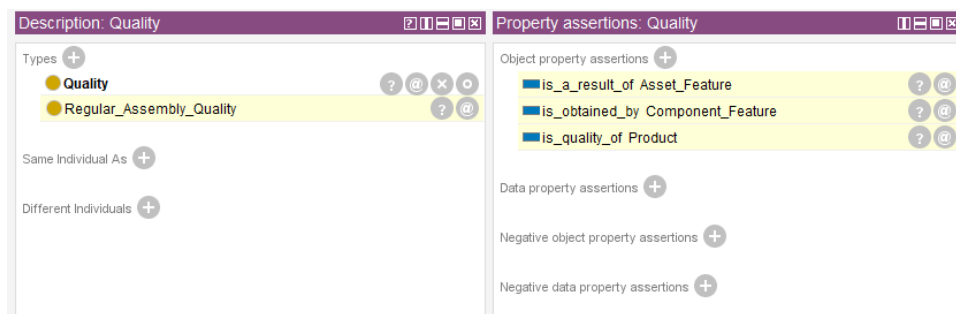


Figure 5-32: Representation of the Property Assertion of Quality with the reasoner active

5.2.2. Repairable_Assembly_Quality case

Now that the first case has been analyzed, it is necessary to move to the second case: the *Repairable Cellphone* case. As stated before this result can be obtained with different combinations of *Component_State* and *Asset_Operating_Condition*, so in the following scenario the case of an *Intermediate_State_Components* and *Bad_Operating_Condition* is going to be considered. The process to create a *Repairable Cellphone* will follow the same cycle presented in the case before, so only the differences respect to the previous case will be considered. Therefore only the Drill Station, the Camera Station and the results obtained from the reasoner will be reported. First the Drill station will be analyzed:

- In this scenario, at the Drill station, the drilling operation performed on the front cover will be different respect to the *Dummy Cellphone* case. In fact in the previous case the holes drilled by the drilling machine were *Conform* to specifics since the shacker was not active and so the front cover does not need any repair process. In the current scenario the shacker is active and so it is being simulated

a malfunctioning of the drilling machine. So by looking at the figures from 5-33 to 5-36 the following relations can be determined:

Drilling_Machine has_feature Drilling_Machine_Feature

Drilling_Machine_Feature → Malfunction value true

Drilling_Machine_Feature settles FrontCover_Feature

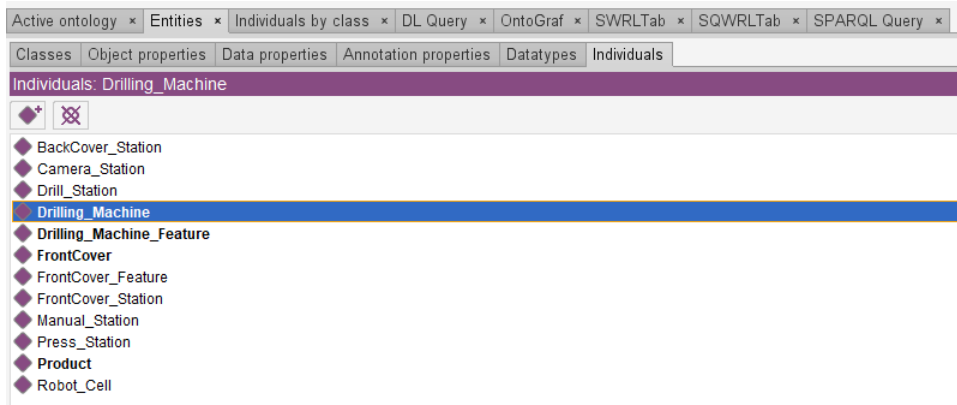


Figure 5-33: Representation of the individual Drilling Machine

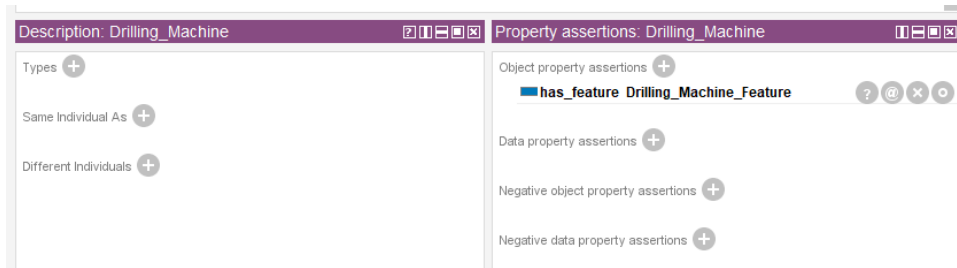


Figure 5-34: Representation of the Property Assertion of Drilling Machine

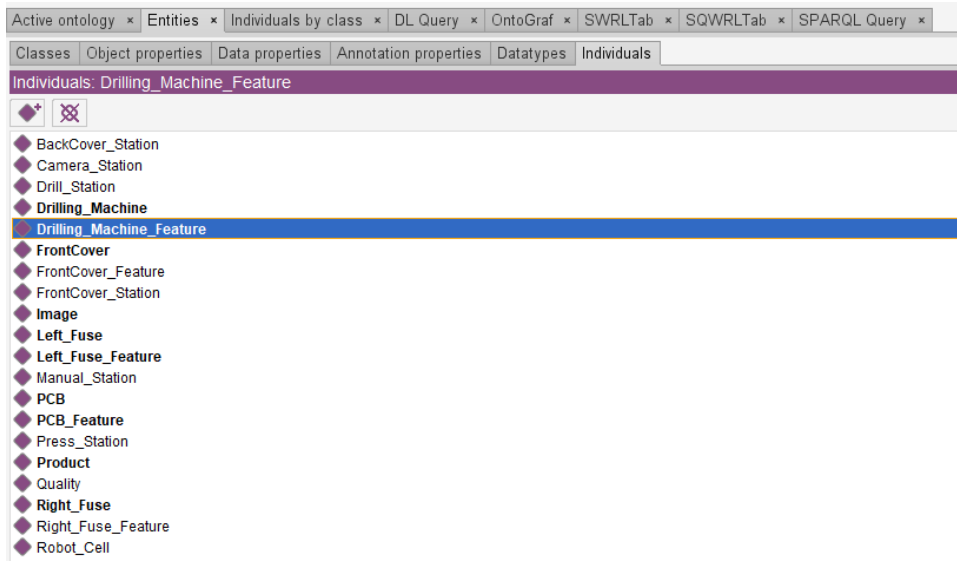


Figure 5-35: Representation of the individual Drilling Machine Feature

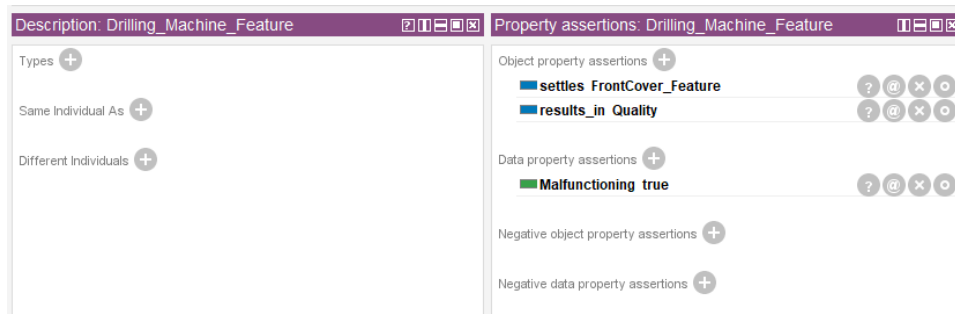


Figure 5-36: Representation of the property assertions of Drilling Machine Feature

- When the Product arrives at the Camera Station, the Assembly is visually inspected by a camera to determine if some defects are present on the product. In this case since the *Repairable_Assembly_Quality* is being considered the PCB is going to be *Not Conform but Repairable* and so a *Square_Image* is present on the PCB, images 5-37 and 5-38. Accordingly to the visual inspection conducted, the following relations can be established:

PCB has_image Image
Image determine PCB_Feature
Image → Square_Image
PCB_Feature brings_to Quality
Product has_quality Quality

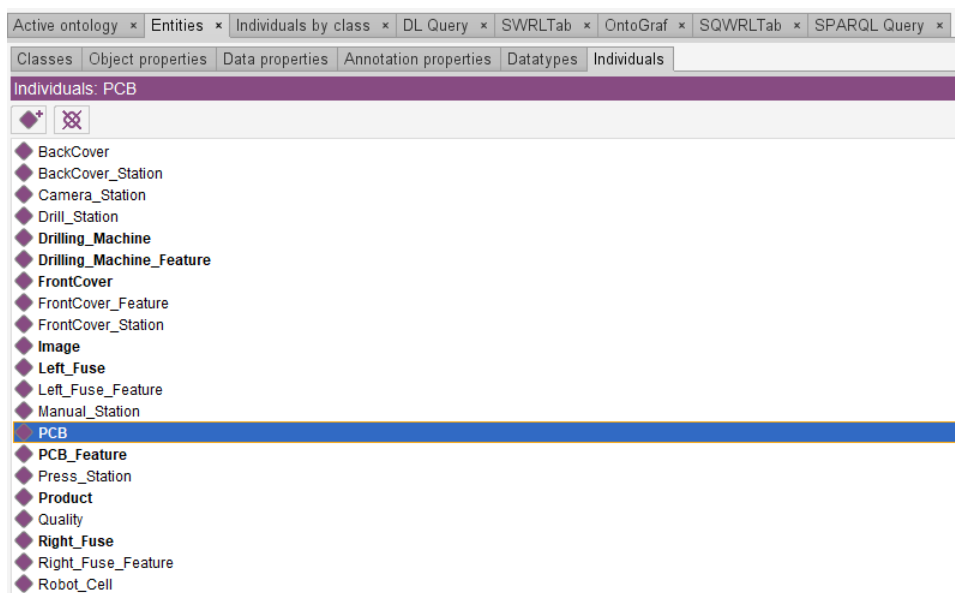


Figure 5-37: Representation of the individual PCB

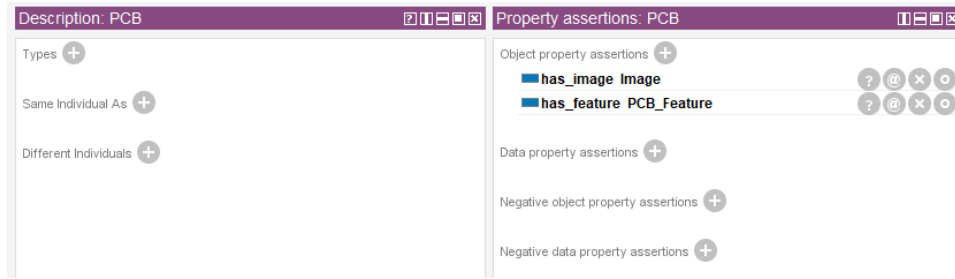


Figure 5-38: Representation of the Property Assertion of PCB

- Now that the Drill Station and the Camera Station have been considered, it is possible to evaluate the differences which are going to be present when the reasoner is activated. In this scenario there are 2 components to be repaired: the front cover and the PCB. In fact the drilling operation executed is sub-optimal since the shacker was active, meaning that the drilling machine had a malfunction, and consequently the front cover needs to be repaired since the holes drilled are not *Conform* to specifics. Concerning the PCB, after the visual inspection at the Camera Station, it is possible to determine that the image found on the component is a Square, so the PCB is defective, but it can be repaired. Therefore by activating the reasoner these are the relations which are going to be found:

Product → *Repairable Cellphone*

PCB_Feature → *Repairable*

PCB_Feature → *Intermediate_State_Component*

Drilling_Machine_Feature → *Bad_Operating_Condition*

Drilling_Machine_Feature → *Bad_State_Asset*

FrontCover_Feature → *Repairable*

Process → *Repairable_Assembly_Process*

Quality → *Repairable_Assembly_Quality*

In the images from 5-39 to 5-50 the results obtained from the reasoner can be checked. In particular in the figure 5-39 and 5-40 it is possible to see the *Drilling_Machine_Feature* and it is possible to verify that the drilling machine has a *Bad_Operating_Condition*. In the images 5-41 and 5-42 it is reported the *FrontCover_Feature* and the fact that the front cover is *Repairable*. In the images 5-43 and 5-44 the *PCB_feature* is reported and it is showed how also the PCB can be *Repaired*. In the images 5-45 and 5-46 the characteristics of the individual *Product* are reported and it is showed how the *Assembly* is a *Repairable_Cellphone*. In the images from 5-47 to 5-50 the *Process* and the

Quality of the production process are reported and it is possible to verify that the *Process* is a *Repairable_Assembly_Process* and the *Quality* is a *Repairable_Assembly_Quality*

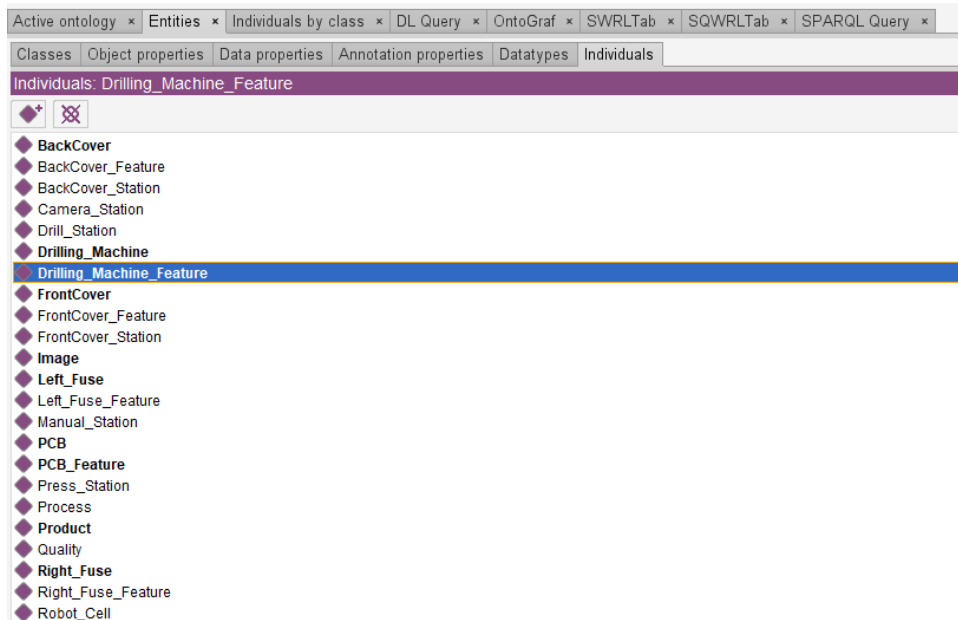


Figure 5-39: Representation of the individual Drilling_Machine_Feature

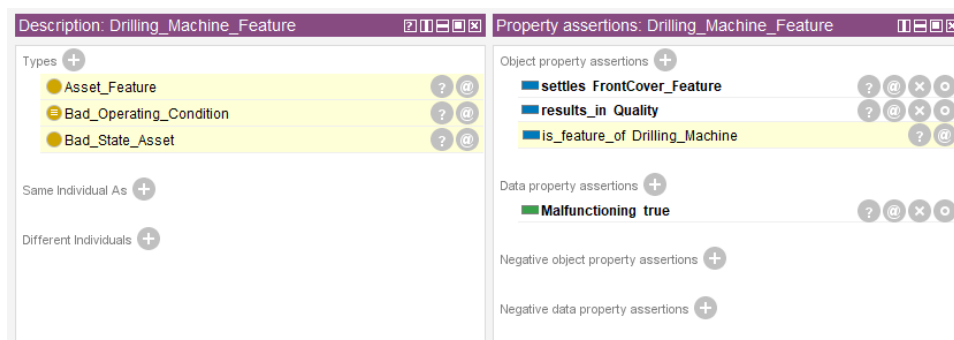


Figure 5-40: Representation of the Property Assertion of Drilling_Machine_Feature with the reasoner active

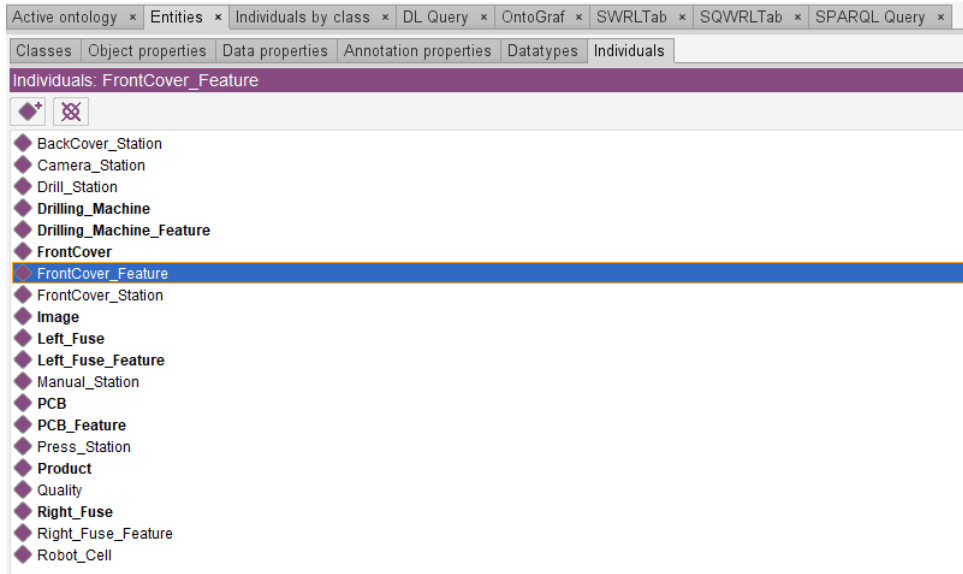


Figure 5-41: Representation of the individual FrontCover_Feature

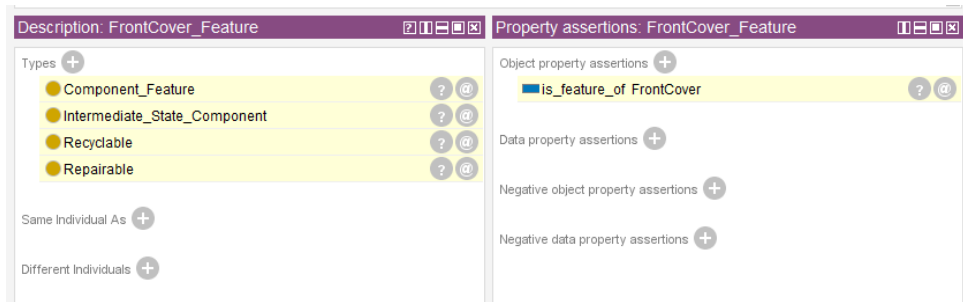


Figure 5-42: Representation of the Property Assertion of FrontCover_Feature with the reasoner active



Figure 5-43: Representation of the individual PCB_Feature

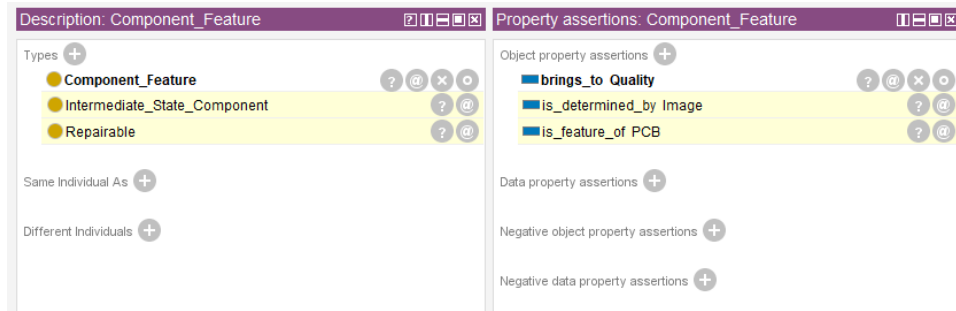


Figure 5-44: Representation of the Property Assertion of PCB_Feature with the reasoner active

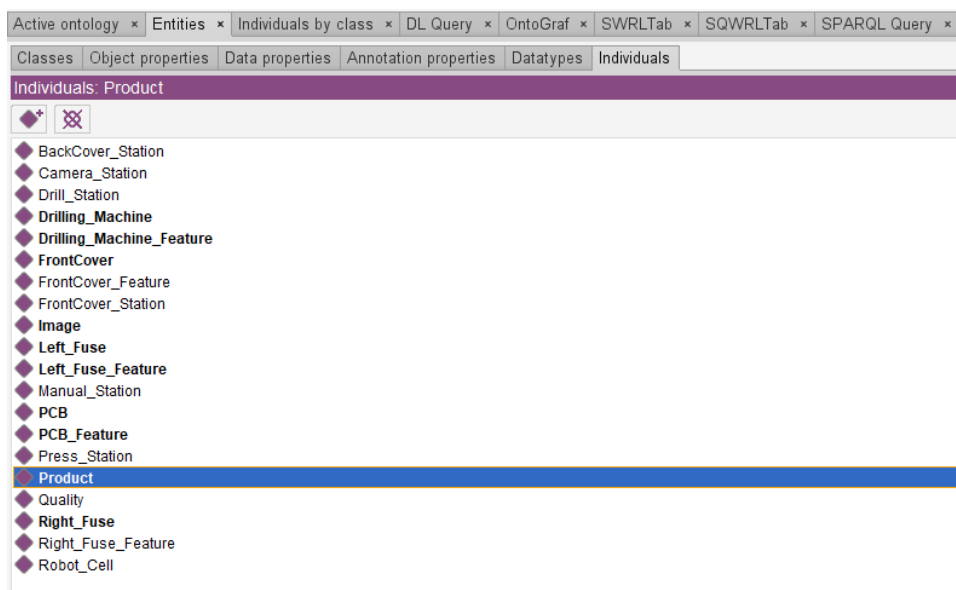


Figure 5-45: Representation of the individual Product at Manual Station

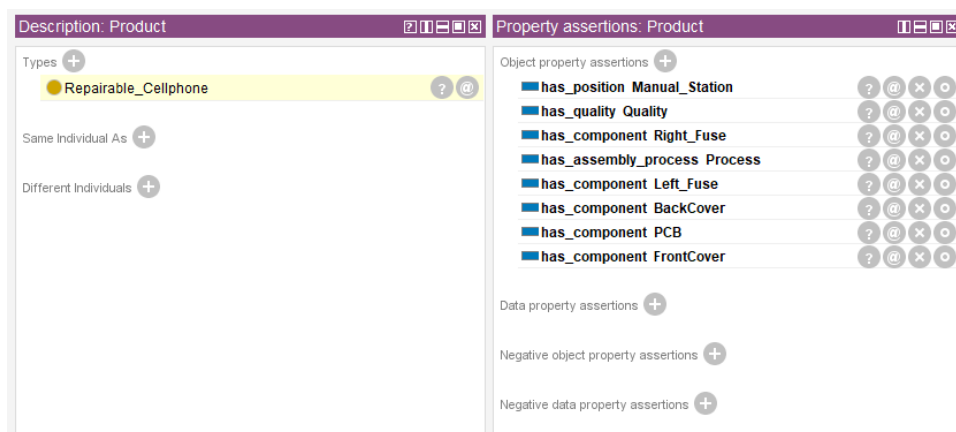


Figure 5-46: Representation of the Property Assertion of Product with the reasoner active

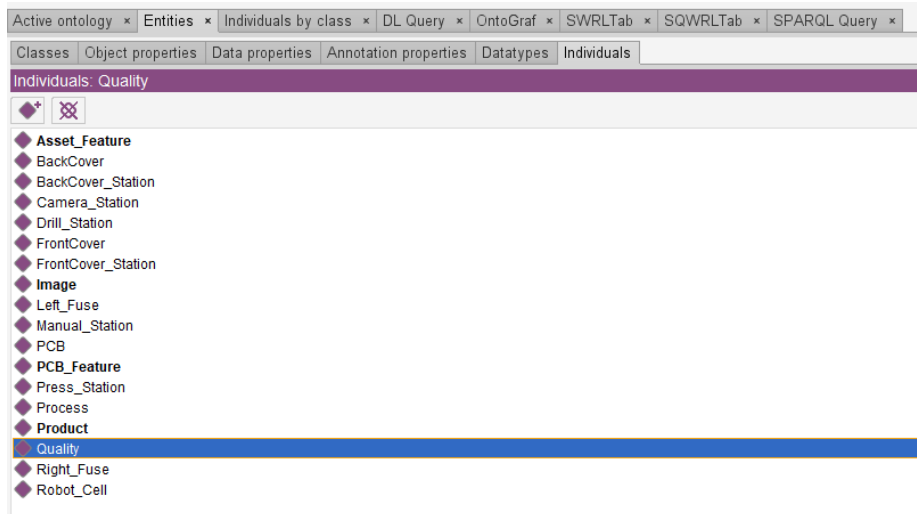


Figure 5-47: Representation of the Individual Quality

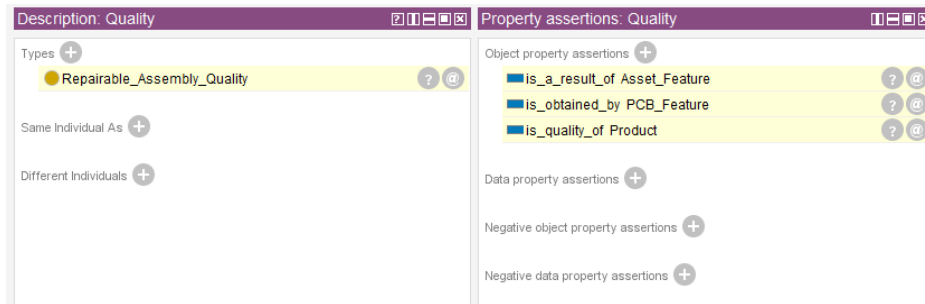


Figure 5-48: Representation of the property Assertion of Quality

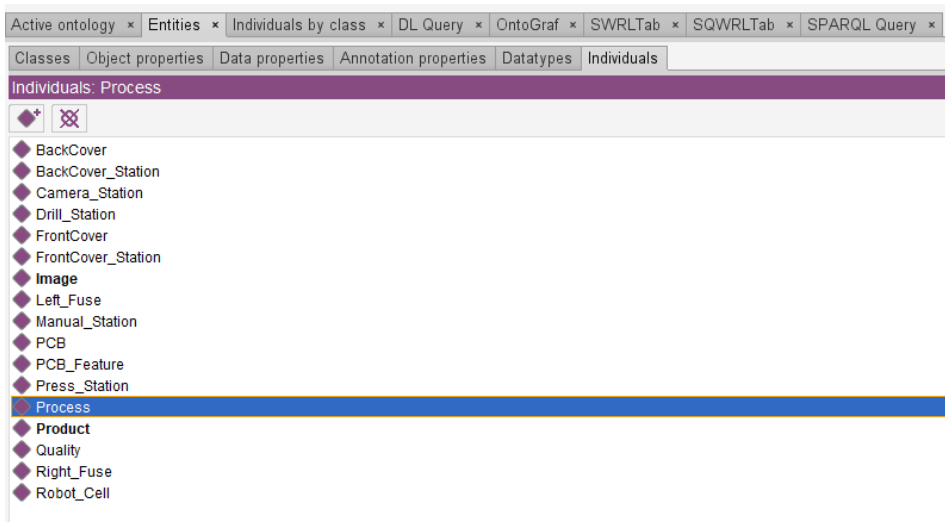


Figure 5-49: representation of the Individual Process

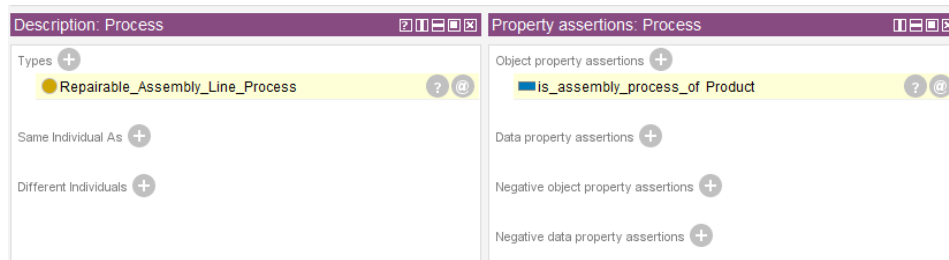


Figure 5-50: Representation of the property Assertion of Process with the reasoner active

5.2.3. Defective_Assembly_Quality case

The last case left to analyze is the *Defective Cellphone* case. As stated before this result can be obtained when the *Asset* is characterized by a *Bad_Operating_Condition* and the *Product* is characterized by the presence of, at least, one *Bad_State_Component*. The process to realize a *Defective Cellphone* will follow the same cycle presented in the cases before, so only the differences respect to the previous cases will be considered. Therefore only the Camera Station and the results obtained from the reasoner will be reported, since in the previous case it has been already treated the case of an *Asset* with *Bad_Operating_Condition*.

- At the Camera Station the condition of the Product is going to be checked accordingly to the condition of the PCB. Since it is being considered the case of the *Defective_Assembly_Quality* the PCB does not conform to specification and it is not repairable. Therefore the image present on the PCB is a *Triangle_Image*. So the following relations can be determined:

PCB has_image Image

Image determine PCB_Feature

Image → Triangle_Image

PCB_Feature brings_to Quality

These relations can be seen in the following images, where in images 5-51 and 5-52 the PCB is represented and in 5-53 and 5-54 the image is represented.

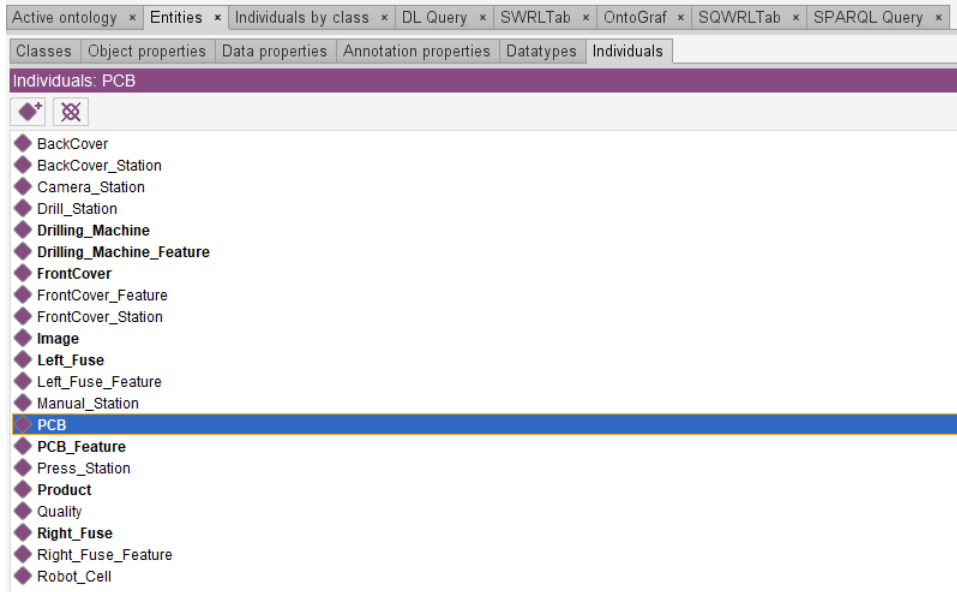


Figure 5-51: Representation of the individual PCB

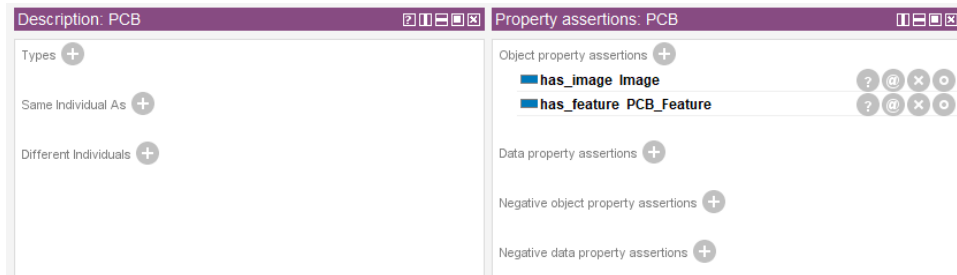


Figure 5-52: Representation of the Property Assertion of PCB

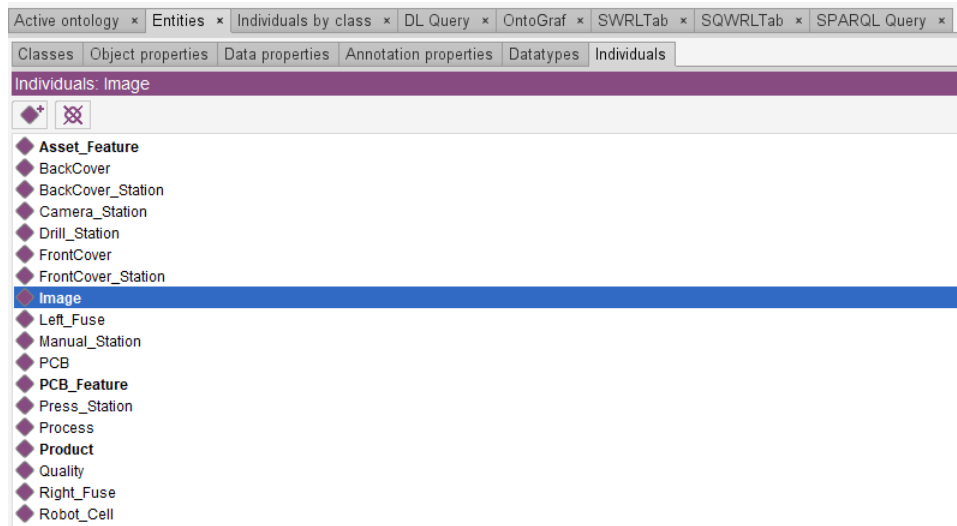


Figure 5-53: Representation of the Individual Image

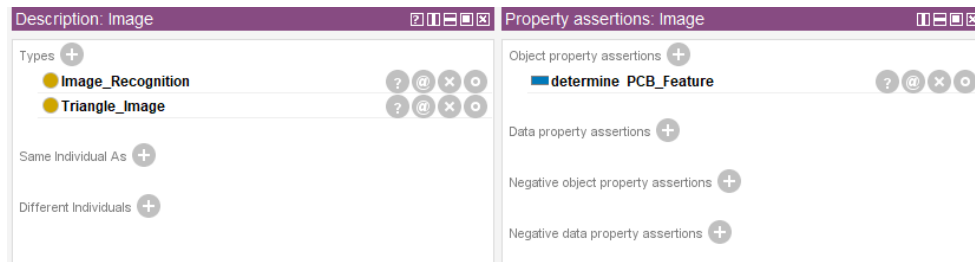


Figure 5-54: Representation of the property assertion of Image

- Now that the Camera Station has been treated, it is necessary to deal with the results obtained when the reasoner is activated. In this scenario the *Product* is characterized by a *Defective_Assembly_Quality* and so at least one component will be characterized by a *Bad_State_Component*. In this case, those components which are recognized as *Bad_State_Component* should not be recycled, since they are defective, instead those components which are *Conform* or *Repairable* should be recycled, since they can be used for another production process. Accordingly to what have been said the following relations can be determined:

Product → *Defective Cellphone*

PCB_Feature → *Not_Repairable*

PCB_Feature → *Bad_State_Component*

Drilling_Machine_Feature → *Bad_Operating_Condition*

Drilling_Machine_Feature → *Bad_State_Asset*

FrontCover_Feature → *Repairable*

Process → *Defective_Assembly_Process*

Quality → *Defective_Assembly_Quality*

FrontCover_Feature → *Repairable*

FrontCover_Feature → *Recyclable*

PCB_Feature → *Not_Repairable*

PCB_Feature → *Not_Recyclable*

Left_Fuse_Feature → *Recyclable*

Right_Fuse_Feature → *Recyclable*

In the images from 5-55 to 5-68 the results obtained from the reasoner can be checked. In particular in the figure 5-55 and 5-56 it is possible to see the *FrontCover_Feature* and it is possible to verify that the front cover has a *Repairable* and *Recyclable* state. In the images 5-57 and 5-58 it is reported the *PCB_Feature* and the fact that the PCB is *Not_Repairable*. In the images from 5-59 to 5-62 the features of the fuses are represented,

where in the images 5-59 and 5-60 the left fuse feature is represented while in the images 5-61 and 5-62 the right fuse feature is represented. In the images 5-63 and 5-64 the characteristics of the individual *Product* are reported and it is showed how the *Assembly* is a *Defective_Cellphone*. In the images from 5-65 to 5-68 the *Process* and the *Quality* of the production process are reported and it is possible to verify that the *Process* is a *Defective_Assembly_Process* and the *Quality* is a *Defective_Assembly_Quality*.

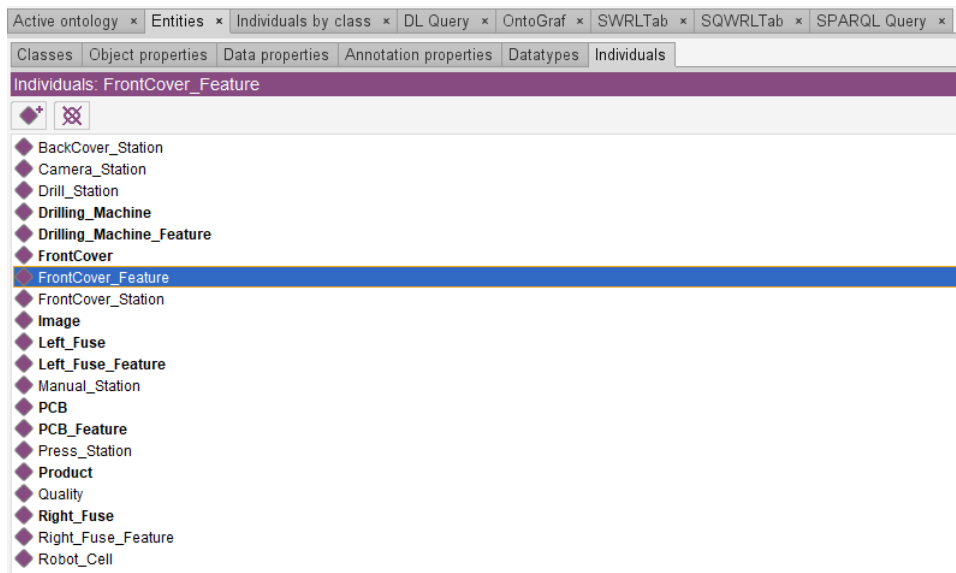


Figure 5-55: Representation of the individual FrontCover_Feature

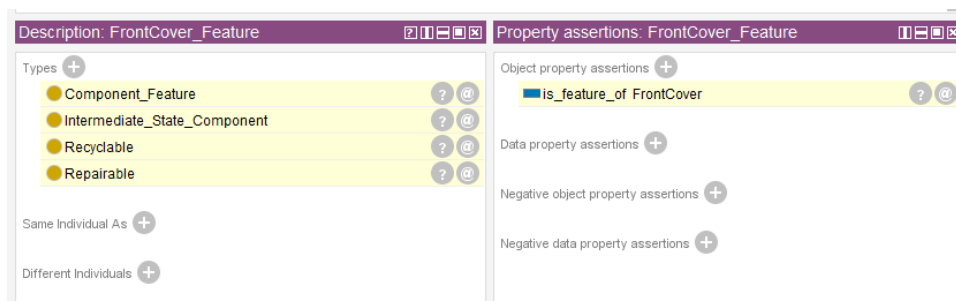


Figure 5-56: Representation of the Property Assertion of FrontCover_Feature



Figure 5-57: Representation of the individual Component_Feature

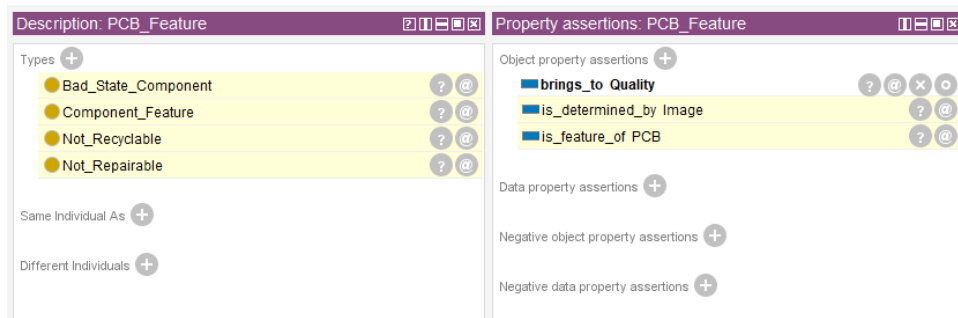


Figure 5-58: Representation of the Property Assertion of PCB_Feature with the reasoner active

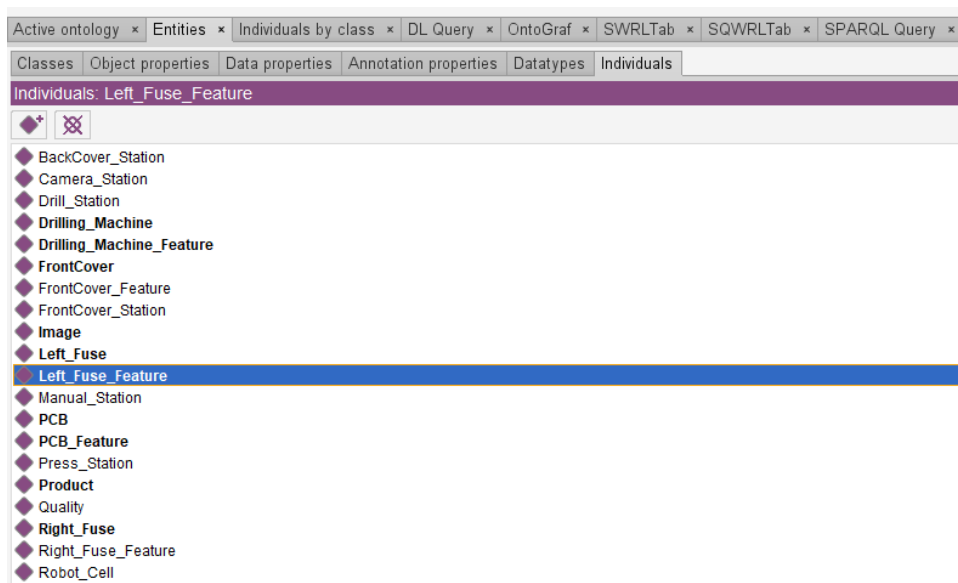


Figure 5-59: Representation of the individual Left_Fuse_Feature

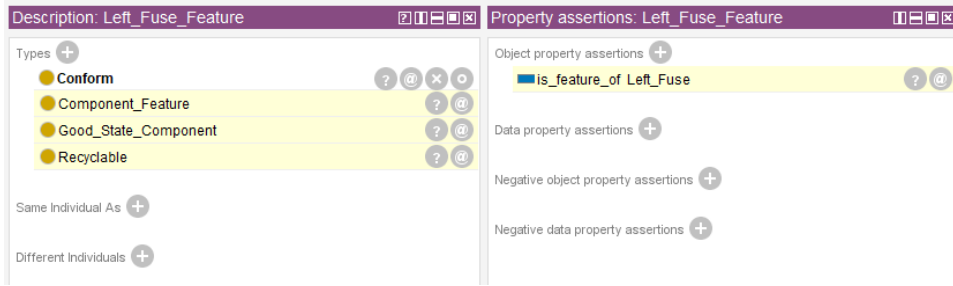


Figure 5-60: Representation of the Property Assertion of Left_Fuse_Feature with the reasoner active

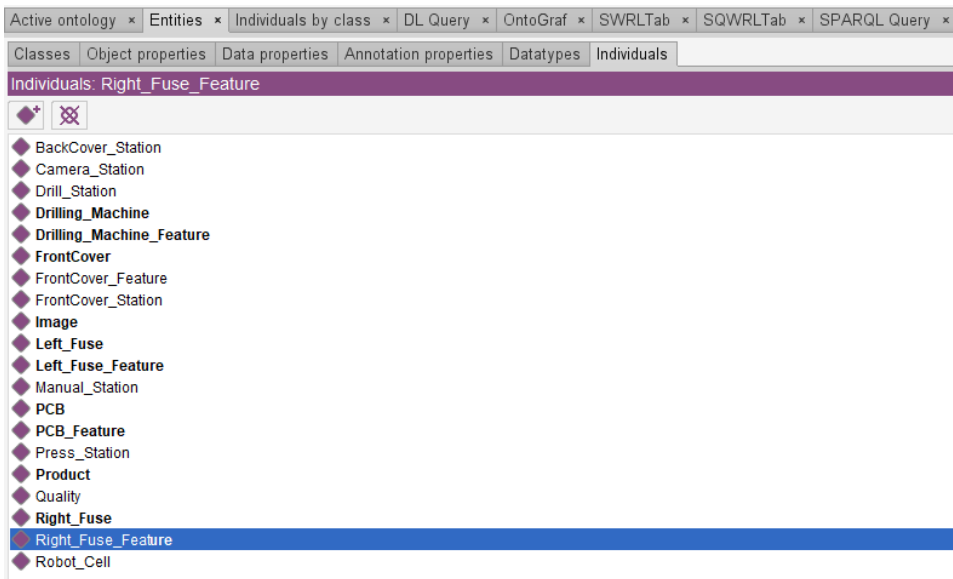


Figure 5-61: Representation of the individual Right_Fuse_Feature

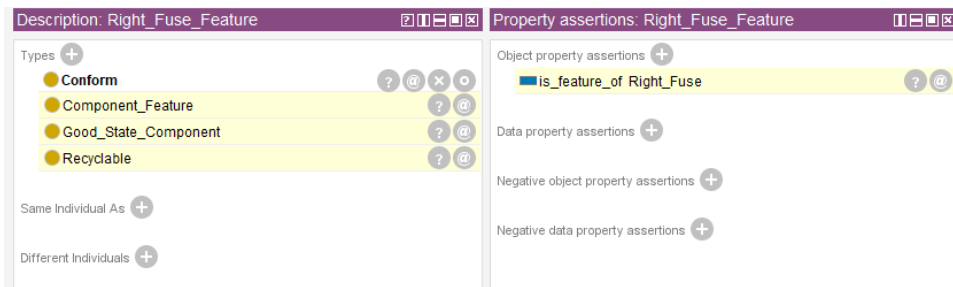


Figure 5-62: Representation of the Property Assertion of Right_Fuse_Feature with the reasoner active

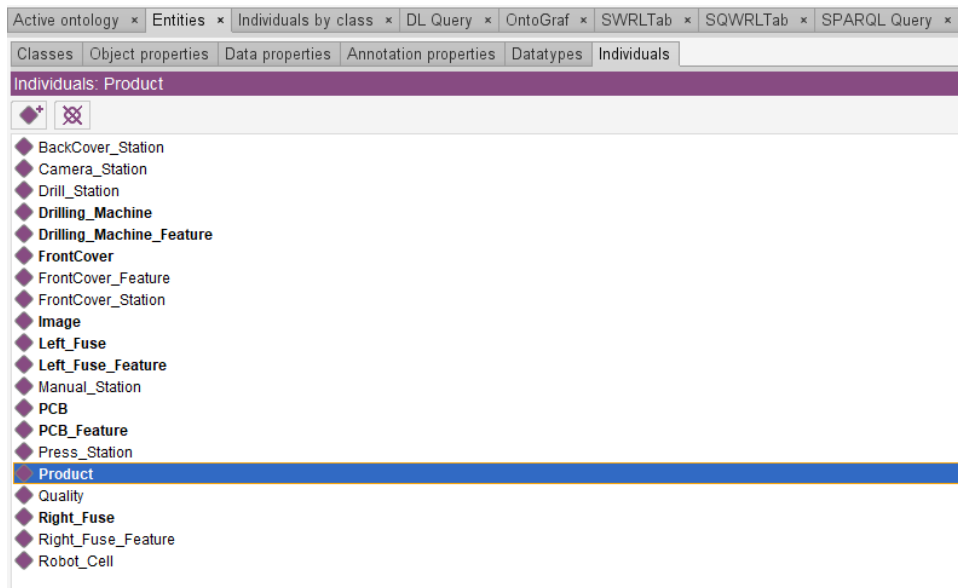


Figure 5-63: Representation of the individual Product at Camera Station

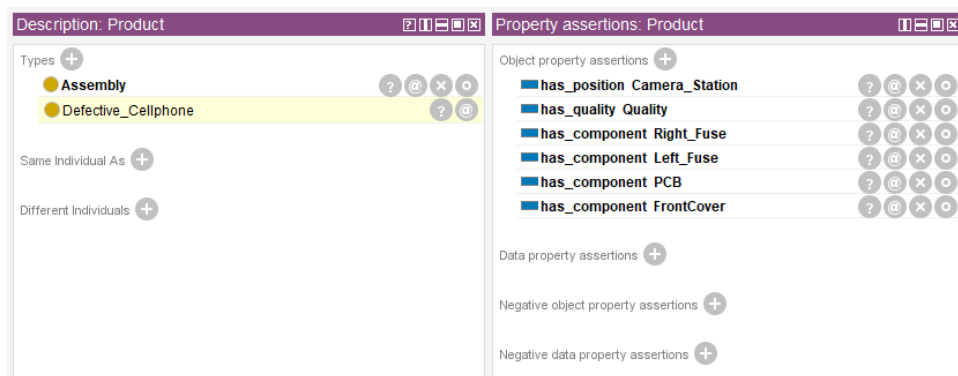


Figure 5-64: Representation of the Property Assertion of Product with the reasoner active

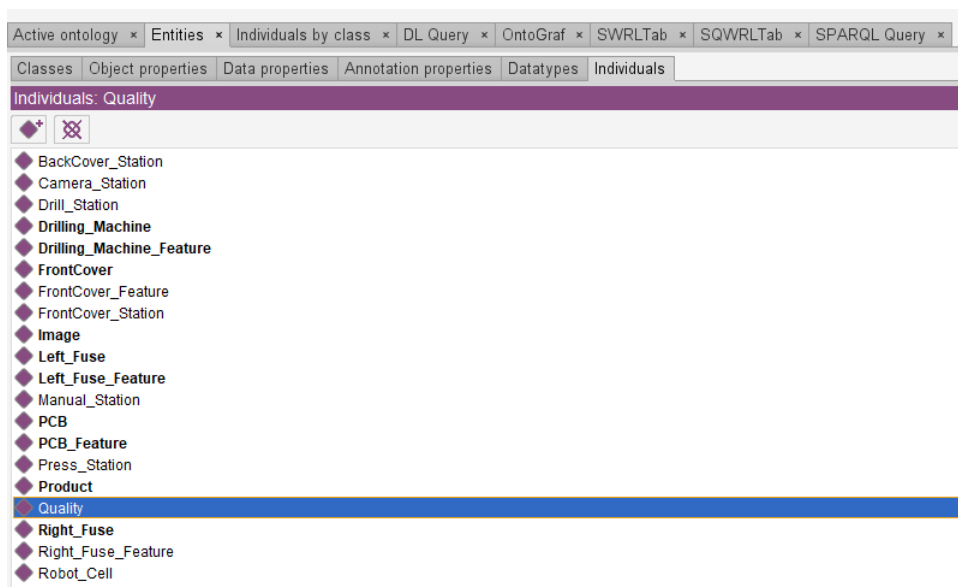


Figure 5-65: Representation of the individual Quality

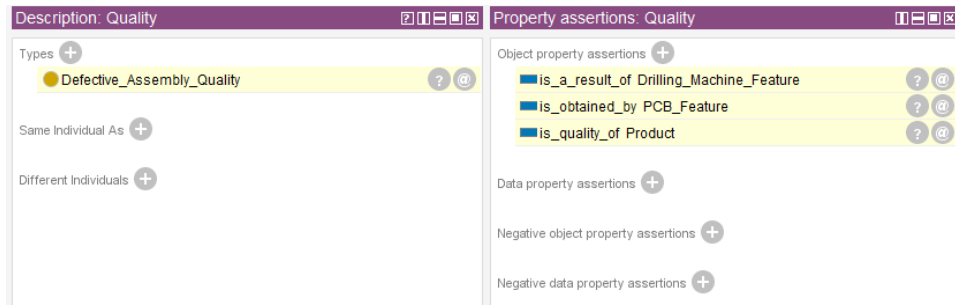


Figure 5-66: Representation of the Property Assertion of Quality with the reasoner active

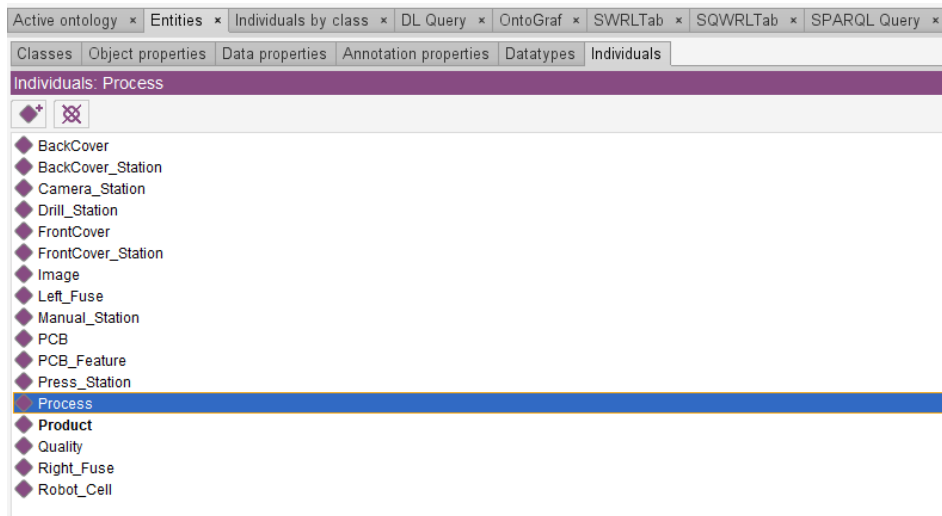


Figure 5-67: Representation of the Individual Process

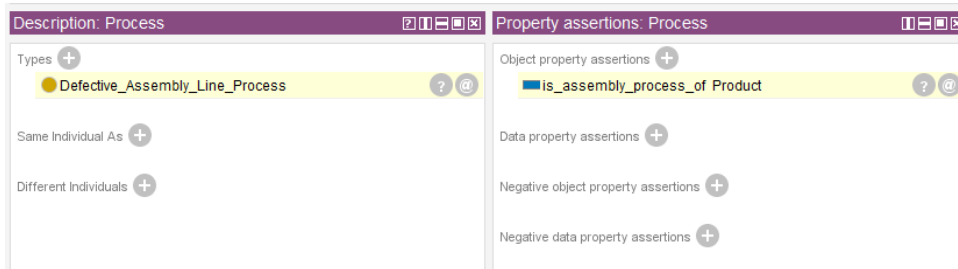


Figure 5-68: Representation of the Property Assertion of Process with the reasoner active

5.2.4. Competency questions

As already mentioned to verify the ontology and to certify that the ontology is able to represent the current knowledge of the system, the ontology needs to answer the competency questions mentioned above. Therefore, in this chapter are presented the Snap SPARQL queries to answer the competency questions introduced in chapter 3. To do so, six individuals have been inserted into the ontology to represent all the types of products that the FML could generate. In particular, these are the elements introduced in the ontology:

- Product 1 is a Dummy Cellphone
 - Product 2, Product 3, Product 4 and Product 5 are Repairable Cellphones
 - Product 6 is a Defective Cellphone
- CQ1 What is the quality of the products that the system realizes?

```

Active ontology x Entities x Individuals by class x DL Query x OntoGraf x SWRLTab x SQWRLTab x SPARQL Query x
SPARQL Query Snap SPARQL Query
Snap SPARQL Query:
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.semanticweb.org/loren/ontologies/2022/5/untitled-ontology-44#>

SELECT ?product ?quality
WHERE {
    ?product :has_quality ?quality .
    ?product rdf:type :Assembly .
    ?quality rdf:type :Quality .
}
ORDER BY ?label
    
```

Figure 5-69: Query to get the quality of the product

From the image above it is possible to see that the results of the query, image 5-69, is a set of qualities, image 5-71. In fact the query executed in figure 5-69 search in the ontology all the elements connected through the relationship "has_quality", in particular are searched all the elements of the Assembly and Quality class that fulfill this relation.

	?product
	:Product4
	:Product3
	:Product2
	:Product1
	:Product6
	:Product5

Figure 5-70: Products found by the query

	?quality
	:Quality4
	:Quality3
	:Quality2
	:Quality1
	:Quality6
	:Quality5

Figure 5-71: Qualities found by the query

From the query of image 5-69 it is not possible to distinguish the quality of the different products, for example it is not possible to distinguish the quality of the Defective Cellphone from the Dummy Cellphone. In fact the query looks for all the qualities that are present in the ontology, so in order to make a

distinction between qualities it is necessary to change the query of image 5-69 in the following way, image 5-72:

```

Active ontology x Entities x Individuals by class x DL Query x SWRLTab x OntoGraf x SQWRLTab x SPARQL Query x
SPARQL Query Snap SPARQL Query
Snap SPARQL Query:
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.semanticweb.org/loren/ontologies/2022/5/untitled-ontology-44#>

SELECT ?product ?quality
WHERE {
  ?product :has_quality ?quality .
  ?product rdf:type :Assembly .
  ?quality rdf:type :Regular_Assembly_Quality
}
ORDER BY ?label

```

Figure 5-72: Query to get the product with Regular_Assembly_Quality

Executing this query brings to the following results, image 5-73 and image 5-74:

Execute	
?product	
:Product1	

Figure 5-73: Product found by the query

?quality	
:Quality1	

Figure 5-74: Quality found by the query

So the query found which of the products in the ontology is characterized by a Regular_Assembly_Quality image 5-74, so which one of the products is a Dummy Cellphone, image 5-73.

- CQ2 Which components realize the product?

```

Active ontology x Entities x Individuals by class x DL Query x OntoGraf x SWRLTab x SQWRLTab x SPARQL Query x
SPARQL Query Snap SPARQL Query
Snap SPARQL Query:
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.semanticweb.org/loren/ontologies/2022/5/untitled-ontology-44#>

SELECT ?product ?component
WHERE {
  ?product :has_component ?component .
  ?product rdf:type :Assembly .
  ?component rdf:type :Component .
}
ORDER BY ?label

```

Figure 5-75: Query to get the components of the product

From the image above it is possible to see that the results of the query, image 5-75, is a set of components, image 5-77. In fact the query executed in figure 5-75 search in the ontology all the elements connected through the relationship

"has_component", in particular are searched all the elements of the Assembly, image 5-76, and Component class, image 5-77, that fulfill this relation.

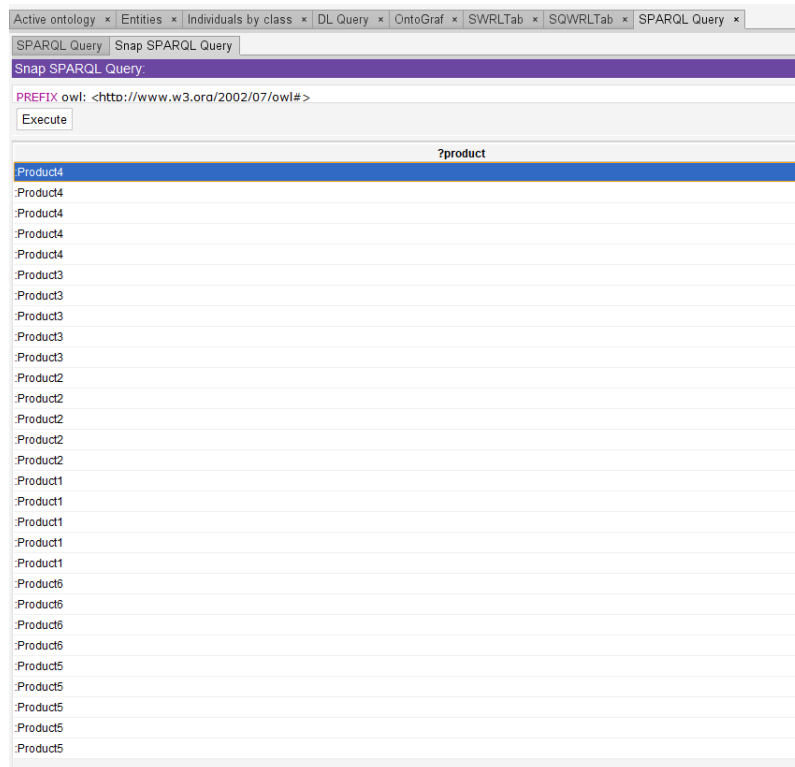


Figure 5-76: Products found by the query

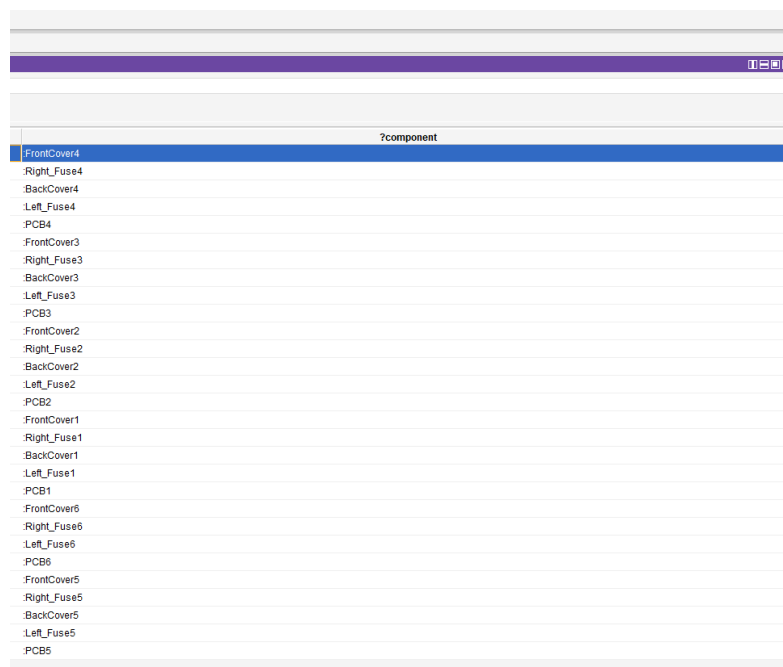
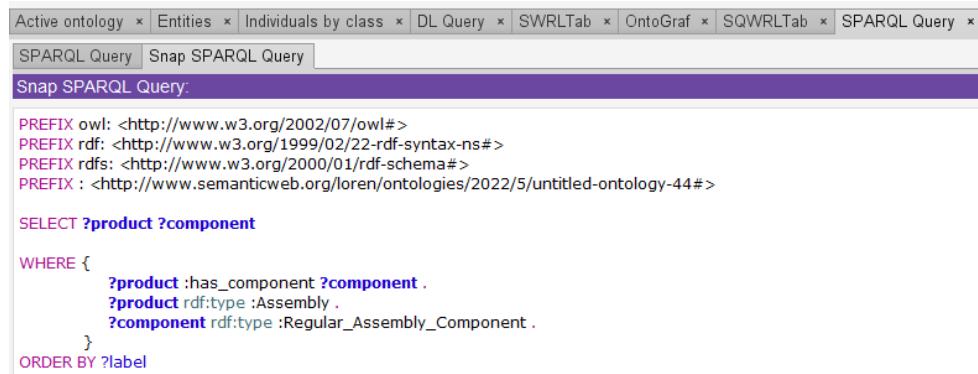


Figure 5-77: Components found by the query

As for the competency question before the query executed does not distinguish the components of the different products, but it simply reports all the

components which are present in the ontology. For this reason in order to make a distinction between components it is necessary to change the query of image 5-75 in the following way:



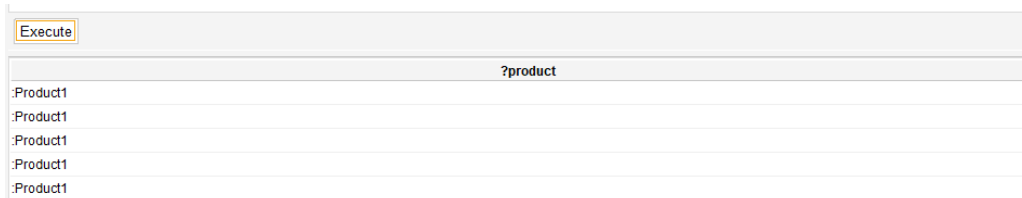
```

Active ontology x | Entities x | Individuals by class x | DL Query x | SWRLTab x | OntoGraf x | SQWRLTab x | SPARQL Query x
SPARQL Query | Snap SPARQL Query
Snap SPARQL Query:
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.semanticweb.org/loren/ontologies/2022/5/untitled-ontology-44#>

SELECT ?product ?component
WHERE {
  ?product :has_component ?component .
  ?product rdf:type :Assembly .
  ?component rdf:type :Regular_Assembly_Component .
}
ORDER BY ?label
  
```

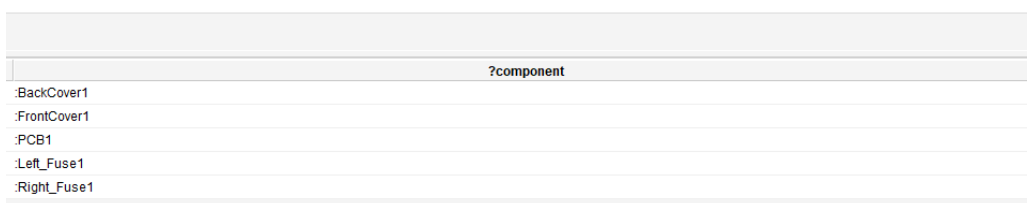
Figure 5-78: Query to determine the Regular_Assembly_Component

Executing this query brings to the following results, image 5-79 and image 5-80:



Execute	?product
	:Product1
	:Product1
	:Product1
	:Product1
	:Product1

Figure 5-79: Product found by the query



?component
:BackCover1
:FrontCover1
:PCB1
:Left_Fuse1
:Right_Fuse1

Figure 5-80: Component found by the query

So the query found which of the products in the ontology is characterized by a Regular_Assembly_Components image 5-80, so which one of the products is a Dummy Cellphone, image 5-79.

- CQ3 Which are the processes required to realize product x?

```

Active ontology x Entities x Individuals by class x DL Query x OntoGraf x SWRLTab x SQWRLTab x SPARQL Query x
SPARQL Query Snap SPARQL Query
Snap SPARQL Query:
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.semanticweb.org/loren/ontologies/2022/5/untitled-ontology-44#>

SELECT ?product ?process
WHERE {
    ?product :has_assembly_process ?process .
    ?product rdf:type :Assembly .
    ?process rdf:type :Process .
}
ORDER BY ?label
    
```

Figure 5-81: Query to determine the processes

From the image above it is possible to see that the results of the query, image 5-81, is a set of processes, image 5-83. In fact the query executed in figure 5-81 search in the ontology all the elements connected through the relationship "has_assembly_process", in particular are searched all the elements of the Assembly and Process class that fulfill this relation.

Execute	?product
	:Product4
	:Product3
	:Product2
	:Product1
	:Product6
	:Product5

Figure 5-82: Product found by the query

?process
:Process4
:Process3
:Process2
:Process1
:Process6
:Process5

Figure 5-83: Process found by the query

In order to make a distinction between processes it is necessary to change the query of image 5-81 in the following way:

```

Active ontology x Entities x Individuals by class x DL Query x SWRLTab x OntoGraf x SQWRLTab x SPARQL Query x
SPARQL Query Snap SPARQL Query
Snap SPARQL Query:
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.semanticweb.org/loren/ontologies/2022/5/untitled-ontology-44#>

SELECT ?product ?process

WHERE {
    ?product :has_assembly_process ?process .
    ?product rdf:type :Assembly .
    ?process rdf:type :Regular_Assembly_Line_Process .
}
ORDER BY ?label

```

Figure 5-84: Query to determine the Regular_Assembly_Line_Process

Executing this query brings to the following results, image 5-85 and image 5-86:

Execute	
	?product
	:Product1

Figure 5-85: Product found by the query

	?process
	:Process1

Figure 5-86: Process found by the query

So the query found which of the products in the ontology is characterized by a Regular_Assembly_Process image 5-86, so which one of the products is a Dummy Cellphone, image 5-85.

- CQ4 Which product/s is/are not feasible considering the current component/asset state?

```

Active ontology x Entities x Individuals by class x DL Query x OntoGraf x SWRLTab x SQWRLTab x SPARQL Query x
SPARQL Query Snap SPARQL Query
Snap SPARQL Query:
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.semanticweb.org/loren/ontologies/2022/5/untitled-ontology-44#>

SELECT ?product ?quality ?asset ?asset_feature ?component ?component_feature
WHERE {
    ?product :has_quality ?quality .
    ?product :has_component ?component .
    ?asset :has_feature ?asset_feature .
    ?asset_feature :results_in ?quality .
    ?component :has_feature ?component_feature .
    ?component_feature :brings_to ?quality .
    ?asset_feature rdf:type :Bad_Operating_Condition .
    ?component_feature rdf:type :Bad_State_Component .
    ?quality rdf:type :Defective_Assembly_Quality .
}
ORDER BY ?label

```

Figure 5-87: Query to determine which product are not feasible in function of the component and asset state

In the image 5-87 the query shown permits to identify which are the elements inside the ontology that belong to the Defective Cellphone class. In particular this search is executed by looking for the elements that belong to the Bad_Operating_Condition class, the Bad_State_Component class and the Defective_Assembly_Quality class that are connected to the element that belong to the Defective_Cellphone class. So when this query is executed it is possible to determine the Defective Cellphone as well as the Bad_Operating_Condition asset, the Bad_State_Component PCB and the Defective_Assembly_Quality quality, as reported in the images 5-88, 5-89 and 5-90:

Execute	
?product	?quality
:Product6	:Quality6

Figure 5-88: Product and Quality found by the query

?asset	?asset_feature
:Drilling_Machine6	:Drilling_Machine_Feature6

Figure 5-89: Asset and Asset_Feature found by the query

?component	?component_feature
:PCB6	:PCB_Feature6

Figure 5-90: Component and Component_Feature found by the query

After having tested the ontology with the competency questions presented above, it is possible to draw some conclusions. First of all, the internal logic of the ontology shows no signs of inconsistency, since it is possible to launch the internal reasoner of the ontology without reporting errors at the logical level. Additionally, the created ontology can represent the knowledge inside the system, such as, what products are made and what characteristics these products have. In fact it is possible to determine the quality of the product, the manufacturing process used or the components that make up this product. After being tested through competency questions, it is possible to integrate the solution deployed in a Flexible Manufacturing Line (FML) at laboratory scale, to verify if the same result could be obtained in a simulated industrial context.

6 Ontology Implementation

The deployment of the integrated solution is pursued to support the application of a ZDM strategy adoption. For this reason, the ontology is made operative by interacting with the assets and sensors installed in the FML to determine the product quality. In order to achieve such result a set of requirements is defined in order to make the solution effective:

- In order to implement the ontology in the FML it is necessary to gather the data from the assets via proper mean/s, in particular a Golang code, collecting directly from onboard devices (like PLC, Programmable Logic Controller or Raspberry Pi) or from centralized storages (like relational or non-relational databases like InfluxDB).
- The quality of the product must be determined by the ontology in functions of the parameters expressed before.
- The data values collected from the line must be used in the ontology to update the individuals present in the ontology accordingly with the new information coming from the shopfloor, i.e., asset health states, if needed.
- The reasoner needs to be launched, inferring properly if the quality of the products is sufficiently good or not
- The results of the reasoner should be made available to a human decision-maker to promptly let her/him be aware of possible infeasibilities or defects, activating a reconfigurability-related decision.

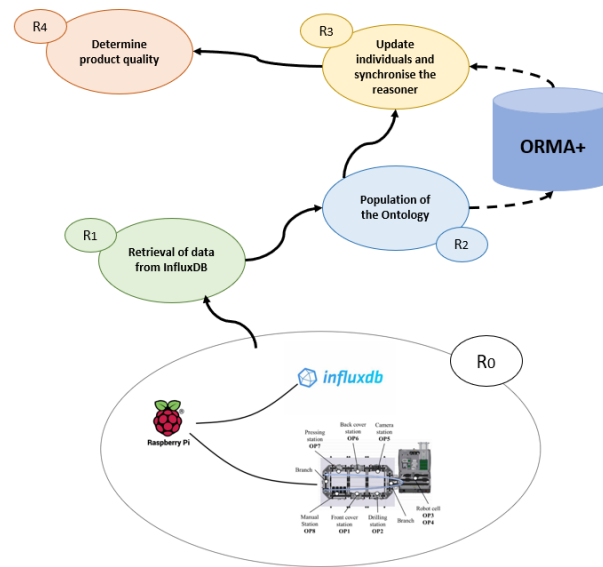


Figure 6-1: Representation of the requirements to determine the product quality

Previously it was stated that the solution deployed relies on Protege as ontology editor, a python code is utilized to synchronize the reasoner and the data collected from the FML in a flexible way via ad-hoc functions and libraries, as OWLREADY2.

In order to meet the first requirement the architecture already present in the laboratory was utilized, as a matter of fact the data has been collected by all the 749 nodes present in the line but only the nodes useful for the development of the solution were kept. In particular only the sensors reporting the passage of the carrier between the stations were kept in order to keep track of the movement of the Product along the line.

For what concerns the second requirement in order to determine the quality of the product it is necessary to introduce some intermediary steps. In particular, in order to determine the quality of the Product it is necessary to collect all the data of the operative process and then utilize these data for the population of the ontology in order to determine the quality of the Product. The quality of the product, as mentioned in the chapter "Validation of the Ontology", is determined by the Asset state and the PCB state, which, in the population, are obtained through two python functions reported in the chapter below, image 6-5 and image 6-6. These functions are executed when the Boolean values correspondent to the arrival of the product to the Drill Station and the Camera Station are detected, thus allowing to determine the quality of the product. In this way the data collected by the FML can be utilized to synchronize the ontology and the shopfloor, since the collected data will be utilized to create individuals in the ontology, which will be updated according to the data collected verifying in this way the third requirement. After the individuals will be created in this way a Hermit reasoner will be utilized in order to determine the quality of the

Product obtained (fourth requirement). The results obtained from the reasoner are made available to an operator allowing him/her to determine the quality of the product and eventually which infeasibilities or defects are present (fifth requirement). The entire architecture is running on a local computer with 16 GB of RAM and an Intel® Core™ i7 of tenth generation.

6.1. Ontology Application

The first step to utilize the ontology in the FML is to collect the data coming from the line, in order to fulfill this activity a Golang code is being utilized to collect all the data coming from the line and send these data to InfluxDB. As soon as all the data correlated to the process are collected on InfluxDB a python code (present in the Appendix B) is executed to retrieve these data and to populate the ontology. As soon as the python code is executed, all the data inside InfluxDB, which have a timestamp inside a specific interval of time, are retrieved but only those data which have a specific NodeID are used to populate the ontology. As a matter of fact only some sensors are going to be considered, since each station of the I4.0 Laboratory has the same embedded digital sensor and actuators in the belt. For this reason each sensor produces a Binary output signal in the form of a logic value that could be only 1 or -1. The sensors that can be found in the belt are:

- Entrance/Exit sensor: is set to 1 if detects a carrier that is entering in a station or is leaving it;
- RFID reading/writing: is the place where the RFID of the carrier is read to know the information about the carrier ID, it writes the type of the piece produced in the associated ID carrier after the work. It is set to 1 if it is busy in the reading or writing activity;
- Position sensor (xBG1): is the sensor used to know if the the carrier is in working position, then is set to 1 if the carrier is there;
- Stopper (MB1): is the mechanism that release the carrier from the working position when the machining is completed, is set to 1 when is activated;
- The belt sensor (xQA A1): it is set to 1 if the belt is moving, 0 if not.

For what concern the modeling of the machine states 'Idle', 'Working Mode' and 'Energy-safe Mode', represented in figure 6-2, it is sufficient to look at the belt of the station, without considering the error at the station:

- Idle: The machine is Idle if the belt is moving, either if there is or not the carrier at the working position;
- Working Mode: The machine is working only if the carrier is in working position and the belt is stopped;
- Energy-Safe Mode: In all the other cases, excluded the error state that will be discussed in the following part, the machine is in Energy-Safe mode, with the belt stopped, no piece in working position and no carrier in the belt.

xQA_A1	xBG1	Machine_State
1	-1	Idle
1	1	Idle
-1	1	Working Mode

Figure 6-2: Representation of the machine states

So only xBG1 values are going to be considered, since this is the node to look for in order to determine if the station is working or is idle. An Exception is the Robot Cell for which there is no xBG1 sensor, so for this station another sensor was considered: the xStart sensor. This sensor indicates when the Robot arm is active and so when the components are added to the front cover.

When the python code is executed, in function of the data collected from the line, the python code will perform the necessary operations to populate the ontology and consequently determine the product quality.

```

lorenzo@ubuntu:~$ cd data_acquisition/
lorenzo@ubuntu:~/data_acquisition$ source bin/activate
(lorenzo@ubuntu:~/data_acquisition) lorenzo@ubuntu:~/data_acquisition$ python3 real_as_monitor.py
* Owlready2 * Warning: optimized Cython parser module 'owlready2_optimized' is not available, defaulting to slower Python implementation
INFO:asynca:Base ontology initialized
INFO:asynca:OPC configuration loaded
INFO:asynca:Local configurations loaded
INFO:asynca:Readed 22715 records.
INFO:asynca:[RobotPLC - 'dIPNo', 210.0, Timestamp('2022-09-01 14:59:00')]
DEBUG:asynca:Using selector: EpollSelector
INFO:asynca:Values from csv loaded
INFO:asynca:Get new Image -> (0, 'TRIANGLE', b'')
INFO:asynca:Get new Image -> (0, 'TRIANGLE', b'')
INFO:asynca:Get new Image -> (0, 'TRIANGLE', b'')
INFO:asynca:Il valore della Immagine era 0
INFO:asynca:Image_value=0, Image_des=TRIANGLE, Image_bytes=b''
INFO:asynca:Get new shacker -> (0, 'SHACKER ON')
INFO:asynca:Get new shacker -> (0, 'SHACKER ON')
INFO:asynca:Get new shacker -> (0, 'SHACKER ON')
DEBUG:asynca:Using selector: EpollSelector
INFO:asynca:Stat_par len= 5
INFO:asynca:Producer created
INFO:asynca:consumer created
INFO:asynca:Producer -> Received FrontCoverPLC xBG1
INFO:asynca:Producer -> Received DrillPLC xBG1
INFO:asynca:Producer -> Received BackCoverPLC xBG1
INFO:asynca:Producer -> Received PressPLC xBG1
INFO:asynca:Producer -> Received RobotPLC xStart
INFO:asynca:Producer -> Received CameraPLC xBG1
INFO:asynca:Producer -> Received ManualPLC xBG1
INFO:asynca:Producer -> Received FrontCoverPLC xBG1
INFO:asynca:Producer -> Received DrillPLC xBG1
INFO:asynca:Producer -> Received ManualPLC xBG1
INFO:asynca:Producer -> Received BackCoverPLC xBG1
INFO:asynca:Producer -> Received CameraPLC xBG1
INFO:asynca:Producer -> Received PressPLC xBG1
INFO:asynca:Producer -> Received RobotPLC xStart
INFO:asynca:Producer -> Received RobotPLC xStart
INFO:asynca:Producer -> Received BackCoverPLC xBG1
INFO:asynca:Producer -> Received FrontCoverPLC xBG1
INFO:asynca:Producer -> Received RobotPLC xStart
INFO:asynca:Producer -> Received DrillPLC xBG1
INFO:asynca:Producer -> Received ManualPLC xBG1
INFO:asynca:Producer -> Received CameraPLC xBG1

```

Figure 6-3: Start of the python code

In the image 6-3 the execution of the python code is shown and it is possible to see from the image that the code proceeds to elaborate all the data coming from the sensors and once the population process is finished, the ontology is updated with the individuals created by the code. The new ontology is saved locally and so it is possible to inspect the results obtained after the population, as showed in the image 6-4.

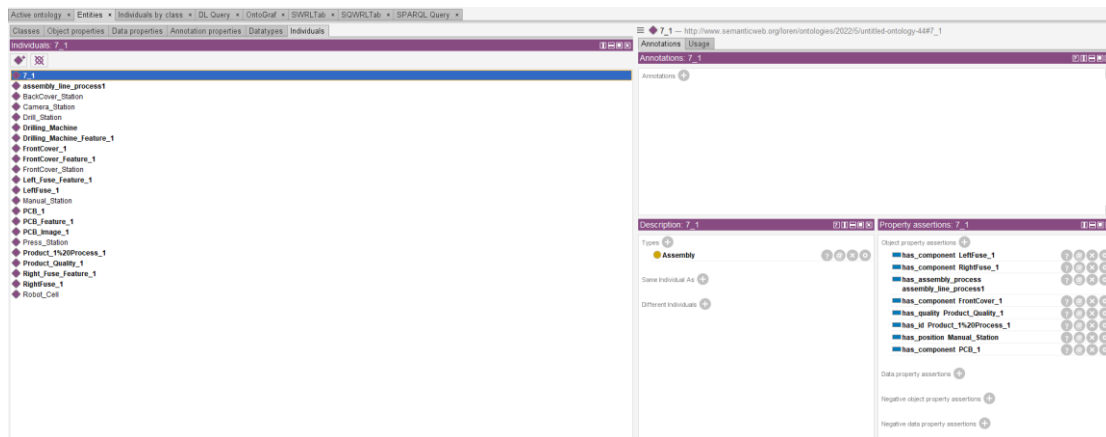


Figure 6-4: Representation of the Individual created with the population

Even though this application of the ontology has potential there are still some limitations which have to be exploited, as a matter of fact to determine the quality of the Product it is necessary to determine if the shaker in the drilling station is active and what is the quality of the Components when they arrive at the Camera Station. However during the elaboration of this thesis the Camera Station was not working and the data corresponding to the activation of the shaker were not individuated and so

Overall, the solution proposed guarantees adequate computational performance, but there are some considerations to make. First of all this process is not a real-time process, as, while the data are retrieved from the sensors it was not possible to execute the python code responsible to read the data from InfluxDB. This is due to the fact that during the data collection the database is continuously refreshed with the new data and so it is not possible for the code to retrieve the data. For this reason the following solution was adopted, all the data contained in InfluxDB are converted in a CSV file, as soon as the process finishes, which is then read by the python code. This solution has to deal also with another aspect concerning the production process: in the FML there are not the necessary instruments to cover the entire process steps that the ontology can perform, including the repair and disassembly process. For this reason only a subset of the potentiality of the ontology can be exploited, since it is not possible to operate on the product in real time. In this way even if the ontology recognizes that the quality of the components is not sufficiently good, in the current physical workplace there is no way to stop the process and so a final product will still be obtained but it will need to be discarded. For these reasons one possible solution can be the creation of an ad hoc CSV file and a new python code which can be utilized to show all the potentialities of the ontology.

6.2. Ontology population: Repair and Disassembly Stations

Accordingly to the limitation of the previous population a new population has been performed without using the CSV file obtained from the FML, but using a CSV file which has been created for this purpose by the author of this thesis work in order to show the potentiality of the ontology. In fact, the ontology developed in this thesis work is already able to operate in real time to manage the repair and disassembly process of an eventual product. Instead the FML in the Laboratory of Industry 4.0 at Politecnico di Milano has not the instruments to do so. For this reason a new code and CSV file have been developed to simulate the behavior of an hypothetical Repair and Disassembly station present in the FML.

The cases that will be presented are the case of a Repairable Cellphone and Defective Cellphone since for the Dummy Cellphone there are no significative differences to be reported (with respect to what already reported in the previous chapter 5).

6.2.1. Repairable Cellphone case:

Concerning the *Repairable Cellphone* case, respect to the previous population, now it is possible to see how the system behaves in the case there is one or more components to

be repaired. In this case it is going to be presented a situation in which the *FrontCover* and the *PCB* need to be repaired.

The first element to analyze is what is obtained after the product pass through the Drilling Station. As a matter of fact the result reported in the ontology are the ones of figure 6-7 and 6-8.

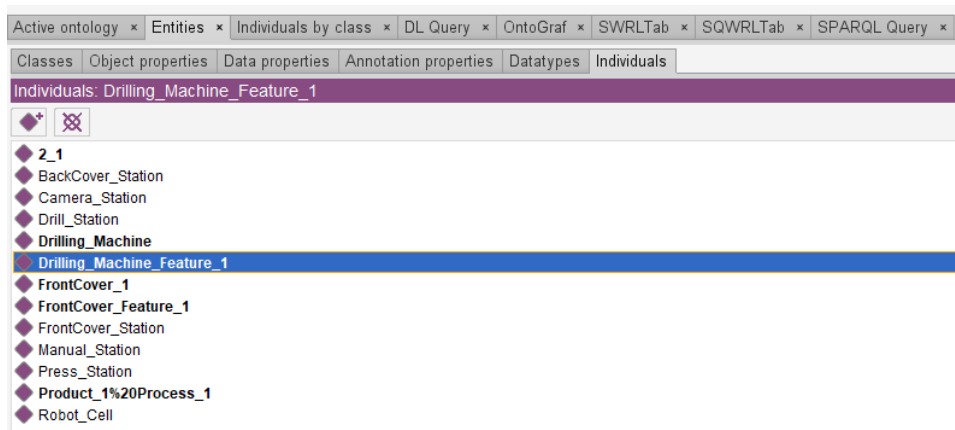


Figure 6-7: Representation of the individual *Drilling_Machine_Feature_1*

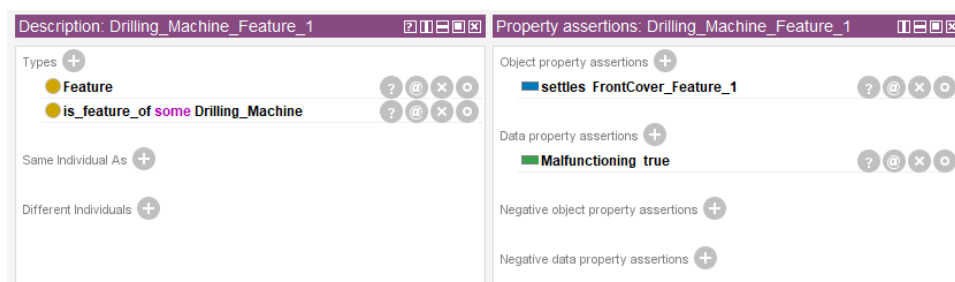


Figure 6-8: Representation of the Property assertion of the *Drilling_Machine_Feature_1*

When the product goes to the next station, the Robot Cell, the reasoner is activated, image 6-9, and the results obtained from the reasoning are showed in the images from 6-10 to 6-13. In the images 6-10 and 6-11 the *FrontCover_Feature* is represented and it is possible to see that the reasoner recognizes the front cover as a repairable component, instead in the images 6-12 and 6-13 the *Drilling_Machine_Feature* is represented and it is possible to verify that the *Asset* has a *Bad_Operating_Condition*.

```
INFO:asyncua:Ontologia_Test_Python.BackCover_Station, Ontologia_Test_Python.Camera_Station, Ontologia_Test_Python.Drill_Station, Ontologia_Test_Python.FrontCover_Station, Ontologia_Test_Python.Manual_Station, Ontologia_Test_Python.Press_Station, Ontologia_Test_Python.Robot_Cell, Ontologia_Test_Python.i_1, Ontologia_Test_Python.Product_1_Process_1, Ontologia_Test_Python.FrontCover_1, Ontologia_Test_Python.FrontCover_Feature_1]
* Owlready2 * Running Hermit...
  java -Xmx2000M -cp /home/lorenzo/data_acquisition/lib/python3.8/site-packages/owlready2/hermit:/home/lorenzo/data_acquisition/lib/python3.8/site-packages/owlready2/hermit/Hermit.jar org.semanticweb.Hermit.cli.CommandLine -c -O -D -I file:///tmp/tnpqwz7k
* Owlready2 * Hermit took 3.185425281524658 seconds
* Owlready * (NB: only changes on entitles loaded in Python are shown, other changes are done but not listed)
INFO:asyncua:Consumer -> Consumed FrontCoverPLC xBG1
WARNING:asyncio:Executing <Task pending name='Task-5' coro=<consumer() running at as_monitor.py:1124> wait_for=<Future pending cb=[<TaskWakeupMethWrapper object at 0x7f114dec2559(0)] created at /usr/lib/python3.8/asyncio/base_events.py:422> created at /usr/lib/python3.8/asyncio/tasks.py:382> took 3.261 seconds
INFO:asyncua:Producer -> Received DrillPLC xBG1
INFO:asyncua:Consumer after get Station=DrillPLC
INFO:asyncua:Consumer after get Node=xBG1
```

Figure 6-9: Activation of the reasoner of the Python code

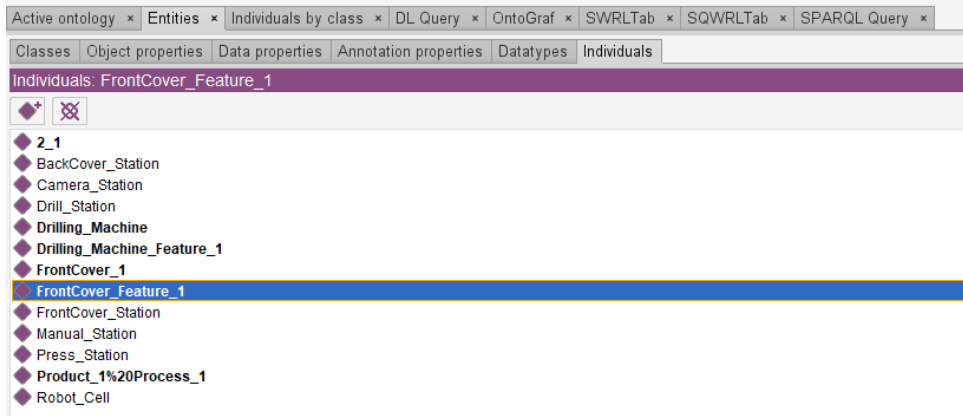


Figure 6-10: Representation of the individual FrontCover_Feature_1

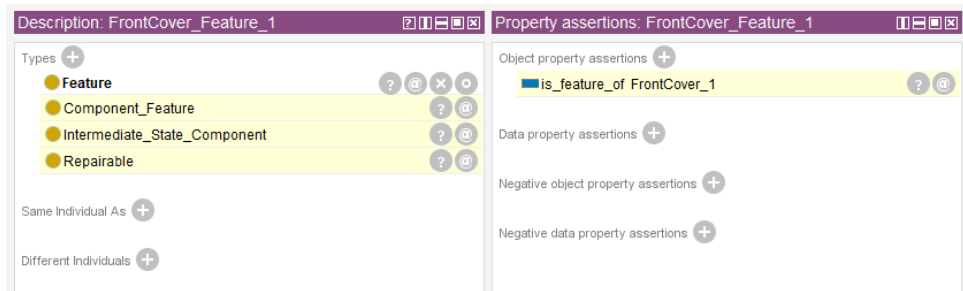


Figure 6-11: Representation of the Property assertion of the FrontCover_Feature_1

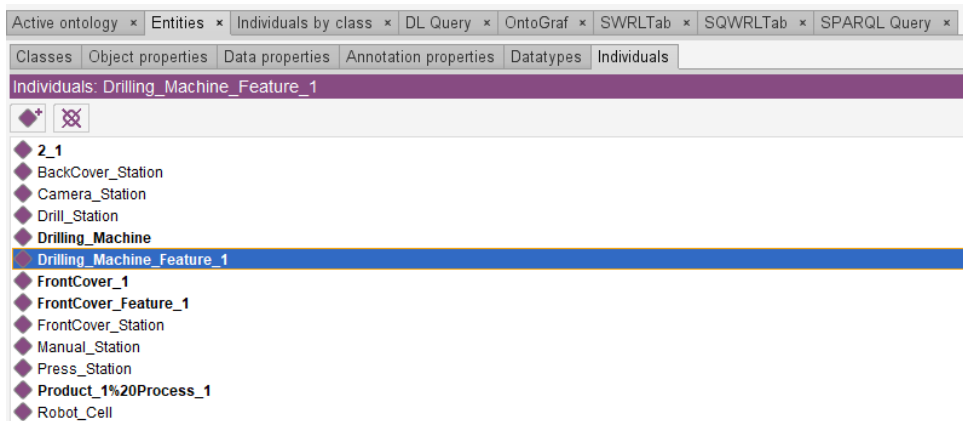


Figure 6-12: Representation of the individual Drilling_Machine_Feature_1

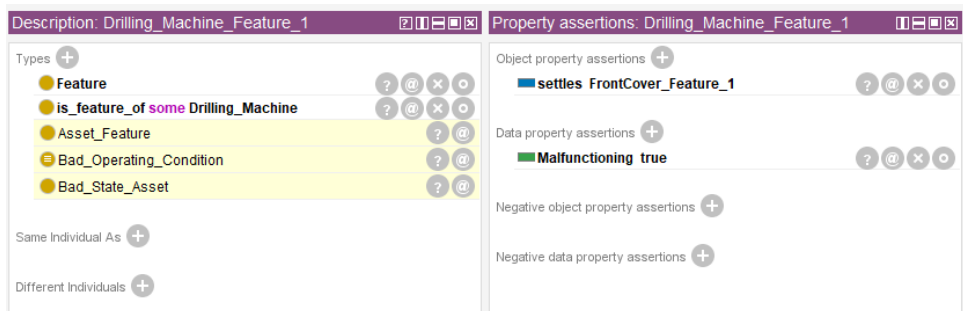


Figure 6-13: Representation of the Property assertion of the Drilling_Machine_Feature_1

When the front cover is recognized as *Repairable*, image 6-11, it is possible to provide to the repair of the component. As a matter of fact the code provides a way to send the component to the Repair Station, a dummy station, located outside of the line, that has been inserted into the code in order to simulate the component repair process, as it is possible to see from the image 6-14.

```
INFO:asyncua:Ontologia_Test_Python.Right_Fuse_Feature_1
INFO:asyncua:The PCB and fuses are inserted at time 2022-09-12 10:12:53.915616
INFO:asyncua:[Ontologia_Test_Python.PCB_1, Ontologia_Test_Python.LeftFuse_1, Ontologia_Test_Python.RightFuse_1]
INFO:asyncua:Consumer -> Rep Producer activated -> Received RepairStation RepairNode FrontCover_1
INFO:asyncua:Consumer -> Rep Producer sended -> Received RepairStation RepairNode FrontCover_1
INFO:asyncua:[Ontologia_Test_Python.BackCover_Station, Ontologia_Test_Python.Camera_Station, Ontologia_Test_Pytho
```

Figure 6-14: Representation of the Repair Station

After the component is sent to the Repair Station, a repair process is simulated where the *Repairable* feature is removed from the component and after a few seconds (since the objective is to simulate the routing, but not the repair time) the *FrontCover* is re-introduced in the line. When the *FrontCover* is re-introduced a new feature is given to the front cover, the *Conform* feature since now the *FrontCover* is repaired. So the result obtained after the repair process are shown in the images 6-15 and 6-16, where the new feature of the front cover is represented.

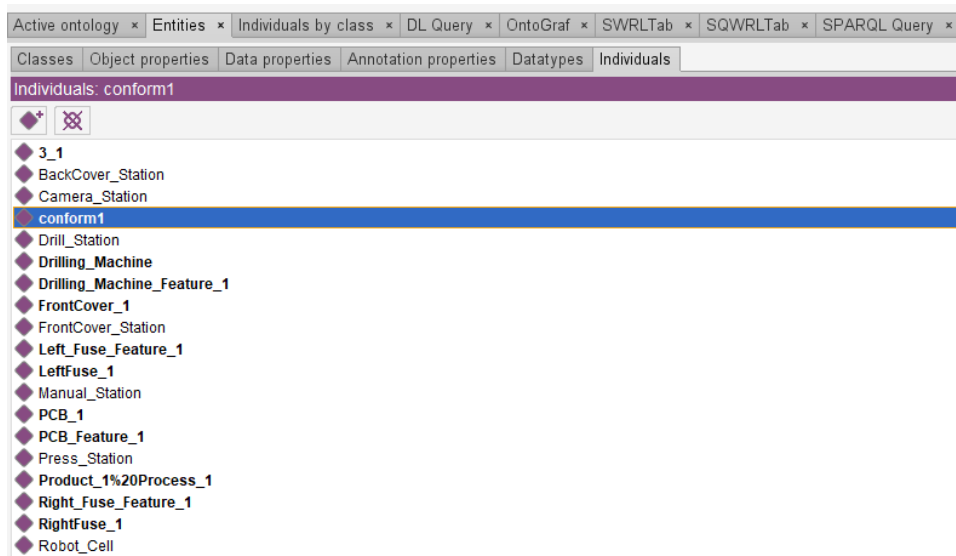


Figure 6-15: Representation of the individual conform_1

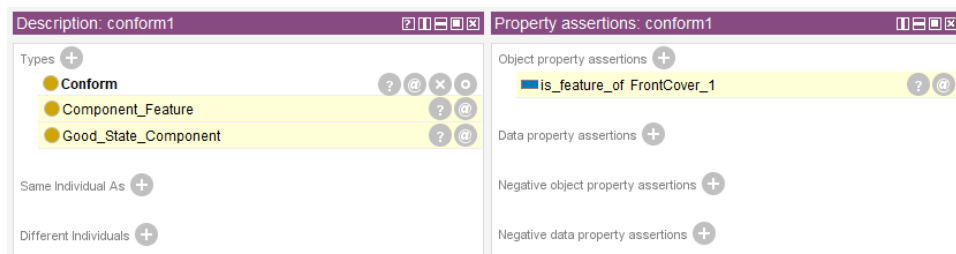


Figure 6-16: Representation of the Property assertion of the conform_1

In the case the *PCB* needs to be repaired the same situation is going to be verified. The results obtained for the repair of the *PCB* are presented in the images from 6-17 to 6-

20. In the images 6-17 and 6-18 the situation before the repair process is represented, while in the images 6-19 and 6-20 the situation before the repair process is represented.

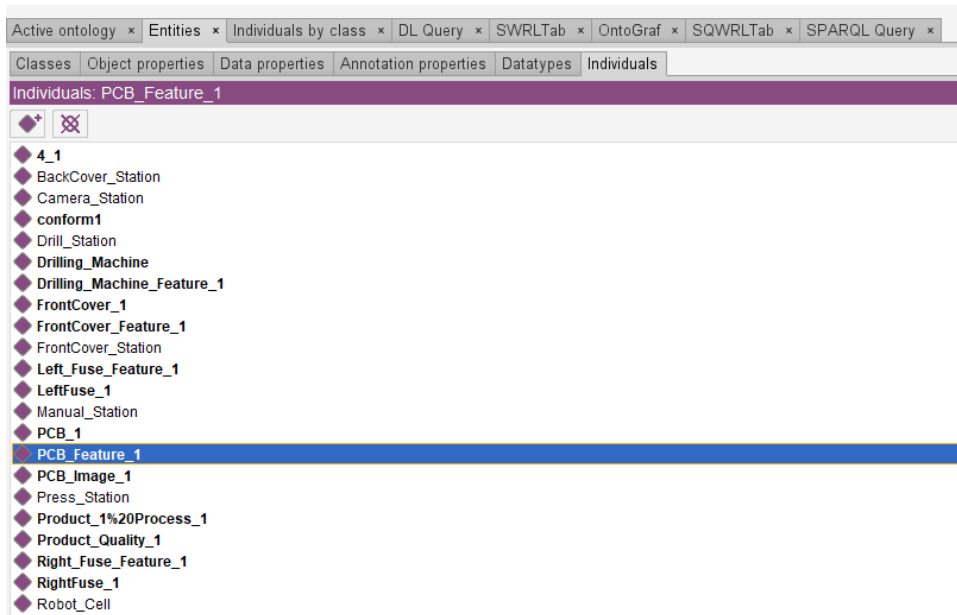


Figure 6-17: Representation of the individual PCB_Feature_1

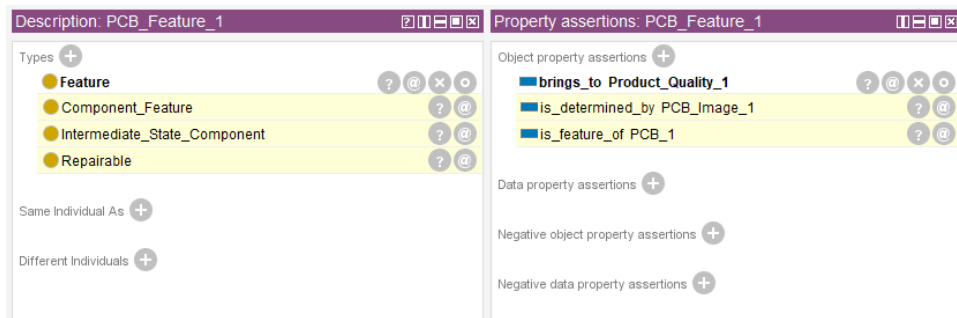


Figure 6-18: Representation of the Property assertion of the PCB_Feature_1

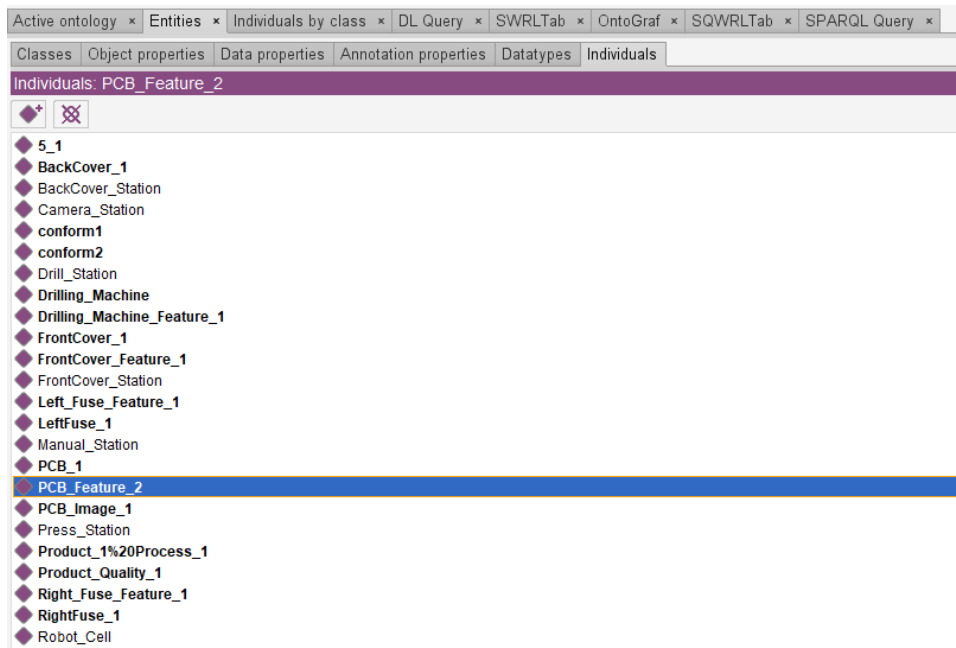


Figure 6-19: Representation of the individual PCB_Feature_2

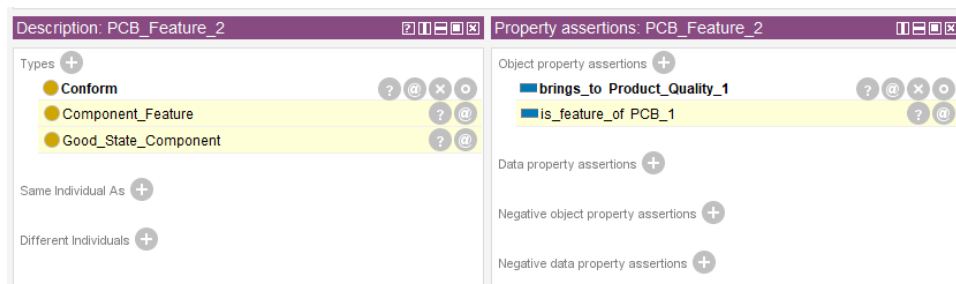


Figure 6-20: Representation of the Property assertion of the PCB_Feature_2 with the reasoner active

Now that the Repair strategy has been analyzed it is necessary to move to the *Defective Cellphone* case in order to see how the population of the ontology changes when it is necessary to apply the disassembly strategy.

6.2.2. Defective Cellphone case

In the *Defective Cellphone* case a disassembly strategy is applied to retrieve those components which have a sufficient good quality to be used for the production of others products, while defective components are discarded.

In order to apply the disassembly strategy two production processes have to be considered: a first production process in which the product obtained is defective and so it has to be discarded, and a second production process in which the good components of the first product are re-used for the second one. Therefore two production processes have been executed and the results of the first one are represented in the images from 6-21 to 6-30. In the images 6-21 and 6-22 the first

Product is represented and, as it is possible to see from the images, it is a *Defective Cellphone*.

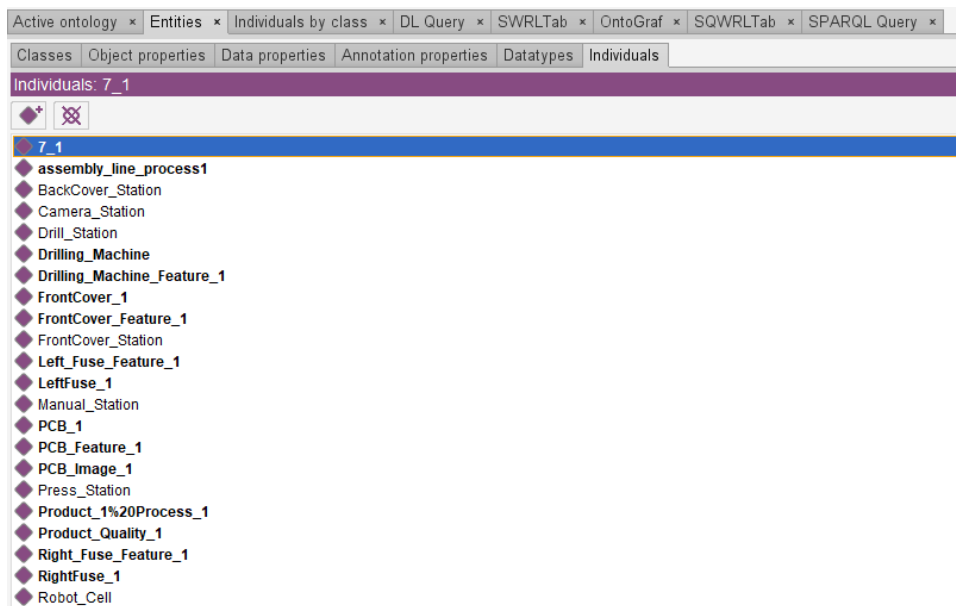


Figure 6-21: Representation of the individual 7_1

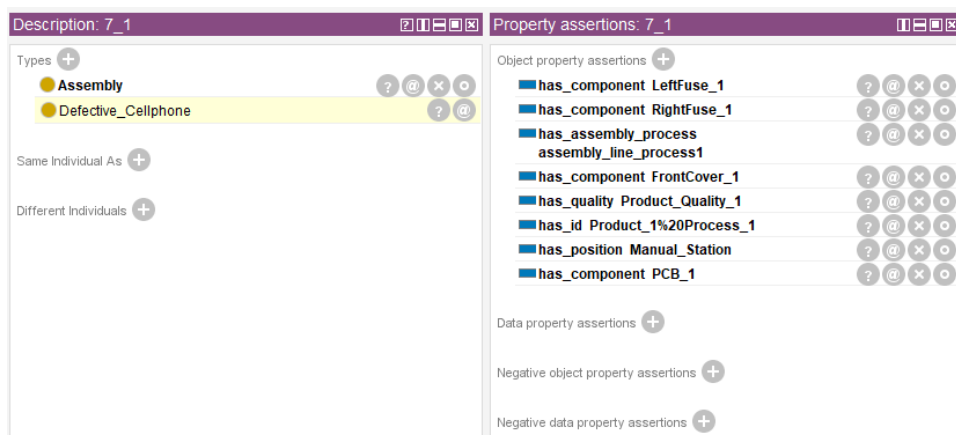


Figure 6-22: Representation of the Property assertion of the 7_1 with the reasoner active

In the images from 6-23 to 6-30 all the components of the Product are analyzed and it is checked the state of each component. In the images 6-23 and 6-24 the *PCB* is checked and it is possible to see that it is a defective component and so it is *not Repairable* and *not Recyclable*. In the images 6-25 and 6-26 the state of the *FrontCover* is checked and it is verified that the front cover is *Recyclable* and so it can be used for the production of a second product. In the images from 6-27 to 6-30 the condition of the *fuses* are checked and it is possible to verify that these components are *Recyclable*.

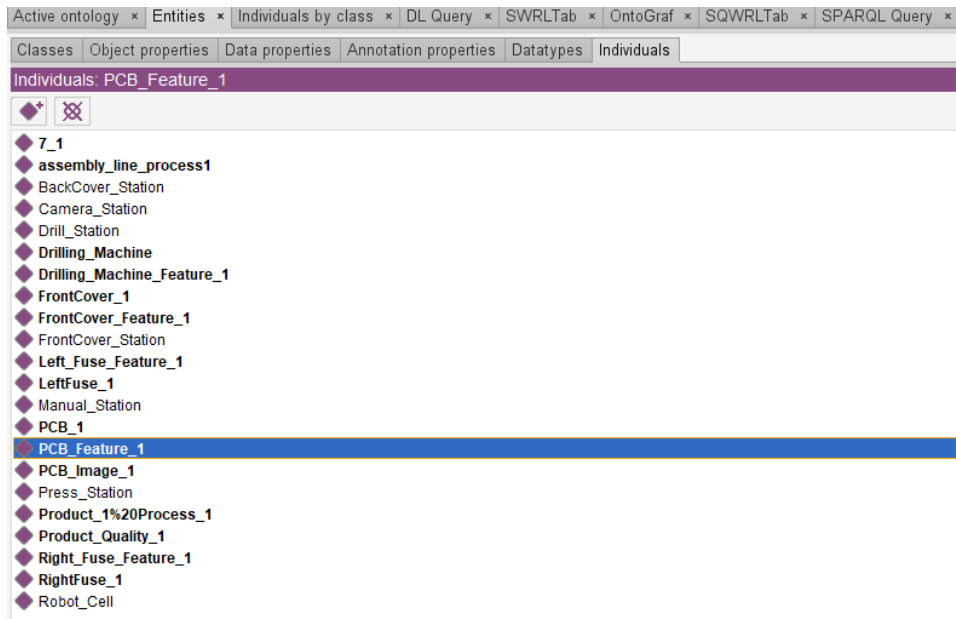


Figure 6-23: Representation of the individual PCB_Feature_1

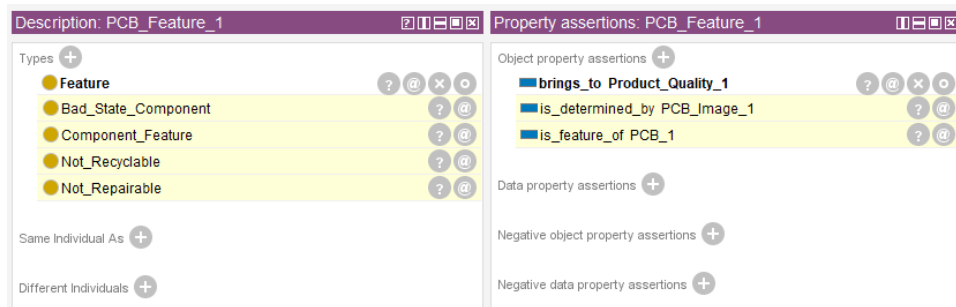


Figure 6-24: Representation of the Property assertion of the PCB_Feature_1 with the reasoner active

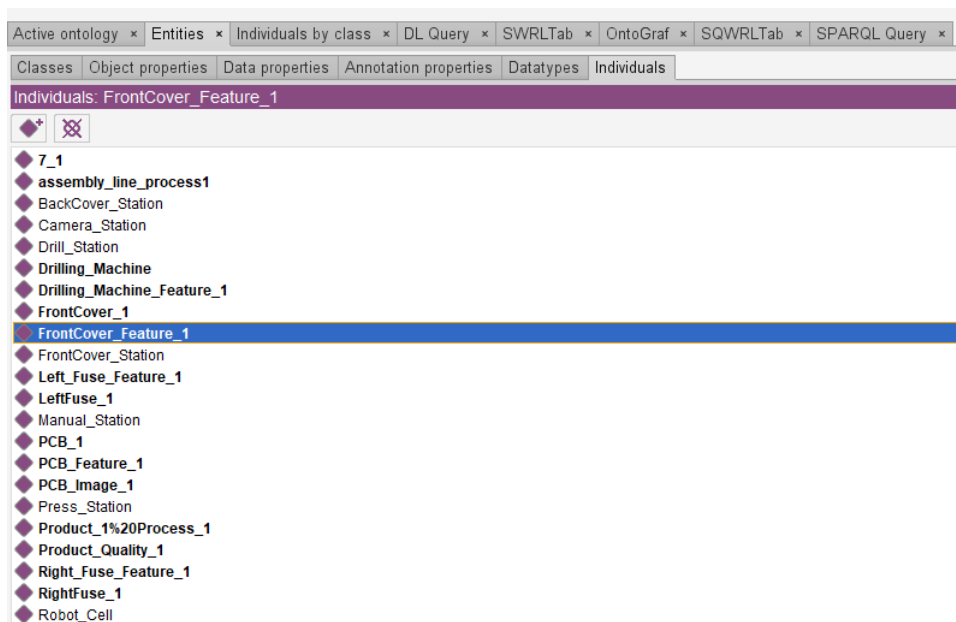


Figure 6-25: Representation of the individual FrontCover_Feature_1

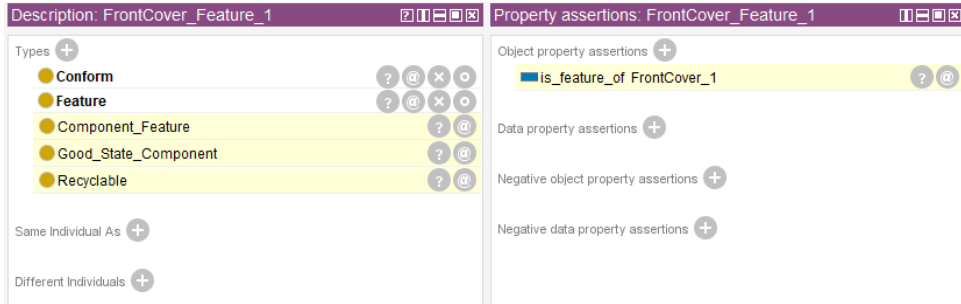


Figure 6-26: Representation of the Property assertion of the FrontCover_Feature_1

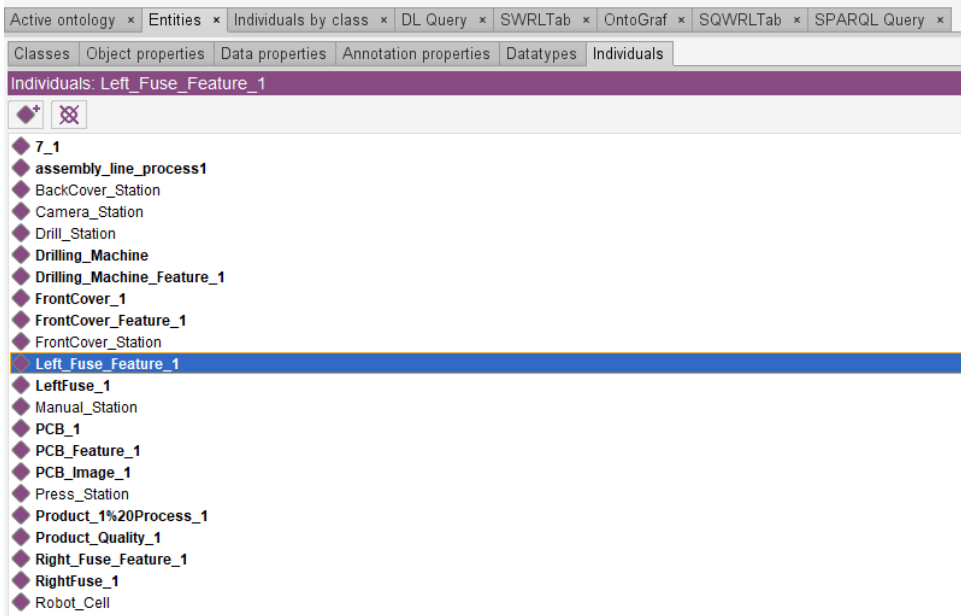


Figure 6-27: Representation of the individual Left_Fuse_Feature_1

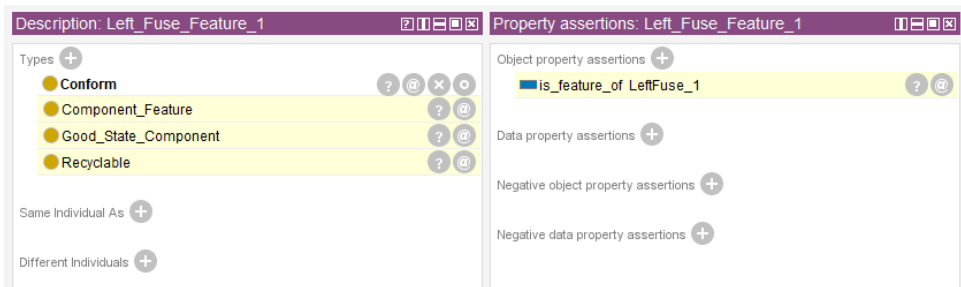


Figure 6-28: Representation of the Property assertion of the Left_Fuse_Feature_1

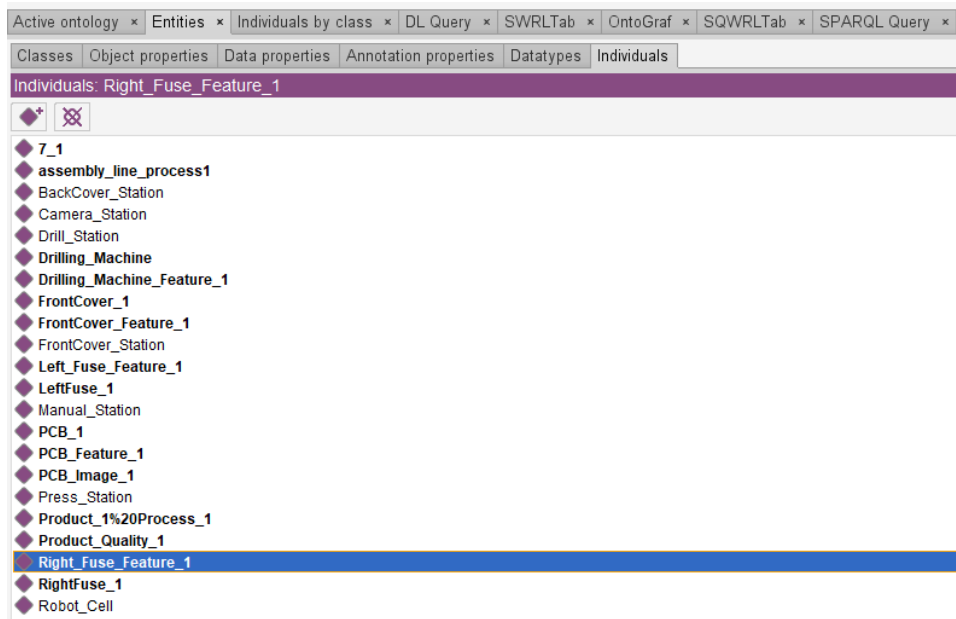


Figure 6-29: Representation of the individual Right_Fuse_Feature_1

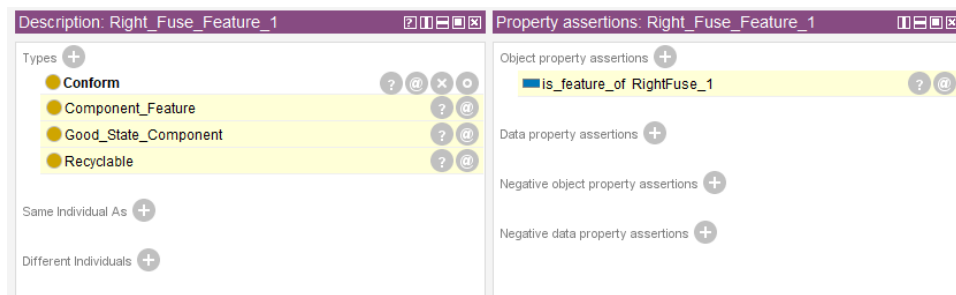


Figure 6-30: Representation of the Property assertion of the Right_Fuse_Feature_1

After the ontology recognizes the Recyclable components a second production process is executed in order to verify if the ontology is able to associate to the new product the recycled components. The result of the second production process is shown in the images 6-31 and 6-32.

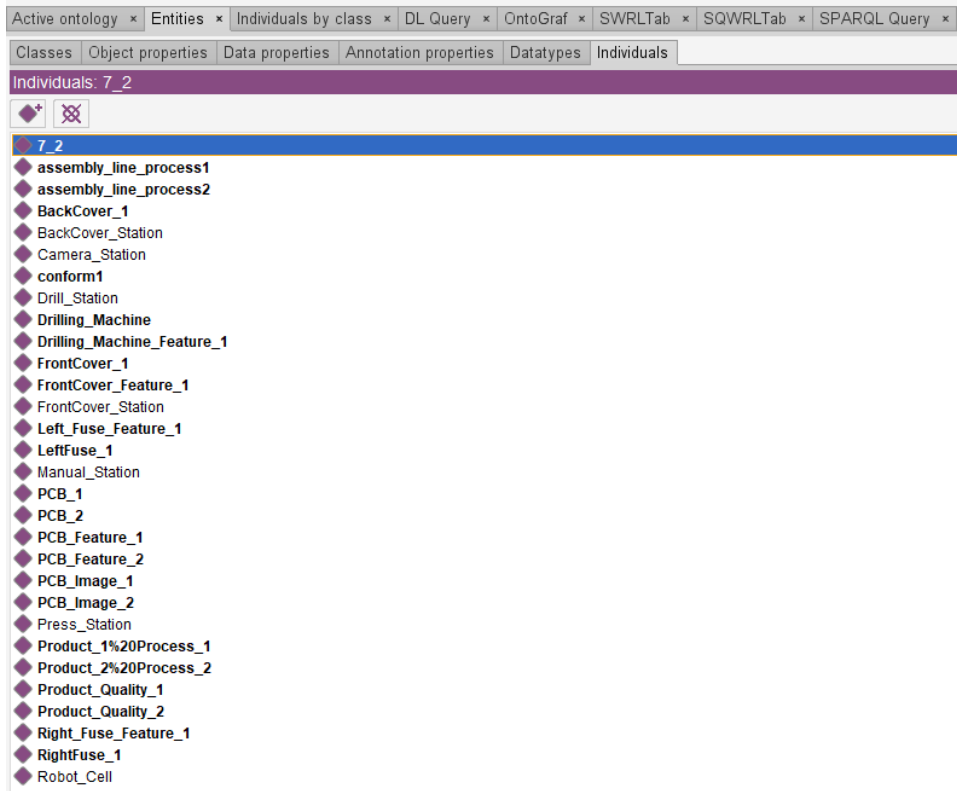


Figure 6-31: Representation of the individual 7_2

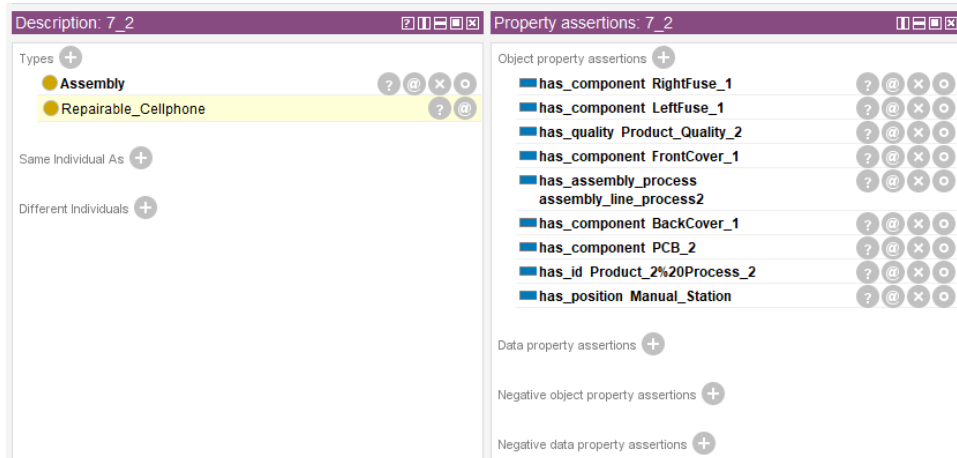


Figure 6-32: Representation of the Property assertion of the individual 7_2 with the reasoner active

As it is possible to see from these images the components which were present for the product before, which were in good state, have been utilized for this product, contrarily to the *PCB* which has been substituted with a new component and the *BackCover* that the previous Product did not have.

7 Result and discussion

In this chapter the results obtained from the validation and the population of the ontology will be discussed. In particular the analysis will concern on the different scenarios presented in the validation of the ontology and the results obtained after the two populations of the ontology.

7.1. Results: Ontology Validation

The validation of the ontology has been executed in order to determine if the logic of the ORMA+ ontology was correct. To verify if the logic inside the ontology was correct three different scenarios were identified: one related to the creation of a *Dummy Cellphone*, one related to the creation of a *Repairable Cellphone* and one related to the creation of a *Defective Cellphone*. These 3 cases are reported below:

- Scenario 1: **Creation of a Dummy Cellphone.** In this scenario a production process has been executed considering a *Good_Operating_State Asset* and a *Product* formed by *Good_State_Components*.
- Scenario 2: **Creation of a Repairable Cellphone.** In this scenario a production process has been executed considering a *Bad_Operating_State Asset* and a *Product* with an *Intermediate_State PCB*.
- Scenario 3: **Creation of a Defective Cellphone.** In this last scenario a production process has been executed considering a *Bad_Operating_State Asset* and a *Product* with a *Bad_State PCB*.

The results obtained from these scenarios enable to make the following consideration:

- The goal of the ORMA + ontology is to help any operator to act accordingly to the results obtained from the ontology. As a matter of fact by running the reasoner when the product arrives at the Manual Station, it is possible to determine the condition of the drilling machine and the state of the components that make up the product, determining, in this way, the quality of the produced product. Thanks this tool it is therefore possible to help/improve decision making by aiding an eventual operator to make the correct decisions, finally and avoiding the delivery of defective products to an eventual customer.

Now that this consideration has been made it is necessary to move on to the part dedicated to the population of the ontology as it represents the practical application of the ontology.

7.2. Results: Population of the ontology

As seen in the chapter “Ontology Implementation” two different population have been executed: one with the data obtained from the line and a second one which simulates a line with the necessary instruments to implement specific actions such as repairing or disassembly.

Starting with the population of the ontology with the data obtained from the line it is necessary to make the following consideration:

- The main outcome it is possible to derive from the first population is that the proposed solution can be implemented in a real industrial context. As a matter of fact the ontology is able to interpret and analyze the data coming from the servers of the FML to obtain the results seen during the ontology validation phase, like the quality of the product or the components that constitutes the product and so on.

Concerning the second population, this was executed with the aim to show all the potentialities of the ontology in the case the line was characterized by the needed instruments to perform a real time operation on the product. So respect to the previous case the following consideration can be made:

- In this case the proposed solution has shown that the ontology is able to execute some real time complex reasonings like the repair or the disassembly of a product. In fact the ontology is able to determine if a product needs to be sent to a repair station, therefore, stopping the production process and waiting for the part to be repaired and reinserted into the line before continuing the production process. At the same time, the ontology is also able to determine if the product obtained is defective and so if it is necessary to execute the disassembly of the product, keeping only those components whose quality is sufficiently good to be re-used for another production process.

7.3. Potentiality of the ORMA+

The application of the ontology proposed in a small scale manufacturing environment like Industry 4.0 Laboratory of Politecnico di Milano, gives a relevant validation of the model itself.

The aim of the ORMA+ ontology is to provide an instrument which can help an eventual operator to improve decision making. It has been, also, proved during the population of ORMA+ that the ontology can have an industrial application. As a matter of fact, in the ontology the ZDM strategy of Detection and Repair have been introduced in order to make the ontology more extended, innovative and complete, introducing a triggering factor and triggering action to deal with specific situations like the repair of one or more components. It is worth also observing that a recycling

activity is also integrated and supported by the ontology through the disassembly; this increases the potentiality in the innovative aspect of the ontology, suggesting a further step beyond the ZDM strategy in its pure definition as found in literature, toward a “sustainable” ZDM strategy.

Furthermore, as mentioned in the Literature Review chapter, in order to make the user of the ORMA+ ontology free to choose which process works the best in the given industrial context, both a process and product oriented approach have been added in the ORMA+ ontology. In fact, in the proposed solution a process based approach is utilized to determine the status of an asset, the drilling machine, in function of the data collected by the FML. The product based approach, instead, consists in determining the state of the PCB, one of the components of the product, in function of what has been observed by the camera of the Camera Station.

Overall, ontologies are expected to play a significant role in the future for several manufacturing industries, given that data, information, and knowledge are expected to continuously rise up and their management and integration is assumed to be more and more a challenging activity. Indeed, the role of an Ontology like the one developed in this work is to increase, through semantic capabilities, the cognitive ability, and this will be an essential feature in the future of manufacturing systems. Therefore, it can be asserted that, with this thesis work, the exploration of the Cognitive Digital Twin has been initially launched. To this last regard, it is worth remarking that the “traditional” understanding of a Digital Twin as a mirroring element of the reality (in this case, the reality is the physical product) is inherently embedded in the solution developed so far in the thesis; a step further, enabled by the same provided solution, is built on the reasoning enabled by the Ontology, this helps to provide the cognitive ability expected in the visionary definitions of a Cognitive Digital Twin.

8 Conclusions

The thesis work carried out was aimed at creating a tool that could be used within an industrial context in order to help an operator make the right choices based on the reasonings produced by the ontology and consequently by the Digital Twin of the product. In order to obtain this solution a Systematic Literature Review has been performed to determine the gaps present in the Literature to motivate this thesis work. Based on the gaps found a Research Design has been carried out to determine the objectives of this thesis work. These objectives have been fulfilled by the ORMA+ ontology in the validation and population phase of this thesis work, where the intrinsic logic of the ontology and the implementation of the ontology in a laboratory scale have been tested. Therefore the work carried out, through the conjunction between the ontology containing the ZDM strategies and the modeled digital twin of the product obtained from the line at the Laboratory, finds an implementable way to monitor the line itself, providing an adequate aid to an eventual operator, thus reducing the possibility of committing any mistakes in the direction of ZDM.

In this Chapter, an analysis of the research contribution and the practice contribution that the proposed solution give is going to be treated. These tasks are carried out by explaining how the implementation of the solution of this thesis work may open the way to providing the exact validated application cases that are necessary to fill the research gaps as well as an explanation of the practical outcomes of the proposed solution. Furthermore, an overview of the limitations of the performed work together with the possible future researches to enrich the solution proposed will be provided.

8.1. Research Contribution

The proposed solution of this thesis work can fulfill the gaps that have been seen in the Literature Review in order to expand the use of ontologies for ZDM related applications. The gaps found in the Literature are addressed by this thesis work in the following way:

- **Main Objective: Creation of a Digital Twin using an Ontology in order to enable a cognitive capability to apply a Detection and Repair strategy to finally pursue the objective of a ZDM.**

- The ontology realized permits to develop a Digital Twin of the product obtained by the FML and, when certain conditions are met, to apply the ZDM strategies of Detect and Repair. As a matter of fact the Detection strategy is applied to determine the state of the product's component or the state of the Asset. In this way it is possible to verify if it is necessary to adopt a Repair strategy or, if the product is defective, if it is possible to disassemble the product in order to retrieve the recyclable components. Thanks to the use of these two strategies, it is therefore possible to validate the proposed solution and to avoid the delivery of defective products to an eventual customer, since it is possible to determine the quality of the product and help the operator to make the correct decisions.

For what concerns the other sub objectives instead:

- ***Secondary Objective 1: In the Literature it is shown how the Repair strategy of the ZDM is not frequently utilized as, in order to use that strategy, is necessary to find a good trade-off between the utility of the strategy and the cost and time that the introduction of that strategy may determine. The proposed solution may then consists in the combined use of the Detect and the Repair strategies to try to solve this trade-off.***
- In this thesis work both these strategies are used together. The Detect strategy was used since this strategy is the most documented and used in the literature, therefore the use of this strategy makes the ontology conform to what was found in the literature. At the same time, however, given the versatile nature of the ontology, the Repair strategy was also introduced in the proposed solution in order to make the ontology more extended, innovative and complete, exploiting a combination of triggering factor, the detection strategy and the triggering action, the Repair strategy. After the validation of the ontology there is no way to tell if this combination helped solving this trade-off analysis, but it has been demonstrated the possibility to integrate the Repair strategy in the entire approach to try to solve this issue.
- ***Secondary Objective 2: It has been observed that in the literature there are several papers in which an application of a ZDM strategy was presented in different sectors, this may make it difficult to understand which tools / techniques can be used and the requirements necessary to obtain a ZDM in a more general sense. To this end the proposed solution may then include a***

standardization of the ZDM terms and definitions, helpful in a sectoral independent definition of ZDM-related concepts.

- The use of an ontology has proven to be fundamental since the ontology in itself involves the introduction of a standardization of the terms and the introduction of a series of definitions, which can make this proposed solution also applicable to other contexts. In fact, one of the key points of the ontology is the Knowledge Reuse; for this reason it is possible to use the knowledge present in the ontology, about ZDM, to construct a new specific solution for any new context.
- ***Sub Objective 3: In the eligible articles, no unified procedure for data collection, management and elaboration is provided in a unified framework. So it can be interesting to provide a solution able to provide such structure.***
- In order to cope with this aspect the solution proposed has been validated in a simulated environment where the population of the ontology is realized with the data obtained by the production line (the FML of the Lab) under analysis and, accordingly to the data collected by the ontology, it is possible to determine the quality of the realized product.
- ***Sub Objective 4: In the eligible articles the approach usually utilized is a product based approach since it is easier to implement due to its nature. For this reason this objective advances the possibility to utilize a process oriented methodology in the proposed solution.***
- In the proposed solution a combination of product and process based approach is utilized. In fact the proposed solution consists in determining the status of an asset, the drilling machine, following the data collected by the FML and then determining the quality of the product made accordingly to the status of the Asset and the state of the PCB, one of the components of the product, in function of what has been observed by the camera of the Camera Station. This choice was done since accordingly to [48] one of the two process can be more or less effective in function of the industrial context of application, for this reason the proposed tool do not impose a specific approach but it allows to adopt the way that best suits the specific context.

8.2. Practice Contribution

Based on what was previously said in the chapter dedicated to the Research Contribution, it was possible to identify a series of shortcomings that led to the definition of some practical objectives that must be fulfilled by the solution found and validated in the Industry 4.0 Laboratory of Politecnico di Milano.

Many of the activities carried out in the Laboratory were aimed at creating a tool whose goal was to model an ontology that would allow the creation of a digital twin of the product obtained following a FML cycle. For this purpose, a work previously performed in the Laboratory based on the creation of an ontology (the ORMA ontology) was taken as a model [23], which had the purpose of monitoring the status of one of the assets present on the Flexible Manufacturing Line, the Drilling Machine. With the use of a set of know-hows such as the BFO [49][69], IAO [70] and CCO [71] ontology and the ORMA ontology [23] it was possible to realize the ORMA + ontology which was implemented and validated in order to determine the quality of the products obtained from the line.

The following practical results have been obtained:

- The realized ontology is able to interpret the data coming from the servers contained in the MLF through the use of a python code which puts the ontology in contact with a local database (InfluxDB) and also allows to use these data in order to perform the population of the ontology.
- Once created, the ontology allows you to determine the quality of the product made by the line using the data provided by it and allows you to identify and classify the characteristics of the product such as its components and their status or the production process which was used.

8.3. Limitations of the current work

The current thesis work has some limitations on the ontological modelling side as well as on the technological deployment side.

As a matter of fact, the ORMA+ ontology, as the ORMA ontology [23], could not manage multiple cycles a product may have. Therefore, ORMA is limited in managing the knowledge in situations where a product may be realized by different machines of the same type in the same working step.

From the technological deployment side, the Digital Twin obtained with this thesis work is a digital copy of the product which requires the presence of an operator who

must make decisions based on the results shown by the ontology. The possibility of giving the ontology the ability to choose which is the best (optimal) action to perform based on results obtained from the ontology was not considered in this work, given the lack of the necessary instruments on the line to perform such operations. Accordingly to this the Repair and Disassembly actions theorized in the validation of the ontology have been just introduced as proof of concept since in the line there is not the possibility to apply these actions.

8.4. Future Researches

Overall this work of thesis with its ability to realize a solution able to determine a Digital Twin with the support of an ontology paves the way to other future researches.

As a matter of fact it has been said throughout the entire thesis work how the solution proposed can be made more complete, if the FML had the necessary actuators to fully exploit the potentialities of the ontology. So one of the possible future application may concern how the proposed solution may work in an industrial context with the needed instruments in order to realize a digital twin able to take decision on its own in real time. A second future development may be the introduction in the proposed solutions of the missing Zero Defect Manufacturing strategies in order to make the solution more complete. Another possibility can be the introduction of other possible monitoring processes, as a matter of fact in the current solution only the Boolean and Energy values coming from the sensors are evaluated. However in an eventual industrial context there may be several other data which may be used in order to perform other monitoring systems, like for example the monitoring of the energetic anomalies or the monitoring of the times taken in order to perform the needed operations. These new kinds of monitoring could be integrated; at the same time, it can be expected that they should be based on the specific domains / sectors of application, for which appropriate knowledge has to be considered to finally proceed with the integration.

9 Bibliography

- [1] S. Ing Tay *et al.*, “An Overview of Industry 4.0: Definition, Components, and Government Initiatives,” 2018. [Online]. Available: <https://www.researchgate.net/publication/332440369>
- [2] B. Caiazzo, M. di Nardo, T. Murino, A. Petrillo, G. Piccirillo, and S. Santini, “Towards Zero Defect Manufacturing paradigm: A review of the state-of-the-art methods and open challenges,” *Computers in Industry*, vol. 134. Elsevier B.V., Jan. 01, 2022. doi: 10.1016/j.compind.2021.103548.
- [3] I. T. Christou, N. Kefalakis, J. K. Soldatos, and A. M. Despotopoulou, “End-to-end industrial IoT platform for Quality 4.0 applications,” *Comput Ind*, vol. 137, May 2022, doi: 10.1016/j.compind.2021.103591.
- [4] F. Psarommatis, “A generic methodology and a digital twin for zero defect manufacturing (ZDM) performance mapping towards design for ZDM,” *J Manuf Syst*, vol. 59, pp. 507–521, Apr. 2021, doi: 10.1016/j.jmsy.2021.03.021.
- [5] D. Powell, M. C. Magnanini, M. Colledani, and O. Myklebust, “Advancing zero defect manufacturing: A state-of-the-art perspective and future research directions,” *Computers in Industry*, vol. 136. Elsevier B.V., Apr. 01, 2022. doi: 10.1016/j.compind.2021.103596.
- [6] “Quality Improvements Towards Zero Defects.”
- [7] C. Caccamo, R. Eleftheriadis, M. C. Magnanini, D. Powell, and Odd Myklebust, “A Hybrid Architecture for the Deployment of a Data Quality Management (DQM) System for Zero-Defect Manufacturing in Industry 4.0,” in *IFIP Advances in Information and Communication Technology*, 2021, vol. 632 IFIP, pp. 71–77. doi: 10.1007/978-3-030-85906-0_8.
- [8] F. Psarommatis, J. Sousa, J. P. Mendonça, and D. Kiritsis, “Zero-defect manufacturing the approach for higher manufacturing sustainability in the era of industry 4.0: a position paper,” *Int J Prod Res*, vol. 60, no. 1, pp. 73–91, 2022, doi: 10.1080/00207543.2021.1987551.
- [9] G. Choi, S. H. Kim, C. Ha, and S. J. Bae, “Multi-step ART1 algorithm for recognition of defect patterns on semiconductor wafers,” *Int J Prod Res*, vol. 50, no. 12, pp. 3274–3287, Jun. 2012, doi: 10.1080/00207543.2011.574502.
- [10] C. F. Kuo, C. T. M. Hsu, C. H. Fang, S. M. Chao, and Y. de Lin, “Automatic defect inspection system of colour filters using Taguchi-based neural network,” *Int J Prod Res*, vol. 51, no. 5, pp. 1464–1476, 2013, doi: 10.1080/00207543.2012.695877.

- [11] C. F. Chien, S. C. Hsu, and Y. J. Chen, "A system for online detection and classification of wafer bin map defect patterns for manufacturing intelligence," *Int J Prod Res*, vol. 51, no. 8, pp. 2324–2338, 2013, doi: 10.1080/00207543.2012.737943.
- [12] D. Mourtzis, E. Vlachou, and N. Milas, "Industrial Big Data as a Result of IoT Adoption in Manufacturing," in *Procedia CIRP*, 2016, vol. 55, pp. 290–295. doi: 10.1016/j.procir.2016.07.038.
- [13] F. Psarommatis, S. Prouvost, G. May, and D. Kiritsis, "Product Quality Improvement Policies in Industry 4.0: Characteristics, Enabling Factors, Barriers, and Evolution Toward Zero Defect Manufacturing," *Frontiers in Computer Science*, vol. 2. Frontiers Media S.A., Aug. 11, 2020. doi: 10.3389/fcomp.2020.00026.
- [14] K. S. Wang, "Towards zero-defect manufacturing (ZDM)-a data mining approach," *Adv Manuf*, vol. 1, no. 1, pp. 62–74, 2013, doi: 10.1007/s40436-013-0010-9.
- [15] F. Psarommatis, G. May, P. A. Dreyfus, and D. Kiritsis, "Zero defect manufacturing: state-of-the-art review, shortcomings and future directions in research," *International Journal of Production Research*, vol. 58, no. 1. Taylor and Francis Ltd., pp. 1–17, Jan. 02, 2020. doi: 10.1080/00207543.2019.1605228.
- [16] F. Eger *et al.*, "Zero Defect Manufacturing Strategies for Reduction of Scrap and Inspection Effort in Multi-stage Production Systems," in *Procedia CIRP*, 2018, vol. 67, pp. 368–373. doi: 10.1016/j.procir.2017.12.228.
- [17] J. Lindström *et al.*, "Towards intelligent and sustainable production systems with a zero-defect manufacturing approach in an Industry4.0 context," in *Procedia CIRP*, 2019, vol. 81, pp. 880–885. doi: 10.1016/j.procir.2019.03.218.
- [18] X. Zheng, J. Lu, and D. Kiritsis, "The emergence of cognitive digital twin: vision, challenges and opportunities," *Int J Prod Res*, 2021, doi: 10.1080/00207543.2021.2014591.
- [19] E. Kharlamov, F. Martin-Recuerda, B. Perry, D. Cameron, R. Fjellheim, and A. Waaler, "Towards Semantically Enhanced Digital Twins," in *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, Jan. 2019, pp. 4189–4193. doi: 10.1109/BigData.2018.8622503.
- [20] D. Plakhotnik, A. Curutiu, A. Zhulavskiy, X. Beudaert, J. Munoa, and M. Stautner, "Framework for coupled digital twins in digital machining," *MM Science Journal*, vol. 2021-November, pp. 5093–5097, Nov. 2021, doi: 10.17973/MMSJ.2021_11_2021158.

- [21] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification," Jan. 2018, vol. 51, no. 11, pp. 1016–1022. doi: 10.1016/j.ifacol.2018.08.474.
- [22] E. Negri, L. Fumagalli, and M. Macchi, "A Review of the Roles of Digital Twin in CPS-based Production Systems," *Procedia Manuf*, vol. 11, pp. 939–948, 2017, doi: 10.1016/j.promfg.2017.07.198.
- [23] A. Polenghi, I. Roda, M. Macchi, and A. Pozzetti, "Ontology-augmented Prognostics and Health Management for shopfloor-synchronised joint maintenance and production management decisions," *J Ind Inf Integr*, vol. 27, May 2022, doi: 10.1016/j.jii.2021.100286.
- [24] C. M. Keet, "An Introduction to Ontology Engineering."
- [25] M. H. Karray, F. Ameri, M. Hodkiewicz, and T. Louge, "ROMAIN: Towards a BFO compliant reference ontology for industrial maintenance," *Appl Ontol*, vol. 14, no. 2, pp. 155–177, 2019, doi: 10.3233/AO-190208.
- [26] A. Polenghi, I. Roda, M. Macchi, A. Pozzetti, and H. Panetto, "Knowledge reuse for ontology modelling in Maintenance and Industrial Asset Management," *J Ind Inf Integr*, vol. 27, May 2022, doi: 10.1016/j.jii.2021.100298.
- [27] F. Ameri, D. Sormaz, F. Psarommatis, and D. Kiritsis, "Industrial ontologies for interoperability in agile and resilient manufacturing," *Int J Prod Res*, vol. 60, no. 2, pp. 420–441, 2022, doi: 10.1080/00207543.2021.1987553.
- [28] V. R. Sampath Kumar *et al.*, "Ontologies for industry 4.0," *Knowledge Engineering Review*, vol. 34, 2019, doi: 10.1017/S0269888919000109.
- [29] C. Sansone, P. Hilletoft, and D. Eriksson, "Critical operations capabilities for competitive manufacturing: A systematic review," *Industrial Management and Data Systems*, vol. 117, no. 5. Emerald Group Publishing Ltd., pp. 801–837, 2017. doi: 10.1108/IMDS-02-2016-0066.
- [30] R. Peres, A. D. Rocha, J. P. Matos, and J. Barata, "GOODMAN Data Model - Interoperability in Multistage Zero Defect Manufacturing," in *Proceedings - IEEE 16th International Conference on Industrial Informatics, INDIN 2018*, Sep. 2018, pp. 815–821. doi: 10.1109/INDIN.2018.8472017.
- [31] F. Eger, C. Reiff, M. Colledani, and A. Verl, "Knowledge Capturing Platform in Multi-Stage Production Systems for Zero-Defect Manufacturing; Knowledge Capturing Platform in Multi-Stage Production Systems for Zero-Defect Manufacturing," 2018.
- [32] F. Fraile, L. Montalvillo, M. A. Rodriguez, H. Navarro, and A. Ortiz, "Multi-tenant data management in collaborative zero defect manufacturing," in *2021 IEEE International Workshop on Metrology for Industry 4.0 and IoT, MetroInd 4.0 and*

- IoT 2021 - Proceedings*, Jun. 2021, pp. 464–468. doi: 10.1109/MetroInd4.0IoT51437.2021.9488534.
- [33] F. Eger, P. Tempel, M. C. Magnanini, C. Reiff, M. Colledani, and A. Verl, “Part variation modeling in multi-stage production systems for zero-defect manufacturing,” in *Proceedings of the IEEE International Conference on Industrial Technology*, Feb. 2019, vol. 2019-February, pp. 1017–1022. doi: 10.1109/ICIT.2019.8754964.
- [34] E. I. Papageorgiou *et al.*, “Short Survey of Artificial Intelligent Technologies for Defect Detection in Manufacturing,” Jul. 2021. doi: 10.1109/IISA52424.2021.9555499.
- [35] R. Teti, “Advanced IT methods of signal processing and decision making for zero defect manufacturing in machining,” in *Procedia CIRP*, 2015, vol. 28, pp. 3–15. doi: 10.1016/j.procir.2015.04.003.
- [36] T. Vafeiadis, D. Ioannidis, C. Ziazios, I. N. Metaxa, and D. Tzovaras, “Towards Robust Early Stage Data Knowledge-based Inference Engine to Support Zero-defect Strategies in Manufacturing Environment,” *Procedia Manuf*, vol. 11, pp. 679–685, 2017, doi: 10.1016/j.promfg.2017.07.167.
- [37] L. Jinzhi, M. Junda, X. Zheng, G. Wang, and D. Kiritsis, “Design Ontology Supporting Model-based Systems-engineering Formalisms,” Oct. 2020, [Online]. Available: <http://arxiv.org/abs/2010.07627>
- [38] M. Armendia *et al.*, “Fingerprint: Machine tool condition monitoring approach for zero defect manufacturing,” *MM Science Journal*, vol. 2021-November, pp. 5247–5253, Nov. 2021, doi: 10.17973/MMSJ.2021_11_2021159.
- [39] F. Psarommatis and D. Kiritsis, “A hybrid Decision Support System for automating decision making in the event of defects in the era of Zero Defect Manufacturing,” *J Ind Inf Integr*, vol. 26, Mar. 2022, doi: 10.1016/j.jii.2021.100263.
- [40] J. Schmidt, F. Grandi, M. Peruzzini, R. Raffaeli, and M. Pellicciari, “Novel robotic cell architecture for zero defect intelligent deburring,” in *Procedia Manufacturing*, 2020, vol. 51, pp. 140–147. doi: 10.1016/j.promfg.2020.10.021.
- [41] D. Mourtzis, J. Angelopoulos, and N. Panopoulos, “Equipment Design Optimization Based on Digital Twin under the Framework of Zero-Defect Manufacturing,” in *Procedia Computer Science*, 2021, vol. 180, pp. 525–533. doi: 10.1016/j.procs.2021.01.271.
- [42] M. C. Magnanini, M. Colledani, and D. Caputo, “Reference architecture for the industrial implementation of zero-defect manufacturing strategies,” in *Procedia CIRP*, 2020, vol. 93, pp. 646–651. doi: 10.1016/j.procir.2020.05.154.

- [43] F. Psarommatis, G. May, and D. Kiritsis, "Predictive maintenance key control parameters for achieving efficient Zero Defect Manufacturing," in *Procedia CIRP*, 2021, vol. 104, pp. 80–84. doi: 10.1016/j.procir.2021.11.014.
- [44] E. Lodgaard and D. Powell, "The changing role of shop-floor operators in zero defect manufacturing," in *Procedia CIRP*, 2021, vol. 104, pp. 594–599. doi: 10.1016/j.procir.2021.11.100.
- [45] N. Nikolaidis, T. Naskos, E. Urkia, and I. N. Metaxa, "'Towards ZDM in danobat, using prediction for equipment RUL,'" in *Procedia Manufacturing*, 2020, vol. 51, pp. 1581–1585. doi: 10.1016/j.promfg.2020.10.220.
- [46] V. Azamfirei, A. Granlund, and Y. Lagrosen, "Multi-layer quality inspection system framework for industry 4.0," *International Journal of Automation Technology*, vol. 15, no. 5, pp. 641–650, Sep. 2021, doi: 10.20965/ijat.2021.p0641.
- [47] J. Peffers, "The Design Science Research Process : A Model for Producing and Presenting Information Systems Research," 2006. [Online]. Available: <http://rightsstatements.org/page/InC/1.0/?language=en>
- [48] F. Psarommatis and D. Kiritsis, "Comparison Between Product and Process Oriented Zero-Defect Manufacturing (ZDM) Approaches," in *IFIP Advances in Information and Communication Technology*, 2021, vol. 630 IFIP, pp. 105–112. doi: 10.1007/978-3-030-85874-2_11.
- [49] M. Almeida *et al.*, "Basic Formal Ontology 2.0 SPECIFICATION AND USER'S GUIDE Co-Authors / Acknowledgments," 2015.
- [50] "Ontology Engineering in a Networked World."
- [51] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology." [Online]. Available: www.unspsc.org
- [52] M. Grr and M. S. Fox, "Methodology for the Design and Evaluation of Ontologies," 1995.
- [53] M. Uschold and M. King, "Towards a Methodology for Building Ontologies," 1995.
- [54] J. Völker, D. Vrandečić, Y. Sure, and A. Hotho, "AEON - An approach to the automatic evaluation of ontologies," *Appl Ontol*, vol. 3, no. 1–2, pp. 41–62, 2008, doi: 10.3233/AO-2008-0048.
- [55] M. Ferndndez, A. Gómez-Perez, and N. Juristo, "METHONTOLOGY: From Ontological Art Towards Ontological Engineering," 1997. [Online]. Available: www.aaai.org
- [56] B. Smith, "On Classifying Material Entities in Basic Formal Ontology." [Online]. Available: <http://ontology.buffalo.edu/smith>

- [57] D. L. Nuñez and M. Borsato, "OntoProg: An ontology-based model for implementing Prognostics Health Management in mechanical machines," *Advanced Engineering Informatics*, vol. 38, pp. 746–759, Oct. 2018, doi: 10.1016/j.aei.2018.10.006.
- [58] D. L. Nuñez and M. Borsato, "An ontology-based model for prognostics and health management of machines," *J Ind Inf Integr*, vol. 6, pp. 33–46, Jun. 2017, doi: 10.1016/j.jii.2017.02.006.
- [59] A. Zhou, D. Yu, and W. Zhang, "A research on intelligent fault diagnosis of wind turbines based on ontology and FMECA," *Advanced Engineering Informatics*, vol. 29, no. 1, pp. 115–125, 2015, doi: 10.1016/j.aei.2014.10.001.
- [60] M. Colledani, W. Terkaj, T. Tolio, and M. Tomasella, "Development of a conceptual reference framework to manage manufacturing knowledge related to products, processes and production systems," in *Methods and Tools for Effective Knowledge Life-Cycle-Management*, Springer Berlin Heidelberg, 2008, pp. 259–284. doi: 10.1007/978-3-540-78431-9_15.
- [61] F. Ameri and C. McArthur, "Semantic rule modelling for intelligent supplier discovery," *Int J Comput Integr Manuf*, vol. 27, no. 6, pp. 570–590, Jun. 2014, doi: 10.1080/0951192X.2013.834467.
- [62] Z. Usman, R. I. M. Young, N. Chungoora, C. Palmer, K. Case, and J. A. Harding, "Towards a formal manufacturing reference ontology," *Int J Prod Res*, vol. 51, no. 22, pp. 6553–6572, Nov. 2013, doi: 10.1080/00207543.2013.801570.
- [63] A. Ghose, M. Lissandrini, E. R. Hansen, and B. P. Weidema, "A core ontology for modeling life cycle sustainability assessment on the Semantic Web," *J Ind Ecol*, vol. 26, no. 3, pp. 731–747, Jun. 2022, doi: 10.1111/jiec.13220.
- [64] H. Jiang, J. Yi, X. Zhu, and Z. Li, "Generating disassembly tasks for selective disassembly using ontology-based disassembly knowledge representation," *Assembly Automation*, vol. 38, no. 2, pp. 113–124, Mar. 2018, doi: 10.1108/AA-04-2016-034.
- [65] B. Zhu and U. Roy, "Ontology-based disassembly information system for enhancing disassembly planning and design," *International Journal of Advanced Manufacturing Technology*, vol. 78, no. 9–12, pp. 1595–1608, Jun. 2015, doi: 10.1007/s00170-014-6704-8.
- [66] P. Borst and H. Akkermans, "An ontology approach to product disassembly," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1997, vol. 1319, pp. 33–48. doi: 10.1007/BFb0026776.
- [67] S. Chen, J. Yi, H. Jiang, and X. Zhu, "Ontology and CBR based automated decision-making method for the disassembly of mechanical products," *Advanced*

Engineering Informatics, vol. 30, no. 3, pp. 564–584, Aug. 2016, doi:
10.1016/j.aei.2016.06.005.

[68] Lamy Jean Baptiste, “Ontologies with Python”, 2021

[69] <https://biportal.bioontology.org/ontologies/BFO>

[70] <https://biportal.bioontology.org/ontologies/IAO>

[71] <https://github.com/CommonCoreOntology/CommonCoreOntologies>

[72] <https://www.industrialontologies.org>

A) APPENDIX: Data Collection

Although the focus of the work is the development of an ontology for the application of the ZDM, a secondary but not negligible part of the work carried out was the collection of data from the machines present on the line in order to be able to verify the functioning of the ontology. This step is in fact essential as the data collected by the various servers can then be entered into the ontology to verify its operations. This process can then enable a method through which it is possible to collect data in real time, which can then be entered into the ontology to perform operations in almost real time. To achieve this objective the steps followed are represented in the image below

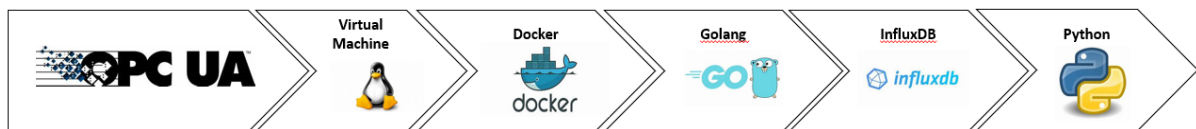


Figure A-1: Steps executed in the thesis work

In order to better understand how it was possible to extract the data from the servers, it is necessary to dwell for a moment on the architecture currently present in the laboratory, the OPC UA.

OPC-UA

OPC UA is a Client-Server protocol, in which possibly more than one Client interfaces itself directly with the Server to acquire data. The Client send one or more requests for services to the Server, which provides a service as an answer, like for example a data read or a subscription and so on.

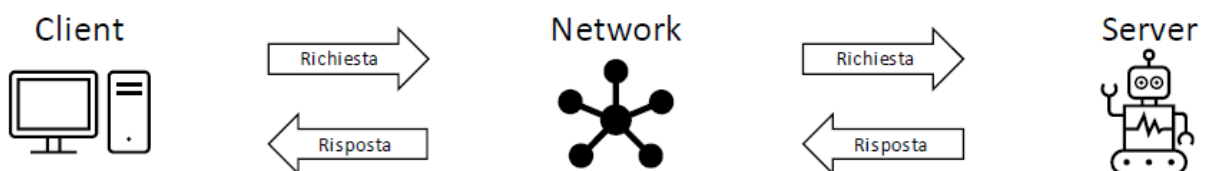


Figure A-2: Representation of the sending and receiving of a request to a server

The exchange of information can happen over TCP (Transmission Control Protocol) or can happen over another application layer protocol which is HTTP (Hypertext Transfer Protocol), which are not going to be considered since in the Lab the communication stack utilized is the TCP one. There are several reasons according to

which OPC UA should be utilized, as a matter of fact in addition to a practical interface for the acquiring of industrial data, OPC UA provides also a framework to execute Information Modelling, to represent and organize the signals which are provided by the server. So Information Modelling can be used to represent complex informations as Objects in an Address Space. The Objects consist in Nodes (characterized by a unique identifier called NodeId) which are linked by References. The class of Nodes to consider is the Variable Node so a Node representing a value that can be read or written. This Node has an associated DataType, which define the actual value of the sensor such as a string, a structure, a float and so on. In this way using OPC UA is possible to access a server current value by investigating the Nodes. Therefore it is possible to associate a node to each signal, making possible to collect signals of interest using the OPC UA specification, which once invoked on a Server permits to acquire the values stored inside its nodes, depending on the library that it is used to interact with OPC UA Servers.

Now that it has been seen how to take advantage of some basic elements of OPC UA there is the need to investigate more deeply on how to explore the Nodes and be able to read their data. In this part it is not necessary to dwell on the technical details that allow to perform a data reading, but a characteristic of OPC UA is treated: its asynchronicity. Previously it was said that OPC UA is a Client-Server protocol and that therefore every time it is necessary to obtain data from a server it is necessary to send a request to the server and wait for its response. This mechanism appears immediate in the case in which there is just a server and a client, but in the case there are more servers the situation becomes more complicated. Specifically, every time a request to a server is made to obtain data, the request is sent to the server which must respond and send a response to the customer, who can make a new request. In the event that there are multiple servers although multiple requests are sent at the same time, the requests do not arrive at the servers at the same time. Let's consider the case of a client and two servers in which the client sends a request to the two servers at the same time. Although the request is sent at the same time, one of the two servers will not receive the request immediately, but will have to wait for the other server to receive the request and respond before receiving its request. However, there is a way to overcome this problem by leveraging the asynchronicity of OPC UA by resorting to a process known as Concurrent Programming. According to the definition reported by Wikipedia, Concurrent Programming means: "Concurrent computing is a form of computing in which several computations are executed concurrently, during overlapping time periods, instead of sequentially, with one completing before the next

starts". Using concurrent programming is possible to sent multiple request to multiple servers permitting the user to work with different tasks at once, since there is not the necessity to wait for the server response each time to send a request to another server

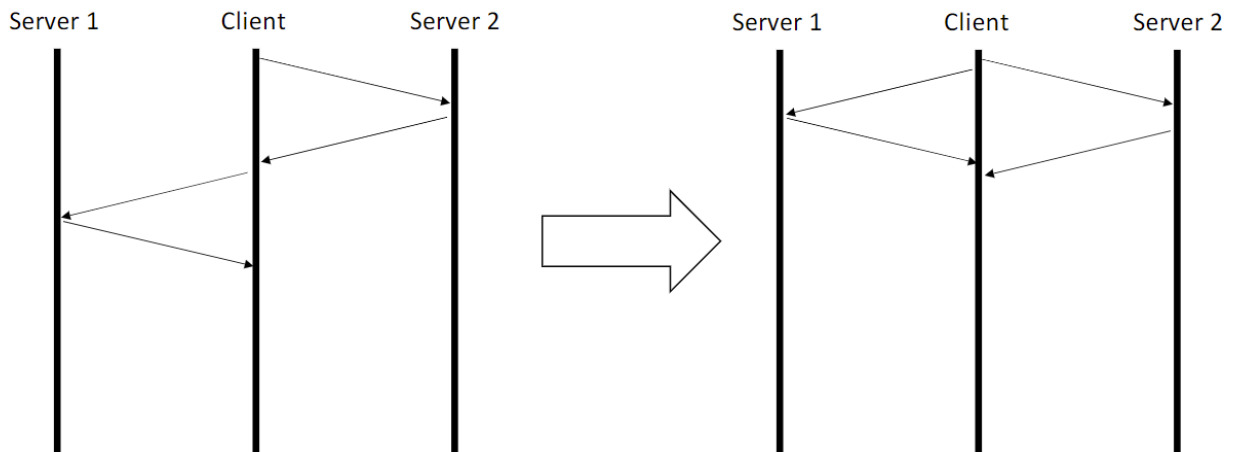


Figure A-3: difference between a non concurrent system (on the right) and concurrent one (on the left)

Work Performed

Now that the basic understanding of OPC UA and concurrent programming has being covered, it is possible to introduce the activities which have been executed. In the Industry 4.0 Lab every station has an Energy and an Operational PLC and every of these two PLCs runs an OPC UA Server. The Energy Server publishes signals related to the energy consumption of the station while the Operational Server publishes mainly Boolean signals related to the presence of carriers in specific positions of the conveyor. The OPC UA servers are on a local network accessible either by connecting at AP-CP-Lab via Wi-Fi but it is also possible to connect an ethernet cable from the personal computer to one of the line's hubs.

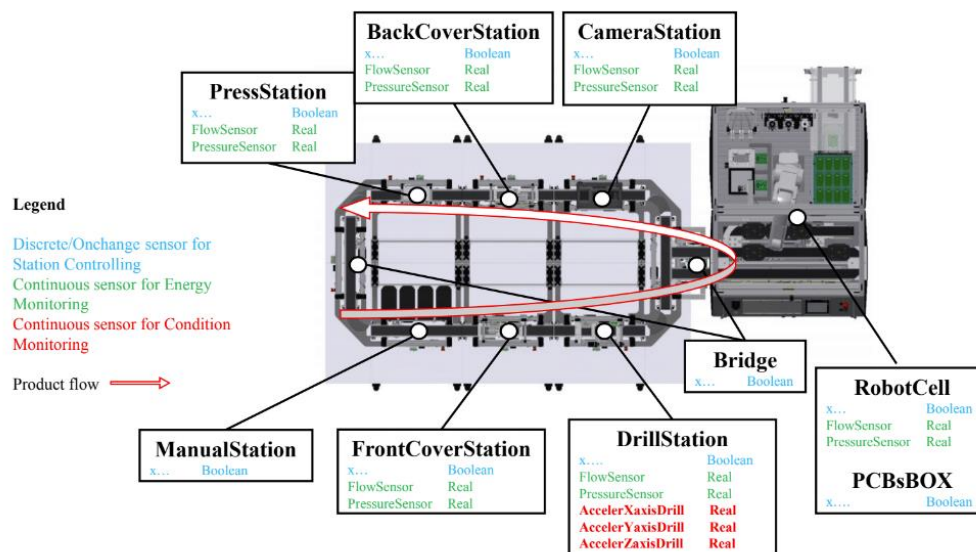


Figure A-4: Taken from 6.1.1, Representation of the Flexible Manufacturing Line in the Industry 4.0 Lab

Linux Virtual Machine

To retrieve the data from the servers a Linux virtual machine has been installed on the computer by using the hypervisor desktop VMware workstation pro. The reasons to use a virtual machine with Linux on it are several, as a matter of fact the Linux environment present different advantages like the lightweight open source operating system, its flexibility, since Linux system provides high performance over different networks and the possibility to use an environment in which there are all the tools necessary for application development, such as the python environment, the Go environment and many others. For all these different reasons it has been decided to utilize a virtual machine with the aim to get all the characteristics reported above and to get a unique environment in which all the needed elements were contained in a tidy manner using Docker features.

Docker

Docker according to <https://www.ibm.com/cloud/learn/docker> is an open source containerization platform which allows to develop, ship, and run applications in the background of the virtual machine. As a matter of fact Docker enables developers to separate your applications from your infrastructure by packing applications into containers, where by container is intended: "A container is a self-contained execution environment that shares the kernel of the host system and which is (optionally) isolated from other containers in the system" where the kernel is the Linux Kernel so the part of an operating system that manages interactions with the hardware of a

computer. To summarize a container is a standardized executable component which combine application source code and operating system libraries and the dependencies required to run the code. In this way it is possible to simplify the delivery of distributed applications, while enabling multiple applications component to share the resources of a single instance of the host operating system in the same way a hypervisor enable multiple virtual machines to share the CPU, the memory and other resources of a single hardware server. So containerization provides several benefits and functionality, for example Docker enhances the native Linux containerization capabilities with technologies that enable:

- Improved—and seamless—portability
- Even lighter weight and more granular updates
- Automated container creation
- Container versioning
- Container reuse
- Shared container libraries

So once the virtual machine has been installed, Docker was installed and different containers were created, one container for Golang, one dedicated to Python and finally one dedicated to InfluxDB were created following the procedure expressed in the Linux official website. Now It is necessary to focus the analysis on the functions performed by the individual containers starting with the one dedicated to Golang.

Golang

Previously It has been introduced the concept of concurrent programming, but it was not stated which are the possible solutions to obtain this result and between all the possible solutions the choice fell on Golang, the Google programming language. The details of the Go syntax are not going to be considered or how is this language compared to other ones, but it will be discussed what are the concepts behind how the protocol can be exploited. In fact the protocol can be used to gather data faster in certain situations using Go programming language and its open source OPC UA Client library gopcua. As a matter of fact Golang provides easy ways to write and launch concurrent programs, but before getting into the how and why, it is necessary to cover the what. There are two primary elements of Go's concurrency model:

- Goroutines: a concurrent thread of execution, which execution is decided by the Golang scheduler, which in order to be spawned it is necessary to put the keyword “go” before a function
- Channels: it is the way through which goroutines communicate each other via message passing. So Channels represent how goroutines share data with one another without the complexity and safety concerns that come with sharing memory. In terms of usage, a channel can essentially be thought of as a typed queue with optional capacity.

Concerning the object of this thesis work to serve a request to the different servers in the line, it is necessary to query the API of the servers, process their data into a consistent format and aggregate all of these data into a single response payload. For this reason, scripts have been created in Golang in order to be able to send a request to the various servers that make up the line in order to be able to collect the energy values of the servers and the various Booleans. Once the script has been executed, the values collected by the server are printed on the virtual machine terminal and on InfluxDB.

InfluxDB

InfluxDB is a database management system developed by InfluxData, Inc and programmed in the Golang programming language. This system was designed for Time Series databases (TSDB) and therefore for databases that are also used to collect and analyze data from sensors or time-stamped protocols over a certain period of time. Once in the database these data must be processed quickly, for this reason a time service known as Network Time Protocol (NTP) is integrated in InfluxDB which ensures time synchronization in all the systems. Nevertheless the data coming from the servers are sent to InfluxDB in which they are collected and stored for further applications like the creation of a dashboard or to be inserted inside the ontology. In order to execute this last step another passage is necessary, as a matter of fact there is the necessity to use a python script in order to retrieve the data from InfluxDB and convert them in a format which is readable for the ontology using the python library `owready2`.

Python

As final step of this operating process a python script was realized in order to convert the data coming from InfluxDB as a CSV file which is then utilized to populate the ontology. As a matter of fact the initial part of the code is dedicated to retrieve the data

coming from the servers which are stored in InfluxDB and convert them into a CSV while the second part of the code is dedicated to use this CSV file to populate the ontology using the functions contained in the owlready2 python library. This part will be seen in greater detail in the next chapter, Appendix B

B) Appendix: Python Code

B.1. Python code for the data extracted by the FML

```
def processa_record(onto, Station, node_id, Timestamp, valore, TS):
    """
    Process Record in the consumer
    """

    global RobotToggle

    try:
        if node_id in lista_bool:
            Boolean = valore
            Energy = None
        else:
            Boolean = "99"
            Energy = valore

        if Station:
            #-----
            #This is the incipit part of the code, here we have all the stations which are present in the line.
            #In function of the data extracted from the csv file it is possible to determine where the assembly
            #is and accordingly what operations need to be performed
            #-----
```

Figure B-1: Initial part of the code

In the image B_1 there is the starting point of the python code. Here the csv file, obtained from the FML, is read and the elements in the csv file are organized in Station, node_id, Timestamp, valore and TS. Where Station is the name of the station, the node_id is the name of the sensor, the Timestamp is the time at which the data is taken, valore is the value of the sensor and TS is the list of objects processed in the FML in a Digital Twin logic. The first *if statement* says that if the name of the node is contained in lista_bool, which is the list of all the sensors recording a boolean value, it will be processed. In case the name of the sensor is not contained in that list then the sensor is not a Boolean sensor, but it is an energy sensor which has valore equal to the energy reported in the csv file.

```
if Station:
    #-----
    #This is the incipit part of the code, here we have all the stations which are present in the line.
    #In function of the data extracted from the csv file it is possible to determine where the assembly
    #is and accordingly what operations need to be performed
    #-----
```

Figure B-2: Reading of the Station in the CSV file

In the image B_2 there is the second *if statement* of the python code, in particular it says that all the stations of the FML are read.

```

if Station == "FrontCoverPLC":
#-----
#This is the sensor dedicated to determine the position of the product, in particular if the
#product has passed through the Frontcover station. This is the first station to be visited
#by the Assembly. Here the front cover is inserted on the product
#-----
    if Boolean == 1.0:
        _logger.info(f"A new product has entered in the line at time {Timestamp}")
        Individual= onto.Assembly(new_piece_st1()) #Here the new product is created using the function new_piece_st1().
                                                #The function operates each time a Boolean equal to 1 is registered
                                                #by creating a new individual named "1_" + the number of Boolean
                                                #equal to 1 have been registered.

        id =onto.Assembly_ID(product_id_st1()) #This is the function to create the id of the product worked by the
                                                #Frontcover station. This function each time a Boolean equal to 1 is
                                                #registered an id is created named "Process_1 Product_" + the number
                                                #of Boolean equal to 1 have been registered.

```

Figure B-3: Creation of an Individual and Id in the FrontCover Station

The first station to be analyzed is the FrontCover station, image B_3. In this station a new *Individual* is going to be created with the name "1_" plus the *Individual object id*. The *object id* is obtained by means of an algorithm which gives a different object id each time a new *Individual* is seen in the FML. In this way the system creates a digital image of the process which is happening in the FML (Digital Twin). The same logic is applied for the *Individual id*, which has name "Process_1_Product_" plus the *Individual object id*. This part of code will be executed in each station of the FML in order to keep track of the movement of *Individual* along the FML

```

Individual.has_id=id
Ts[Station+"_"+str(id)]=Timestamp
_logger.info(Individual)
_logger.info(Individual.has_id)
Individual.has_component=[] #This element is introduced in the code in order to make possible for the product
                             #to have more than one component

front_cover= onto.Component(FrontCover()) #Here the individual front cover is created, so each time a product
                                             #arrives at the front cover station a new frontcover is retrieved

_logger.info(id)
Individual.has_component.append(front_cover) #Here the front cover is inserted on the product, which now have
                                             #just one component

_logger.info(Individual.has_component)
_logger.info(f"The front cover has been inserted at time {Timestamp}")
front_cover.has_feature=onto.Feature(frontcover_feature()) #Since the front cover is the element that should be
                                                            #drilled at the Drill Station it is necessary to
                                                            #introduce one element dedicated to describe the quality
                                                            #of the operation conducted by the drilling machine

```

Figure B-4: Adding of the front cover component to the Individual

Since we are dealing with the FrontCover Station the individual front cover is going to be created and is going to become a component of *Individual*. As a matter of fact in the image B_4 it is possible to see that *Individual* has now a component which is front cover. The individual front cover has a feature, namely *frontcover_feature*, which is attributed to the individual because in the next station, the Drill Station, the front cover is going to be drilled and so the quality of the drilling process must be recorded in the *frontcover_feature*.

```

_logger.info(front_cover.has_feature)
Individual.has_position=onto.FrontCover_Station #Here the position of the product is determined, so if for some reason
                                                  #the product does not go to the next station it is possible to check
                                                  #if the product passed for the FrontCover station

_logger.info(Individual.has_position)
else:
    _logger.info("The piece is not at the front cover station")

```

Figure B-5: FrontCover Station position of the individual

After the operations described in the images B_3 and B_4 are done then a position is attributed to Individual, as a matter of fact in the image B_5 Individual has a position named FrontCover_Station.

```
if Station == "FrontCoverEnergy": #This is the sensor dedicated to determine the energy value of the FrontCover station so
    #this sensor just reports the energy of the station and not if the product is passing by
    if not (node_id in lista_bool):
        _logger.info("The energy of the machine is",Energy)
if Station == "DrillPLC":
```

Figure B-6: Representation of the FrontCover Energy station

In the image B_6 is shown that if the name of the station is FrontCoverEnergy then only the energy value of the station is reported.

```
if Station == "DrillPLC":
#-----
#This is the sensor dedicated to determine the position of the product, in particular if the
#product has passed through the Drill station. This is the second station to be visited by
#the Assembly. Here the front cover is drilled by the drilling machine
#-----

if Boolean == 1.0:
    Individual= onto.Assembly(new_piece_st2()) #This function has the same behaviour of the function we have seen in the
    #station before, new_piece_st1(). This function is needed to define the
    #product that arrives at the Drill Station. As for the case before, each
    #time a Boolean equal to 1 is registered a new individual is created with
    #the name "2_" + the number of Boolean equal to 1 have been registered.

    id =onto.Assembly_ID(product_id_st2()) #This is the function to create the id of the product worked by the
    #Drill station which is made in the same way of the one at the FrontCover
    #station. As a matter of fact each time a Boolean equal to 1 is registered
    #an id is emitted, named "Process_1 Product_" + the number of Boolean
    #equal to 1 have been registered.

    Individual.has_id=id
    TS[Station+"_"+str(id)]=Timestamp
    front_cover= onto.Component("FrontCover"+"_"+str(FC))
    _logger.info(id)
    Individual.has_component.append(front_cover)
    _logger.info(Individual.has_component)
    front_cover.has_feature=onto.Feature("FrontCover_Feature_" + str(FCF))
    _logger.info(front_cover.has_feature)
    Individual.has_position= onto.Drill_Station
    Asset= onto.Drilling_Machine
    if onto.Assembly("1_" + str(st1)).has_id == onto.Assembly("2_" + str(st2)).has_id: #This passage is done in order to execute the
    #tracking of the product, as a matter of fact
    #if the id is generated and it is the same of
    #the one obtained at the FrontCover station
    #then it is possible to eliminate the individual
    #before and keep the new one
```

Figure B-7: Representation of the Drill Station

The next station to analyze is the Drill Station, here the code shown in image B_7 has the same initial structure of the FrontCover Station. The difference respect to the FrontCover Station is reported in the lower half of image B_7 where the id of the two individuals created in the FrontCover and Drill station are confronted. This passage is necessary in order to execute a simple tracking process where if the IDs are the same only the entity created in the latter station is kept, as seen in the image B_8. This is done because in case of error it is possible to check the id of Individual and see which was the last station visited by the Individual.

```

destroy_entity(onto.Assembly("1_" + str(st1))) #This is the passage expressed before, since the id of the individuals are
#the same then it is possible to eliminate the individual produced before
#and keep the one produced now

sh_value,sh_str=get_new_shacker(shacker_repository) #This is the function introduced in order to simulate the behavior of the
#shaker, as a matter of fact it reports if the shacker is active or not

_logger.info(sh_str)
if sh_str == "SHACKER OFF": #If the shacker is off then the behaviour of the drilling machine is optimal and so the feature
#of the drilling machine will show no malfunction

    Asset_Feature= onto.Feature(drilling_machine_feature())
    Asset.has_feature=Asset_Feature
    Malfunctioning= False
    Asset_Feature.Malfunctioning=Malfunctioning
    Asset_Feature.settles= onto.Feature("FrontCover_Feature_" + str(FCF))

else: #This is the opposite case so the shacker is active and the drilling machine behaviour will be sub optimal
#so the feature of the drilling machine will show a malfunction. The piece obtained then needs to be
#repaired

    Asset_Feature= onto.Feature(drilling_machine_feature())
    _logger.info(Asset_Feature)
    Asset.has_feature=Asset_Feature
    _logger.info(Asset.has_feature)
    Malfunctioning=True
    Asset_Feature.Malfunctioning= Malfunctioning
    _logger.info(Asset_Feature.Malfunctioning)
    Asset_Feature.settles= onto.Feature("FrontCover_Feature_" + str(FCF))
    _logger.info(Asset_Feature.settles)
else:
    _logger.info("The piece is not at the drill station")

```

Figure B-8: Representation of the Shacker function

In image B_8 it is also shown the elements corresponding to the analysis of the shacker. If the shacker is not active then the individual *Drilling_Machine* has a feature which is characterized by Malfunctioning equal to false otherwise it has value true.

```

if Station == "DrillEnergy": #This is the sensor dedicated to determine the energy value of the Drill station so
#this sensor just reports the energy of the station and not if the product is passing by

if not (node_id in lista_bool):
    #_logger.info(onto.Manual_Station_Operating_Time)
    _logger.info("The energy of the machine is",Energy)

```

Figure B-9: Representation of the Drill Station Energy

In the image B_9 is shown that if the name of the station is DrillEnergy then only the energy value of the station is reported

```

if Station == "RobotPLC":
#-----
#This is the sensor dedicated to determine the position of the product, in particular if the
#product has passed throught the Robot Cell. This is the third station to be visited by the
#Assembly. Here the pcb and the fuses are inserted by the robot arm on the frontcover
#-----

if (Boolean != 1.0) and (RobotToggle == True): #This element have been introduced in the code because of the
#configuration of the csv file. In particular, as expressed in
#the thesis we are going to consider the xStart sensor, which
#is characterized by positive and negative values. In the positive
#values there are many values equal to 1 so it is necessary to
#set these conditions to make the code work in the right way.
#First if a 1 is recorded and the global variable "RobotToggle"
#is false then the variable is set to true and the needed
#operations are performed. So as soon as a negative value is
#recorded and the "RobotToggle" is true the variable is set
#to false to make possible the following code for the next population

    RobotToggle = False
if (Boolean == 1.0) and (RobotToggle == False):
    RobotToggle = True
    Robot_Cell_Arrival=Timestamp
    Individual= onto.Assembly(new_piece_st3()) #This function is the same we have seen in the stations before. It
#define the product that arrives at the Robot Cell. As for the cases
#before, each time a Boolean equal to 1 is registred a new individual
#is created with the name "3_" + the number of Boolean equal to 1 have
#been registred.

```

Figure B-10: Representation of the Robot Cell

In the image B_10 it is presented the mechanism introduced in the code in order to read the value of the sensor xStart, following the modalities described in 6.1.1.

```

RobotToggle = False
if (Boolean == 1.0) and (RobotToggle == False):
    RobotToggle = True
    Robot_Cell_Arrival=Timestamp
    Individual= onto.Assembly(new_piece_st3()) #This function is the same we have seen in the stations before. It
    #define the product that arrives at the Robot Cell. As for the cases
    #before, each time a Boolean equal to 1 is registered a new individual
    #is created with the name "3_" + the number of Boolean equal to 1 have
    #been registred.

    id =onto.Assembly_ID(product_id_st3()) #This is the function to create the id of the product worked by the
    #Robot Cell which is made in the same way of the ones at the stations
    #before. Each time a Boolean equal to 1 is registred an id is emitted
    #named "Process_1 Product_" + the number of Boolean equal to 1 have
    #been registred

```

Figure B-11: Individual and id creation in the Robot Cell

```

Individual.has_id=id
if onto.Assembly("3_" + str(st3)).has_id == onto.Assembly("2_" + str(st2)).has_id: #This passage is same before to execute the
    #tracking of the product, if the id is
    #generated and it is the same of the one
    #obtained at the Drill station then it is
    #possible to eliminate the individual before
    #and keep the new one

    destroy_entity(onto.Assembly("2_" + str(st2))) #This is the passage expressed before, since the id of the individuals are
    #the same then it is possible to eliminate the individual produced before
    #and keep the one produced now

    TS[Station+"_"+str(id)]=Timestamp
    _logger.info(Individual)
    _logger.info(Individual.has_id)
    Individual.has_position=onto.Robot_Cell
    _logger.info(Individual.has_position)

```

Figure B-12: Comparison of the Individuals id and Individual "2_" + str(st2) destruction

In the images B_11 and B_12 the same operations executed in the previous stations are presented and this part will be overlooked.

```

_logger.info(pcb)
pcb.has_feature= onto.Feature(PCB_feature()) #As expressed in the thesis the pcb is the component which has
    #an image on the back and so the feature of the pcb will be
    #determined accordingly

_logger.info(pcb.has_feature)
Individual.has_component.append(pcb)
sh_value,sh_str=get_new_shacker(shacker_repository)
left_fuse = onto.Component(LeftFuse())
left_fuse.has_feature= onto.Conform(left_fuse_feature()) #In the ontology there is not a way to determine
    #the characteristics of the fuses and so they are
    #considered conform by default

_logger.info(left_fuse.has_feature)
right_fuse = onto.Component(RightFuse())
right_fuse.has_feature= onto.Conform(right_fuse_feature())
_logger.info(right_fuse.has_feature)
_logger.info(f"The PCB and Fuses are inserted at time {Timestamp}")

```

Figure B-13: pcb and fuses insertion

In the following image, figure B_13, the insertion of the pcb and the fuses is treated. It is also possible to see that for the left and right fuse the feature associated is Conform since, as described in the image, there is no way in the ontology to determine the feature of these 2 elements.

```

if Station == "RobotEnergy":
    #This is the sensor dedicated to determine the energy value of the Drill station so
    #this sensor just reports the energy of the station and not if the product is passing by

    if not (node_id in lista_bool):
        #_logger.info(onto.Manual_Station_Operating_Time)
        _logger.info("The energy of the machine is",Energy)

```

Figure B-14: Representation of the Robot Energy station

In the image B_14 is shown that if the name of the station is DrillEnergy then only the energy value of the station is reported

```

if Station == "CameraPLC":
#-----
#This is the sensor dedicated to determine the position of the product, in particular if the
#product has passed through the Camera station. This is the fourth station to be visited by the
#Assembly. Here the pcb is inspected to determine its state
#-----

if Boolean == 1.0:
    Camera_Station_Arrival=Timestamp
    Individual= onto.Assembly(new_piece_st4()) #This function is the same we have seen in the stations before. It
        #define the product that arrives at the Camera station. As for the cases
        #before, each time a Boolean equal to 1 is registered a new individual
        #is created with the name "4_" + the number of Boolean equal to 1 have
        #been registered.

    id =onto.Assembly_ID(product_id_st4()) #This is the function to create the id of the product worked by the
        #Robot Cell which is made in the same way of the ones at the stations
        #before. Each time a Boolean equal to 1 is registered an id is emitted
        #named "Process_1 Product_" + the number of Boolean equal to 1 have
        #been registered

    Individual.has_id=id
    Im_value,Im_string,Im_bit= get_new_image(image_repository) #This is the function introduced in order to simulate the behavior of the
        #camera inspection, as a matter of fact it reports what is the image on
        #the pcb

    TS[Station+" "+str(id)]=Timestamp
    _logger.info(Individual.has_id)
    Individual.has_position= onto.Camera_Station
    if onto.Assembly("4_" + str(st4)).has_id == onto.Assembly("3_" + str(st3)).has_id: #This passage is same before to execute the
        #tracking of the product, if the id is
        #generated and it is the same of the one
        #obtained at the Drill station then it is
        #possible to eliminate the individual before
        #and keep the new one

destroy_entity(onto.Assembly("3_" + str(st3))) #This is the passage expressed before, since the id of the individuals are
        #the same then it is possible to eliminate the individual produced before
        #and keep the one produced now

front_cover= onto.Component("FrontCover"+" "+str(FC))
left_fuse = onto.Component("LeftFuse"+" "+ str(LF))
right_fuse = onto.Component("RightFuse"+" "+ str(RF))
pcb = onto.Component("PCB"+" "+ str(PC))
front_cover.has_feature= onto.Feature("FrontCover_Feature_" + str(FCF))
left_fuse.has_feature= onto.Conform("Left_Fuse_Feature_" + str(LFF))
right_fuse.has_feature= onto.Conform("Right_Fuse_Feature_" + str(RFF))
asset_feature= onto.Feature("Drilling_Machine_Feature_" + str(DMF))
pcb_feature= onto.Feature("PCB_Feature_" + str(PBF))
pcb.has_feature= pcb_feature
Individual.has_component.append(front_cover)
Individual.has_component.append(pcb)
Individual.has_component.append(left_fuse)
Individual.has_component.append(right_fuse)
_logger.info(Individual.has_component)

```

Figure B-15: Representation of the Individual and id creation in the Camera Station

```

destroy_entity(onto.Assembly("3_" + str(st3))) #This is the passage expressed before, since the id of the individuals are
        #the same then it is possible to eliminate the individual produced before
        #and keep the one produced now

front_cover= onto.Component("FrontCover"+" "+str(FC))
left_fuse = onto.Component("LeftFuse"+" "+ str(LF))
right_fuse = onto.Component("RightFuse"+" "+ str(RF))
pcb = onto.Component("PCB"+" "+ str(PC))
front_cover.has_feature= onto.Feature("FrontCover_Feature_" + str(FCF))
left_fuse.has_feature= onto.Conform("Left_Fuse_Feature_" + str(LFF))
right_fuse.has_feature= onto.Conform("Right_Fuse_Feature_" + str(RFF))
asset_feature= onto.Feature("Drilling_Machine_Feature_" + str(DMF))
pcb_feature= onto.Feature("PCB_Feature_" + str(PBF))
pcb.has_feature= pcb_feature
Individual.has_component.append(front_cover)
Individual.has_component.append(pcb)
Individual.has_component.append(left_fuse)
Individual.has_component.append(right_fuse)
_logger.info(Individual.has_component)

```

Figure B-16: Representation of Components insertion

In the images B_15 and B_16 are contained the operations already treated in the stations before and so they will be overlooked.

```

_logger.info(Individual.has_component)
if Im_string == "CIRCLE": #if the image on the pcb is a circle it means that the condition of
                          #the pcb are optimal and so the piece will not need any repair action
                          #or it does not have to be discarded. After the check of the camera
                          #it is possible to determine the quality of the product, which is the
                          #result of the state of the pcb and the state of the front cover

    _logger.info(Im_string)
    image = onto.Blank_Image(PCB_image())
    _logger.info(image)
    quality = onto.Quality(product_quality())
    Individual.has_quality= quality
    pcb.has_image = image
    image.determine = onto.Feature("PCB_Feature_" + str(PBF))
    pcb_feature.brings_to= quality
    asset_feature.results_in= quality

```

Figure B-17: Operations in the case of a Circle image on the PCB

```

if Im_string == "SQUARE": #if the image on the pcb is a square it means that the condition of
                          #the pcb are sub optimal and so the piece will need a repair action
                          #in order to sell the product. After the check of the camera
                          #it is possible to determine the quality of the product, which is the
                          #result of the state of the pcb and the state of the front cover

    _logger.info(Im_string)
    image = onto.Square_Image(PCB_image())
    _logger.info(image)
    quality = onto.Quality(product_quality())
    Individual.has_quality= quality
    pcb.has_image = image
    image.determine = onto.Feature("PCB_Feature_" + str(PBF))
    pcb_feature.brings_to= quality
    asset_feature.results_in= quality
if Im_string == "TRIANGLE": #if the image on the pcb is a triangle it means that the condition of
                             #the pcb is bad and so the piece can not be repaired and it needs to
                             #be discarded. After the check of the camera
                             #it is possible to determine the quality of the product, which is the
                             #result of the state of the pcb and the state of the front cover

    _logger.info(Im_string)
    image = onto.Triangle_Image(PCB_image())
    _logger.info(image)
    quality = onto.Quality(product_quality())
    Individual.has_quality= quality
    _logger.info(Individual.has_quality)
    pcb.has_image = image
    image.determine = onto.Feature("PCB_Feature_" + str(PBF))
    _logger.info(pcb.has_image)
    _logger.info(image.determine)
    pcb_feature.brings_to= quality
    asset_feature.results_in= quality
else:
    _logger.info("The piece is not at the camera station")

```

Figure B-18: Operations in the case of a Square and Triangle image on the PCB

In the images B_17 and B_18 it is possible to see the elements introduced in the python code to determine the pcb images. The pcb images determines the relationships present in the images B_17 and B_18 like the quality of the Individual or the relationship between the quality and the image on the pcb. In particular, even if the quality of the Individual is determined in this moment, there are not the actuators necessary to operate on the Individual in real time. For this reason, even if the Individual is defective, the operations of the next stations will be executed even if it means to waste components and energy.

```

if Station == "BackCoverPLC":
#-----
#This is the sensor dedicated to determine the position of the product, in particular if the
#product has passed through the Backcover station. This is the first station to be visited
#by the Assembly. Here the back cover is inserted on the product
#-----

if Boolean == 1.0:
    BackCover_Station_Arrival=Timestamp
    Individual= onto.Assembly(new_piece_st5()) #This function is the same we have seen in the stations before. It
    #define the product that arrives at the Camera station. As for the cases
    #before, each time a Boolean equal to 1 is registered a new individual
    #is created with the name "5_" + the number of Boolean equal to 1 have
    #been registered.

    id =onto.Assembly_ID(product_id_st5()) #This is the function to create the id of the product worked by the
    #Robot Cell which is made in the same way of the ones at the stations
    #before. Each time a Boolean equal to 1 is registered an id is emitted
    #named "Process_1 Product_" + the number of Boolean equal to 1 have
    #been registered

```

Figure B-19: Individual and id creation in the BackCover Station

```

Individual.has_id=id
TS[Station+"_"+str(id)]=Timestamp
_logger.info(Individual.has_id)
if onto.Assembly("5_" + str(st5)).has_id == onto.Assembly("4_" + str(st4)).has_id: #This passage is same before to execute the
#tracking of the product, if the id is
#generated and it is the same of the one
#obtained at the Robot Cell then it is
#possible to eliminate the Individual before
#and keep the new one

destroy_entity(onto.Assembly("4_"+str(st4))) #This is the passage expressed before, since the id of the individuals are
#the same then it is possible to eliminate the individual produced before
#and keep the one produced now

```

Figure B-20: id comparison in the BackCover Station

```

Individual.has_position= onto.BackCover_Station
front_cover= onto.Component("FrontCover"+"_"+str(FC))
left_fuse = onto.Component("LeftFuse"+"_" + str(LF))
right_fuse = onto.Component("RightFuse"+"_" + str(RF))
pcb = onto.Component("PCB"+"_" + str(PC))
front_cover.has_feature= onto.Feature("FrontCover_Feature_" + str(FCF))
left_fuse.has_feature= onto.Conform("Left_Fuse_Feature_" + str(LFF))
right_fuse.has_feature= onto.Conform("Right_Fuse_Feature_" + str(RFF))
Individual.has_component.append(front_cover)
Individual.has_component.append(left_fuse)
Individual.has_component.append(right_fuse)
Individual.has_component.append(pcb)
quality=onto.Quality("Product_Quality_" + str(PrQ))
Individual.has_quality= quality
Im_value,Im_string,Im_bit= get_new_image(image_repository)
_logger.info(Im_string)
back_cover= onto.Component(BackCover()) #Since in the FML there are not the actuators necessary to operate in real time
#the back cover is inserted even if the product is defective

back_cover.has_feature=onto.Conform()
Individual.has_component.append(back_cover)
pcb.has_feature= onto.Feature("PCB_Feature_" + str(PBF))
pcb_feature= onto.Feature("PCB_Feature_" + str(PBF))
pcb_feature.brings_to = quality

if Im_string == "CIRCLE":
    pcb.has_image = onto.Blank_Image("PCB_Image_" + str(PCBI))
    pcb_image= onto.Blank_Image("PCB_Image_" + str(PCBI))
    pcb_image.determine= pcb_feature
    _logger.info(Individual.has_component)
    _logger.info(Individual.has_position)

if Im_string == "SQUARE":
    pcb.has_image = onto.Square_Image("PCB_Image_" + str(PCBI))
    pcb_image= onto.Square_Image("PCB_Image_" + str(PCBI))
    pcb_image.determine= pcb_feature
    _logger.info(Individual.has_component)
    _logger.info(Individual.has_position)

```

Figure B-21: Components insertion in the BackCover Station


```

if Im_string == "TRIANGLE":
    pcb.has_image = onto.Triangle_Image("PCB_Image_" + str(PCBI))
    pcb_image= onto.Triangle_Image("PCB_Image_" + str(PCBI))
    pcb_image.determine= pcb_feature
    _logger.info(Individual.has_component)
    _logger.info(Individual.has_position)
else:
    _logger.info("The piece is not at the back cover station")

```

Figure B-22: Operations to execute in case of a Triangle image

Accordingly the operations shown in the images B_19-B_22 are going to be executed even if the Individual is defective. These operations are the same described for the stations before so they have already been examined. The same goes for the images B_23-B_28 where the Press Station and the Manual Station are going to be treated

```

if Station == "PressPLC":
#-----
#This is the sensor dedicated to determine the position of the product, in particular if the
#product has passed throught the Press station. This is the first station to be visited by the
#Assembly. Here the product is pressed to assembly all the components of the product
#-----

if Boolean == 1.0:
    Press_Station_Arrival=Timestamp
    Individual= onto.Assembly(new_piece_st6()) #This function is the same we have seen in the stations before. It
#define the product that arrives at the Camera station. As for the cases
#before, each time a Boolean equal to 1 is registred a new individual
#is created with the name "5_" + the number of Boolean equal to 1 have
#been registred.

    id =onto.Assembly_ID(product_id_st6()) #This is the function to create the id of the product worked by the
#Robot Cell which is made in the same way of the ones at the stations
#before. Each time a Boolean equal to 1 is registred an id is emitted
#named "Process_1 Product_" + the number of Boolean equal to 1 have
#been registred

    Individual.has_id=id
    TS[Station+"_"+str(id)]=Timestamp
    _logger.info(Individual.has_id)
    Individual.has_position= onto.Press_Station
    Im_value,Im_string,Im_bit= get_new_image(image_repository)
    _logger.info(Im_string)
    if onto.Assembly("6_" + str(st6)).has_id == onto.Assembly("5_" + str(st5)).has_id: #This passage is same before to execute the
#tracking of the product, if the id is
#generated and it is the same of the one
#obtained at the BackCover station then it is
#possible to eliminate the individual before
#and keep the new one

        destroy_entity(onto.Assembly("5_"+str(st5))) #This is the passage expressed before, since the id of the individuals are
#the same then it is possible to eliminate the individual produced before
#and keep the one produced now

```

Figure B-23: Individual and id creation and id comparison in the Press Station

```

front_cover= onto.Component("FrontCover"+"_"+str(FC)) #Since in the FML there are not the actuators necessary to operate in real time
#the all the operations below are executed even if the product is defective

left_fuse = onto.Component("LeftFuse_" + str(LF))
right_fuse = onto.Component("RightFuse_" + str(RF))
pcb = onto.Component("PCB_" + str(PC))
back_cover= onto.Component("BackCover_" +str(BC))
quality=onto.Quality("Product_Quality_" + str(PrQ))
Individual.has_quality= quality
front_cover.has_feature= onto.Feature("FrontCover_Feature_" + str(FCF))
left_fuse.has_feature= onto.Conform("Left_Fuse_Feature_" + str(LFF))
right_fuse.has_feature= onto.Conform("Right_Fuse_Feature_" + str(RFF))
pcb.has_feature= onto.Feature("PCB_Feature_" + str(PBF))
pcb_feature= onto.Feature("PCB_Feature_" + str(PBF))
Individual.has_component.append(front_cover)
Individual.has_component.append(left_fuse)
Individual.has_component.append(right_fuse)
Individual.has_component.append(pcb)
Individual.has_component.append(back_cover)
pcb_feature.brings_to= quality

if Im_string == "TRIANGLE":
    pcb.has_image = onto.Triangle_Image("PCB_Image_" + str(PCBI))
    pcb_image= onto.Triangle_Image("PCB_Image_" + str(PCBI))
    pcb_image.determine= pcb_feature

if Im_string == "CIRCLE":
    pcb.has_image = onto.Blank_Image("PCB_Image_" + str(PCBI))
    pcb_image= onto.Blank_Image("PCB_Image_" + str(PCBI))
    pcb_image.determine= pcb_feature
    _logger.info(Individual.has_component)
    _logger.info(Individual.has_position)

if Im_string == "SQUARE":
    pcb.has_image = onto.Square_Image("PCB_Image_" + str(PCBI))
    pcb_image= onto.Square_Image("PCB_Image_" + str(PCBI))
    pcb_image.determine= pcb_feature

```

Figure B-24: Components insertion in the Press Station

```

else:
    _logger.info("The piece is not at the press station")

if Station == "PressEnergy": #This is the sensor dedicated to determine the energy value of the Drill station so
#this sensor just reports the energy of the station and not if the product is passing by

if not (node_id in lista_bool):
    #_logger.info(onto.Manual_Station_Operating_Time)
    _logger.info("The energy of the machine is",Energy)

```

Figure B-25: Representation of the Press Energy Station

```

if Station == "ManualPLC":
#-----
#This is the sensor dedicated to determine the position of the product, in particular if the
#product has passed through the Press station. This is the last station to be visited by the
#Assembly. Here the product is removed from the line
#-----

if Boolean == 1.0:
    Manual_Station_Arrival=Timestamp
    Individual= onto.Assembly(new_piece_st7()) #This function is the same we have seen in the stations before. It
#define the product that arrives at the Camera station. As for the cases
#before, each time a Boolean equal to 1 is registered a new individual
#is created with the name "7_" + the number of Boolean equal to 1 have
#been registered.

id =onto.Assembly_ID(product_id_st7()) #This is the function to create the id of the product worked by the
#Robot Cell which is made in the same way of the ones at the stations
#before. Each time a Boolean equal to 1 is registered an id is emitted
#named "Process_1 Product_" + the number of Boolean equal to 1 have
#been registered

Individual.has_id=id
TS[Station+"_"+str(id)]=Timestamp
_logger.info(Individual.has_id)
Individual.has_position= onto.Manual_Station
In_value,In_string,Im_bit= get_new_image(image_repository)
if onto.Assembly("7_" + str(st7)).has_id == onto.Assembly("6_" + str(st6)).has_id: #This passage is same before to execute the
#tracking of the product, if the id is
#generated and it is the same of the one
#obtained at the BackCover station then it is
#possible to eliminate the individual before
#and keep the new one

destroy_entlty(onto.Assembly("6_" +str(st6))) #This is the passage expressed before, since the id of the individuals are
#the same then it is possible to eliminate the individual produced before
#and keep the one produced now

```

Figure B-26: Individual and id creation and id comparison in the Manual Station

```

_logger.info(Individual)
front_cover= onto.Component("FrontCover"+"_"+str(FC)) #This is the last station of the FML. Here the product is extracted and its
#state is evaluated. So after this station we are able to determine the quality
#of the obtained product

left_fuse = onto.Component("LeftFuse_" + str(LF))
right_fuse = onto.Component("RightFuse_" + str(RF))
pcb = onto.Component("PCB_" + str(PC))
back_cover= onto.Component("BackCover_"+str(BC))
quality=onto.Quality("Product_quality_" + str(PrQ))
Individual.has_quality= quality
front_cover.has_feature= onto.Feature("FrontCover_Feature_" + str(FCF))
left_fuse.has_feature= onto.Conform("Left_Fuse_Feature_" + str(LFF))
right_fuse.has_feature= onto.Conform("Right_Fuse_Feature_" + str(RFF))
pcb.has_feature= onto.Feature("PCB_Feature_" + str(PBF))
pcb_feature= onto.Feature("PCB_Feature_" + str(PBF))
Individual.has_component.append(front_cover)
Individual.has_component.append(left_fuse)
Individual.has_component.append(right_fuse)
Individual.has_component.append(pcb)
Individual.has_component.append(back_cover)
pcb_feature.brings_to= quality
_logger.info(Individual.has_component)
_logger.info(Individual.has_position)
process = onto.Assembly_Line_Process()
Individual.has_assembly_process= process
_logger.info(Individual.has_assembly_process)
quality.is_connected_to= process

if Im_string == "TRIANGLE":
    pcb.has_image = onto.Triangle_Image("PCB_Image_" + str(PCBI))
    pcb_image= onto.Triangle_Image("PCB_Image_" + str(PCBI))
    pcb_image.determine= pcb_feature

if Im_string == "CIRCLE":
    pcb.has_image = onto.Blank_Image("PCB_Image_" + str(PCBI))
    pcb_image= onto.Blank_Image("PCB_Image_" + str(PCBI))
    pcb_image.determine= pcb_feature

```

Figure B-27: Components insertion in the Manual Station

```

if Im_string == "CIRCLE":
    pcb.has_image = onto.Blank_Image("PCB_Image_" + str(PCBI))
    pcb_image= onto.Blank_Image("PCB_Image_" + str(PCBI))
    pcb_image.determine= pcb_feature

if Im_string == "SQUARE":
    pcb.has_image = onto.Square_Image("PCB_Image_" + str(PCBI))
    pcb_image= onto.Square_Image("PCB_Image_" + str(PCBI))
    pcb_image.determine= pcb_feature
else:
    _logger.info("The piece is not at the press station")

if Station == "ManualEnergy": #This is the sensor dedicated to determine the energy value of the Drill station so
#this sensor just reports the energy of the station and not if the product is passing by

if not (node_id in lista_bool):
    #_logger.info(onto.Manual_Station_Operating_Time)
    _logger.info("The energy of the machine is",Energy)

_logger.info(list(onto.Individuals()))

save_function(onto)

```

Figure B-28: Representation of the Manual Energy Station

B.2. Python code for an hypothetical enhanced FML

For enhanced FML is intended a line characterized by the necessary actuators to operate in real time to a product, which is being processed by the line. The ontology developed in this thesis work is already able to operate in real time to manage the repair and disassembly process of an eventual product. Instead the FML in the Laboratory of Industry 4.0 at Politecnico di Milano has not the actuators to do so. For this reason a new code has been developed to simulate the behavior of an hypothetical Repair and Disassembly station present in the FML.

The new python code is able to show all the potentialities of the ontology has the same structure of the one used to elaborate the data coming from the FML, but there are

some evolutions. The first one is the introduction in the code of the Repair and Disassembly station, while the second one is the activation of the reasoner after each time a product goes to a different station. In this chapter will be treated only the enhanced features of the Repair and Disassembly station.

```

if Station == "FrontCoverPLC":
    if Boolean == 1.0:
        _logger.info(f"A new product has entered in the line at time {Timestamp}")
        Individual= onto.Assembly(new_piece_st1())
        # _logger.info(f"Dopo Assembly")
        id =onto.Assembly_ID(product_id_st1())
        Individual.has_id=id
        TS[Station+"_"+str(id)]=Timestamp
        _logger.info(Individual)
        _logger.info(Individual.has_id)
        Individual.has_component=[]

        if len(front_cover_list) > 0:
            destroy_entity(onto.Assembly("_" + str(st7)))
            front_cover= onto.Component("FrontCover"+"_"+str(FC))
            _logger.info(f"The front cover has been inserted at time {Timestamp}")
            #onto.Quality_Specificaton=[]
            _logger.info(id)
            Individual.has_component.append(front_cover)
            _logger.info(Individual.has_component)
            front_cover.has_feature=onto.Feature("FrontCover_Feature_" + str(FCF))
            _logger.info(front_cover.has_feature)
            Individual.has_position=onto.FrontCover_Station
            _logger.info(Individual.has_position)

        else:
            front_cover= onto.Component(FrontCover())
            _logger.info(f"The front cover has been inserted at time {Timestamp}")
            #onto.Quality_Specificaton=[]
            _logger.info(id)
            Individual.has_component.append(front_cover)
            _logger.info(Individual.has_component)
            front_cover.has_feature=onto.Feature(frontcover_feature())
            _logger.info(front_cover.has_feature)
            Individual.has_position=onto.FrontCover_Station
            _logger.info(Individual.has_position)

```

Figure B-29: Front Cover recycle at the FrontCover Station

In the image B_29 the Disassembly process is being treated. As a matter of fact concerning the FrontCover Station there may be some recycled components to deal with. If there are recycled components then the `front_cover_list` is different from 0, so it is not necessary to create a new front cover individual but it is possible to use the one already inside the ontology. As a matter of fact it has been considered the case of an already existing population in which the Individual obtained was a discarded product and so the good state components have been recycled. Instead if the `front_cover_list` is equal to 0 no pieces are recycled and so it is necessary to utilize a new front cover. The same procedure is utilized for the Robot Cell where the fuses are eventually re-inserted on the product as it is possible to see in the image B_30

```

if len(left_fuse_list) > 0:
    left_fuse_list.pop(0)
    left_fuse = onto.Component("LeftFuse"+"_" + str(LF))
    left_fuse.has_feature= onto.Conform("Left_Fuse_Feature_" + str(LFF))
else:
    left_fuse = onto.Component(LeftFuse())
    left_fuse.has_feature= onto.Conform(left_fuse_feature())
    _logger.info(left_fuse.has_feature)
if len(right_fuse_list) > 0:
    right_fuse_list.pop(0)
    right_fuse = onto.Component("RightFuse"+"_" + str(RF))
    right_fuse.has_feature= onto.Conform("Right_Fuse_Feature_" + str(RFF))
else:
    right_fuse = onto.Component(RightFuse())
    right_fuse.has_feature= onto.Conform(right_fuse_feature())
    _logger.info(right_fuse.has_feature)
_logger.info(f"The PCB and fuses are inserted at time {Timestamp}")
Individual.has_component.append(left_fuse)
Individual.has_component.append(right_fuse)
_logger.info(Individual.has_component)
if len(front_cover_list) > 0:
    front_cover_list.pop(0)
    front_cover= onto.Component("FrontCover"+"_" +str(FC))
    Individual.has_component.append(front_cover)
    front_cover.has_feature= onto.Feature("FrontCover_Feature_" + str(FCF))

```

Figure B-30: Fuses recycle at the Robot Cell

Now the last element to consider is when these components are sent to the Disassembly Station. This part is treated in the following images

```

if Im_string == "TRIANGLE":
    _logger.info(f"The Product does not need this operation")
    front_cover= onto.Component("FrontCover"+"_" +str(FC))
    left_fuse = onto.Component("LeftFuse"+"_" + str(LF))
    right_fuse = onto.Component("RightFuse"+"_" + str(RF))
    pcb = onto.Component("PCB"+"_" + str(PC))
    quality=onto.Quality("Product_Quality_" + str(PrQ))
    front_cover.has_feature= onto.Feature("FrontCover_Feature_" + str(FCF))
    left_fuse.has_feature= onto.Conform("Left_Fuse_Feature_" + str(LFF))
    right_fuse.has_feature= onto.Conform("Right_Fuse_Feature_" + str(RFF))
    pcb.has_feature= onto.Feature("PCB_Feature_" + str(PBF))
    pcb_feature= onto.Feature("PCB_Feature_" + str(PBF))
    Individual.has_quality= quality
    Individual.has_component.append(pcb)
    Individual.has_component.append(front_cover)
    Individual.has_component.append(left_fuse)
    Individual.has_component.append(right_fuse)
    pcb.has_image = onto.Triangle_Image("PCB_Image_" + str(PCBI))
    pcb_image= onto.Triangle_Image("PCB_Image_" + str(PCBI))
    pcb_image.determine= pcb_feature
    pcb_feature.brings_to= quality
    process= onto.Assembly_Line_Process()
    Individual.has_assembly_process=process
    _logger.info(Individual.has_assembly_process)
    quality.is_connected_to= process
    #
    # Disassembly
    #
    value = "FrontCover"+"_" +str(FC)
    p_tuple = ("DisassemblyStation",
              "front_cover",
              datetime.datetime.now(),

```

Figure B-31: Representation of the Disassembly Station 1/2

```

#
# Disassembly
#
value = "FrontCover"+"_" +str(FC)
p_tuple = ("DisassemblyStation",
           "front_cover",
           datetime.datetime.now(),
           value)
_logger.info(f"Consumer -> Dis Producer activated -> Received DisassemblyStation front_cover {value}")
dis_queue.put_nowait(p_tuple)
_logger.info(f"Consumer -> Dis Producer sendend -> Received DisassemblyStation front_cover {value}")
#
value = "RightFuse"+"_" + str(RF)
p_tuple = ("DisassemblyStation",
           "right_fuse",
           datetime.datetime.now(),
           value)
_logger.info(f"Consumer -> Dis Producer activated -> Received DisassemblyStation right_fuse {value}")
dis_queue.put_nowait(p_tuple)
_logger.info(f"Consumer -> Dis Producer sendend -> Received DisassemblyStation right_fuse {value}")
#
value = "LeftFuse"+"_" + str(LF)
p_tuple = ("DisassemblyStation",
           "left_fuse",
           datetime.datetime.now(),
           value)
_logger.info(f"Consumer -> Dis Producer activated -> Received DisassemblyStation left_fuse {value}")
dis_queue.put_nowait(p_tuple)
_logger.info(f"Consumer -> Dis Producer sendend -> Received DisassemblyStation left_fuse {value}")

```

Figure B-32: Representation of the Disassembly Station 2/2

As represented in the images B_31 and B_32 when there is a pcb with a *triangle image*, meaning that the product must be discarded, then the good state components are sent to the Disassembly station. In this case the tuple to fill is that Station is the Disassembly Station, the node_id is the component to recycled and the value is the id of the component to be recycled. So after these components are sent to the Disassembly Station then the lists seen in the images B_29 and B_30 are filled with the respective components.

For what concerns the Repair station, the following image will be considered

```

def processa_record(onto, Station, node_id, Timestamp, valore, TS, rep_queue, dis_queue):
    """
    Process Record in the consumer
    """
    try:
        if node_id in lista_bool:
            Boolean = valore
            Energy = None
        else:
            Boolean = "99"
            Energy = valore

        rep_current_object = None
        if Station == "RepairStation":
            _logger.info(f"Processa_record per riparazione -> station = {Station}, node_id = {node_id}, timestamp = {Timestamp}, valore = {valore}")
            Station = node_id
            rep_current_object = valore
            Boolean = 1.0

        _logger.info(f"Processa_record -> station = {Station}, node_id = {node_id}, timestamp = {Timestamp}, valore = {valore}")

```

Figure B-33: Representation of the start of the code

In the image B_33 the initial part of the code is presented and it is possible to see the structure of the Repair station. The first element to consider is rep_current_object which at the start of the execution is equal to none, meaning that no piece needs to be repaired. Then it is possible to evaluate the Repair Station structure which is composed by station, node_id, timestamp and valore. The Station is always equal to Repair Station, the node_id is the station in which the component must be re-inserted and rep_current_object is the piece which needs to be repaired. In the following images the application of the Repair Station is going to be seen.

From the image B_34 it is possible to see that when the Individual arrives at the Robot Cell two situations may happen: in the first case there is a front cover which is coming

back from the Repair Station and so since it has been repaired the front cover has not the *Repairable* feature anymore but now it has the *Conform* feature. In the second case there are no pieces coming back from the Repair station and so no repaired pieces have to be processed.

```

if Station == "RobotCellPLC":
    if Boolean == 1.0:
        if rep_current_object:
            # Back from repair station
            _logger.info(f"Questo è il caso di un oggetto {rep_current_object} che rientra")
            front_cover.has_feature = onto.Conform("FrontCover_Feature_" + str(FCF))
            _logger.info(f"{front_cover.has_feature}")
            _logger.info(f"ok")
        else:
            # Normal piece
            Robot_Cell_Arrival=Timestamp
            Individual= onto.Assembly(new_piece_st3())
            #_logger.info(f"Dopo Assembly")
            id =onto.Assembly_ID(product_id_st3())
            Individual.has_id=id
            if onto.Assembly("3_" + str(st3)).has_id == onto.Assembly("2_" + str(st2)).has_id:
                destroy_entity(onto.Assembly("2_" + str(st2)))
                TS[Station+"_"+str(id)]=Timestamp
                _logger.info(Individual)

```

Figure B-34: Front Cover retrieval from the Repair Station

```

if len(front_cover_list) > 0:
    front_cover_list.pop(0)
    front_cover= onto.Component("FrontCover"+"_"+str(FC))
    Individual.has_component.append(front_cover)
    front_cover.has_feature= onto.Feature("FrontCover_Feature_" + str(FCF))
else:
    front_cover= onto.Component("FrontCover"+"_"+str(FC))
    Individual.has_component.append(front_cover)

if sh_str == "SHACKER ON":
    # Normal piece, shacker on to repair
    value = "FrontCover"+"_"+str(FC)
    p_tuple = ("RepairStation",
              "RobotCellPLC",
              datetime.datetime.now(),
              value)
    _logger.info(f"Consumer -> Rep Producer activated -> Received RepairStation RepairNode {value}")
    destroy_entity(onto.Feature("FrontCover_Feature_" + str(FCF)))
    rep_queue.put_nowait(p_tuple)
    _logger.info(f"Consumer -> Rep Producer sendd -> Received RepairStation RepairNode {value}")
else:
    # SHAKER OFF
    # Normal piece, shacker off, nothing to repair
    front_cover.has_feature= onto.Feature("FrontCover_Feature_" + str(FCF))

```

Figure B-35: Sending of the front cover to the Repair Station

In the image B_35, in case the shacker was active, the front cover needs to be repaired and so it is sent to the Repair station. In this case it is possible to see the tuple that must be filled in order to send the component to the Repair Station. Station is always Repair Station, the node_id is equal to Robot Cell, so the front cover must re-enter in the Robot Cell and the value is equal to the id of the front cover individual. It is also possible to see that the frontcover_feature is eliminated in the case the front cover is repaired. Now the component has not the *Repairable* feature anymore since it is sent to be repaired and instead, as shown in image B_34, it has the *Conform* feature now. This is one of the two parts of the code in which the components are sent to the Repair Station, the second one is the BackCover Station.

```

if Station == "BackCoverPLC":
    if Boolean == 1.0:
        if rep_current_object:
            # Back from repair station
            _logger.info(f"Questo è il caso di un oggetto {rep_current_object} che rientra")
            pcb_feature = onto.Conform(PCB_feature())
            pcb.has_feature = pcb_feature
            back_cover = onto.Component(BackCover())
            back_cover.has_feature = onto.Conform()
            Individual.has_component.append(back_cover)
            pcb_feature.brings_to = onto.Quality("Product_Quality_" + str(PrQ))
            _logger.info(f"{pcb.has_feature}")
            _logger.info(Individual)
            pcb.has_image = onto.Square_Image("PCB_Image_" + str(PCBI))
            pcb_image = onto.Square_Image("PCB_Image_" + str(PCBI))
            pcb_feature.brings_to = quality
            _logger.info(Individual.has_component)
            _logger.info(Individual.has_position)

        else:
            # Normal piece
            BackCover_Station_Arrival = Timestamp
            Individual = onto.Assembly(new_piece_st5())
            id = onto.Assembly_ID(product_id_st5())
            Individual.has_id = id
            TS[Station + "_" + str(id)] = Timestamp
            _logger.info(Individual.has_id)
            if onto.Assembly("5_" + str(st5)).has_id == onto.Assembly("4_" + str(st4)).has_id:
                destroy_entity(onto.Assembly("4_" + str(st4)))
                Individual.has_position = onto.BackCover_Station
                front_cover = onto.Component("FrontCover_" + str(FC))
                left_fuse = onto.Component("LeftFuse_" + str(LF))
                right_fuse = onto.Component("RightFuse_" + str(RF))
                pcb = onto.Component("PCB_" + str(PC))
                front_cover.has_feature = onto.Feature("FrontCover_Feature_" + str(FCF))

```

Figure B-36: Retrieval of the pcb from the Repair Station

```

else:
    # Normal piece
    BackCover_Station_Arrival = Timestamp
    Individual = onto.Assembly(new_piece_st5())
    id = onto.Assembly_ID(product_id_st5())
    Individual.has_id = id
    TS[Station + "_" + str(id)] = Timestamp
    _logger.info(Individual.has_id)
    if onto.Assembly("5_" + str(st5)).has_id == onto.Assembly("4_" + str(st4)).has_id:
        destroy_entity(onto.Assembly("4_" + str(st4)))
        Individual.has_position = onto.BackCover_Station
        front_cover = onto.Component("FrontCover_" + str(FC))
        left_fuse = onto.Component("LeftFuse_" + str(LF))
        right_fuse = onto.Component("RightFuse_" + str(RF))
        pcb = onto.Component("PCB_" + str(PC))
        front_cover.has_feature = onto.Feature("FrontCover_Feature_" + str(FCF))
        left_fuse.has_feature = onto.Conform("Left_Fuse_Feature_" + str(LFF))
        right_fuse.has_feature = onto.Conform("Right_Fuse_Feature_" + str(RFF))
        Individual.has_component.append(front_cover)
        Individual.has_component.append(left_fuse)
        Individual.has_component.append(right_fuse)
        Individual.has_component.append(pcb)
        quality = onto.Quality("Product_Quality_" + str(PrQ))
        Individual.has_quality = quality
        Im_value, Im_string, Im_bit = get_new_image(image_repository)
        _logger.info(Im_string)

    if Im_string == "SQUARE":
        # Normal piece, SQUARE, to repair
        value = "PCB_" + str(PC)
        p_tuple = ("RepairStation",
                  "BackCoverPLC",
                  datetime.datetime.now(),
                  value)
        _logger.info(f"Consumer -> Rep Producer activated -> Received RepairStation RepairNode {value}")
        destroy_entity(onto.Feature("PCB_Feature_" + str(PBF)))

```

Figure B-37: Sending of the pcb to the Repair Station

As it is possible to see from the images B_36 and B_37 the situation is the same expressed for the Robot Cell but with the pcb in this case. As a matter of fact if the image on the pcb is a square then the pcb needs to be repaired and so it is sent to the repair station. In this case the tuple is compiled in the following way: the station is the Repair Station, the node_id is the BackCover Station and the value is the id of the back cover individual. After the product is sent to the Repair Station the pcb feature is eliminated and instead the *Conform* one is given, as illustrated in the image B_36.

List of Figures

Figure 1-1: Evolution in time of the Industrial Revolutions	1
Figure 1-2: Representation of the pragmatic and semantic side of an ontology	10
Figure 2-1: Retrieval of articles from the databases	14
Figure 2-2: Distribution of the collected articles	16
Figure 2-3: Percentages of ZDM articles which deals	17
Figure 2-4: Representation of the sectors in which ZDM is applied	18
Figure 2-5: Representation of which ZDM strategy is applied in each article	18
Figure 2-6: Distribution of the eliminated articles after the 1° screening	20
Figure 2-7: Distribution of the articles after the 2° screening	20
Figure 2-8: Representation of the eligible documents	21
Figure 4-1: Representation of the methontology method	33
Figure 4-2: Entity class of the BFO ontology	35
Figure 4-3: Continuant class of the BFO ontology	36
Figure 4-4: Occurrent class of the BFO ontology	36
Figure 4-5: Independent Continuant class of the BFO ontology	37
Figure 4-6: Specifically Dependent Continuant class	38
Figure 4-7: Material Entity class of the BFO ontology	38
Figure 4-8: Complete representation of the BFO ontology	39
Figure 4-9: Identifier class of the IAO ontology	40
Figure 4-10: Specifically Dependent Continuant class of the IAO ontology	41
Figure 4-11: Planned Process class of the IAO ontology	41
Figure 4-12: Artifact and Transducer class of the CCO ontology	42
Figure 4-13: Artifact Function class of the CCO ontology	42
Figure 4-14: Artifact Identifier class of the CCO ontology	43
Figure 4-15: Artifact Function Specification and Quality Specification	43
Figure 4-16: Assembly class of the ORMA+ ontology	47
Figure 4-17: Assembly Position class of the ORMA+ ontology	48
Figure 4-18: Component class of the ORMA+ ontology	50

Figure 4-19: Has_component property of the ORMA+ ontology	51
Figure 4-20: : Has_position property of the ORMA+ ontology	51
Figure 4-21: Asset class of the ORMA+ ontology	52
Figure 4-22: Failure Mode and Effect Analysis and	53
Figure 4-23: Asset Component Class of the ORMA+ ontology	54
Figure 4-24: Asset Function class of the ORMA+ ontology.....	55
Figure 4-25: Failure Cause class of the ORMA+ ontology	56
Figure 4-26: Has_failure_cause property of the ORMA+ ontology	56
Figure 4-27: Has_failure_event property of the ORMA+ ontology.....	57
Figure 4-28: Has_failure_mode_and_effect property of the ORMA+ ontology.....	57
Figure 4-29: Has_failure_mode_code property of the ORMA+ ontology	57
Figure 4-30: Has_function property of the ORMA+ ontology	57
Figure 4-31: Quality class of the ORMA+ ontology	58
Figure 4-32: Physical Object Quality class of the ORMA+ ontology.....	58
Figure 4-33: Feature class of the ORMA+ ontology.....	60
Figure 4-34: Specification class of the ORMA+ ontology	61
Figure 4-35: Component Specifics and Process Specifics class of the ORMA+ ontology	62
Figure 4-36: Component_Specifics class of the ORMA+ ontology.....	63
Figure 4-37: Has_feature property of the ORMA+ ontology	64
Figure 4-38: Has_quality property of the ORMA+ ontology	64
Figure 4-39: Has_specifics property of the ORMA+ ontology	64
Figure 4-40: Process class of the ORMA+ ontology	66
Figure 4-41: Image Recognition class of the ORMA+ ontology	67
Figure 4-42: Process Boundary class of the ORMA+ ontology	68
Figure 4-43: Transducer class of the ORMA+ ontology	69
Figure 4-44: Has_assembly_process property of the ORMA+ ontology	70
Figure 4-45: Has_monitoring property of the ORMA+ ontology.....	70
Figure 4-46: Has_process_specifics property of the ORMA+ ontology.....	70
Figure 4-47: Has_state property of the ORMA+ ontology.....	71
Figure 4-48: determine property of the ORMA+ ontology.....	71

Figure 4-49: Results_in property of the ORMA+ ontology	71
Figure 4-50: Brings_to property of the ORMA+ ontology	71
Figure 5-1: Representation of the Flexible Manufacturing Line in the Industry 4.0 Lab [23]	72
Figure 5-2: Representation of the cases deal with the ORMA+ ontology	75
Figure 5-3: Representation of the individual Product.....	76
Figure 5-4: Property assertion of product.....	77
Figure 5-5: Product at the Drill_Station	78
Figure 5-6: Property Assertion of Product.....	78
Figure 5-7: Individual drilling machine	78
Figure 5-8: Property assertion of drilling machine.....	78
Figure 5-9: Representation of the individual Drilling Machine Feature	79
Figure 5-10: Representation of the property assertions of Drilling Machine Feature .	79
Figure 5-11: Product individual at the Robot Cell.....	80
Figure 5-12: Property assertion of Product.....	80
Figure 5-13: Representation of the Image individual.....	81
Figure 5-14: Representation of the Property Assertion of Image	81
Figure 5-15: Representation of the individual PCB	82
Figure 5-16: Representation of the Property Assertion of PCB	82
Figure 5-17: Representation of the individual PCB_Feature.....	82
Figure 5-18: Representation of the Property assertion of PCB_Feature	83
Figure 5-19: Representation of the individual Product at BackCover_Station.....	83
Figure 5-20: Representation of the Property Assertion of Product	84
Figure 5-21: Representation of the individual Product at Press_Station.....	84
Figure 5-22: Representation of the Property Assertion of Product	85
Figure 5-23: Representation of the individual Product at Manual_Station	86
Figure 5-24: Representation of the Property Assertion of Product with the reasoner active	86
Figure 5-25: Representation of the individual PCB_Feature.....	87
Figure 5-26: Representation of the Property Assertion of PCB_Feature with the reasoner active	87

Figure 5-27: Representation of the individual Drilling_Machine_Feature	87
Figure 5-28: Representation of the Property Assertion of Drilling_Machine_Feature with the reasoner active.....	88
Figure 5-29: Representation of the individual Process	88
Figure 5-30: Representation of the Property Assertion of Process with the reasoner active	88
Figure 5-31: Representation of the individual Quality	89
Figure 5-32: Representation of the Property Assertion of Quality with the reasoner active	89
Figure 5-33: Representation of the individual Drilling Machine	90
Figure 5-34: Representation of the Property Assertion of Drilling Machine	90
Figure 5-35:Representation of the individual Drilling Machine Feature	90
Figure 5-36: Representation of the property assertions of Drilling Machine Feature .	91
Figure 5-37: Representation of the individual PCB	91
Figure 5-38: Representation of the Property Assertion of PCB	92
Figure 5-39: Representation of the individual Drilling_Machine_Feature	93
Figure 5-40: Representation of the Property Assertion of Drilling_Machine_Feature with the reasoner active.....	93
Figure 5-41: Representation of the individual FrontCover_Feature	94
Figure 5-42: Representation of the Property Assertion of FrontCover_Feature with the reasoner active	94
Figure 5-43: Representation of the individual PCB_Feature.....	94
Figure 5-44: Representation of the Property Assertion of PCB_Feature with the reasoner active	95
Figure 5-45: Representation of the individual Product at Manual Station	95
Figure 5-46: Representation of the Property Assertion of Product with the reasoner active	95
Figure 5-47: Representation of the Individual Quality	96
Figure 5-48: Representation of the property Assertion of Quality.....	96
Figure 5-49: representation of the Individual Process	96
Figure 5-50: Representation of the property Assertion of Process with the reasoner active	97
Figure 5-51: Representation of the individual PCB	98

Figure 5-52: Representation of the Property Assertion of PCB	98
Figure 5-53: Representation of the Individual Image	98
Figure 5-54: Representation of the property assertion of Image	99
Figure 5-55: Representation of the individual FrontCover_Feature	100
Figure 5-56: Representation of the Property Assertion of FrontCover_Feature	100
Figure 5-57: Representation of the individual Component_Feature	101
Figure 5-58: Representation of the Property Assertion of PCB_Feature with the reasoner active	101
Figure 5-59: Representation of the individual Left_Fuse_Feature	101
Figure 5-60: Representation of the Property Assertion of Left_Fuse_Feature with the reasoner active	102
Figure 5-61: Representation of the individual Right_Fuse_Feature	102
Figure 5-62: Representation of the Property Assertion of Right_Fuse_Feature with the reasoner active	102
Figure 5-63: Representation of the individual Product at Camera Station	103
Figure 5-64: Representation of the Property Assertion of Product with the reasoner active	103
Figure 5-65: Representation of the individual Quality	103
Figure 5-66: Representation of the Property Assertion of Quality with the reasoner active	104
Figure 5-67: Representation of the Individual Process	104
Figure 5-68: Representation of the Property Assertion of Process with the reasoner active	104
Figure 5-69: Query to get the quality of the product.....	105
Figure 5-70: Products found by the query	105
Figure 5-71: Qualities found by the query	105
Figure 5-72: Query to get the product with Regular_Assembly_Quality	106
Figure 5-73: Product found by the query	106
Figure 5-74: Quality found by the query.....	106
Figure 5-75: Query to get the components of the product	106
Figure 5-76: Products found by the query	107
Figure 5-77: Components found by the query	107

Figure 5-78: Query to determine the Regular_Assembly_Component	108
Figure 5-79: Product found by the query	108
Figure 5-80: Component found by the query	108
Figure 5-81: Query to determine the processes	109
Figure 5-82: Product found by the query	109
Figure 5-83: Process found by the query	109
Figure 5-84: Query to determine the Regular_Assembly_Line_Process	110
Figure 5-85: Product found by the query	110
Figure 5-86: Process found by the query	110
Figure 5-87: Query to determine which product are not feasible in function of the component and asset state	110
Figure 5-88: Product and Quality found by the query	111
Figure 5-89: Asset and Asset_Feature found by the query	111
Figure 5-90: Component and Component_Feature found by the query	111
Figure 6-1: Representation of the requirements to determine the product quality ...	113
Figure 6-2: Representation of the machine states	115
Figure 6-3: Start of the python code	116
Figure 6-4: Representation of the Individual created with the population	116
Figure 6-5: Representation of the images collected at the camera station	117
Figure 6-6: Representation of the behavior of the shaker at the drilling station	117
Figure 6-7: Representation of the individual Drilling_Machine_Feature_1	119
Figure 6-8: Representation of the Property assertion of the Drilling_Machine_Feature_1	119
Figure 6-9: Activation of the reasoner of the Python code	119
Figure 6-10: Representation of the individual FrontCover_Feature_1	120
Figure 6-11: Representation of the Property assertion of the FrontCover_Feature_1	120
Figure 6-12: Representation of the individual Drilling_Machine_Feature_1	120
Figure 6-13: Representation of the Property assertion of the Drilling_Machine_Feature_1	120
Figure 6-14: Representation of the Repair Station	121
Figure 6-15: Representation of the individual conform_1	121

Figure 6-16: Representation of the Property assertion of the conform_1.....	121
Figure 6-17: Representation of the individual PCB_Feature_1.....	122
Figure 6-18: Representation of the Property assertion of the PCB_Feature_1.....	122
Figure 6-19: Representation of the individual PCB_Feature_2.....	123
Figure 6-20: Representation of the Property assertion of the PCB_Feature_2 with the reasoner active	123
Figure 6-21: Representation of the individual 7_1.....	124
Figure 6-22: Representation of the Property assertion of the 7_1 with the reasoner active	124
Figure 6-23: Representation of the individual PCB_Feature_1.....	125
Figure 6-24: Representation of the Property assertion of the PCB_Feature_1 with the reasoner active	125
Figure 6-25: Representation of the individual FrontCover_Feature_1	125
Figure 6-26: Representation of the Property assertion of the FrontCover_Feature_1.....	126
Figure 6-27: Representation of the individual Left_Fuse_Feature_1	126
Figure 6-28: Representation of the Property assertion of the Left_Fuse_Feature_1 ..	126
Figure 6-29: Representation of the individual Right_Fuse_Feature_1	127
Figure 6-30: Representation of the Property assertion of the Right_Fuse_Feature_1 ..	127
Figure 6-31: Representation of the individual 7_2.....	128
Figure 6-32: Representation of the Property assertion of the individual 7_2 with the reasoner active	128
Figure A-1: Steps executed in the thesis work	144
Figure A-2: Representation of the sending and receiving of a request to a server	144
Figure A-3: difference between a non concurrent system (on the right) and concurrent one (on the left).....	146
Figure A-4: Taken from 6.1.1, Representation of the Flexible Manufacturing Line in the Industry 4.0 Lab.....	147
Figure B-1: Initial part of the code	151
Figure B-2: Reading of the Station in the CSV file	151
Figure B-3: Creation of an Individual and Id in the FrontCover Station	152
Figure B-4: Adding of the front cover component to the Individual.....	152
Figure B-5: FrontCover Station position of the individual.....	152

Figure B-6: Representation of the FrontCover Energy station.....	153
Figure B-7: Representation of the Drill Station	153
Figure B-8: Representation of the Shacker function	154
Figure B-9: Representation of the Drill Station Energy.....	154
Figure B-10: Representation of the Robot Cell	154
Figure B-11: Individual and id creation in the Robot Cell.....	155
Figure B-12: Comparison of the Individuals id and Individual "2_" + str(st2) destruction.....	155
Figure B-13: pcb and fuses insertion.....	155
Figure B-14: Representation of the Robot Energy station.....	155
Figure B-15: Representation of the Individual and id creation in the Camera Station	156
Figure B-16: Representation of Components insertion	156
Figure B-17: Operations in the case of a Circle image on the PCB.....	157
Figure B-18: Operations in the case of a Square and Triangle image on the PCB.....	157
Figure B-19: Individual and id creation in the BackCover Station.....	158
Figure B-20: id comparison in the BackCover Station	158
Figure B-21: Components insertion in the BackCover Station.....	158
Figure B-22: Operations to execute in case of a Triangle image	159
Figure B-23: Individual and id creation and id comparison in the Press Station	159
Figure B-24: Components insertion in the Press Station	160
Figure B-25: Representation of the Press Energy Station	160
Figure B-26: Individual and id creation and id comparison in the Manual Station..	160
Figure B-27: Components insertion in the Manual Station	161
Figure B-28: Representation of the Manual Energy Station	161
Figure B-29: Front Cover recycle at the FrontCover Station	162
Figure B-30: Fuses recycle at the Robot Cell.....	163
Figure B-31: Representation of the Disassembly Station 1/2.....	163
Figure B-32: Representation of the Disassembly Station 2/2.....	164
Figure B-33: Representation of the start of the code.....	164
Figure B-34: Front Cover retrieval from the Repair Station	165

Figure B-35: Sending of the front cover to the Repair Station	165
Figure B-36: Retrieval of the pcb from the Repair Station.....	166
Figure B-37: Sending of the pcb to the Repair Station	166

Acknowledgement

Having reached the end of this path, I think it is my duty to thank all those who have allowed me, in one way or another, to reach this important milestone.

My first thoughts go to Professor Marco Macchi, whom I thank for offering me the opportunity to do this particularly stimulating thesis work, thus allowing me to get involved with such an innovative and interesting topic.

I thank Adalberto Polenghi, who, thanks to his invaluable advice and his great availability, accompanied me throughout my thesis process, proving to be fundamental for the achievement of this goal.

I thank also go to all the members of the Industry 4.0 Laboratory of the Politecnico di Milano who welcomed me into the laboratory and, above all, for their willingness to give me a hand and an opinion even and above all in times of difficulty.

I also thank all my friends; even if I don't name you one by one, you know what you do for me and how important you are. A big thank you goes in particular to my longtime friends: Filippo, Simone, Claudio and Marco with whom I had fantastic experiences and who still continue to give me huge smiles and laughter. Without you and your support this result would not have had the same significance, so this achievement is also your merit. A big thank you also goes to all those people who have shared this path with me. Thanks to you, it was possible to discover a beautiful city like Milan, passing moments full of happiness, such as lessons done together, evenings spent on Milano Navigli, exam days and many other moments that I treasure deeply. For these reasons, a heartfelt thanks goes to Alessandra, Angela, Daniele, Jacopo, Luca, Mattia, Valeria and Vittoria.

I thank my family, which has always been one of my cornerstones. A special thanks goes to my grandparents, Guglielmo and Graziella, who have never stopped being my number 1 fans, but unfortunately could not attend this milestone. Finally, I thank my parents: Pierfrancesco and Angela. These years of university have reminded me how important they are to me and how they never stopped believing in me.