Proceedings of DET2011
7th International Conference on Digital Enterprise Technology
Athens, Greece
28-30 September 2011

KNOWLEDGE MANAGEMENT FRAMEWORK, SUPPORTING MANUFACTURING SYSTEM DESIGN

Konstantinos Efthymiou
University of Patras
efthymiu@lms.mech.upatras.gr

Kosmas Alexopoulos
CASP
alexokos@casp.gr

Platon Sipsas
CASP
psipsas@casp.gr

Dimitris Mourtzis
University of Patras
mourtzis@lms.mech.upatras.gr

George Chryssolouris
University of Patras
xrisol@mech.upatras.gr

ABSTRACT

Knowledge, in the contemporary economy, represents a fundamental issue. Information is being increasingly distributed across individual workers, work teams and organizations. Therefore, the ability to create, acquire, integrate and deploy knowledge has become a significant organizationalfactor. In particular, estimates suggest that a 55% to 75% of the engineering design activity comprises a reuse of previous design knowledge in order for a new design problem to be addressed. The aim of the current paper is the description of a knowledge management framework, able to support a factory's planning throughout its lifecycle from design to dismantling. The main core of the proposed framework is the ontology stored in the second component of the framework, i.e. the semantics based repository. On top of that, there is another component, the knowledge association engine, being responsible for performing similarity measurements and the inference rule definition and execution. Finally, two use cases, focusing on the design phase, are presented in order to show the applicability of the proposed framework.

KEYWORDS

Knowledge Based Engineering, Ontology, Manufacturing, Key Performance Indicators

1. INTRODUCTION

Knowledge Management (KM) refers to a range of practices and techniques used by organizations to identify, represent and distribute information, knowledge, know-how, expertise and other forms of knowledge for leverage, utilization, reuse and transfer of knowledge across the enterprise [Chryssolouris et al. 2008].

PDM / PLM systems provide an integration of product data with the use of conventional database approaches. The ERP systems are less product-focused and emphasize more on the integration of the business processes. In the last years two modules namely the CATIA V5 Knowledgeaware and the Siemens Teamcenter v9. have emerged and are mainly in support of product knowledge management. Both of the modules use the concept of the so-called templates or archetypes that focus on reusing parameterized, modular part designs. Both the EPR and the PDM/PLM systems are still considered weak in knowledge management, especially for the process related knowledge management; while the integration of knowledge based tools into PDM, such as the Distributed Open Intelligent PDM system [Kim et al. 2001] are few [Gao et al. 2003]. Another similar approach that may act complementarily to the existing PLM/PDM

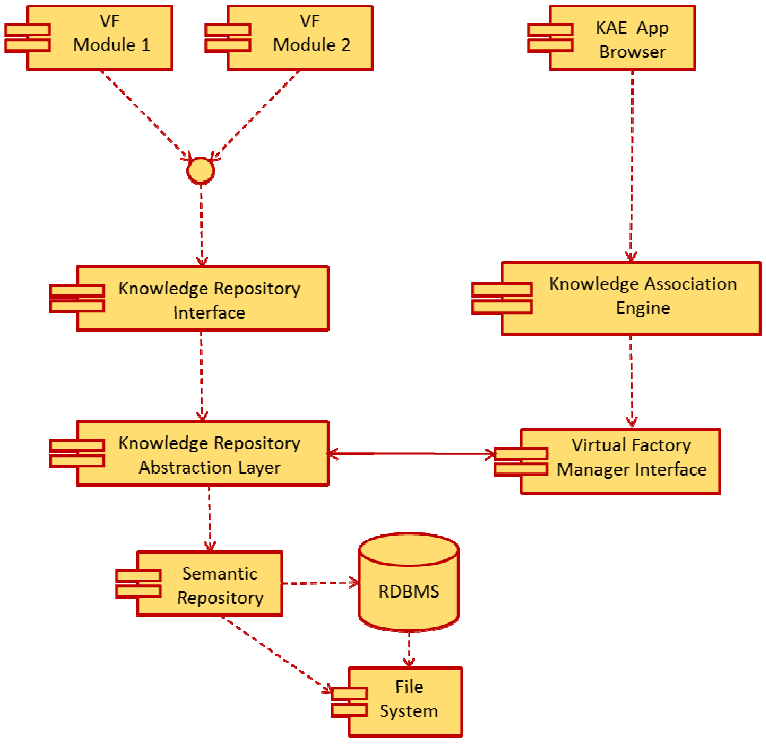

Figure 1 – Knowledge Framework Components Architecture

systems and may reduce time and cost of the early design phases has been proposed in [Papakostas et al. 2010]. In particular, this work introduces a new theoretical framework for analyzing and classifying knowledge, related to existing assembly configuration, in terms of manufacturing attributes such as time, cost and flexibility.

Product cost estimation during the early design phase of the product development is a process of great significance since it facilitates the cost control and influences positively the enterprise. A great obstacle of estimating the product cost, during its design phase, is the lack of information. Knowledge based systems mostly following case based reasoning approaches, attempt to overcome this problem and provide a first estimation of the product cost. The DATUM project (Design Analysis Tool for Unit-cost Modeling [Scanlan et al. 2006] aimed at the development of a knowledge based system, capable of estimating the cost of an aircraft's engine and its subcomponents and generating a process plan for the manufacturing of the engine. A similar application to the automotive industry has been developed in the context of the MyCar project [Makris et al. 2010]. In particular, a knowledge based system for subassembly cost estimation has been implemented, following a hybrid methodology utilizing case based reasoning and regression analysis [Mourtzis et al. 2011].

The proposed knowledge based framework is addressing knowledge management through the factory lifecycle, from design to dismantling phase. However, this paper emphasizes more on the design phase of a manufacturing system by providing two use cases from the ice cold merchandisers industry. The remainder of this paper is structured as follows. Section 2, provides the core elements of the knowledge based framework, namely, the ontology, the repository and the association engine. Two use cases, related to manufacturing systems design, are presented in Section 3. Section 4, concludes the paper and proposes future research paths.

2. KNOWLEDGE BASED FRAMEWORK

2.1 ARCHITECTURE

The aim of this work is to model the existing data with the use of the Semantic Web technology in order for useful knowledge to be extracted. For this reason, a Knowledge Repository (Dhavalkumar Thakker et al, 2010) is developed following its architecture and it is being explained.

A manufacturing ontology is created in the OWL language (W3C, 2010) and it is then placed in a Semantic Repository (SR). The SR is a web based module used to saving and querying semantic data (W3C, 2011). Moreover, it has a built in reasoning engine to support rule inference over the deposited

data. The SR can be accessed through the Semantic Repository Abstraction Layer (SRAL).

The SRAL is implemented utilizing the Enterprise Java Bean technology and allows the disconnection of the Semantic Repository from the other Knowledge Repository components. Exploiting the Semantic Web data model and the reasoning engine, a web application is built that enables the extraction of knowledge from the SR data.

This application is named Knowledge Association Engine (KAE), due to its ability to operate over semantic data and to extract useful knowledge from it. Apart from the KAE, there are several other Virtual Factory (VF) modules that exchange data with the SR through the Knowledge Repository Interface (KRI) service; howerer,these are beyond the scope of this paper.

Finally, the data in the SR is synchronized with that from another module, known as the Virtual Factory Manager. The synchronization is succeeded with the implementation of the Virtual Factory Manager Interface (VFMI).

The Virtual Factory Manager is an information exchanging platform to support new and existing software for manufacturing design and management. Concerning its data, it provides concurrent data access and consistency, evolving factory data management and that of safety in cases of malfunction. In this way, it enhances the data exchange among different software tools. Moreover, it has platform independent interfacing capabilities, acceptable response time and semantic web functionalities, enabling each software tool to communicate with this service and handle its data by using the semantic web technology. To further enhance its usability, this platform has a versioning system for its data and provides a set of methods for versioning management and structured data retrieval. In the case of the Knowledge Repository, it is the aforementioned set of methods used. The KR interacts with the VF Manager in order to retrieve structured data exploiting its versioning system. In order to synchronize its data with the VF Manager platform while preserving the versioning management and the data structures, the KR has its own software module, namely the VFMI. The VFMI has been designed to interact with both the KR application and the KR Semantic Repository and to perform data synchronization between the KR Semantic Repository and the VF Manager platform every time that isrequired.

2.2. MANUFACTURING ONTOLOGY

The current ontology aims to cover the manufacturing domain, to model the structure and the relationships among the primarily physical and virtual entities of a manufacturing system (Chryssolouris, 2006). The purpose of the ontology is to represent plant, product, orders, performance indicators and define their interconnections, in order to support the modelling and the analysis of alternative plant configurations useful for the different phase of the factory's lifecycle, such as design and planning. The ontology will answer questions, concerning the assessment of manufacturing performance indicators for alternative plant configurations or alternative task planning activities, providing this way, a decision support mechanism. The overall ontology scheme is illustrated with the help of figure 1. On the left side, the product and class orders are presented while in the center of the figure, the plant class is illustrated and beyond it, the manufacturing attributes structure is depicted. The ontology's basic scheme is further analysed in the following paragraphs.

2.2.1. Plant Hierarchy

The factory's structure is represented with the use of the plant's hierarchy. In the current ontology, a more coherent approach to plant hierarchy, following a four level hierarchy (Chryssolouris, 2006), is adopted and includes:

- Factory level
- Job shop level
- Work center level
- Resource level

The factory is the highest level in the hierarchy and corresponds to the system as a whole. The factory further consists of job shops that represent a group of workcenters. The workcenters are considered as a set of resources that perform similar manufacturing processes. A resource is regarded as a generic entity that can be a machine, a human worker or a storage area. In the present ontology, each level is modeled as a class

2.2.2. Product Hierarchy

A generic and abstract structure of the product is proposed for the needs of the ontology whereby the Product is the main class. This concept represents the actual finished goods produced by the plant.

2.2.3. Orders Hierarchy

Corresponding to the plant hierarchy there is also the workload hierarchical breakdown. Orders are broken down into jobs, which in turn, consist of a number of tasks. An order corresponds to the overall production facility and is divided into jobs that based on their specifications can be processed only by a suitable job shop. A job consists of tasks that can be released only to one workcenter. Tasks can

be dispatched to more than one of the work center's parallel resources (Chryssolouris and Lee, 1994). Based on this concept, the orders hierarchy is modelled with the following classes.

2.2.4. Performance Indicators Structure

The four most important attributes, used for making decision in manufacturing during design, planning, operation and in general during the entire factory lifecycle, are cost, time, flexibility and quality (Chryssolouris, 2006). Consequently, the main classes of the performance indicators are classified into time, cost, quality and flexibility. Then the classes are further analyzed with more specific subclasses and instances that are associated with the plant hierarchy and order hierarchy classes with the use of rules and ontology relationships. In the current study, flexibility has three subclasses, namely capacity, operational and product flexibility, cost subclasses are modeled based on the Active Based Costing method while Time includes, production rate and flowtime. The aim of the performance indicators hierarchy is to provide a general scheme for classifying manufacturing attributes and associating them with plant and order classes, facilitating the assessment of performance indicators for different level of the plant hierarchy.

The cost modelling is based on the Activity Based Costing (ABC) method. So the main classes and their structure follow the ABC modelling.

The class of Time is specialized by the subclass of Production Rate. The scheme can be easily enhanced with other specializations of the time related performance indicators, such as lead time, process time etc.

High flexibility or low sensitivity to a change provides a manufacturing system with three principal advantages. It is convenient to think of these advantages as having ariseb from the various types of flexibility that can be summarized in three main categories as in (Chryssolouris, 2006), namely those of product, capacity and operation flexibility.

2.3. SEMANTIC REPOSITORY

The semantic repository, which is built could be paralleled with o a database. Its purpose is to store data and to allow its external retrieval through queries. However, the data is stored following the Semantic Web model, written in the OWL language. Furthermore, the SR provides a built in OWL reasoned. This is a module, enabling the application of rules over the stored data. The inference of the rules can be realized when querying the SR and retrieving data from it.

For the implementation of the SR, it is the Java Technology used, combined with the Jena Framework (Jena, 2011). A web application is built in the Java language using the Jena Framework to provide the semantic functionalities. The Jena Framework is a set of Java libraries implementing basic functionalities for semantic data storage and querying, following the W3C standards (Mc Bride, 2011). Moreover, it provides a built in rule engine, which executes rules over the stored semantic data. The language of the rules is the Jena Rule language, which is created to support this framework. Among others, this framework is selected for the implementation of the SR for the following two main reasons. First of all, it is an open source and very flexible framework, providing Java support and secondly, it is the only open source framework with built in support for rules that are freely distributed for research purposes.

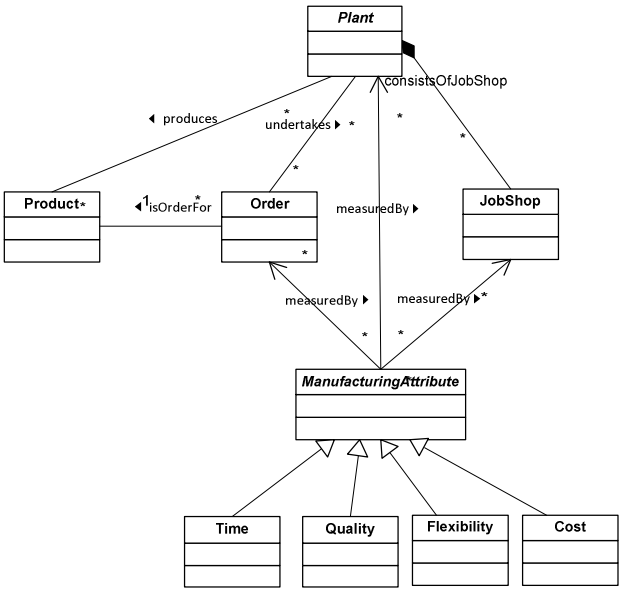

Figure 2 –Basic classes of the proposed manufacturing ontology and their relationships

2.4. KNOWLEDGE REPOSITORY ABSTRACTION LAYER

The role of this component is to decouple the Semantic Repository (off the shelf component) from the rest of the Knowledge Repository components. In this matter, it could be easier to change or switch the repository module, if needs be, in the future. This is rather important since this technology is not well established yet, and the probability of a change in the selected semantic repository module, is likely to happen.

This module has been implemented as an Enterprise Java Bean (EJB) application and is embedded in the apache tomcat (The Apache Software Foundation, 2011) using the openEJB plugin. It was developed as a stateful session bean, meaning that each client has one instance in memory. It has direct access to the SR and moreover, it can directly communicate

with the built-in rule engine in the SR. All other applications communicate with the SR through it.

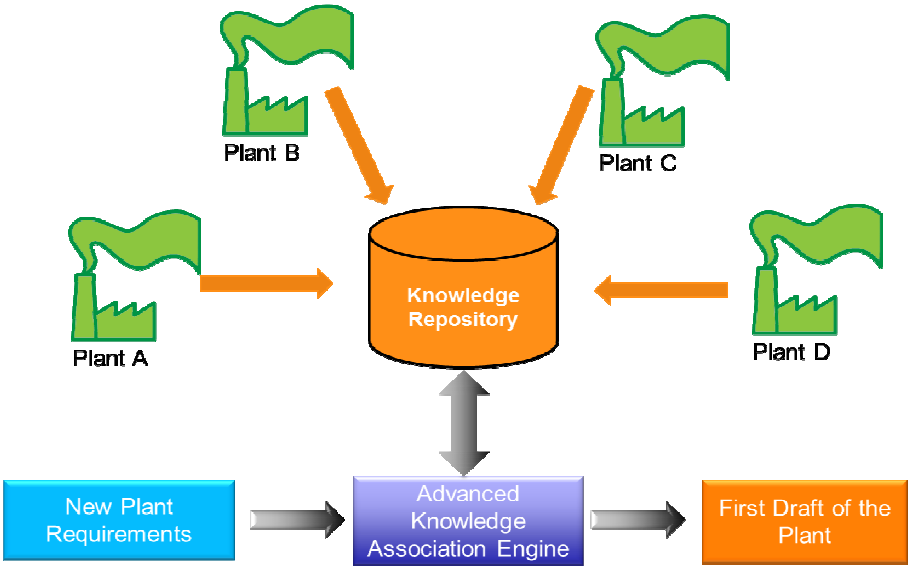

Figure 3 – Retrieval of past plant configuration solution

2.5. VIRTUAL FACTORY MANAGER INTERFACE (VFMI)

The VFMI is the module responsible for the data synchronization between the VF Manager and the SR. It has been developed as an EJB application, also embedded in the apache tomcat. The synchronization mechanism has been implemented as follows. When a Knowledge Application needs to access data from the VF Manager, it communicates with the VFMI, requiring the repository name by which a specific version of VF Manager data exists in the KR. The VFMI searches if the required version exists in the SR and replies with the name of this version's repository. If the version does not exist, the VFMI requires the data from the VF Manager and creates a new repository in o the SR. Then, it replies to the name of this new repository.

2.7. KNOWLEDGE ASSOCIATION ENGINE

The aim of the Knowledge Association Engine (KAE) is to extract useful knowledge using the existing data found in the SR. It is developed as a web application with a server side and a client side part. The client side part is actually the Graphical User Interface (GUI) of the application, by which the users are allowed to have access to the application services. The server side part is that where all the services of the application have been implemented (Carroll et al, 2003).

The Java technology along with some Java Script open source libraries are used for the implementation of this web application.. In this way, a user's only requirement is a web browser with an internal Java Script engine, and all well known browsers meet this requirement by their default. No other external software is required for installation. The entire application is hosted in an apache tomcat server and it has already been tested with the apache tomcat server v6.0 (The Apache Software Foundation, 2011).

The Enterprise Java Beans (EJB) technology is used for communicating with the SR. Through this communication, the KAE is able to have access to the data stored in the SR.

2.7.1. Similarity Measurement

Past or alternative plant configurations are assessed with the use of KPIs. In other words, each configuration is characterized by a set of KPIs. This characterization is not restricted only to a plant level, but it can be performed at a jobshop and workcenter level as well. So, in the context of the ontology, each class belonging to a plant hierarchy is associated with classes of the performance indicators structure. In particular, the connection *ismeasuredby,* determines the relationship between the classes' plant hierarchy and KPIs structure. During the design of a new plant, a jobshop or a workcenter, engineers search for past solutions that have fulfilled requirements similar to those of the new plant. Requirements are expressed in terms of KPIs, such as cost, time, quality and flexibility. The similarity measurement functionality facilitates a designer to search efficiently, in terms of time, past configuration solutions. Firstly, the designer defines

the level of the plant hierarchy and the product or the part that is to be

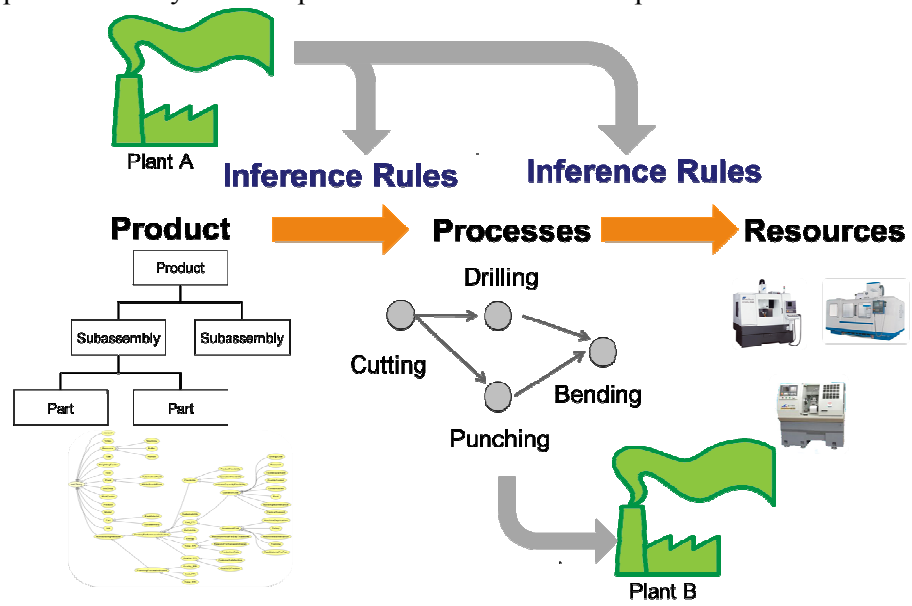

Figure 4 – Equipment selection based on inference rules

produced. Afterwards, the engineer determines the KPIs that represent the requirements of the new plant/jobshop/workcenter. Finally, the weighting factors for each KPI are also defined, determining this way, the significance of each of them, in the similarity measurement procedure. The paragraphs hereafter present the main algorithm behind the similarity measurement.

There is a great number of similarity measures that exist and apply to specific demands. The nearest neighbour technique, a widely used technology in the cased based reasoning retrieval process, computes the similarity between past cases (in our cases plant configurations) stored in the repository and the new case i.e. new plant configuration, based on weight features. In the current approach, the following form of similarity measurement is used.

$$Sim_{Global}(T,S_j)=\sum_{i=1}^{n} w_i \times f(x_i^t, x_i^s) \qquad (1)$$

Where:

- T: the new (Target) case (configuration)
- S: the past (Source) case (configuration)
- j: one of the past cases (configurations)
- n: the number of KPIs characterize each case (configuration).
- i: one of the n individual configuration KPIs.
- w_i: the weighting factor for the i^{th} KPI.
- f: the similarity function.
- x_i^t : the i_{th} KPI of the target (new) case (configuration).
- x_i^s : the i_{th} KPI of the source (past) case (configuration).

As a similarity function, a slight modification of the Minkowski measure is employed, providing normalized values ranging from 0 to 1, where 1 means a total similarity in contrast to 0 meaning total dissimilarity.

$$f(T,S_j)=\left(1-\sqrt{\left(T_i-S_i/T_i\right)^2}\right) \qquad (2)$$

Thus, each one of the past (source) cases (configurations) is compared with the new (target) case (configurations). The past case (configuration) with the greatest similarity value, i.e. nearest to 1 is selected to be the proposed solution used as a reference design configuration.

2.7.2 Rule Editing and Inference

The rule management of the SR is supported by the KAE application. Rules are statements, which are applied to the data existing in the SR so as to create a new useful knowledge. Considering that for the SR's implementation, the Jena Framework has been used, the rules have to follow this approach. So, in the KAE application, the Jena Rule Language is

supported for the writing of rules. For the rule execution and validation the built-in support of the Jena Framework is used for the implementation.

The Jena Framework includes a general purpose rule-based reasoner. This reasoner supports rule-based inference over RDF graphs and provides forward chaining, backward chaining and a hybrid execution model. To be more exact, there are two internal rule engines; one forward chaining RETE engine and one tabled datalog engine - they can be run separately or the forward engine can be used to prime the backward engine, which in turn, will be used for answering to queries.

Rules in the Jena Rule Language have two parts, that of the head and the body. These two parts have an arrow between them, with direction from head to body. Each part consists of a list of terms, the body terms also called premises while the head terms are also referred to as conclusions. Each term is either a triple pattern, an extended triple pattern or a call to a built-in primitive.

For the execution of rules there is a module called rule engine. The latter are used to deriving additional RDF assertions, which are entailed from some base RDF together with any optional ontology information and the axioms and rules associated with them. The primary use of this mechanism is to support the use of languages, such as the RDFS and OWL, which allow additional facts to be inferred from instance data and class descriptions.

The KAE application allows the user to store rules in the SR and to activate them. For the execution, a rule engine is used. The Jena Framework, used for the SR's implementation of the SR, is in support of a built in rule engine. In this way, in order to export new data for the execution of these rules, the KAE application sends SPARQL queries (Mc Carthy, 2005) to the SR with a parameter indicating that the rule engine should also be used.

The added value of the KRE application is the combination of the rules with the similarity measurement mechanism. The similarity measurement mechanism internally uses a similarity algorithm, which is applied to the data that have been retrieved by the SR. The kind of data to be retrieved from the SR by the similarity measurement function, is controlled by the rules. Next, in order to be better explained how this works, an example has been given. Supposing that a user desires to do a similarity measurement among all the plants stored in the SR that produce ice cold merchandisers, then he needs to use the rule shown in Table 1.

Table 1 - Rule example

```
[IceColdMerchandiser:
 ( ?kpi base:measures ?plant )
 ( ?plant base:produces ?product )
 ( ?product rdf:type base:Ice_Cold_Merchandiser )
->
( ?kpis base:smMeasures ?product )
 ( ?product <base#smElements> ?plant )
( ?product base:smType base:Plant ) ]
```

In the head of this rule, a list of terms exists in a triple format. This list selects the plants' Key Performance Indicators (KPIs) and those of plants which produce the type of an ice-cold merchandiser product. In the body of the rule, the selected KPIs are connected with this product by the 'smMeasures' predicate; this product is connected with the selected plants by the 'smElements' predicate and also, this product is connected with the 'Plant' semantic URI by the 'smType' predicate. Those three predicates are used by the similarity algorithm and have the following meanings. The 'smMeasures' means that for the similarity level selection, the object of this triple should be included. The subject of this clause should always be a set of KPIs. The 'smElements' predicate means that if this similarity level is selected, the data of the similarity algorithm is the object of this triple. Finally, the 'smType' predicate should always connect this similarity level with the semantic class type of the similarity data. In our case, the similarity data belongs in the Plant semantic class.

3. USE CASES

This section describes two use cases from the ice cold merchandisers industry, presenting the applicability of the approach to real industrial problems. The first use case focuses on the design of a new production line. Past production lines were stored in the Knowledge Base and those most similar to the requirements of the new line production lines were retrieved and proposed for the new line, with the help of the Knowledge association engine. The second use case emphasizes on defining, in a systematic way, the knowledge related to the equipment required for the manufacturing of a product that emphasizes on the problem's technological side without taking into account time and cost constraints. The knowledge stored is later on retrieved for the selection of equipment to be used for the production of a new product. In the second use case, the rules functionality provided by the knowledge association engine, is mainly used.

3.1. PAST SOLUTIONS RETRIEVAL

The specific case study emphasizes on the knowledge management during the design of a new plant. The ICM plant building projects present a great similarity. This means that new plant designs may be based in an existing plant. Two phases can be defined within the use case: a. plant configuration definition, and b. similar plant configuration retrieval. During the first phase, knowledge engineers update the knowledge repository with ontology instantiations. Each instantiation provides information, related to the plant configuration, the KPIs of this configuration and the products that are produced. Plant configuration includes the plant's hierarchy, consisting, of the jobshops, the workcenters and the resources. Each plant is characterized by KPIs, indicating the former's performance. In the current use case, the KPIs are, investment cost, throughput and YYY. The second phase concerns the retrieval of a similar plant configuration. First, the engineer defines the requirements, i.e. the KPIs and defines their target and the relevant weight factors. Then the plant configuration that is closer to the target KPI values is proposed to the engineer as a draft solution. The entire phase is illustrated within figure 1.

3.2. RESOURCE IDENTIFICATION

The Second use case emphasizes on the definition and storage of knowledge utilizing inference rules. In particular, two types of inference rules are utilized.

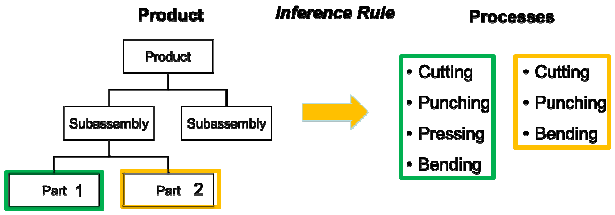

Figure 5 – Inference Rules associating product with required processes

The first type of rules associates a product with processes, while the second one connects processes with resources. In this way, the knowledge is systematically stored in the knowledge repository and is easily accessible by all the company departments all over the world. Consequently, engineers can query KR for the resources that are required for a specific process or the process required for a product and the determined resources and processes to be provided.

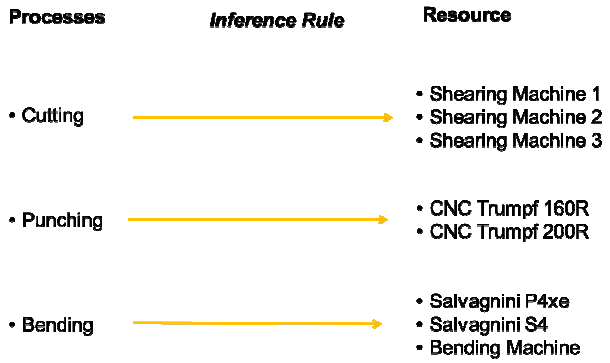

Figure 6 – Inference rules mapping processes with the required resources

4. CONCLUSIONS

This paper proposed a knowledge based framework that aimed to support the planning of a factory throughout its lifecycle, with special emphasis having been given to the design phase. The knowledge framework was mainly based on three pillars, namely that of ontology, the knowledge repository and the knowledge association engine. Ontology is responsible for modelling the knowledge domain, in our case the factory planning knowledge. The knowledge repository stores knowledge by utilizing the semantics technology and knowledge association engine, discovers new knowledge with the similarity functionality and defines new knowledge with the inference rules functionality. Finally, two use cases are presented illustrating the potentiality and the applicability of the approach. In particular, the use cases show how the past knowledge, in terms of past similar problems and inference rules, can be utilized for the design of a new assembly line. The specific use cases are in a preliminary phase and will be further enhanced in future showing the capabilities of a knowledge manager to its full extent.

5. ACKNOWLEDGMENTS

This work has been partially supported by the research project Virtual Factory Framework (VFF) NMP2 2010-228595 funded by the European Union Seventh Framework Programme (FP7/2007-2013).

REFERENCES

Chryssolouris, G., "Manufacturing Systems: Theory and Practice", 2nd Edition, Springer-Verlag, New York, 2006.

Dhavalkumar Thakker, Taha Osman , Shakti Gohil and Phil Lakin, "A Pragmatic Approach to Semantic Repositories Benchmarking", L. Aroyo et al. (Eds.): ESWC 2010, Part I, LNCS 6088, pp. 379–393, 2010.

Carroll, J.J., Dickinson, I., Dollin, C., Seaborne, A., Wilkinson, K. and Reynolds, D., "Jena: Implementing the Semantic Web Recommendations", *Proceedings of the 13th international World Wide Web conference*, 2003, pp 74-83

Jena, "Jena – A Semantic Web Framework for Java", SourceForge.com, 2011, Retrieved: 15.06.2011, <http://www.openjena.org/>

Mc Bride, B., "An Introduction to RDF and the Jena RDF API", 2010, Retrieved: 15.06.2011, <http://jena.sourceforge.net/tutorial/RDF_API/>

Mc Carthy, P., "Search RDF data with SPARQL", IBM, developerWorks, 2005, Retrieved: 15.06.2011, <http://www.ibm.com/developerworks/xml/library/j-sparql/>

The Apache Software Foundation, "Apache HTTP Server Project", The Apache Software Foundation, 2011, Retrieved: 15.06.2011, <http://httpd.apache.org/>

The Apache Software Foundation, "Apache Tomcat 6.0", The Apache Software Foundation, 2011, Retrieved: 15.06.2011, <http://tomcat.apache.org/tomcat-6.0-doc/index.html>

W3C, "OWL Web Ontology Language - Use Cases and Requirements", W3C, 2004a, Retrieved: 15.06.2011, <http://www.w3.org/TR/webont-req/#onto-def>

W3C, "OWL Web Ontology Language - Reference", W3C, 2004b, Retrieved: 15.06.2011, <http://www.w3.org/TR/owl-ref/>

W3C, "SPARQL Query Language for RDF", W3C, 2008a, Retrieved: 15.06.2011, <http://www.w3.org/TR/rdf-sparql-query/>

W3C, "SPARQL Update", W3C, 2008b, Retrieved: 15.06.2011, <http://www.w3.org/Submission/SPARQL-Update/ >

W3C, "Web Services Activity", W3C, 2011, Retrieved: 15.06.2011, <http://www.w3.org/2002/ws/>

W3C, "XML Schema Part 1: Structures Second Edition", W3C, 2004c, Retrieved: 15.06.2011, http://www.w3.org/TR/xmlschema-1/

Kim Y, Kang S, Lee S, and Yoo S, "A distributed, open, intelligent product data management system". *International Journal of Computer Integrated Manufacturing*, Vol. 14, No.2, 2001, pp. 224–235

Gao JX, Aziz H, Maropoulos PG, and Cheung WM, "Application of product data management technologies for enterprise integration", *International Journal of Computer Integrated Manufacturing* Vol. 16 No 7 2003, pp. 491−500

Scanlan J, Rao A, Bru C, Hale P, Marsch R (2006) DATUM Project: Cost Estimating Environment for Support of Aerospace Design Decision Making. Journal of Aircraft Vol. 43 No.4 pp.1022−1028

Papakostas N, Efthymiou K, Chryssolouris G, Stanev S, Ovtcharova J, Schafer K, Conrad RP, and Eytan A, "Assembly Process Templates for the Automotive Industry",*(CATS 10), 3rd CIRP Conference on Assembly Technologies and Systems*, Trondheim, Norway, 2010 pp. 151-156

Mourtzis D, Efthymiou K, Papakostas N, "Product cost estimation during design phase", *44th CIRP International Conference on Manufacturing Systems*, Madison, USA 2011

Chryssolouris G, Mourtzis D, Papakostas D, Papachatzakis V, and Xeromeritis S, "Knowledge Management in Selected Manufacturing Case Studies", *Methods and Tools for Effective Knowledge Life-Cycle-Management, A.Bernard, S.Tichkiewitch (eds.),Part3*, pp. 521-532, Springer 2008

Makris S, Michalos G, Efthymiou K, Georgoulias K, Alexopoulos K, Papakostas N, Eytan A, Lai M, Chryssolouris G, "Flexible assembly technology for highly customisable vehicles", *(APMS 10), International Conference Competitive and Sustainable Manufacturing, Products and Services, Como, Italy (2010)*