Proceedings of the First International Workshop on Ontology and Semantic Web for Manufacturing (OSEMA 2011)

Collocated with the
8th Extended Semantic Web Conference
Hersonissos, Crete, Greece, May 29, 2011

Edited by
Alexander García
Carlos Toro
Luis Ramos
Lutz Schröder

.

July 1, 2011

Preface

This volume contains the papers presented at OSEMA2011 (http://www.osema.org.ve/): the First International Workshop on Ontology and Semantic Web for Manufacturing, held on May 29, 2011 in Hersonissos, Crete, Greece. There were 6 submissions. Each submission was reviewed by three program committee members. One was withdrawn and 5 were accepted. We would like to thank our peer reviewers for carefully reviewing the submissions and giving constructive feedback.

July 1, 2011

Bremen

Alexander García
Carlos Toro
Luis Ramos
Lutz Schröder

Program Committee

Aristeidis Matsokis	Laboratory for Computer Aided Design and Production, Switzerland
Aziz Bouras	University Claude Bernard Lyon II, France
David Baxter	University of Cranfield, England
Dong Yang	Shanghai jiao Tong University, China
Grubic Tonci	University of Cranfield, England
John Bateman	University of Bremen, Germany
Jürgen Angele	Ontoprise, Germany
Kristina Shea	Technische Universität München, Germany
Oliver Eck	Department of Computer Science, HTWG Konstanz, Germany
Parisa Ghoudous	University Claude Bernard Lyon I, France
Richard Gil Herrera	University Simón Bolivar. Venezuela
Sylvere Krima	National Institute of Standards and Technology (NIST), USA
Yuh-Jen Chen	National Kaohsiung First University of Science and Technology, Taiwan

Contents

Linking Data for Industrial Knowledge Management—A Case Study

Katariina Nyberg, Matias Frosterus, and Eero Hyvönen

Semantic Computing Research Group (SeCo)
Aalto University, Dept. of Media Technology, and
University of Helsinki, Dept. of Computer Science
http://www.seco.tkk.fi/
firstname.lastname@tkk.fi

Abstract. Manufacturing companies face the challenge of maintaining documentation and knowledge about their projects and products, scattered in heterogenous, distributed databases, represented in different formats and languages, and hosted in mutually incompatible systems. At the same time, the knowledge needs to be accessed on a global level from different perspectives and user groups, such as project planners, designers, and maintenance personnel. This paper presents a case study, based on real datasets of a major international diesel engine and power plant manufacturer, where these problems are addressed simultaneuosly by harmonizing the datasets from different sources using RDF, and by linking them together into a global repository using shared resources. Based on the global RDF store, services for both human and machine users, such as a faceted search engine and a SPARQL end-point, can be provided to support access from different perspectives to the company knowledge base.

1 Introduction

The Semantic Web[1] and Linked Data [3] provide an RDF-based[2] framework for global linking of data on the web, using heterogeneous datasets produced by independent actors in a distributed environment. On a company and an intranet scale, the semantic web provides a solution approach to problems of managing scattered, hard-to-find heterogenoues data, too. Using the methods and tools of the semantic web one can provide structure and meaning for the data, and facilitate the creation of intelligent user-interfaces and visualizations for it. The linked data principles allow also for integration between the company data and external data stores, such as the Linked Open Data cloud[3].

In the following, we first show a general publication pipeline, through which heterogeneous data originating from different datasets can be harmonized using the RDF data model, validated and corrected in a metadata editor if needed, and published instantly as a faceted semantic portal, with interfaces for both human users and the machine. After this, application of the pipeline to the contents of large international diesel engine and power plant manufacturer is described. In conclusion, contributions of the work are summarized, some related work discussed, and future research proposed.

[1] http://www.w3.org/standards/semanticweb/
[2] http://www.w3.org/RDF/
[3] http://linkeddata.org

2 Publication Process

An overview of the process of utilizing the Semantic Web and linked data approach in an industrial knowledge management context is shown in Figure 1. On the left, there are different kinds of datasets from different parts of the organization in different formats and for different needs. The first step is to transform the heterogeneous data into RDF form, and give URIs to the different resources, whose descriptions involve properties. Some resources, such as projects, shipments, and installations, often already have some form of ID numbers to be used as a basis for the local names in URIs. However, ID formats can differ and uniqueness of ID numbers between different datasets is not necessarily guaranteed. Furthermore, resources such as employees and buildings, may lack any unambigous ID numbers.

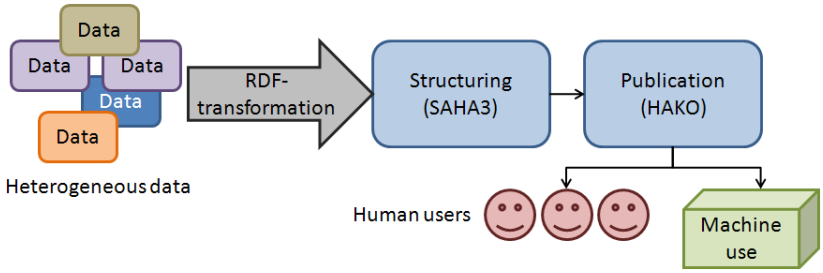

Fig. 1. Overview of the process of utilizing the linked data paradigm in an industrial knowledge management context

Once the data of each dataset is in RDF format, they can be merged together into a coherent whole, which in the case of RDF means technically simply taking the union of the datasets. However, when linking data in this way, two semantic data alignment problems must be solved before this is feasible. First, the schemas used in different datasets must be aligned (e.g. Dublin Core and in-house metadata schemas). Second, the vocabularies used in filling out the metadata schema values (e.g., persons, motor types, locations, etc.) must be aligned. In the linked data paradigm, standard properties such as owl:sameAs, rdfs:subclassOf, rdfs:subpropertyOf, and skos:narrowMatch are typically used for this.

In our process model, the aligned and merged data is imported into the SAHA 3 [8] metadata editor, which can be used to validate the data and make (manual) corrections to it, if needed.

For the publication of the data we used HAKO [8], a faceted search engine generator for publishing a SAHA3 project as a readily usable, faceted portal with machine usable APIs. The RDF data in SAHA3 is instantly available in HAKO, which is then configured to produce a portal matching the needs of the end-user in a few seconds. The publisher simply specifies 1) the classes whose instances are to be searched, and 2) what properties form the search facets for these instances. An example of the faceted

search portal can be seen in Figure 3. The facets are on the left and the search results, correspoding to the current selection of facet categories, on the right. There is also the possibility of using traditional, text-based search.

For machine use, SAHA3 and HAKO have two machine APIs: one for using the content as an ONKI ontology service [11] for annotation work, and one for using the content via a SPARQL end-point, used by other applications and HAKO itself.

3 Case Study: Making Data Available for Humans and Machines

Our case study concerns knowledge management of project ja product documentation in a major international manufacturer that produces diesel engines and power plant solutions for its customers. The idea is to apply the publicaton process of the previous section to large amounts of heterogenous company datasets. This case study aims at making it easier for the company's employees with different responsibilities and perspectives, such as project management or plant maintenance, to find and browse through the data in multiple ways and gain efficient access to the desired information.

There are three aspects to consider, when making data available in RDF form: 1) the data itself, 2) the metadata (including schemas), and 3) shared vocabularies for the domain of the (meta)data, i.e. ontologies [4]. These three aspects are considered in the following sections 3.1– 3.3.

3.1 Converting the Datasets into RDF

We used the company's Enterprise Resource Planning (ERP) system for obtaining the following datasets containing information about the power plant stations.

Plant Operation Manuals This data was in the form of numerous XML files containing parts of power plant manuals, such as plant operation manuals. They were grouped according to different power plant projects. The schemas for these XML files were splintered into numerous different DTD files, which made it difficult to find a consistent way to parse the files into other formats. Most of the XML files contained text fragments of the manuals for plant operations. The materials were in zip files that were named with a global unique identifier (GUID) corresponding to the project that the XML files described. We were also provided with the actual plant manuals for each plant project in PDF form. By analyzing the PDF files, we were able to identify the corresponding manual's page number for each XML file.

Power Plant Project Information This dataset was a large spreadsheet file that contained information about the personnel in plant projects, i.e. the people that had worked in a given project as project managers, controllers, or engineers. In addition, the data contained technical information about the power plant projects, such as diesel engines used and fuel types needed. The data also contained information about the country of registration of the plant and important project dates.

The dataset was converted into RDF form in such a way, that each row in the spreadsheet was turned into a resource representing a plant project, and each column corresponded to a property of that resource. If the information in a column represented an entity, such as a person or a place, then the column content was turned into a resource, and the resulting property was an object property of the resources corresponding to the rows. If the information in the column was a numerical value, such as the engine quantity or a date, then the column contents were represented as a literal property.

Block Diagrams of Plant Systems A PDF file containing 14 different block diagrams was received from the company. These diagrams are an abstract and general representation of the inner workings of a plant. Each diagram represents a subsystem of the plant, such as the lubrication oil system or the steam generation system. The blocks in a diagram are connected to each other, labeled, and most of them are marked with a three-lettered code. Figure 2 shows as an example an extract from a block diagram that describes the lube oil system. The information contained in the diagrams, such as the different connections between individual blocks, are intended for visual use by human readers, and therefore an RDF representation of them was created by hand using the SAHA 3 editor.

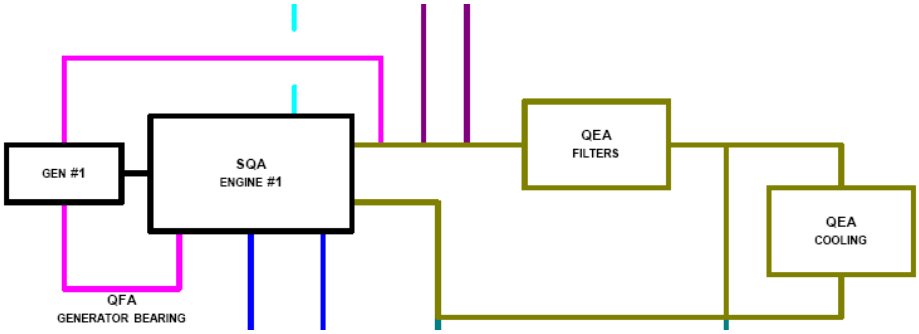

Fig. 2. A part of one of the block diagrams. The concepts in this block diagram detail the lube oil system in a power plant.

Other Data The links between the XML files and the projects were established by yet another spreadsheet file. Each row contained the project ID used in the power plant project information spreadsheet file (described above), and the project GUID that identified the respective XML files.

3.2 Creating Metadata Schemas

We created an RDF metadata schema for representing the XML files, where an RDF instance of a document class was created for each XML file. The properties of the

document class included the text contents of the XML file, its file name, and the power plant project it belonged to. In addition to this, a link to the manual and the respective page number were recorded as literal properties of the resource.

The schema for describing the contents of the power plant project information spreadsheet file followed mostly its structure. Some of the columns were turned into literal properties and others into object properties. In the case of an object property column, a resource of the type corresponding to the column was created, and the column content was represented as its label.

A schema for representing the 14 block diagrams was created, too. Here the individual blocks are represented as resources corresponding to the underlying ontological concepts. Information about a block, such as the three letter code, its label, and its connection to other blocks, were represented by the properties of the resource.

3.3 Shared Vocabularies

The ontologies that described people, places and mechanical information on power plants, such as fuel type, were created and populated from the spreadsheet file, when transforming it into RDF form.

A power plant system ontology was manually created based on the concepts presented in the block diagrams. The relations between the concepts express a consequential and other connections. For example, in Figure 2 the concept of the filters (QEA) is connected to the concept of the engine (SQA), indicating that a text passage that mentioning the filters might possibly also more generally apply to the concept of the engine without expressing it explicitly. There is part of relation from each concept to the block diagram (subsystem) it belongs to.

The manual text fragments corresponding to the XML files were analyzed, the concepts mentioned in the block diagrams were extracted from the text, and the links between the XML file resources and the concepts were established. Since the XML files belonged to a certain plant project, the projects and the concepts were linked, too.

3.4 Making the Data Searchable

In its original form, the underlying data is sorted and searched for according to a hierarchical classification of the individual power plant projects. As a result, it is not easily accessible to human users with complex access needs, such as, searching for a project documentation based on the personnel involvement. Browsing through a spreadsheet file with a lot of columns, makes it hard for a user to see the possible connections that exist between the projects. By transforming the data dumps into RDF form and making them compatible with each other, we managed to create a network of data, which allows for the data's information to be used to its full potential. Because of this, complex and perchance unexpected connections in the data can be found.

The idea was to create a system that would make it easy to answer questions, such as: "Which power plants are in the execution state of their life cycle and use dual fuel for power production?" or "Do I know anyone who has worked with John Smith on a power plant?" For this purpose, we published all of the resulting RDF in SAHA3. It was

easy to view the results in it and see if all the connections between different data were done correctly. The same RDF could be used with HAKO for configuring a faceted search for the data.

We configured HAKO in such a way, that the plant projects were the instances of the search. There was a total of 93 projects shown. As facets for the plant projects we chose all the relevant properties of the plant projects. Figure 3 shows the HAKO portal and how the amount of projects is narrowed down to ten according to a plant's fuel type and state. It gives an answer to the first of the above mentioned questions. To answer the second question, a user could clear the search by removing the selections, and narrow it by clicking on the name "John Smith" in one of the property lists that mention persons.

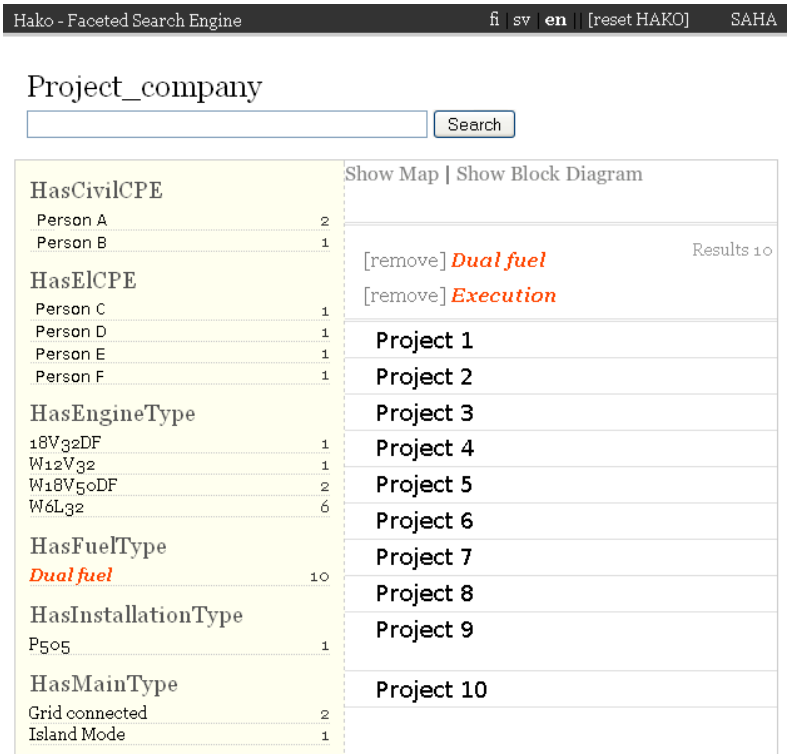

Fig. 3. HAKO faceted search portal (note that the project and person names have been edited out)

3.5 Block Diagram and Map Facets

Aside from the text-based facets already provided by HAKO, the nature of the block diagrams lends itself intuitively to a graphical, block based facet interface. This allows for easy access to the relevant manuals for maintenance and engineering personnel based

on the actual structure of a certain power plant system, such as the lube oil system in Figure 2. The block diagrams being general means that no customization is needed for different installations regardless of the specific decisions made in the construction.

The graphical block diagram facet was built into HAKO and placed above the results view into a tab system, which allows for easy integration of multiple graphical facets, as needed by a given system. We also implemented a map view, which is a worthwhile graphical facet for an international company that has projects in different parts of the world. It allows for an easy way of restricting the search results to arbitrary geographical areas and is useful in gaining an overview of a certain region's projects, especially when combined with the status information. "Show Map" and "Show Block Diagram" in Figure 3 open the map and block diagram view respectively when they are clicked on.

4 Discussion and Related Work

Contributions This paper presented a Semantic Web and Linked Data -based model and tools for publishing project and product documentation in an industrial company. From a human viewpoint, the idea is to aggregate and link related heterogeneous datasets, and make the whole data cloud more easily accessible from different perspectives using faceted search and browsing. At the same time, the content can be published for other services to use based on Linked Data principles, i.e. as a SPARQL endpoint, as a dereferenceable URI service, or as an RDF dump.

To evaluate the approach, a case study with real data from a manufacturing company was carried out with promising first results. The data dumps provided by the company contained information about persons and their roles in the projects. This is useful information for knowledge management, when special know-how needs to be found inside the company. Using HAKO it would be easy for the management of the company to see who has worked with whom and in what projects, and what experience they therefore now have. In a similar vein, also other additional perpectives to the project and product documentation datasets can be provided, using a single RDF-based knowledge repository. At the moment, end-user tests are being planned by the company in order to test the usefulness of the case study system in practice.

Related Work Kobilarov et al. [7] describe how the heterogeneous data from various sources in the BBC was made accessible using the tools of the Semantic Web, and linking to datasets in the LOD cloud, such as the DBPedia. They argued that interlinking data is beneficial to the users of the company's web page and the company itself at large. Antezena et al. [2] summarize different ways for supporting knowledge management in biology, discuss the emerging role of the Semantic Web, and introduce projects for knowledge management, such as BioGateway. It intergrates diverse datasets and offers a graphical interface and a SPARQL endpoint for its users [1].

Pollit [10] argues that the knowledge structures, on which the search target depends on, provide the facets with which the search can be narrowed down. This approach has been adapted for semantic faceted search, where the key design criteria have been to create a search interface for arbitrary RDF data [6, 5, 9].

Acknowledgements This work is part of the National Semantic Web Ontology project in Finland[4] FinnONTO (2003–2012), funded currently by the National Technology and Innovation Agency (Tekes) and a consortium of 35 public organizations and companies.

References

1. Erick Antezana, Ward Blondé, et al. Structuring the life sciences resourceome for semantic systems biology: lessons from the biogateway project. In A. Burger, A. Paschke, P. Romano, and A. Spendiani, editors, *Semantic Web Applications and Tools for Life Sciences, SWAT4LS 2008*, volume 435. CEUR Workshop Proceedings, http://CEUR-WS.org, 2008.
2. Erick Antezana, Martin Kuiper, and Vladimir Mironov. Biological knowledge management: the emerging role of the semantic web technologies. *Briefings in Bioinformatics*, 10(4):392–407, 2009.
3. Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data—the story so far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 2009.
4. Tom Gruber. Ontology. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*. Springer–Verlag, 2009.
5. Michiel Hildebrand, Jacco van Ossenbruggen, and Lynda Hardman. /facet: A browser for heterogeneous semantic web repositories. In Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora Aroyo, editors, *The Semantic Web - ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*, pages 272–285. Springer–Verlag, 2006.
6. E. Hyvönen, S. Saarela, and K. Viljanen. Application of ontology-based techniques to view-based semantic search and browsing. In *Proceedings of the First European Semantic Web Symposium*. Springer–Verlag, 2004.
7. Georgi Kobilarov, Tom Scott, et al. Media meets semantic web — how the BBC uses DBpedia and linked data to make connections. In *Proceedings of the 6th European Semantic Web Conference (ESWC 2009)*, pages 723–737. Springer–Verlag, 2009.
8. Jussi Kurki and Eero Hyvönen. Collaborative metadata editor integrated with ontology services and faceted portals. In *Workshop on Ontology Repositories and Editors for the Semantic Web (ORES 2010), the Extended Semantic Web Conference ESWC 2010, Heraklion, Greece*. CEUR Workshop Proceedings, http://CEUR-WS.org, 2010.
9. Eyal Oren, Renaud Delbru, and Stefan Decker. Extending faceted navigation for RDF data. In Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora Aroyo, editors, *The Semantic Web - ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*, pages 559–572. Springer–Verlag, 2006.
10. A Steven Pollit. The key role of classification and indexing in view-based searching. Technical report, University of Huddersfield, UK, 1998. http://www.ifla.org/IV/ifla63/63polst.pdf.
11. Jouni Tuominen, Matias Frosterus, Kim Viljanen, and Eero Hyvönen. ONKI SKOS server for publishing and utilizing SKOS vocabularies and ontologies as services. In *Proceedings of the 6th European Semantic Web Conference (ESWC 2009)*, 2009. Springer–Verlag.

[4] http://www.seco.tkk.fi/projects/finnonto/

Supporting Runtime Decision Making in the Production Automation Domain Using Design Time Engineering Knowledge

Thomas Moser[1], Wikan Danar Sunindyo[1], Munir Merdan[2] and Stefan Biffl[1]

[1] Christian Doppler Laboratory
"Software Engineering Integration for Flexible Automation Systems"
Vienna University of Technology, Vienna, Austria
[2] Automation Control Institute
Vienna University of Technology, Vienna, Austria
{thomas.moser, wikan.sunindyo, munir.merdan, stefan.biffl}@tuwien.ac.at

Abstract. Complex production automation systems are often represented as multi-agent systems which need to be reconfigured correctly and efficiently to adapt to new requirements. While the system knowledge is available at design time in form of workshop layouts, product trees or production strategies, at runtime this knowledge is often not available and therefore not used at all for operational decision making. In this paper we describe an engineering ontology used for the representation of design time engineering knowledge for supporting runtime decisions. We evaluate the proposed approach using three scenarios from the production automation domain. Major result was that the explicitly available design time knowledge can provide valuable input to runtime decisions.

Keywords: Runtime decision making; design time knowledge, engineering ontology.

1 Introduction

Complex software-intensive systems in production automation need to be flexible to adapt to changing business situations and to become more robust against relevant classes of failures. Production automation systems consist of components, for which a general design and behavior is defined during the design phase, but much of the specific design and behavior is defined during implementation, deployment, and run time with a range of configuration options. The "Simulator for Assembly Workshops" (SAW) [1] simulates complex reconfigurable production automation systems to maximize the overall system output by scheduling sequences of transport and machine tasks over 100 times faster than the actual hardware at VUT's ACIN lab[1]. Figure 1 (left hand side) illustrates an example assembly workshop layout that consists of software-controlled manufacturing components: transport components such as conveyor belts (dark green), crossings (light green), and stoppers (yellow/red circles); and assembly machines (colored rectangles with round corners); product parts are trans-

[1] Automation & Control Institute; http://www.acin.tuwien.ac.at

ported on pallets (colored rectangles; colors represent the target machines). SAW has been validated with real hardware components in an assembly workshop lab to ensure that the simulation outcome is relevant for real production automation systems.

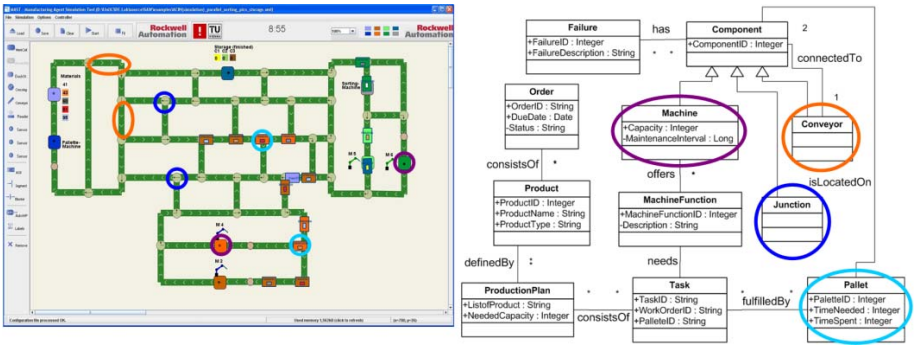

Figure 1. SAW Simulator and underlying data model.

Engineers, who want to adapt the system at runtime, need information from software models that reflect dependencies between components at design and run time, e.g., the workshop layout, customer orders and assembly procedures that translate into needs for machine function capacities over time; and the coordination of tasks for redundant machines in case of a failure. During development, design-time software models like data-oriented models (e.g., class or EER diagrams) or workflow-oriented models (e.g., sequence diagrams or state charts) are the basis to derive run-time models. But these models are often not provided in machine-understandable format to reflect on changes at runtime, i.e., the knowledge is modeled using an explicit human-understandable way but cannot be accessed by components automatically. Domain and software experts are needed to integrate the fragmented views (e.g., propagating model changes into other models, cross-model consistency checks) from these models, which often is an expensive and error-prone task due to undetected model inconsistencies or lost experience from personnel turnover.

Practitioners, especially designers and quality assurance (QA) personnel, want to make reconfigurable software-intensive systems (which like SAW consist of components defined by general design-time behavior, derived run-time configuration, and run-time specific behavior enactment) more robust against important classes of failures: machine failures, misuse from invalid supply, and failure-related changes in machine capacities at runtime. QA people could benefit from more effective and efficient tool support to check system correctness, by improving the visibility of the system defect symptoms (e.g., exceptions raised from assertions).

Challenges to detect and locate defects at run-time originate from the different focus points of models: e.g., components and their behavior are defined at design time, while configurations may change at runtime and violate tacit engineering assumptions defined in the design-time models. Without an integrated view on relevant parts of both design time and runtime models inconsistencies from changes and their impact are harder to evaluate and resolve between design and run time. Better integrated engineering knowledge can improve the quality of decisions for run-time changes to

the system, e.g., better handling severe failures with predictable recovery procedures, lower level of avoidable downtime, or better visibility of risks before damage occurs.

In this paper we present an approach to improve the support for runtime decision making with an engineering ontology. This engineering ontology provides a better integrated view on relevant engineering knowledge in typical design time and runtime models, which were originally not designed for machine-understandable integration. The engineering ontology can contain schemes on all levels and instances, data, and allows reasoning to evaluate rules that involve information from several models that would be fragmented without machine-understandable integration. The major advantage of using the engineering ontology for representing and querying the domain-specific engineering knowledge is the fact that ontologies are well suited to model logical relationships between different variables in axioms which can be used later for the derivation of assertions based on measured runtime data. We illustrate and evaluate the engineering ontology approach with three exemplary scenarios (change of conveyor directions, machine reconfiguration and machine maintenance preparation) from the production automation domain. Major result was that the explicitly available design time knowledge can provide valuable input to runtime decisions.

The remainder of this paper is structured as follows: Section 2 summarizes related work on Production Automation Systems, on the representation of design time knowledge and on the support of runtime decisions. Section 3 identifies the research issues and introduces the use case. Section 4 presents the approach; and finally section 5 discusses the findings, concludes the paper and presents further work.

2 Related Work

This section summarizes related work on Production Automation Systems, and on the representation of design time knowledge and the support of runtime decisions.

2.1 Production Automation Systems

By offering modularity and decentralizing system control, multi-agent-based approaches are recognized as a promising way to reduce complexity and increase flexibility of manufacturing systems [2, 3]. In this context, an agent is an intelligent entity placed in a manufacturing environment in order to supervise particular units and make decisions that influence the environment as well as its state. Agents communicate and negotiate with each other in order to perform the operations based on the available local information or in order to solve possible conflicts. However, manufacturing systems typically consist of heterogonous units, which use different types of data and data structures, and it is not easy to ensure the uninterrupted flow of information between and sometimes through the controlled levels. In order to ensure the correct understanding of the exchanged messages, agents must have the same presentation of the environment, or at least that part of the shared environment about which they are exchanging information with each other. Ontologies have been developed and investigated for quite a while in artificial intelligence and natural language processing to facilitate knowledge sharing and reuse [4]. They are of vital importance for enabling

knowledge interoperations between agents and, at the same time, a fluent flow of different data between different entities.

Various ontologies have been developed to capture particular fields in the manufacturing domain: the OZONE ontology [5] is devoted to constructing scheduling systems, the Enterprise Ontology aims to define the overall activities of an organization [6], the TOVE Ontology focuses on the enterprise modeling [7], the "Machine Shop Information Model" is intended for representing and exchanging machine shop data, initially between manufacturing execution, scheduling, and simulation systems [8], the Process Specification Language (PSL) covers generic process representation common to manufacturing applications [9]. On the other hand, ontologies like MASON [10] or ADACOR [11] could be classified as general-purpose manufacturing ontologies. An interesting standardization initiative has been started by the ONEIDA consortium establishing the framework for both the hardware and the software interoperability at all enterprise levels. Product data, which encapsulates intellectual property along with appropriate semantic information, is collected from the manufacturer and integrators in order to set a searchable repository and ease the work of related intelligent repository agents [12]. Complementary work has been reported by Lopez and Lastra: they merged several ontologies for mechatronic devices reference models (covering both the hardware and the software features) and the IEC 61499 reference model respectively into an ontology for an Automation Objects reference model [13].

However, by now an ontology is missing that will support the usage of design time engineering knowledge for supporting runtime decisions as well as is able to provide system knowledge to agents. The application of agent technology does not bring any advantages if the used agents are not intelligent. Considering ontologies as an intelligent way to manage knowledge, the integration of both technologies brings advantages such as extensibility and communication, and enables agents to agree on the meaning of common concepts they use with any other agent in an open environment [14].

2.2 Representing Design Time Knowledge and Supporting Runtime Decisions

An ontology is a representation vocabulary for a specific domain or subject matter, e.g., production automation. More precisely, it is not the vocabulary as such that qualifies as an ontology, but the (domain-specific) concepts that the terms in the vocabulary are intended to capture [15].

Manufacturing Execution Systems (MES), as a state-of-the-art in production workshops, link plan management and workshop control in an enterprise, which has the advantage to be integrated or interfaced with ERPs. Long [16] constructed a MES ontology which provides a formal specification of the concepts in the MES domain.

The infrastructure of MDA provides an architecture for creating models and meta-models, defining transformations between these models, and managing meta-data. Although the semantics of models are structurally defined by its meta-model, the mechanisms to describe the semantics of a domain are rather limited compared to machine-understandable representations using, e.g., knowledge representation languages like RDF[2] or OWL[3]. In addition, MDA-based languages do not have a know-

[2] Resource Description Framework: http://www.w3.org/RDF/

ledge-based foundation to enable reasoning (e.g., for supporting QA), which ontologies provide [17].

Beyond traditional data models like UML class diagrams or entity relationship diagrams, ontologies provide methods for integrating fragmented data models into a common model without losing the notation and style of the individual models [18]. The idea of using design time engineering knowledge to support runtime decision making have been already introduced by Moser et al. [19]. The authors proposed to collect design models information from production automation systems, such as the workshop layout, customer orders, product tree, or assembly procedures, and integrate them with the run-time information by using an ontology-based approach.

Andreolini et al. proposed to use models and frameworks for supporting runtime decisions in the context of web-based service systems [20]. The problems behind runtime decisions are how to detect significant and non-transient load changes of a system resource and how to predict its future load behavior. The authors described, tested and tuned the two-phase strategy to overcome the problems and integrating the strategy into a framework to support runtime decisions in a cluster web system and in a locally distributed Network Intrusion Detection System.

3 Research Issues and Use Case

In this section, we describe a real-world use case on system adaptation, e.g., to accommodate runtime failures. The use case is based on a Java simulation of an adaptive system that has been validated with a hardware version available at VUT's ACIN lab, the so-called Simulator for Assembly Workshops" (SAW). In the simulation context we collect evidence to which extent a richer and better integrated semantic knowledge base can translate into more accurate faster and cheaper decision making. The general SAW architecture consists of three major layers: the business process layer, the workshop system coordination layer and the machines in the workshop.

SAW simulates complex reconfigurable production automations systems to maximize the overall system output by scheduling sequences of transport and machine tasks over 100 times faster than the actual hardware. In the workshop, each machine has a set of specific functions, e.g., drilling or painting. Production parts are put on pallets and delivered to the machines via conveyor belts. In production automation systems, conveyor belts, junctions, and sensors can be represented by software agents that are working together and form a multi-agent system. By configuring an agent, the behavior of the real hardware can be specified as well. A junction connects two or more conveyor belts and follows the configuration of the software agents; select the correct outgoing conveyor belt for a pallet carrying a work piece. Sensors help the software agents to sense if pallets are in close proximity or help agents counting passing pallets to detect an overloaded conveyor belt and move to a backup strategy.

We defined three scenarios here according to the usage of SAW to solve possible problems that could occur also when using real hardware.

Runtime Decision 1 – Change of Conveyor Directions. The first scenario is related to the way of solving problems caused by conveyors failure. The failures of

[3] Web Ontology Language: http://www.w3.org/2007/OWL

conveyors and especially of conveyors that connect machines to each other, may lead to unreachable machines, hence may result in the failure of the system to produce products at all. One possible option to solve this problem is to change the direction of other unbroken conveyors in order to be used as a substitution of the broken conveyor. By changing other conveyors' direction, it is expected that a new route can be formed; hence the connection between different machines may become available again. The runtime decision that can be taken is to decide which conveyor directions should be changed in order to fulfill the new goal. The operator should also consider which conveyors are available and have a connection to the other machines directly or indirectly in order to identify possible new routes.

Runtime Decision 2 – Machine Reconfiguration. The second scenario in the production automation system that we want to go through is the possibility to reconfigure machines. Usually machines do not only offer one machine function but several different machine functions that can be reconfigured. However it takes time to reconfigure the machine functions inside one machine, e.g., to disassemble and reassemble the machine functions to another machine. If a machine fails, certain machine functions are not available anymore, but there is still exists the possibility to transfer the machine function to another machine which is still working well. In this case, the operator should calculate whether another machine should be reconfigured to offer the needed machine function or not, e.g., to check whether the repair time is smaller than the time needed for reconfiguration, and to check whether there are any products ordered that require the machine function at all.

Runtime Decision 3 – Machine Maintenance Preparation. The third scenario is regarding the maintenance time that is needed by machines in the production automation system. After a certain amount of time running, machines need to undergo maintenance. In order to prepare the maintenance mode, the operator should calculate the number of orders, and therefore the related time needed for all machine functions required for a certain order, which can be produced before the machine needs to go into maintenance mode.

Based on the use case and the three derived exemplary runtime decisions, we derive the following research issues:

RI-1. Efficiency and effectiveness of the proposed engineering ontology. Investigate to what extent the integrated design time and runtime data models do facilitate queries regarding data originating from more than one different data model. Compare the number of queries required for retrieving specific information using the engineering ontology approach with a traditional (e.g., database-based) approach using multiple data sources with potentially heterogeneous data schemata. Can the proposed engineering ontology cope with a potentially very large number of possible solutions for specific scenarios, and how does this affect the answer time of complex queries to the engineering ontology?

RI-2. Scalability of the proposed engineering ontology. Since the evaluated workshop layout is manageably small, investigate the scalability of the proposed engineering ontology approach. Does the ontology area concept [21] support the structuring and therefore the usability of a potentially large and fast-growing engineering ontology? How to support specific stakeholders in working with parts of the engineering ontology, e.g., by providing tool support for specific tasks?

For investigating the research issues we gathered requirements from the use cases in the production automation domain. Based on these use cases we designed the architecture of the engineering ontology and the evaluation regarding the support for the three different runtime decisions.

4 Supporting Runtime Decisions using the engineering ontology

This section describes the architecture and contained design and runtime knowledge of the used engineering ontology, as well as the application of the engineering ontology for supporting the three runtime decisions introduced in the previous section.

4.1. Engineering Ontology Architecture

In this section, we describe the engineering ontology, a set of relevant information elements about components in machine-understandable format using OWL DL ontology syntax. Components can query the engineering ontology at run time to retrieve information for decision making, e.g., enriching and filtering failure information or runtime coordination of machine workloads due to changes of the available machine capacities.

The engineering ontology provides a place for storing design-time information that seems valuable for supporting run-time decisions of components, especially in the case of handling failures or unplanned situations (but not transformed into run-time code or configuration to limit their complexity).

Components can query the engineering ontology at run time with query languages like SPARQL[4] (SPARQL Protocol and RDF Query Language), which provide the components with the full expressive power of ontologies, including the ability to derive new facts by reasoning. In addition, components can feed back interesting observations into the run-time information collection of the engineering ontology and therefore help to improve the design-time models (e.g., by improving estimated process properties with analysis of actual run-time data) and/or check the information based on a certain set of assertions. Furthermore, valuable deployment information can also be stored in the engineering ontology in order to support and enhance for further deployments.

Figure 2 shows the three different layers involved in SAW: a) the *business layer* for production planning to fulfil customer orders by assigning optimal work orders to the workshop; b) the *workshop layer* for coordinating the complex system of transport elements and machines to assemble smaller basic products into larger more comprehensive products according to the work orders; and c) the *operation layer* for monitoring the individual transport system elements and machines to ensure their contributions to the workshop tasks. Those three layers are divided into two parts based on the time those layers worked on, namely design time (development) and run time (usage).

[4] www.w3.org/TR/rdf-sparql-query/

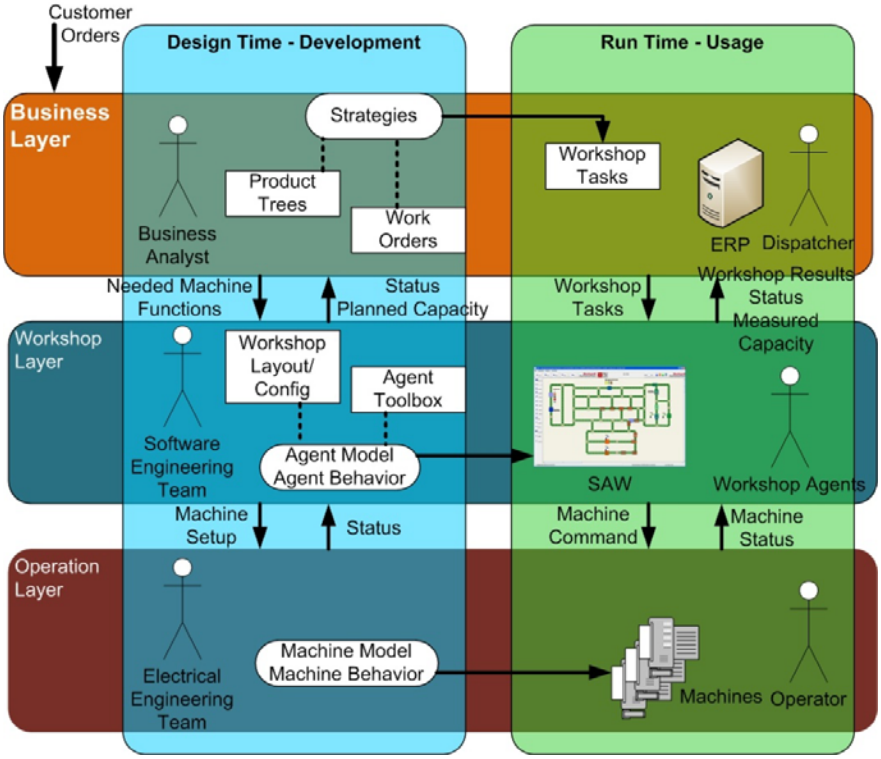

Figure 2. Engineering Ontology Overview.

In prior work [21], we proposed a data modelling approach that helps structuring large ontologies using ontology building blocks, so-called *"Ontology Areas"*. An ontology area is a part of an ontology, which is meaningful for a stakeholder and which helps ontology users to manage a complex ontology. The combination of all needed ontology area represents the overall ontology for supporting the original engineering process. An ontology area is a subset of an ontology as a building block that can solve a certain task. The ontology can be broken into ontology areas based on several aspects, for example by time, volatility, layer and roles. Figure 2 shows the breakdown of ontology into several ontology areas based on the stakeholder layers (business, workshop, operation) and the time the models are mostly used (design time and run time). Some parts of the data mode are much more volatile than others, e.g., run-time process measurements compared to design-time workshop layout.

4.2 Runtime Decision 1 – Change of Conveyor Directions

A Change-Direction-Algorithm (CDA) is used to handle a breakdown of conveyors which might lead to unreachable destinations (machines). The CDA is able to find the best stable configuration of the transport system by changing the directions of specific conveyors [22].

Each component of the transport system (e.g. conveyor belts, nodes, etc.) is represented and controlled by a corresponding Automation Agent [23] — an autonomous semantic entity responsible for the maintenance of the local data described in its world model. Besides the Automation Agents we also introduced the Contact Agent (CA) [24], which is created at the start-up of the system and is always active. Its main responsibilities are to supervise the functionality of the system and in the case that one part of the system collapses this agent considers its influence on the system performance and, if significant, undertakes particular steps in order to bring the system back into the optimal state.

The system uses the CDA and reacts on failures as follows:

1) The hardware of the system detects the failure using sensors. The low-level control informs the corresponding high-level control through the low-level communication interface [25].
2) Based on the given information the agent updates its knowledge base and informs all related agents about the detected failure. Each node has to re-calculate its routing table.
3) Furthermore, the agent also informs the CA, which is responsible for the overall system functionality.
4) The CA starts the CDA and compares its results with the actual system state.
5) In the case that the CDA recommends a new configuration, the CA updates its ontology, requests conveyor agents of concerned conveyors to change directions, and informs node agents to update their system representation.
6) Each node agent will recalculate and update its routing table. Having accurate information and an up-to-date world model of the system is of primary importance for nodes. Due to their role to receive pallets coming from input conveyors and —according to their destinations — to route them to the appropriate output conveyors.

Listing 1 shows the Prolog code (we use the Prolog notation for simplicity reasons) for detecting routes between two machines. There are 4 conveyors that connect machine A and machine B, machine B and junction C, junction C and junction D, and machine A and junction D. The connection between two nodes is defined bidirectional, i.e., whether there is a conveyor from a node to another node or vice versa. The routes from a node to another node are specified either as a connection directly from the second node to the first node (backtrack) or if there is a connection from the first node to the next node on the list of the nodes on the route to second node. This is done recursively. Hence the route from machine A to machine B can be discovered by using the query *route(machineA,machineB,Y).* This query returns two results, namely machine A → junction D → junction C → machine B and machine A → machine B, which means if the conveyor from machine A to machine B is failing, there is the possibility to use an alternative route.

This query requires design time workshop layout information regarding conveyors that are connected to machines or to other conveyors to check possible routes and change conveyor directions if required.

```
conveyor('machineA','machineB').
conveyor('machineB','junctionC').
conveyor('junctionC','junctionD').
conveyor('junctionD','machineA').

connect(NodeX,NodeY) :- conveyor(NodeX,NodeY),
  conveyor(NodeY,NodeX).

route(Node1,Node2,Way) :- go(Node2,Node1,[],Way).

go(Node,Node,Oldway,[Node|Oldway]).
go(Node1,Node2,Oldway,Way) :- connect(Node1,ANode),
  member(ANode,Oldway),
  go(ANode,Node2,[Node1|Oldway],Way).

?- route(machineA,machineB,Y).
Y = [machineA, junctionD, junctionC, machineB].
Y = [machineA, machineB].
```

Listing 1. CDA Query for detecting routes between two machines.

4.3 Runtime Decision 2 – Machine Reconfiguration

The second scenario deals with the possible reconfiguration of machines. Since there may be available modular assembly machines that can perform more than one different machine function, there always is the option to reconfigure a certain machine; either to provide a machine function in case of a failure of the original machine providing that function, or to provide an additional instance of this machine function to increase the throughput. However, this reconfiguration does not only may take essential time, but may also be not required at all, since it depends on the type and number of (future) orders to be produced. All of this information needs to be taken into consideration before machine reconfiguration should take place, and since these considerations may become quite complex, automation support is required for an efficient and effective machine reconfiguration process.

Listing 2 shows the Prolog code for calculating the reconfiguration time of a machine and comparison with the repair time needed to repair the failed machine providing the required machine function, so the operator can decide whether to reconfigure an alternative machine to provide the required machine function or wait for the repair of the failed machine. We have the design time information regarding the time needed to disassemble the machine function mf1 from machine A and also regarding the time required to assemble machine function mf1 to machine B. Hence, the reconfiguration time is calculated as sum of disassembly and assembly time of machine function mf 1. By applying the query *reconfigure(mf_1)*, we can check whether the time required to reconfigure an alternative machine is smaller than the time needed to repair the failed machine, which is true for the used exemplary values.

This query requires design time machine configuration information regarding the time required to disassemble and assemble machine function from certain machines, as well information regarding standard repair times of machines. Then we can com-

pare the time needed to reconfigure an alternative machine to provide a required machine function or to repair failed machine.

```
disassembly('machineA','mf_1',100).
assembly('machineB','mf_1',200).

reconfiguration_time(MachineFunction,Time) :-
  disassembly(_,MachineFunction,DisassemblyTime),
  assembly(_,MachineFunction,AssemblyTime),
  Time is DisassemblyTime + AssemblyTime.

repair_time('mf_1',500).

reconfigure(MachineFunction) :-
  reconfiguration_time(MachineFunction,TimeA),
  repair_time(MachineFunction,TimeB),
  TimeA =< TimeB.

?- reconfigure(mf_1).
true.
```

Listing 2. Query for calculating the reconfiguration time of a machine.

4.4 Runtime Decision 3 – Machine Maintenance Preparation

A third relevant run-time decision in the production automation scenario is the decision when to perform machine maintenance tasks in order to keep a certain minimum level of production output. This decision could also be taken during design time, resulting in a decreased ability to react to new or changing environment conditions (e.g., failures, reconfiguration). Since system flexibility supports operational efficiency in production automation, the decision when to perform machine maintenance tasks (e.g., cleaning, refurbishment) and the preparations for these tasks (i.e., emptying the machine buffers) could be taken by the machines themselves taking into account the state of other machines and workshop environment conditions. The idea is to coordinate the maintenance tasks of a set of related machines to minimize the impact on the overall production process. This planned maintenance should also be reported to a controlling system (e.g., an ERP system) in order to allow in-time reaction to the future capacity changes.

Listing 3 shows the Prolog code for calculating the total runtime of a machine. By using this code, the operator can calculate the total time to run all machine functions of a specific machine. The operator can check whether the total time is not exceeding the maximal runtime of the machine. By using the query *overuse('machineA')*, we can check whether the machine A overuses the maximal runtime allowed by the simulation workshop. In the used example, the total time is still below the maximal allowed runtime.

This query requires design time machine configuration information regarding the different times needed to run each machine function of particular machines.

```
run('machineA','mf_1',50).
run('machineA','mf_2',75).
run('machineA','mf_3',100).

list_of_run_time(List,X) :-
  findall(T,run(X,MF,T),List).

list_sum([], 0).

list_sum([Head | Tail], TotalSum) :-
  list_sum(Tail, Sum1),
  TotalSum is Head + Sum1.

total(X,Time) :-
  list_of_run_time(List,X),
  list_sum(List,Time).

maximal_runtime('machineA',300).

overuse(MachineName) :-
  total(MachineName,TimeA),
  maximal_runtime(MachineName,TimeB),
  TimeA > TimeB.

?- overuse('machineA').
false.
```

Listing 3. Query for calculation the total runtime of a machine.

5 Discussion and Conclusion

Engineers of complex software-intensive systems such as the “Simulator of Assembly Workshops” (SAW) system, who want to adapt the system at runtime, need information from software models that reflect dependencies between components at design and run time, e.g., the workshop layout, customer orders and assembly procedures that translate into needs for machine function capacities over time; and the coordination of tasks for redundant machines in case of a failure. Without an integrated view on relevant parts of both design-time and run-time models inconsistencies from changes and their impact are harder to evaluate and resolve between design and run time. Better integrated engineering knowledge can improve the quality of decisions for run-time changes to the system, e.g., better handling severe failures with predictable recovery procedures, lower level of avoidable downtime, and better visibility of risks before damage occurs.

In this paper, we presented an approach to improve support for runtime decision making using an engineering ontology. This engineering ontology provides a better integrated view on relevant engineering knowledge in typical design-time and runtime models, which were originally not designed for machine-understandable integration. We illustrated and showed the feasibility of the engineering ontology approach with three exemplary scenarios (change of conveyor directions, machine reconfiguration

and machine maintenance preparation) from the production automation domain, an extensive empirical evaluation will be performed in our future research work.

Based on this evaluation, we addressed the following research issues:

RI-1. Efficiency and effectiveness of the proposed engineering ontology. The engineering ontology provides a better integrated view on relevant engineering knowledge contained in typical design-time and run-time models in machine-understandable form to support runtime decisions. Another benefit is the possibility to define assertions in the engineering ontology which are checked based on the run-time information input of the running components. Further, the quality of information presented to an operator is improved since all information both from design-time as well as from run-time is available, leading to more intelligent run-time analysis and decision support.

RI-2. Scalability of the proposed engineering ontology. Typically, the engineering ontology can become very large and complex compared to the basic data model (such as used in a data base to automate run-time processes) if they include several aspects on a domain and some parts of the data model are volatile. In this paper, we proposed to use a data modeling approach based on ontology building blocks, so-called "Ontology Areas" [21], which allow solving tasks with smaller parts of the overall ontology. Ontology Areas also improve the efficiency of data collection task for decision making by lowering the cognitive complexity for designers and users of the ontology.

Future Work. While the engineering ontology can be seen as a comprehensive ontology, which stores and uses engineering knowledge both at design time and run time, more manageable, their application needs the effort of designers for structuring the overall ontology and for building task-specific smaller ontologies. Thus we will conduct empirical studies on the effort needed to design and use the engineering ontology. Future work could include human-subject experiments to assess complexity and efficiency more rigorously.

Acknowledgments. This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria; and also by the FIT–IT: Semantic Systems program, an initiative of the Austrian federal ministry of transport, innovation, and technology (bm:vit) under contract FFG 815132.

References

1. Merdan, M., Moser, T., Wahyudin, D., Biffl, S.: Performance evaluation of workflow scheduling strategies considering transportation times and conveyor failures. International Conference on Industrial Engineering and Engineering Management. IEEE (2008) 389-394
2. Jennings, N.R., Bussmann, S.: Agent-based control systems: Why are they suited to engineering complex systems? Control Systems Magazine, IEEE **23** (2003) 61-73
3. Sycara, K.P.: Multiagent systems. AI magazine **19** (1998) 79-92
4. Kulvatunyou, B., Cho, H., Son, Y.J.: A semantic web service framework to support intelligent distributed manufacturing. Int. J. Know.-Based Intell. Eng. Syst. **9** (2005) 107-127
5. Smith, S.F., Becker, M.A.: An ontology for constructing scheduling systems. Working Notes of 1997 AAAI Symposium on Ontological Engineering. AAAI Press (1997)

6. Uschold, M., King, M., Moralee, S., Zorgios, Y.: The Enterprise Ontology. Knowl. Eng. Rev. **13** (1998) 31-89
7. Fox, M.S., Barbuceanu, M., Gruninger, M.: An organisation ontology for enterprise modeling: Preliminary concepts for linking structure and behaviour. Computers in Industry **29** (1996) 123-134
8. McLean, C.R., Lee, Y.T., Shao, G., Riddick, F.: Shop data model and interface specification. NISTIR 7198. National Institute of Standards and Technology (2005)
9. Grüninger, M., Kopena, J.B.: Planning and the Process Specification Language. WS2 ICAPS 2005 (2005) 22-29
10. Lemaignan, S., Siadat, A., Dantan, J.-Y., Semenenko, A.: MASON: A Proposal For An Ontology Of Manufacturing Domain. IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications. IEEE Computer Society (2006) 195-200
11. Leitão, P., Restivo, F.: ADACOR: A holonic architecture for agile and adaptive manufacturing control. Computers in Industry **57** (2006) 121-130
12. Vyatkin, V.V., Christensen, J.H., Lastra, J.L.M.: OOONEIDA: an open, object-oriented knowledge economy for intelligent industrial automation. Industrial Informatics, IEEE Transactions on **1** (2005) 4-17
13. Lopez Orozco, O.J., Martinez Lastra, J.L.: Using semantic web technologies to describe automation objects. International Journal of Manufacturing Research **1** (2006) 482-503
14. González, E.J., Hamilton, A.F., Moreno, L., Marichal, R.L., Muñoz, V.: Software experience when using ontologies in a multi-agent system for automated planning and scheduling. Softw. Pract. Exper. **36** (2006) 667-688
15. Chandrasekaran, B., Josephson, J.R., Benjamins, V.R.: What are ontologies, and why do we need them? IEEE Intelligent Systems and Their Applications **14** (1999) 20-26
16. Long, W.: Construct MES Ontology with OWL. ISECS International Colloquium on Computing, Communication, Control, and Management (CCCM '08), Vol. 1 (2008) 614-617
17. Baclawski, K., Kokar, M.K., Kogut, P.A., Hart, L., Smith, J., Letkowski, J., Emery, P.: Extending the Unified Modeling Language for Ontology Development. International Journal of Software and Systems Modeling (SoSyM) **1** (2002) 142-156
18. Hepp, M., De Leenheer, P., De Moor, A., Sure, Y.: Ontology Management: Semantic Web, Semantic Web Services, and Business Applications. Springer-Verlag (2007)
19. Moser, T., Schatten, A., Sunindyo, W.D., Biffl, S.: A Run-Time Engineering Knowledge Base for Reconfigurable Systems. Technical Report (available at: http://bit.ly/hOr9Tf), Vienna, Austria (2009)
20. Andreolini, M., Casolari, S., Colajanni, M.: Models and framework for supporting runtime decisions in Web-based systems. ACM Trans. Web **2** (2008) 1-43
21. Biffl, S., Sunindyo, W.D., Moser, T.: Bridging Semantic Gaps Between Stakeholders in the Production Automation Domain with Ontology Areas. 21st International Conference on Software Engineering & Knowledge Engineering (SEKE 2009). KSI (2009) 233-239
22. Koppensteiner, G., Merdan, M., Hegny, I., Weidenhausen, G.: A change-direction-algorithm for distributed multi-agent transport systems. Mechatronics and Automation, 2008. ICMA 2008. IEEE International Conference on (2008) 1030-1034
23. Vallée, M., Kaindl, H., Merdan, M., Lepuschitz, W., Arnautovic, E., Vrba, P.: An automation agent architecture with a reflective world model in manufacturing systems. IEEE International Conference on Systems, Man and Cybernetics. IEEE Press (2009) 305-310
24. Merdan, M.: Knowledge-based Multi-Agent architecture applied in the assembly domain (Available at:http://www.ub.tuwien.ac.at/diss/AC05040230.pdf). PhD Thesis, VUT (2009)
25. Merdan, M., Lepuschitz, W., Hegny, I., Koppensteiner, G.: Application of a communication interface between agents and the low level control. Autonomous Robots and Agents, 2009. ICARA 2009. 4th International Conference on (2009) 628-633

Ontological Knowledge Based System for Product, Process and Resource Relationships in Automotive Industry

Muhammad Baqar Raza, Robert Harrison

Wolfson School of Mechanical and Manufacturing Engineering
Loughborough University, Loughborough, LE11 3TU, UK.

Abstract. This paper provides solutions to the real industrial need of adding knowledge layer to commercial PLM systems. PLM systems can be enhanced to be used as Knowledge Management tools to solve the semantic interoperability problem of heterogeneous data. Large amounts of product and machine component data exists in under-utilised databases due to the inability of existing integration approaches to systematise and relate the available information. The information about products, processes and resources is managed in PLM systems but it is not linked relationally among each other for complex decision making purposes. With the help of ontologies, knowledge based services serve a new layer of manufacturing management. The main objective of PLM as a KM tool is to improve the capabilities of technology intensive organisations to monitor and respond to technological and product changes and fully utilise the information stored in PLM systems through explicit mapping among products, processes and resources.

Key words: Product Lifecycle Management (PLM), Knowledge Management (KM), Ontology, Knowledge Based (KB) System, Semantic Web Services.

1. Introduction:

Dynamism and uncertainty are posing greater challenges for the continually changing manufacturing world. Industries have to enhance their strategy in order to respond efficiently to changing customer requirements and market needs [1]. The automotive industry is often described as “the engine of Europe” [2]. One of the key areas within the lifecycle of automotive manufacturing is powertrain and powertrain assembly systems. Such systems are supported by a number of engineering tools e.g. CAD, CAM and CAPP [3]. These tools are typically developed for individual system requirements in order to decrease lead time and increase customisation.

Current automation systems fail to meet business requirements. Assembly machines are designed to rapidly assemble different variants of products. Any change in product necessitates checking whether it is possible to assemble the new product on the existing machines. The answer to this question is not a straight forward one. Neither does exist an explicit mapping among products, processes and resources in the present day PLM systems, nor is the capability to define relational constraints in the form of rules and axioms. This is because the current PLM systems are product-focussed and processes are defined as a subset of products. Therefore a separate application, e.g. Process Designer (PD), has to be used in parallel with PLM systems to properly control the key area of process management in assembly systems. The focus of the research is to define assembly processes as relational constraints between product features and machine capabilities.

2. Specific Automotive Challenges

Technological innovation has brought about considerable changes in automotive industry. Powertrains (product), assembly processes (process) and powertrain assembly automation machines (resource) development in the automotive industry are very complex tasks. Life of the assembly machines surpass to the life of the products made out of them. Heavy investment can go unutilised or wasted when the new / changed product is introduced. Launching a new variant of the product in automotive industry is a huge challenge because of rippling effect of product change to several other domains. Product, Process and Resource (PPR) are the key elements of engineering domain in any automotive industry. Processes link products and resources, as shown in Figure1.

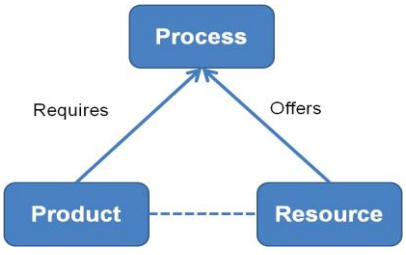

Fig. 1. Processes link products with resources

The assembly lines, such as powertrain assembly line for automotive engine, have a limited capacity to produce a variety of products. The built-in capability has to be limited to justify investment and is a trade-off between the unpredictable changes and the increased cost of flexibility. Designing and reconfiguring automotive assembly lines is an extensive process requiring expertise knowledge, business intelligence and involving several cross-functional domains. It becomes inevitable to use the best ICT tools and infrastructure such as PLM systems to manage and utilise information [4]. The information about Products, Processes and Resources is structured in the PLM systems, however, useful decision making process cannot be supported. This is because there are no pre-defined constraints of machine mechanisms available for product assembly through process parameters. The PPR domains in current PLM systems are managed independently presenting a challenge to relate them let alone defining dependency constraints, rules and axioms. The unavailability of the processes' and in turn, resources' constraints at the conceptual phase of the product design is a major discrepancy which results in target delays at later stages of program management. New strategies are required especially in the ICT systems in automotive sector due to rapid products' and consequent processes' changes to meet new business trends. There is a necessity of the processes' and resources' capability information to be available for a particular assembly system upfront so that the decision for manufacturing / assembling of the possible varieties of the products could be made rapidly and confidently.

3. Status and Scope of PLM

It has been recognised that current PLM implementations are document oriented, with no customisable data models and facing many inter-enterprise integration difficulties [6]. Therefore appropriate technology solutions for lifecycle support of PPRs are imperatively required to facilitate efficient implementation of PLM systems.

3.1. PLM Systems

A typical use case scenario in UK automotive industry is presented here. PLM system i.e. Teamcenter at Ford is used to manage manufacturing and assembly data in terms of validation of business processes w.r.t. time, cost, productivity, robustness, etc.

- Teamcenter (TC) manages manufacturing / assembly processes' design within Manufacturing Structure Editor (MSE) module of the tool as shown in Fig. 2.
- Planning starts with a top level structure that reflects the actual Bill of Process (BOP) from a generic BOP in the form of hierarchy of process steps. TC arranges BOP structures by managing data in three key areas as shown below:

Fig. 2. Product, Process and Resource in Teamcenter

Teamcenter manages PPR information separately by providing individual tabs in 'MSE module' whereby the information is not actually linked as shown above. The capability of Teamcenter in manufacturing/assembly data management does fulfil the industry needs in terms of information management, however, it limits the usability for change management. TC manages products with processes and plant (resources) as static records of data, nevertheless, it fails to associate and relate the three domains explicitly which results in manual efforts for any decision making activity. In general, current PLM approaches do not enable product and their under pinning resource systems and associated processes to be readily changed. Whenever there is any change in the product, it is a paramount concern to determine how this change affects associated processes and machines. Without explicit definition of relational knowledge, it is difficult to compare, contrast and critically scrutinise effects of product changes to processes and resources.

3.2. Current Inefficiencies

An important decision in the life of a complex product assembly is the selection of manufacturing / assembly processes for optimum use of resources and must be decided quickly and reliably to avoid extra costs. Ontologies and knowledge based tools can help the decision makers in the selection of an appropriate manufacturing / assembly process by matching the required attributes of products with available skills of machines through assembly processes. In this context, the aim of the current project is to explore the opportunity to build and use a relational KB system to capture and reuse knowledge and provide decision support in product change scenario.

Presently, the relationships among PPRs are not explicitly available in any commercially available PLM system. Every time there is a change in the engine design, the process engineers have to manually check all the stations to determine the potential changes to be made in the assembly line. The current reconfiguration approach is largely based on the skill and knowledge of domain engineers rather than the efficient use of already available information. Whenever there is any change in the product it is then essentially engineers' responsibility to examine, verify and validate the needs of the reconfigured system to support the new product [5].

The current PLM data is converted to rich semantic data by adding relationships among the three domains. The prevalent 'PLM Resources' are defined as ontological concepts and converted to knowledge elements by adding properties as well as relations with other concepts. For instance, adding properties of the concept 'PLM-Resource' in such a way that it is linked and interconnected to processes and products e.g. in the developed KB system, the concept 'PLM-Resource' has a property '*hasProduct*' as 'PLM-Product'; another property defined is '*performs*' as 'PD-Process', furthermore, 'PD-Process' property is defined *'hasSteps*' as 'Steps' and 'PD-Process' '*makes*' 'Sub-assemblies' from 'PLM-Product_Parts' and so on. An example of a concept and its instance in the KB system is shown below:

Concept PLM Resource	***Instance PLM Resource***
name ofType string	*name* hasValue "Station 500-CSA"
performs ofType PD Process	*performs* hasValue Crankshaft Assembly
hasProduct ofType PLM Product	*hasProduct* hasValue Crank Sub-Assembly

3.3. Improving PLM

The research focuses on improving PLM support infrastructure by exploring the use of ontologies. Within multi-faceted complex production environments, the use of ontologies has a great potential to aid knowledge management [5]. Ontologies also assist in laying down foundations for Service Oriented Architecture (SOA) which can be used to query PLM data through services. The scope of the research includes: (i) applications / database integration and (ii) PPR relations & mapping. To achieve seamless flow of data across applications and overcome the problem of semantic heterogeneity, ontologies can be used as a common language across several domains and information sources in large scale manufacturing industries as shown in Figure 3.

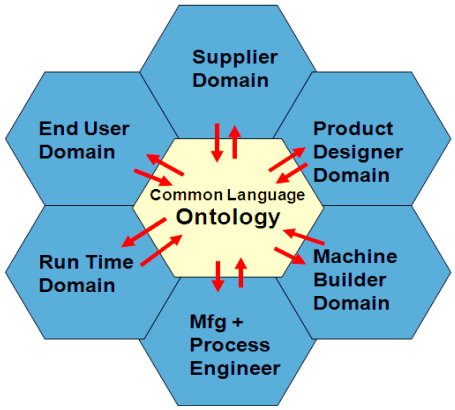

Fig. 3. Ontologies as a common language for the enterprise

Ontologies are not only useful for achieving semantic interoperability on the web but also to coordinate a range of disparate expertise for large organisations [8]. Ontological enrichment of existing data eliminates latency in the knowledge stream among concerned stake holders and supply chain partners within and across organisational boundaries. Ontologies provide the foundation on which a KB or an expert application system is built. The first phase of the work focussed on capturing and structuring data from Teamcenter in the form of ontologies. In addition to this, services are being developed to link this data into the enterprise systems to aid scheduling of the implementation of the line and order of appropriate parts from suppliers.

4. Proposed Research Concept

This research paper summarises ongoing research efforts on the development of new knowledge based powertrain assembly automation systems for automotive industry. The research focuses on knowledge integration by establishing relationships from multiple sources of information to solve a complex task. The main purpose of the ontology is to explicitly define relationships among products, processes and resources and made this information available, through queries, in the form of web services. The suggested framework is shown below in Figure 4.

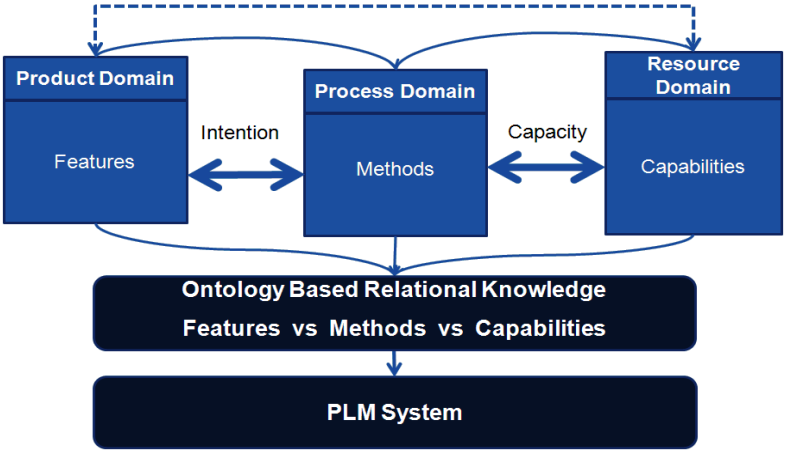

Fig. 4. Ontological knowledge based system in engineering domain

Ontologies have been considered one of the most efficient methodologies to develop semantic driven knowledge based systems [7]. The ontological knowledge based system combines: (i) object oriented approach and (ii) first order predicate logic. Real world objects (products, processes and machines) are populated by defining instances of concepts in the ontologies. In this way a formal and explicit definition of relationships is established along with correct information models. In case of product change, first directly affected resources are retrieved and then rules and axioms are applied to check whether the changed product can be assembled on the affected resources.

4.1. Characteristics of Ontology-Assisted PLM System

Modern businesses need to make complex decisions which require a lot of information analysis and processing. Such decisions must be made quickly and reliably. To automate (fairly) the task of assembly line design and/or reconfiguration, product and resource link points need to be defined at early stages of design and made available easily to be searched, analysed and implemented on 'when and where required' basis [5]. These link points are defined in ontologies and made available through semantic web services thus integrating PLM systems efficiently into common factory floor information platform. As a result, ontologies created a centralised relational knowledge base of Bill of Material (BOM) with its Bill of Process (BOP) and machine Bill of Resource (BOR) thus ontological based connections and mapping among BOM, BOP and BOR is formally established and efficiently exploited.

4.2. Reasoning Framework in WSML Ontology

By reasoning about the information using applied knowledge, ontologies and KB systems help domain engineers in decision making activities. In the project, Web Services Modeling Language (WSML) is used to build ontologies which is relatively a new language [8] based on logic-based knowledge representation formalisms, namely Description Logics [9] and Logic Programming [10]. It specifies XML and RDF serialisations to be compatible with existing web standards [8]. The WSML syntax is split into two parts: (i) the conceptual syntax, and (ii) logical expression syntax. The general logical expression syntax for WSML has a first-order logic style. Additionally, WSML provides extensions based on 'F-Logic' as well as 'Logic Programming' rules and database-style integrity constraints [11]. WSML has the usual first-order connectives, apart from first-order constructs, WSML supports logic programming rules of the form "$H :- B$", with the typical restrictions on the 'head' and 'body' expressions, H and B. Based upon this, data was gathered, rules and restrictions formulated and translated into ontology. For example, the rule of 'length' for powertrain is to use the standard length to minimize tooling cost. If greater length is required for additional power requirements, the maximum length cannot be exceeded without assembly feasibility study. In WSML ontology, it transforms to:

If Length > x *Then* actionA ***AND*** *If* Length ≤ x *Then* actionB

An example of axiom formulation in WSML, used in the KB system, is as follows:

Axiom Station50

Defined by ?x member of Product and

If Product length < station Y-axis capability *and* Product width < station X-axis capability *and* Product Height < station Z-axis capability *and* Processes required within System capability *and* Product Weight < = max allowable weight on station

Then Implies ?x member of Station50.

A series of these types of rules helped in quick evaluation of machine constraints with changing products and overcome the difficulties in pinpointing the engineering problem rather than working with human judgement and uncertain assumptions.

5. Case Study – Implementing Ontological Knowledge Based System at Ford Production Facility

This research study is part of a wider research project, Business Driven Automation (BDA) project [12] at MSI research institute, Loughborough University, in collaboration with Ford Motor Company, UK. Ford's DVM4 diesel engine assembly line at Dagenham plant, UK, was considered as the case study in this research. The authors have proposed and implemented a new framework where information stored in PLM system is converted into knowledge and presented as web accessible services throughout the company as well as supply chain partners. The case study was planned on four major steps:

- Study current PLM system and formulise needs
- Develop a standardised method to capture, structure and organise data
- Structuring and organising information in "subject-problem-solution" format i.e. a knowledge based (KB) system
- Make this data available to all stake holders in the form of services through a user friendly form

Ontologies have been used to capture and structure data so that information can be presented in a consistent manner and seamless communication be made possible. The scope of the ontologies developed for the current research is focussed on designing and reconfiguring assembly line against a new or changed product based on Component Based (CB) technology. In CB technology, any particular station of the engine assembly line can be decomposed to basic building blocks of modules of mechanisms (Components) which are independent to each other and can perform one operation independently. Different modules can be combined together to make a new station with changed process capabilities. These mechanisms are the building blocks of the extendable resources. Processes are the way the resources are and can be used. This relational dependency has been translated into ontologies as shown in Figure 6. Machines and their smaller functional units, associated with the assembly operations they perform, are converted to ontology thus obtaining complete knowledge of one of the zones on powertrain assembly line. In this way, products and processes are explicitly linked to the resources (machines) in ontological knowledge based system which is linked to the PLM system through semantic web services.

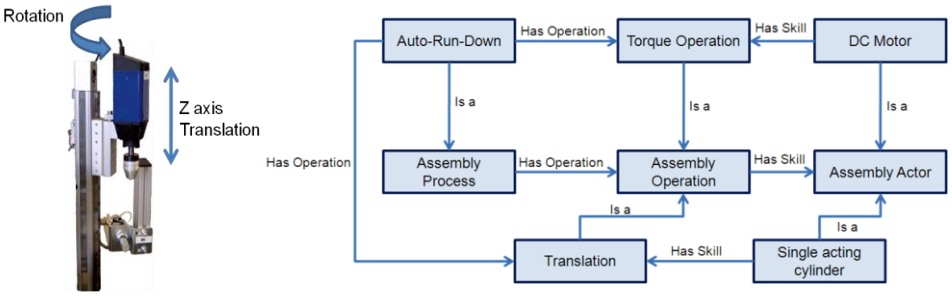

Fig. 6. Auto-Run-Down Equipment vs Auto-Run-Down Equipment Ontology

In the current research, Web Services Modeling Ontology (WSMO) has been used to define ontologies of products, processes and machines, once the concepts are defined then properties are associated with the concepts and relations among different concepts established. Based upon these concepts, relations among products, processes and resources are established. For example, a certain workstation performs particular assembly tasks on specific products to achieve a distinct objective. With the help of

this knowledge in ontologies, a quick evaluation of many potential configurations of resources is possible as well as the best suited one for a changed product. Therefore the ontologies of machines and equipments were developed, disparate data structured and PPR information linked to each other by adding semantics to the data for decision making purposes. In this way, information of one of the zones of the engine assembly line at Dagenham powertrain assembly plant was converted into ontology, PPR linked to each other through concepts and properties, as shown in Figure 7:

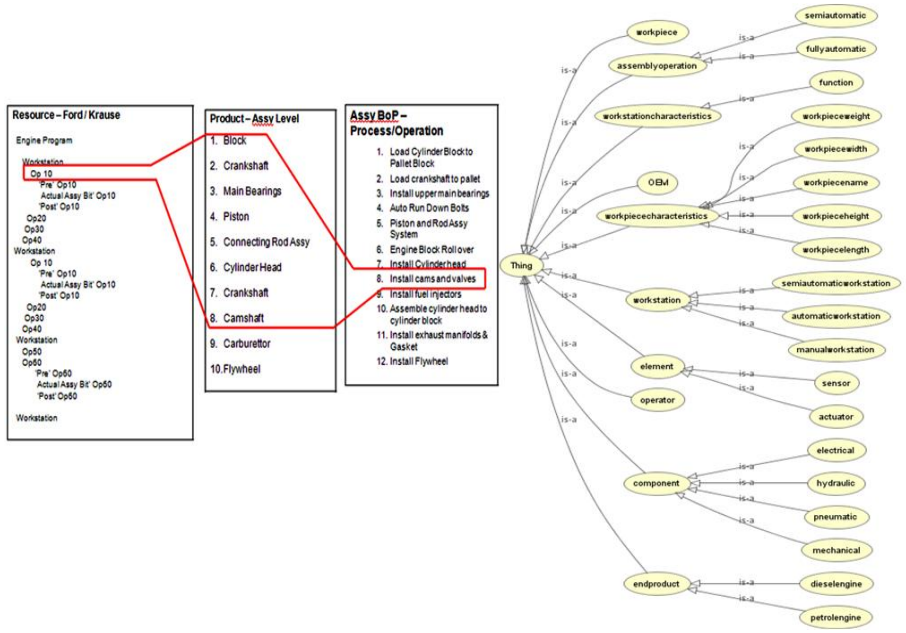

Fig. 7. Ontological mapping and linking PPR in KB system

Key concepts introduced into the ontology of the line are ‘PLM Resource’, ‘PLM Product’, ‘PLM Resource Characteristics’, ‘PLM Product Parts’ ‘PLM Process’ ‘PLM Process Steps’, OEM, Operator (manual resource) etc.

The ontological knowledge based PLM system is designed to be used for as complex an activity as an assembly line design and reconfiguration task. A typical use case scenario is rapid constraints evaluation for assemblibility of changed product against existing machines. Currently, product change management at Ford is dealt with the help of Teamcenter which requires skills of experienced engineers and

human-centred information retrieval and processing. With the help of knowledge based PLM, Ford engineers can first automatically retrieve the affected stations and process steps and then verify effects of product change automatically with the help of application of rules and axioms defined in the ontology. These rules are based upon relational constraints of products with resources e.g. spatial restrictions of different workstations imposed upon product are transformed to cognitive knowledge in the ontological KB system. End users of the developed KB system can access the ontology with the help of a user form to find, locate, add, remove, change and compare information as well as navigate to PLM information sources. The current knowledge based system can act as both an HTTP server, to provide information to end users, and an HTTP client, to collect information from Teamcenter.

6. Results and Contribution

One of the top priorities of Ford is to establish well integrated relationships among Products, Processes and Resources to provide lifecycle support to Powertrain automation systems. The software applications available in the market are mostly generalised so that these may fit in with most of the scenarios and business models in the world. Lack of advance, open and specific solutions always requires designing and building of new automation systems from scratch. Thus a fundamental drawback is addressed and solution provided to help automotive engineers to quickly evaluate effects of product changes to processes and automation resources.

There are no platform independent application tools available for modelling the PPR information explicitly neither does any tool exist to link PPR relational information unequivocally. The developed knowledge based system is believed to be first of its kind for assembly line design / reconfiguration activity which is open, extendable, interoperable and platform (hardware & software) independent. The ontological knowledge management provides a clear added value to PLM systems by using the existing information efficiently. Furthermore, adding a layer of knowledge services bears virtually no additional cost to the existing infrastructure as well as practically least training is required to learn and effectively use the developed KB

application. Some of the benefits of a PLM tool with ontologies are a greater rigour to process planning & design, greater capability to reuse 'resources' as well as faster cloning of existing machines and processes to cater for the changes in the product. Similarly, using services, knowledge from the PLM can be linked to the ontologies. With the help of ontology, this process is becoming smoothened and helping Ford engineers to perform parametrical relationship analysis between engine and workstation with relevant assembly processes through ontologies.

7. Conclusion / Future Work

This paper describes how existing PLM systems can be used as a KM tool to solve the semantic interoperability problem of heterogeneous data. The research proposed a rigorous model with well-defined meanings of PPRs entities in engineering domain. The development of a series of ontologies to both represent and capture this data will rapidly improve the production process in large scale manufacturing/assembly processes. In the next phase of the project, work will be carried out on updating the ontologies automatically as the line or especially the product changes. New concepts, properties and values of properties will be extracted from the legacy systems such as PLM systems and added to the ontology automatically so the ontology will be dynamically updated. The continuation of the work consists of including other downstream application tools in the ontology as well as enhancing the scope from line designing / reconfiguration to other knowledge intensive activities including line simulations and resource productivity analyses.

Acknowledgements

The authors gratefully acknowledge the support of 'UK EPSRC' and Loughborough University's 'IMCRC' through Business Driven Automation (BDA) project. We would like to thank all the project participants and engineers especially 'Dunton Technical Centre' & 'Dagenham Assembly Plant', of Ford Motor Company, UK.

References

[1] Harrison R, AW Colombo, Collaborative Automation from Rigid Coupling Towards Dynamic Reconfigurable Production Systems. 16th IFAC World Control Congress, 2005, Prague.

[2] Harrison R., A.A. West and L.J. Lee. Lifecycle engineering of future automation systems in automotive powertrain sector, 2006, IEEE International Conference on Industrial Informatics 2006, p 305-310.

[3] Sharma A, Collaborative product innovation, integrating elements of CPI via PLM framework, Computer Aided Design, 2005. 37 (13) p.1425-1434

[4] Qui, R. G. "A Service-oriented Integration Frame-work for Semiconductor Manufacturing Systems." International Journal Manufacturing Technology and Management 10(2/3): 177-191. 2007.

[5] Raza M.B., Harrison R., Kirkham T., Knowledge Based Flexible and Integrated PLM System at Ford – Journal of Information & Systems Management , Volume 1, Number 1, March 2011.

[6] Marek Obitko, Vladimir Marik, "Ontologies for Multi- Agent Systems in Manufacturing Domain", Proceedings of the 13th International Workshop on Database and Expert Systems Applications IEEE (DEXA'02), 2002.

[7] Hausser, R., The four basic ontologies of semantic interpretation. The 10th European-Japanese Conference on Information Modeling and Knowledge Bases, Finland, 2000, pp. 21-40.

[8] D. Roman, U. Keller, H. Lausen, R. L. Jos de Bruijn, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web Service Modeling Ontology. Applied Ontology, 1(1):77–106, 2005.

[9] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. The Description Logic Handbook. Cambridge University Press, 2003.

[10] M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. JACM, 42(4):741–843, 1995.

[11] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, D. Fensel. The Web Service Modeling Language WSML. WSML Deliverable D16.1v0.2, 2005. Available from: http://www.wsmo.org/TR/d16/d16.1/v0.2/

[12] BDA project at Loughborough University: http://www.lboro.ac.uk/departments/mm/research/manufacturing-systems/dsg/bda/index.htm

A Semantic Web Representation of a Product Range Specification based on Constraint Satisfaction Problem in the Automotive Industry

Fadi Badra[1], François-Paul Servant[2], and Alexandre Passant[1]

[1] Digital Enterprise Research Institute
National University of Ireland, Galway, Ireland
firstname.lastname@deri.org
[2] Renault SA
13 avenue Paul Langevin, 92359 Plessis Robinson, France
francois-paul.servant@renault.com

Abstract. Product Range Specification (PRS) in the automotive world is one of the most complex PRS that exists in industrial contexts. PRS plays therefore a key role in the information system of an automaker: related data pervades many systems, and numerous applications are using it. This is the case at Renault, where PRS is modelled as a Constraint Satisfaction Problem. In this paper, we study how to represent the objects, concepts and services related to such a PRS using Semantic Web standards. Plugging them into a Linked Data based architecture enables with new ways to access corresponding data and tools in the whole car manufacturing and selling process.

Keywords: Constraint Satisfaction Problem, Product Range Specification, Enterprise Linked Data, RDF(S), Configuration

1 Introduction and Motivation

In order to cut the cost of accessing data and exchanging it between systems — both internally and externally — Renault, one of the world's largest automakers[3], is considering the use of Linked Data principles and Semantic Web technologies in its information system. Several prototypes were done in the past years [12], and the first operational application based on Linked Data at Renault was released in early 2010, enabling the vision of "Linking Enterprise Data" [16] in the company. To go one step further, Renault aims at building a Semantic Web compliant representation of the objects, concepts and services related to its Product Range Specification (PRS). PRS is used to specify the set of all possible car configurations that an automaker can sell. Several reasons motivate the representation of PRS into Renault's current Linked Data infrastructure. First, PRS impacts many business tasks, and is a core component of Renault's information system. Product diversity being huge and complex, dedicated tools and

[3] http://renault.com

services are required to handle it. Moreover, PRS related data pervades many systems, and must be used in many applications which, for instance, must refer to subsets of the range (such as the "Twingo petrol with air conditioning"), and must state things about them (starting price, features, available options, etc.). Hence, enabling easier access to PRS data using the Linked Data principles, and making the PRS functionalities available as REpresentational State Transfer (REST) services, would be very rewarding. In addition, representing PRS in Semantic Web format would provide efficient ways to share data with industrial partners.

On the other hand, PRS data is not plain vanilla relational data. That could be a pain point within a Linked Data framework. In the evaluation of Linked Data for Renault, we need therefore to carefully check how this difficulty can be overcome. If the deployment of Semantic Web technologies is to be expanded at Renault, we need to represent and manipulate the objects of the PRS in the Web's Resource Description Framework (RDF). If this turns out to be impossible, RDF cannot be the solution for data modelling at Renault.

From a general perspective, Renault's PRS can be modelled as a Constraint Satisfaction Problem (CSP). Renault has developed several tools, based on a compiled representation of this CSP, for the many PRS-related questions that need to be answered in day to day operation of business. Thus, instead of representing PRS using Semantic Web technologies, we decided to represent Constraint Satisfaction Problems as such. This also bridges the gap between CSP and Semantic Web, and facilitates exchanges with academic partners, for instance for benchmark purposes. Yet, so far, we limited ourselves to the kind of CSP used at Renault (CSP with finite domain variables), and do not pretend to cover the whole question of CSP in RDF, which may come at a later stage.

From Renault's perspective, requirements on the representation of PRS objects include that the representation language is a published W3C standard and that open-source Java tools support this standard (both for producing and consuming it). Furthermore, its syntax must be serialisable in RDF, or at least allow a serialisation of the main PRS objects, so that the latter can be exchanged with Renault's partners. Succinctness and readability of the syntax is also a requirement in order to limit data transmission bandwidth and to enable Semantic-Web agnostic developers to understand the representation format. Hence, the proposed solution must find the right trade-off between strict standard compliance (no extensions to existing standards are required to be implemented by the Semantic Web client for reading the data), usability and conciseness of the description. Finally, the proposed solution should build upon the existing PRS infrastructure. Note that the purpose of this work is strictly about defining a representation of PRS data, not about providing tools for reasoning over it: all reasoning tasks on the PRS knowledge base are to be left to the existing CSP reasoner at Renault, which is already highly efficient and optimised.

Overall, this paper discusses how we represented PRS objects using Semantic Web technologies, and how we applied this in the industrial context of car manufacturing. The next section introduces Renault's Product Range Specification

and shows how it is modelled as a constraint network. Section 3 presents some use cases of applications that need to represent, store, manipulate, or exchange different PRS objects. Section 4 gives some related work and discusses our modelling choices regarding a Semantic Web representation of the objects of the PRS. Section 5 presents an RDF Schema representation of the main PRS objects. Section 6 compares this representation with its corresponding OWL representation and in section 6 with its corresponding SPARQL representation. Section 7 highlights a RESTful API developed to provide the functionalities developed over the PRS. Finally, section 8 concludes the paper and gives some future work.

2 Product Range Specification

2.1 Modelling vehicle diversity

PRS is used by Renault to specify all possible car configurations and comes as a lexicon, i.e., a set of discrete variables representing the descriptive features or attributes of a vehicle (body type, engine type, gearbox type, colour, etc.), together with a set of constraints that restrict the possible combinations of variable assignments. A particular vehicle is uniquely defined by a tuple of values, one and only one per variable. Constraints invalidate some of the possible car configurations to reflect industrial, engineering or marketing imperatives. The purpose of the PRS is to specify Renault's vehicle diversity, that is, the set of the distinct cars that Renault can build. The size of this set being very big (exceeding 10^{20}), it cannot be enumerated and it must therefore be defined in intention. Yet, product range is not only huge, it is also complex, because of numerous technical, commercial and legal constraints: should every combination of distinctive features and options be possible, the number of distinct vehicles would be in the order of 10^{25} rather than 10^{20}. To address these issues, fast and reliable reasoning services are required. To this aim, the PRS vocabulary and constraints are expressed as a constraint network and a CSP solver is used for reasoning over this constraint network [3].

2.2 Modelling PRS as a constraint network

A constraint network consists of a finite set of variables $\mathcal{X}$ such that each variable $x \in \mathcal{X}$ is associated with a finite domain $D(x)$ denoting the set of values allowed for x, and a finite set of constraints $\mathcal{C}$ that restricts the values the variables can simultaneously take. A constraint $c \in \mathcal{C}$ is a subset of the Cartesian product on domains: $c \subseteq D(x_1) \times D(x_2) \times \ldots \times D(x_n)$. A solution to a constraint network is the assignment of a value to each variable such that all the constraints are satisfied. A constraint network is said to be satisfiable if it admits at least a solution. To a constraint network is associated the constraint satisfaction problem, which task is to determine if such a constraint network is satisfiable. Renault's PRS is modelled as a constraint network, in which a solution (i.e., an assignment of all variables of $\mathcal{X}$) completely defines a particular vehicle. A vehicle range is

defined by a partial assignment of variables of $\mathcal{X}$. Therefore, some constraints will be represented as Boolean expressions on fluents. A *fluent* is a pair (x,A), where $x \in \mathcal{X}$ and $A \subseteq D(x)$. A fluent is *elementary* when A is a singleton (it thus represents an assignment of x).

3 Use cases

3.1 The Bill of Materials

The Bill of Materials, which is the process of defining the parts used in each vehicle of the range, is organised in "generic parts": a generic part is a function fulfilled by a part. For instance, the steering wheel, as a function, is a generic part which may be fulfilled by different steering wheels, as parts, depending on the vehicle. The relationships between the PRS, the generic parts and the corresponding real parts are defined by Boolean expressions called "use cases": the use case of a part is a Boolean expression (over the variables of the PRS) specifying on which vehicles the part is used. This definition of the references of parts corresponding to a given "generic part" is equivalent to defining a new variable whose part references constitute the list of possible values. These values have to be defined with respect to the variables of the PRS[4]. Basically: any vehicle has one and only one steering wheel, as part.

3.2 Accessing after-sales documentation

One user wants to find, for instance, how to remove the gearbox of the car under repair. Assuming methods are tagged with their subject, this looks like a simple SPARQL query — and the first part of the question, indeed, is a simple SPARQL query, such as "select documents where the subject is gearbox removal". Yet, filtering on the vehicle is more tricky: each document has a "condition" property, which points to the set of vehicles for which the document in question is relevant. This condition must be evaluated against the vehicle to decide whether the document must be returned by the query or not. In other words, the service accessing the technical documentation has two input parameters, one being the vehicle (possibly only partially defined with regards to PRS variables), the other one being a standard SPARQL query over documents described with metadata. We could even consider that any vehicle has its own SPARQL endpoint that provides access to its documentation.

3.3 Exchanging PRS related information with partners

It is not uncommon that automotive constructors need to exchange information related to their respective PRS. This occurs for instance when one of them assembles in its own plants cars conceived by the other one, or sell under its own brand cars conceived and / or assembled by the other one. In such cases, each

[4] See [3] for a description of the controls needed for the Bill of materials.

one has its own definition of the PRS. The core of it originates of course from the constructor who conceived the model. Yet, even before adding its own marketing constraints, the other one will first rephrase this PRS using the terms it is used to. This corresponds to the definition of new variables and attached values, based on the original ones, though the use of Boolean expressions . Several variables can be involved in the definition of a given value. Here's a real world example, regarding the seats and their features (such as "adjustable height with lumbar support") : one of the constructor uses the two variables "driver's seat" and "passenger's seat", while the other one uses "left seat" and "right seat". Cross definition of the values involves the variable "traffic flow direction".

This boils down to the creation of translation tables between PRS vocabularies. Tools to assist in the creation of such translations needs (for the GUI) information about the variables and values (typically what we put into a RDF description, such as labels, etc.). CSP based computation is necessary to control the validity of the translation table.

4 Representing constraints on the Semantic Web

Constraint satisfaction is an important reasoning paradigm in artificial intelligence. Constraints are essentially declarative, which makes them very well suited for knowledge sharing and reuse [10]. In [11], an XML format is proposed that can be used to represent CSP instances, as well as quantified or weighted CSP instances. While no standard formalism currently exist to represent constraints on the Semantic Web, several languages have been proposed to extend existing Semantic Web standards in order to express various types of constraints.

4.1 Representing constraints in OWL

Different constraint languages based on OWL/SWRL have been proposed. For example, CIF/SWRL [6] extends SWRL with quantifiers and nested implications in order to express complex range-restricted constraints. [9] further extends CIF/SWRL to add disjunction, negation in rule antecedents and the ability to use any OWL description in the scope of quantifiers. A complementary approach can be found in [5], where OWL is extended to support arithmetic constraints. But these languages are not currently Semantic Web standards and no implementation is to be found. Besides, the semantics of OWL does not seem well suited for constraint checking because it makes the open world assumption and it does not adopt the unique name assumption. Indeed, though a constraint can be represented in many different ways, including mathematical inequalities, logical formulas or matrices, it can be seen conceptually as the set of all legal compound labels for a set of variables [14]. In this regard, constraint satisfaction is very close to database theory, and a constraint satisfaction problem can even be expressed as a database-theoretic problem [15]. This has important implications regarding the choice of the formalism to use to represent CSPs. In

particular, constraint satisfaction problems typically apply a closed world assumption[5] and a unique name assumption[6] as in database modelling, which is usually not the case on the Semantic Web [8]. For this reason, providing OWL with a constraint-checking mechanism would require in some cases to grant subsequent axioms with alternative semantics [13]. For example, value constraints can be expressed using range restrictions but can not be checked using direct OWL inference. The following axiom could be used to state that a car must have its `version` variable set, and its value is either `generic` or `other`:

$$\texttt{Car} \sqsubseteq \exists\, \texttt{version}.\{\texttt{generic}, \texttt{other}\}$$

Yet, due to the open world assumption, there is no way to ensure that the constraint is not violated on a particular dataset using strict OWL semantics, since a missing value for the variable `version` in an OWL knowledge base would not cause logical inconsistency.

4.2 Representing constraints in SPARQL

In the Semantic Web community, propositions have been made to assimilate constraints to the SPARQL queries that can be triggered on a given RDF dataset to check for their validity [1, 4, 13]. For example, the above constraint could be expressed as the following SPARQL ASK query:

```
ASK { NOT EXISTS {
  ?x :version ?y . FILTER (?y != :generic && ?y != :other)
} }
```

In this approach, a constraint is modelled as the evaluation of a graph pattern on a dataset, and checking its validity amounts to running the query. One of the main benefits of this approach lie in the expressivity of the subsequent constraint language (the SPARQL query language has the expressive power of the relational algebra, as shown in [2]). Besides, SPARQL/SPIN [4] provides an RDF Schema for SPARQL queries so that they can be serialised in RDF. SPIN is not a Semantic Web standard but provides an implementation[7] on top of the Jena API, which is already used by Renault.

4.3 Representing constraints in RIF

As Renault's main requirement is to use a Semantic Web standard to represent Boolean expressions, we also considered the W3C's Rule Interchange Format (RIF)[8]. In particular, RIF's dialect RIF-BLD could fairly suit Renault's needs

[5] The closed world assumption states that if a fact is not explicitly stated, then it is assumed to be false.

[6] The unique name assumption states that two different identifiers can not refer to the same individual.

[7] http://www.topquadrant.com/topbraid/spin/api

[8] http://www.w3.org/TR/rif-core/

as an interchange format for constraints but it still suffers from a lack of implementations, and at the time of writing no consistent RDF serialisation is to be found. It can be hoped that future developments on RIF would include defining a dedicated dialect for constraints, as well as an RDF serialisation format.

4.4 Discussion

Regarding the user's requirements, RDF(S), SPARQL/SPIN and OWL appear to be the best candidates to represent PRS objects. Our hypothesis here is that an RDF(S) vocabulary might be sufficient to express the main PRS objects, all the more so that Renault has no need for any constraint-checking mechanism. For these reasons, we started with a direct translation of the main PRS objects into a lightweight RDF Schema, and then studied how this representation translates to OWL and SPARQL/SPIN.

5 An RDF(S) representation of the main PRS objects

5.1 Variables and their domains

The basic building blocks of the PRS is a set of variables and their associated domains of values. A variable can be seen as a function associating a value to

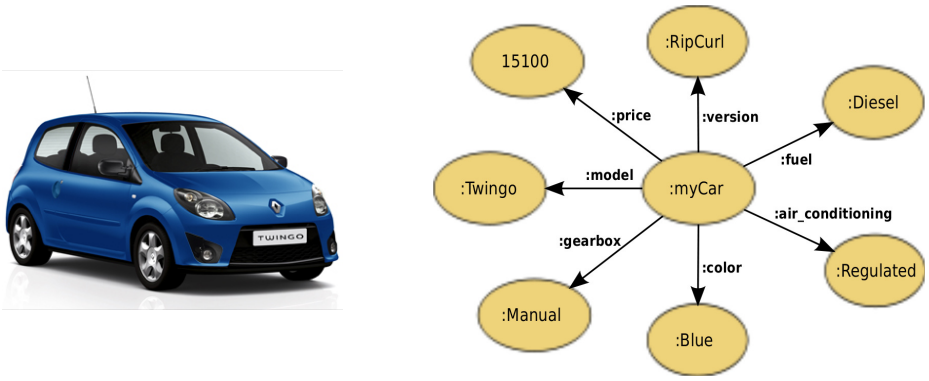

Fig. 1. The RDF representation of a particular car.

an object (here a vehicle). Therefore, it is modelled as an RDF property:

```
csp:variable a rdf:Property ;
  rdfs:domain csp:Solution .
```

The domain of values of a given variable is specified by the `rdfs:range` property:

```
:fuel rdfs:subPropertyOf csp:variable ;
  rdfs:label "The car fuel type." ;
  rdfs:domain :Vehicle ;
  rdfs:range [ owl:oneOf (:Diesel :Petrol :Electric) ] .
```

In this example, the variable `fuel` can take only 3 different values: `Diesel`, `Petrol` or `Electric`. A particular vehicle is assigned a URI and described using the values it takes for different variables (Fig. 1).

5.2 Fluents

A fluent (x,A), where $x \in \mathcal{X}$ and $A \subseteq D(x)$ is the association of a variable and a subset of its domain of values. Two properties are introduced to model a fluent in RDF: the property `csp:var` links a fluent to its variable and the property `csp:val` links a fluent to its different values. For example, the fluent (`fuel`,{`Petrol`,`Diesel`}) is represented by the RDF graph given in Fig. 2. In this figure, the empty node represents a blank node, but this blank node can be replaced by a URI in order to provide the fluent with an identifier.

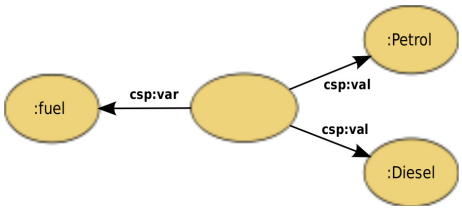

Fig. 2. The RDF representation of the fluent (`fuel`,{`Petrol`,`Diesel`}).

5.3 Boolean expressions

To represent Boolean expressions on fluents, a class `csp:BoolExpr` is introduced, along with 3 subclasses `csp:And`, `csp:Or`, and `csp:Not`, that model logical sub-expressions. The Boolean operators $\wedge$, $\vee$, and $\neg$ can be viewed as functions taking Boolean expressions as arguments. Therefore, three properties `csp:and`, `csp:or`, and `csp:not` are introduced, to link an operator to its arguments (Fig. 3). The subject of each property can either be a blank node or a URI, thereby enabling each Boolean expression to be given an identifier.

5.4 Subsets of the product range

This vocabulary can be used to represent subsets of the product range. For example, the set of cars with manual gearbox and either diesel or petrol type of

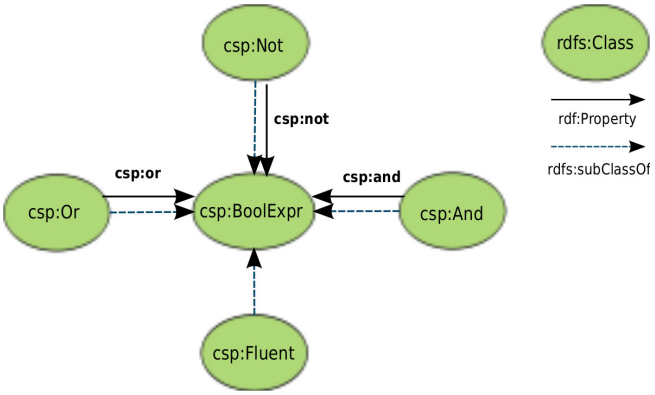

Fig. 3. An RDF Schema to represent Boolean expressions.

fuel corresponds to the conjunction of fluents:

$$(\texttt{gearbox}, \{\texttt{Manual}\}) \wedge (\texttt{fuel}, \{\texttt{Petrol}, \texttt{Diesel}\})$$

which is written in RDF (here with Turtle syntax) as:

```
:myCarSet a csp:And ;
  csp:and
    [csp:var :gearbox ; csp:val :Manual],
    [csp:var :fuel ; csp:val :Petrol,:Diesel] .
```

5.5 Constraints given in intension

Constraints are given either in intension or in extension. Constraints given in intension are represented using Boolean expressions on fluents. An example of such constraint expressed in propositional logic is: $\texttt{Electric} \implies \texttt{NoGearbox}$. This constraint states that electric cars have no gearbox. We could have introduced a representation of the Boolean operator $\implies$ in our vocabulary, but it would be only syntactic sugar since all Boolean operators can be rewritten using solely the $\wedge$, $\vee$, and $\neg$ operators. Indeed, the previous constraint can be rewritten as the following Boolean expression on fluents:

$$\neg(\texttt{fuel}, \{\texttt{Electric}\}) \vee (\texttt{gearbox}, \{\texttt{NoGearbox}\})$$

This Boolean expression is represented in RDF as:

```
:myConstraint a csp:BoolExpr ;
  csp:or
    [csp:not [csp:var :fuel ; csp:val :Electric]],
    [csp:var :gearbox ; csp:val :NoGearbox] .
```

5.6 Constraints given in extension

Other constraints are given in extension, and simply consist in the list of all valid combinations of values of a set of variables. Consider for example the compatibility array given in Tab. 1 for the variables `fuel` and `gearbox`. In this

gearbox \ fuel	Diesel	Petrol	Electric
Manual	x	x	
Automatic	x	x	
NoGearbox			x

Table 1. A compatibility array for the variables `fuel` and `gearbox`.

array, the crosses specify the valid combinations of values of these two variables. So the interpretation of such a compatibility array is that uncrossed combinations are illegal, i.e., no solution of the constraint network should include one of these combinations. For some compatibility arrays, it is also required that each crossed combination must be satisfiable, i.e., there must exist at least one solution of the constraint network that includes this combination. A list of valid combinations of values for a set of variables is represented in RDF(S) by a list of tuples, i.e., an RDF list of conjunctions of elementary fluents. The presence of the additional requirement regarding the satisfiability of each crossed combination is modelled by adding an attribute `csp:isSatisfiable` that points to a Boolean literal. So the compatibility array given in Tab. 1 is represented in RDF by:

```
:myRelation a csp:Relation ;
  csp:isSatisfiable
    "true"^^<http://www.w3.org/2001/XMLSchema#Boolean> ;
  csp:supports (
    [csp:and
      [csp:var :fuel ; csp:val :Diesel],
      [csp:var :gearbox ; csp:val :Manual]]
    [csp:and
      [csp:var :fuel ; csp:val :Petrol],
      [csp:var :gearbox ; csp:val :Manual]]
    [csp:and
      [csp:var :fuel ; csp:val :Diesel],
      [csp:var :gearbox ; csp:val :Automatic]]
    [csp:and
      [csp:var :fuel ; csp:val :Petrol],
      [csp:var :gearbox ; csp:val :Automatic]]
    [csp:and
      [csp:var :fuel ; csp:val :Electric],
      [csp:var :gearbox ; csp:val :NoGearbox]]) .
```

6 Comparison with OWL and SPARQL/SPIN

The syntax of language presented so far seems to be very close to the OWL RDF syntax and the transformation from our language to OWL is quite straightforward. For example, the properties `csp:or`, `csp:and` and `csp:not` can be translated into the OWL properties `owl:intersectionOf`, `owl:unionOf` and `owl:complementOf`. A fluent can also be represented in OWL as an existential restriction. For example, conjunction of fluents

$$(\texttt{gearbox}, \{\texttt{Manual}\}) \wedge (\texttt{fuel}, \{\texttt{Petrol}, \texttt{Diesel}\})$$

which was introduced in section 5.4 to represent the set of cars with manual gearbox and either diesel or petrol type of fuel can be written in OWL as:

```
:myCarSet rdf:type owl:Class ;
            owl:equivalentClass
            [ rdf:type owl:Class ;
              owl:intersectionOf (
              [ rdf:type owl:Restriction ;
                owl:onProperty :fuel ;
                owl:someValuesFrom
                [ rdf:type owl:Class ;
                  owl:oneOf (:Petrol :Diesel)]]
              [ rdf:type owl:Restriction ;
                owl:onProperty :gearbox ;
                owl:someValuesFrom
                [ rdf:type owl:Class ;
                  owl:oneOf (:Manual)]])] .
```

However, Renault has currently no need of using OWL semantics for constraint satisfaction as the reasoning is done by its external CSP solver, and the OWL RDF syntax is somewhat cumbersome. Thus, adopting OWL RDF syntax only as a serialisation means might be of limited interest. On the other hand, providing a complete representation of a CSP instance in OWL that would enable constraint satisfaction by the means using OWL inference would require supplementary axioms (Tab. 2). These axioms correspond to a set of constraints that are implicitly added by the CSP solver at execution time but are not explicitly represented in the definition of the CSP instance. They would be needed in OWL e.g., to make the unique name assumption and to ensure that in any solution of the CSP a variable is assigned one and only one value.

A translation to SPARQL/SPIN syntax would be also quite straightforward. A Boolean expression corresponds to a SPARQL abstract query. The `csp:and`, `csp:or` and `csp:not` operators translate respectively to a set of basic graph patterns, a UNION of basic graph patterns, and a NOT EXIST filter expression. For example, the above conjunction of fluents could be expressed as the following SPARQL query, in which `?this` is the special SPIN variable that binds to the current instance of the class the constraints apply to:

```
ASK {
  ?this :gearbox :Manual .
  { ?this :fuel :Diesel } UNION { ?this :fuel :Petrol }
}
```

The SPIN serialisation of this query gives:

```
:myCarSet    a     sp:Ask ;
     sp:where ([ sp:object :Manual ;
                 sp:predicate :gearbox ;
                 sp:subject spin:_this
               ] [ a     sp:Union ;
                 sp:elements (([ sp:object :Diesel ;
                                 sp:predicate :fuel ;
                                 sp:subject spin:_this
                               ]) ([ sp:object :Petrol ;
                                 sp:predicate :fuel ;
                                 sp:subject spin:_this
                               ]))
               ]) .
```

However, expressing PRS constraints in SPARQL/SPIN seems a bit artificial since no constraint-checking mechanism is needed by Renault, which relies on its own CSP solver to check whether a constraint (or a set of constraints) is satisfied or not.

Constraint	**Additional Axiom**	**Description**
Unique Name Assumption	`DifferentIndividuals`	Two different variable (resp., value) identifiers can not refer to the same individual.
var at most 1	`FunctionalProperty`	In any solution of the CSP a variable is assigned at most one value.
var at least 1	`SomeValuesFrom`	In any solution of the CSP a variable is assigned at least one value.

Table 2. Some additional axioms that would need to be added to an OWL knowledge base to enable constraint satisfaction using OWL inference.

7 Adhering to the Linked Data principles

One last step is required to seamlessly use these Boolean expressions in a Linked Data architecture: recognising them as first class citizens, a status they truly deserve, as they do represent very concrete "real world things", i.e. precisely characterised sets of vehicles, in the case of Renault PRS. To do so, we also need to assign them URIs, and — as per compliance with the Linked Data principles — making those URIs dereferenceable, and returning information about corresponding subset of the range (first of all their definition in RDF) when they

are accessed. However, as the possible number of distinct cars (and therefore of subsets of the range) is so huge, we cannot use URIs that are truly opaque: the server in charge of returning information about them must be able to reconstruct the Boolean expression from the URI (otherwise, we would need a database with an infinite number of lines to store them), without storing all the possibilities in an internal database or RDF store. All services to PRS can then be integrated on the enterprise Linked Data bus as REST services. As an illustration, and to highlight the interest of this compliance to Linked Data principles, let us consider the configuration question.

Configuration is a very classical and challenging problem regarding PRS. It is defined as the process of choosing interactively a vehicle by defining its features, the CSP solver ensuring that only possible choices are proposed to the user [7]. The configuration process can be seen as a traversal of Linked Data provided by a REST service. At any step, the configuration state (i.e, the list of choices already done) corresponds to a Boolean Expression over the variables of the PRS (typically, a conjunction of elementary fluents). Following our model, a URI provides, in particular, the links to the following step in the configuration process (i.e. the choices that can be picked up), as well as any appropriate marketing information (including price, description of features, etc.)

The HTML page presented to the user can be built from that data. Some RDFa in the page maintains the link to the "real world thing" it is about — a car that is partially defined, in other word, a subset of the range. Which is a very interesting thing to take care of: it captures indeed the exact expression of the customer's choice at the moment. This opens some interesting possibilities, in particular from a marketing point of view. One obvious example is related to recommendations (considering for instance the association of configuration with Facebook's OpenGraph[9] protocol and its "like" buttons). It makes no doubt also that the inclusion of RDFa data linking to open description of products in e-commerce context, such as GoodRelations[10], will soon be useful for the structured information it can provide to search engines.

8 Conclusion and Future Work

In this paper, an RDF(S) representation of the main objects related to Renault's Product Range Specification have been provided. We discussed the various requirements to build this, as well as our modelling decisions and the way our model relates with other Semantic Web standards such as OWL or SPARQL. In addition, we discussed design patters for representing PRS on the Web, and their applicability for end-user applications.

While our model focus on the particular PRS representation at Renault, future work may include extensions to the model to cope with other PRS needs, as well as using the related model as an interoperability format between various CSP solvers. Yet, while being specific to our particular use-case, we demonstrated

[9] http://ogp.me
[10] http://www.heppnetz.de/projects/goodrelations

in this paper how Linked Data principles and Semantic Web technologies can be used to model one of the core components of the manufacturing and commercial process in the automotive industry.

Acknowledgements. The work presented in this paper is funded in part by Science Foundation Ireland under grant number SFI/08/CE/I1380 (Lion 2) and by a research grant from Renault SA.

References

1. Faisal Alkhateeb, Jean-François Baget, and Jérôme Euzenat. Constrained Regular Expressions in SPARQL. Technical report, EXMO - INRIA Rhône-Alpes, Grenoble, France, 2007.
2. Renzo Angles and Claudio Gutierrez. The Expressive Power of SPARQL. In *Proceedings of the 7th International Conference on The Semantic Web*, ISWC '08, pages 114–129, Berlin, Heidelberg, 2008. Springer-Verlag.
3. J.M. Astesana, L. Cosserat, and H. Fargier. Constraint-based modeling and exploitation of a vehicle range at Renault's: Requirement analysis and complexity study. In *ECAI2010, Configuration Workshop Proceedings*, 2010.
4. Christian Fürber and Martin Hepp. Using SPARQL and SPIN for Data Quality Management on the Semantic Web. In *Business Information Systems, 13th International Conference*, volume 47 of *LNBIP*, pages 35–46. Springer, 2010.
5. Hak-Jin Kim, Wooju Kim, and Myungjin Lee. Semantic Web Constraint Language and its application to an intelligent shopping agent. *Decision Support Systems*, 46(4):882 – 894, 2009. IT Decisions in Organizations.
6. Craig McKenzie, Peter M. D. Gray, and Alun D. Preece. Extending SWRL to Express Fully-Quantified Constraints. In *RuleML 2004*, volume 3323 of *LNCS*, pages 139–154. Springer, 2004.
7. Bernard Pargamin. Vehicle Sales Configuration: the Cluster Tree Approach. In *ECAI2002, Configuration Workshop Notes*, 2002.
8. Peter F. Patel-Schneider and Ian Horrocks. A comparison of two modelling paradigms in the Semantic Web. *Journal of Web Semantics*, 5(4):240 – 250, 2007.
9. Alun Preece, Stuart Chalmers, Craig McKenzie, Jeff Z. Pan, and Peter M.D. Gray. A semantic web approach to handling soft constraints in virtual organisations. *Electronic Commerce Research and Applications*, 7(3):264 – 273, 2008.
10. Alun D. Preece, Kit ying Hui, W. A. Gray, Philippe Marti, Trevor J. M. Bench-Capon, Zhan Cui, and Dean M. Jones. Kraft: An agent architecture for knowledge fusion. *Int. J. Cooperative Inf. Syst.*, 10(1-2):171–195, 2001.
11. Olivier Roussel and Christophe Lecoutre. XML Representation of Constraint Networks: Format XCSP 2.1. *CoRR*, abs/0902.2362, 2009.
12. François-Paul Servant. Semantic Web Technologies in Technical Automotive Documentation. In *Proceedings of OWLED 2007*, volume 258 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.
13. Jiao Tao, Evren Sirin, Jie Bao, and Deborah L. McGuinness. Integrity Constraints in OWL. In *AAAI 2010*. AAAI Press, 2010.
14. Edward Tsang. *Foundations of Constraint Satisfaction*. Academic Press, London and San Diego, 1993.
15. Moshe Y. Vardi. Constraint satisfaction and database theory: a tutorial. In *Proceedings of ACM PODS '00*, pages 76–85, New York, NY, USA, 2000. ACM.
16. David Wood, editor. *Linking Enterprise Data*. Springer, 2010.

A Knowledge Dashboard for Manufacturing Industries

S. Mazumdar[1], A. Varga[2], V. Lanfranchi[2] and F. Ciravegna[2]

[1]Information School
[2]OAK Group, Department of Computer Science
University of Sheffield, Regent Court – 211 Portobello Street,
S1 4DP Sheffield, UK
s.mazumdar@sheffield.ac.uk
{a.varga, v.lanfranchi, f.ciravegna}@dcs.shef.ac.uk

Abstract. The manufacturing industry offers a huge range of opportunities and challenges for exploiting semantic web technologies. Collating heterogeneous data into semantic knowledge repositories can provide immense benefits to companies, however the power of such knowledge can only be realised if end users are provided visual means to explore and analyse their datasets in a flexible and efficient way. This paper presents a high level approach to unify, structure and visualise document collections using semantic web and information extraction technologies.

Keywords: Semantic Web, Information Visualisation, User Interaction.

1 Introduction

Modern manufacturing is a complex domain where productivity and efficiency are strongly affected by a broad range of factors such as site locations, cultural values, management decisions and communication capabilities. For example, large manufacturing organizations are usually globalised, with facilities geographically distributed, making use of multiple manufacturing machines, interacting with several suppliers and warehouses. Also, a recent trend in large organisations has been the presence of dynamic, interdisciplinary working groups and communities of practice who require rapid, flexible customisation of information to their specific needs [1]. At the same time, the information they generate needs to be shared with the rest of the organisation, and hence, must be presented to other communities in ways that can be easily understood (and correctly interpreted) and reused [2].

The underlying commonality between these phenomena is information availability: if information is captured, stored and shared between different departments and locations then efficient communication can be reached and stronger support for managerial decisions can be provided. Unfortunately this information is often collected in a wide variety of formats (e.g., text files, images, PDF documents) and

dispersed in independent repositories, including shared directories, local and company-wide databases, ad hoc information systems, etc. Critical knowledge may be hidden in the huge amount of manufacturing data, and the cost of exhaustively identifying, retrieving and reusing information across this fragmentation is very high and often a near impossible task.

This paper presents how Semantic Web and Information Extraction (IE) technologies can be adopted to unify such collections of documents and formalize their knowledge content, bringing together information from different domains, which can feed into organisational knowledge. Visualisation techniques can then be applied on top of the semantically structured data to explore, contextualise and aggregate it, offering multiple perspectives on the information space and provide analytic tools that could support users in spotting trends and identifying patterns and relationships. In order to achieve this goal two steps are required:

- Knowledge Acquisition: acquiring information from different documents and corpora and semantically structuring it in a semi-supervised manner.
- Knowledge Visualisation: creating multiple views over the semantic knowledge space.

Our methodology is innovative compared to previous literature (analysed in Section 2) as it defines the Knowledge Acquisition and Visualisation steps at an abstract level: the use of ontologies to extract, structure and visualise information make our approach flexible, reusable and extensible.

The Knowledge Acquisition and Visualisation steps will be described in details in Section 3, before providing implementation details (Section 4) and discussing conclusions and future work (Section 5).

The following scenario (taken from SAMULET[1], an existing research project on advanced manufacturing in the aerospace industry in which the authors are involved) has been considered as a foundation for the work: in a manufacturing industry a huge number of components are produced every day based on design data provided by Design departments, and are reused in other divisions of the company. When these components are produced manufacturing data is collected such as manufacturing time, location of the plant and of the manufacturing machine, type of component and details (possibly linked to design data). Additional information includes the person and machine responsible for the production, manufacturing costs and so on. This data is collected in a wide variety of formats (e.g. Excel spreadsheets, images, Word Documents), stored in independent repositories and often distributed using personal channels (such as e-mails, or shared network drives).

Manufacturing data are essential to resolving any issue that may arise on a component, in order to be able to clearly identify the driving factors behind the issue and to discover any significant trends or patterns related to individual manufacturing units/machines/personnel. Identifying non-obvious patterns in the data is fundamental to increasing productivity and efficiency: for example, a consistently poorly performing machine may be over-shadowed by a well performing manufacturing unit

[1] SAMULET (Strategic Affordable Manufacturing in the UK with Leading Environmental Technology), http://www.rolls-royce.com/investors/news/2009/280709_research_factories.jsp Last Accessed 14/04/2011

– data analysis and visualisation would help in spotting such trends and support putting corrective measures in place.

2 Related work

Our approach aims to provide a consistent and coherent environment for knowledge exploration in the manufacturing domain, encompassing knowledge acquisition and knowledge visualisation techniques. Related work in both these areas is now analysed, with particular emphasis on the adoption in the manufacturing domain.

2.1 Knowledge Acquisition

Traditional machine learning (ML) approaches for knowledge acquisition in manufacturing started to gain much attention only in recent years [3-10], mostly because the majority of the ML algorithms and tools require skilled individuals to understand the output of ML process [3]. However there has been some work on using traditional ML techniques for specific areas (such as fault detection, quality control, maintenance, engineering design, etc.) employing classification [6,7], clustering [8] and association rule mining [9,10] algorithms [3-5]. Classification algorithms were used for categorising data into different classes, for example classifying defects in the semi-conductor industry [5]. [6] employed a hybrid approach combining neural networks and decision tree classification algorithms for recognising false classifications in control chart pattern recognition (CCPR) thus facilitating quality control. [7] used decision tree algorithms for producing classification rules which were then saved in the competitive decision selector (CDS) knowledge bases enabling efficient job shop scheduling. Clustering algorithms were also used to group similar data into clusters, for example clustering the orders into batches for speeding up the product movement within a warehouse [5]. [8] applied fuzzy c-means clustering algorithm for identifying changes in traffic states thus improving the traffic management systems. Association rule mining algorithms were used to identify relationships among the attributes describing the data. [9] used association rule mining for detecting the source of assembly faults, thus improving the quality of assembly operations. [10] extracted association rules from historical product data to identify the limitations of the manufacturing processes. This information can then be used to improve the quality of the product and identify the requirements for design change.

Despite the increased interest, most of these approaches still lack portability and require a large amount of annotated data to achieve high performance, which is usually tedious and costly [13] to obtain. Furthermore recent advances in domain adaptation show that traditional Machine Learning (ML) approaches for IE are no longer the best choices [11,12]. These algorithms work only well when the format, writing style in which the data (e.g. manufacturing time, location of the plant and the machine) is presented is similar across different corpora [11,12]. In dynamic and heterogeneous corpora, these ML based systems need to be rebuilt for each corpus or format, making them impractical in many scenarios [11], such as the one presented in

this paper. To enable effective knowledge capture in manufacturing our approach employs an adaptable IE framework based on domain adaptation techniques, as presented in Section 3.

2.2 Knowledge Visualisation

Information visualisation techniques have been extensively adopted in the manufacturing domain to display and illustrate different processes such as simulation of model verification and validation, planning, decision making purposes and so on [14, 20]. Though most simulation results are based on data models, visualisations are essential to efficiently communicate information to end-users [15]. For example visualising CAD (Computer Aided Design) models enriched with performance scores provides analysts insights into the performances of different manufacturing units; alternative techniques provide ways for manufacturing units to validate their products against software models [14] (to evaluate compliance of manufacturing units to design).

Commercial tools generally focus on 3D visualisations of manufacturing models, factories, machines and so on. Examples of such commercially available tools used in the manufacturing industry include Rockwell's FactoryTalk[2] (remote monitoring of manufacturing processes); Autodesk's 3ds Max[3] and Maya[4] (modelling of product designs, animation, virtual environments); VSG's OpenInventor[5] (3D Graphics toolkit for developing interactive applications); DeskArtes ViewExpert[6] (viewing, verifying, measuring CAD data); Oracle's AutoVue[7] (Collaboration tool to annotate 3D or 2D models). These 3D commercial tools are also adopted in other industries like gaming, animation and so on [17]. However the high cost of 3D hardware and software makes this option unfeasible for smaller companies [16].

3D visualisation techniques have also been investigated in academic works, such as Cyberbikes, a tool for interaction with and exploration using head-mounted displays. [21] presents another example of 3D visualisation, providing factory floor maps which use animations to convey real-time events.

Using visualisations to communicate high-quality data in manufacturing scenarios can greatly reduce the amount of time and effort taken by engineers to resolve an issue: in a study by [18], engineers provided with animated visualisations combining several steps of a simulation could substantially reduce their analysis time. [22] discusses how factory map visualisation based navigation can often provide means to significantly reduce the cognitive load on analysts monitoring a typical manufacturing factory, when compared to list-based navigation of factory machines and their performances. Our approach takes inspiration from this latter works in aiming to

[2] FactoryTalk, http://www.rockwellautomation.com/rockwellsoftware/factorytalk/ Last Accessed 14/04/2011

[3] AutoDesk 3ds Max, http://usa.autodesk.com/3ds-max/ Last Accessed 14/04/2011

[4] AutoDesk Maya, http://usa.autodesk.com/maya/ Last Accessed 14/03/2011

[5] VSG OpenInventor, http://www.vsg3d.com/open-inventor/sdk Last Accessed 14/04/2011

[6] DeskArtes ViewExpert, http://www.deskartes.com/ Last Accessed 14/04/2011

[7] Oracle AutoVue, http://www.oracle.com/us/products/applications/autoVue/index.html Last Accessed 14/04/2011

provide efficient visualisation techniques that will reduce engineers cognitive workload and facilitate knowledge analysis.

3. Adding semantics to the manufacturing domain

Given the large scale and the heterogeneity both in data types and data formats, automatic techniques are required to process the data, unifying the document collections and formalising their knowledge content. In the following we distinguish between data, information and knowledge as proposed in [27]. Namely, data refers to the basic raw unit without any implicit meaning, information refers to data enhanced with context and perspective, and knowledge is information connected by patterns and relations. In our case the outcome of our Information Extraction framework is considered knowledge as it extracts entities and relations and assigns semantic meaning to them.

Our approach (shown in Figure 1) is therefore based on the use of a common knowledge representation in the form of ontologies describing the manufacturing domain. The ontologies are created manually so that the high-level ontology covers the generic manufacturing scope (common concepts and relationships between them), and the local ontologies (interlinked by the over-arching high-level ontology) capture the information specific to the different corpora. An adaptable Information Extraction framework considering the high-level ontology then extracts the common concepts across the corpora, thus avoiding ontology mapping and integration (see Section 3.1).

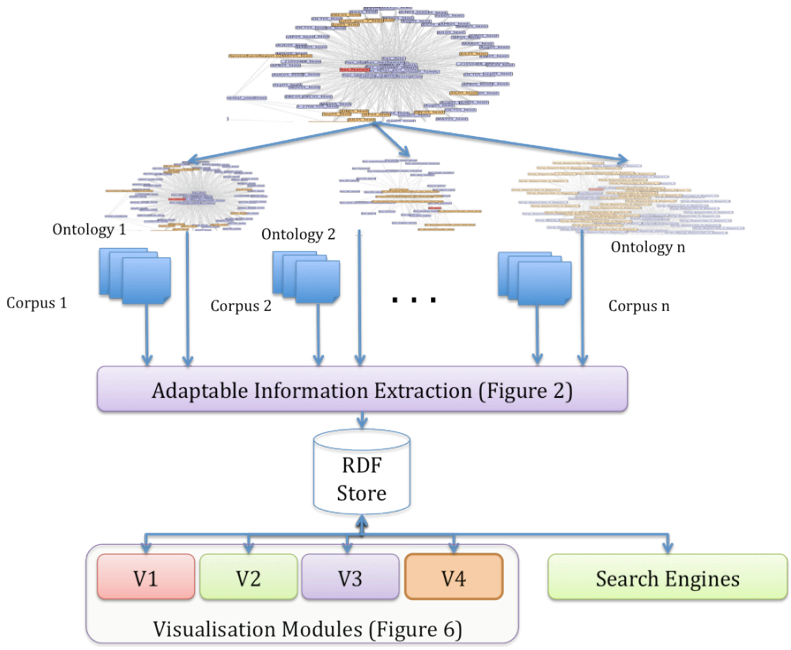

Figure 1 - The knowledge acquisition and visualisation process

The extracted information is then stored in a RDF store and available for query and visualisation (see Section 3.2).

3.1 Adaptable Information Extraction framework

The adaptable Information Extraction (IE) framework runs in a semi-supervised manner over the (automatically converted) textual versions of the documents in each corpus, extracting the relevant entities and relations and mapping them to the ontological concepts. The IE process is composed of two steps:

- Manual annotation of a subset of data by domain experts for training purposes.
- Unsupervised domain adaptation and annotation of the remaining documents using a Support Vector Machine (SVM)[8] [25] classifier.

Whilst this approach is common in literature [11,12] the novelty is in the portability of the classifier between different corpora with minimal supervision (using only a small amount of human annotations). For each new corpus (and document type) the initial classifier is augmented applying a feature representation approach [11,12] inspired on [29]. That is, the words from all the corpora are first clustered into semantic topics using Latent Dirichlet Allocation [28] topic model. Then new semantic features consisting of a set of most probable topics for each word are added to the classifier. This approach makes our IE system flexible and adaptable, enabling efficient knowledge acquisition across corpora.

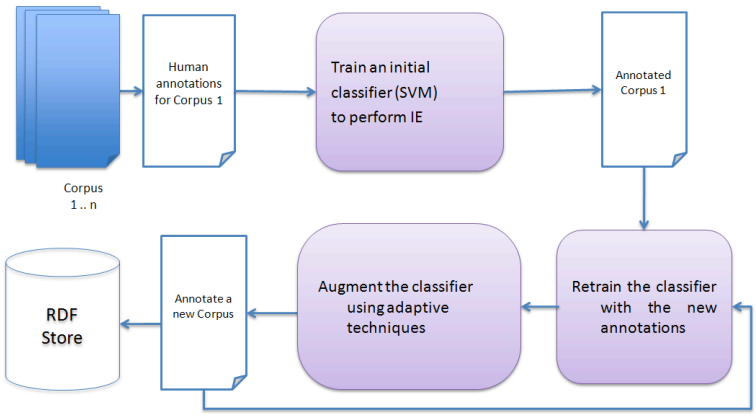

Figure 2 - The adaptable information extraction process

The extracted knowledge (ontology-based annotations) is then stored in a triple store in the form of RDF triples and used later for semantic visualisation. The current implementation of the IE framework also applies a terminology recognition [26] module for domain specific information extraction (e.g. type of component) within the SAMULET project, however the scope of the IE system is more generic and

[8] LibSVM tool: http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/ Last Accessed 14/04/2011

allows extracting domain independent entities and relations too (e.g. person, time, location).

3.2. Knowledge Dashboard

Our approach focuses on providing multiple knowledge visualisations at different granularity levels, using a semantic knowledge dashboard to support users in quickly gathering a broad insight of their datasets from differing perspectives. This approach is based on a set of interlinked ontologies (as explained in Section 3), which structure the knowledge from the different corpora and define relations between found entities. For each semantic entity type in the knowledge space a set of possible visualisations is defined: automatic inferences are then made on the type of entities and relations stored in the knowledge space to create the visualisation widgets. These visualisations can be customised to suit the user task, needs and preferences.

A dashboard interaction paradigm has been chosen as it provides large amounts of information in one interface, without compromising on clarity [23] and it is an increasingly common visualisation paradigm thanks to its adoption by several well-known websites like igoogle[9] and BBC[10]. Such an approach offers the possibility of dynamically choosing the best visualisation tool for the task in hand, as differently represented data can reveal different insights.

A detailed scenario is now presented to highlight the features of the knowledge dashboard and the interaction possibilities. In our hypothetical scenario, a manufacturing engineer (Bruce) working at a large aerospace organisation has access to six types of documents from different departments:

- Machine Performance Reports - describing operational performances of machines at manufacturing sites;
- Site Performance Reports - describing the overall performance of manufacturing sites;
- People Pages – websites of various individuals and authors of the reports;
- Machine Testing Reports – describing the findings of laboratory testing on machines at manufacturing sites;
- Quality Documents – reports discussing the outcome of various quality tests on manufactured products.
- Service Event Reports – reports discussing various service and maintenance operations conducted on engines over their lifetime.

These different report types have been analysed using our adaptable IE framework and semantic knowledge has been extracted and stored in a unified knowledge base. Visualisation ontologies have been defined for the different entities and relations and for the user preferences. These ontologies are used by the knowledge dashboard to automatically build the knowledge space and visualisation widgets. The selections of the visualisations are based on various features such as user preferences, usage history, current task, scale of retrieved datasets and types of data. The visualisation ontologies are essentially classifications of existing visualisations based on these

[9] iGoogle interface, http://www.google.com/ig, Last Accessed 04/03/2011
[10] BBC interface, http://www.bbc.co.uk/, Last Accessed 04/03/2011

parameters. Once a dataset is retrieved, the visualisation ontologies are used to infer the most effective visualisations for the dataset and users.

In our scenario, Bruce is investigating a condition where a lot of enquiries have been made to the manufacturing teams while service engineers were inspecting compressors of several engines during maintenance. Bruce first selects the relevant document sets from the combo boxes provided in the query interface. He then selects the filters 'Regime' and 'Component' and enters his query ('maintenance' and 'compressor' respectively).

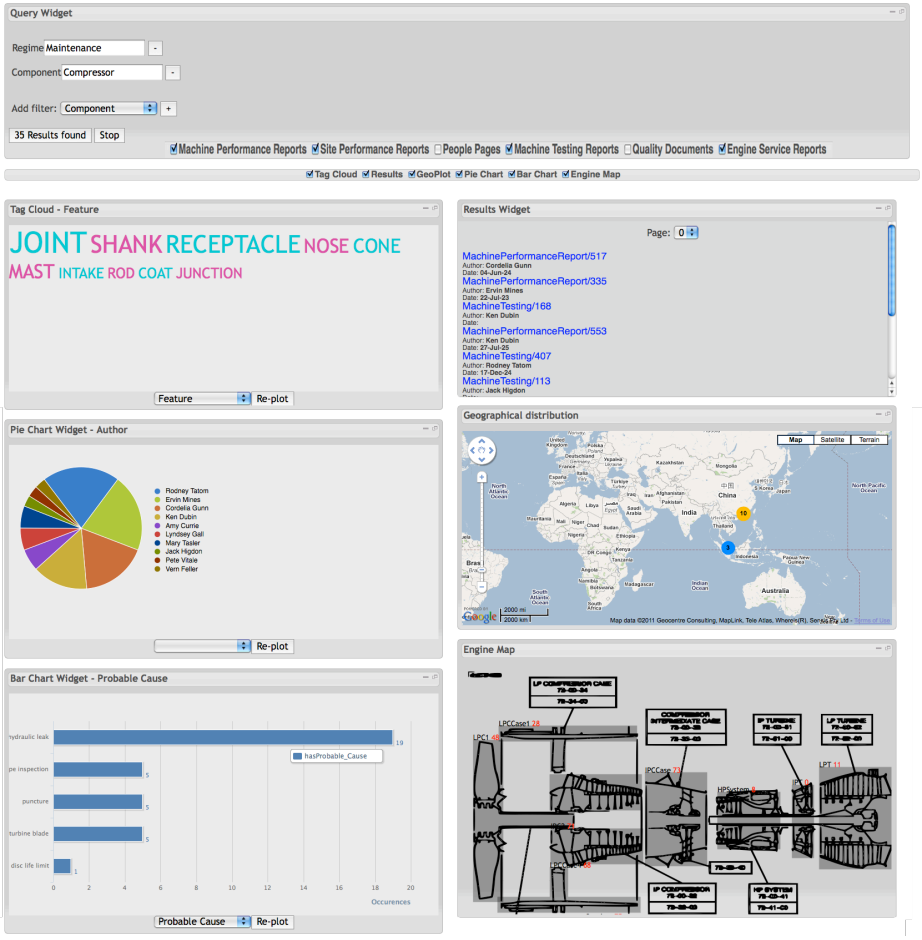

Figure 3 - Knowledge Dashboard

This initiates several queries to be sent to the backend from the interface. Bruce is then provided with several widgets (Figure 3), each of which present different facets (powered by different relations in the underlying semantic knowledge): the tag cloud informs Bruce that out of all the documents retrieved, most of the discussion has been related to features 'joint', 'Shank' and 'Receptacle' – this could indicate which

manufacturing machines might be responsible. The bar chart indicates that most of the documents discussing engine service events have also discussed 'hydraulic leaks'. The engine map groups the documents by the components they discuss – this shows how documents discussing 'compressors' also refer to other related components. These components are then displayed as grey areas, along with counts of how many documents have been found for each component. The pie chart provides a plotting by document authors – this enables Bruce to contact authors for further information and advice. The geographical plot provides the locations of manufacturing sites that are responsible for producing the components being described in the datasets. Using such visualisations, Bruce can now answer several common questions often asked during investigations: Where are the manufacturing machines located? What parts of an engine have the machines manufactured? What are the features of the parts that are being manufactured? Who are responsible for the manufacturing sites? From the multiple visualisation layers a summation of the knowledge emerges that can highlight previously unseen trends, patterns and issues/relations.

Thanks to the semantic knowledge and the background ontologies, the document collections can be visualised at different levels of granularity. For example an encompassing visualisation is achieved by displaying the whole document collections and comparing them, to show a high level view on the available facets without having to look at the individual document instances. The widgets are interactive, allowing zooming and selecting the preferred granularity level, from document to instance level. This follows the well-known principle of "overview first, zoom and filter, then details-on-demand" [24].

In our scenario if Bruce needs to analyse the performances of the organization from a manufacturing unit point-of-view, he can explore the knowledge space using a geographic view, then zooming in on an individual manufacturing site to reveal the site's floor plan along with the positions of the manufacturing machines. This floor plan is then enriched with performance statistics of the machines, extracted from the Site and Machine performance reports, as shown in Figure 4.

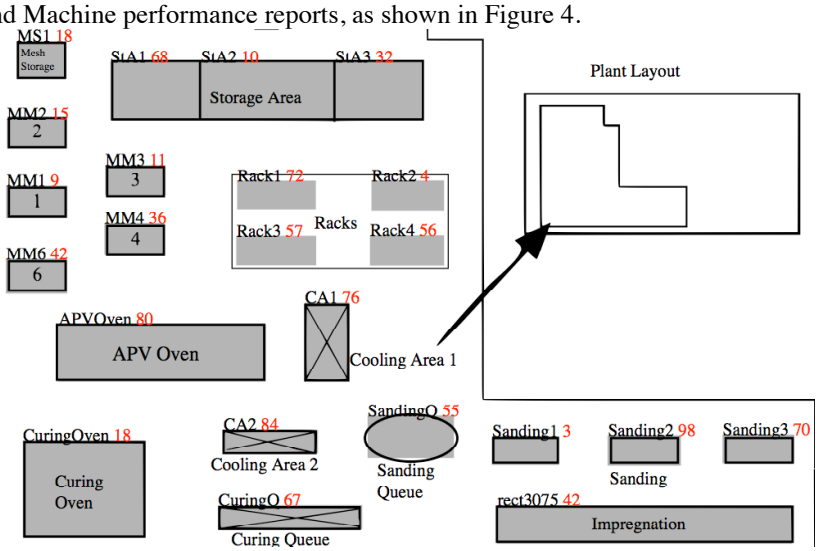

Figure 4 -Floor plan visualisation of knowledge instances

Users can also choose to look at the information from the product point-of-view, by clicking on sensitive areas of the engine, which loads a detailed view of the area of interest, enriched with instances from the documents returned as shown in figure 5. The documents are now grouped into different sections, which are shown as shaded areas- the numbers beside each section indicate the number of retrieved documents related to that section.

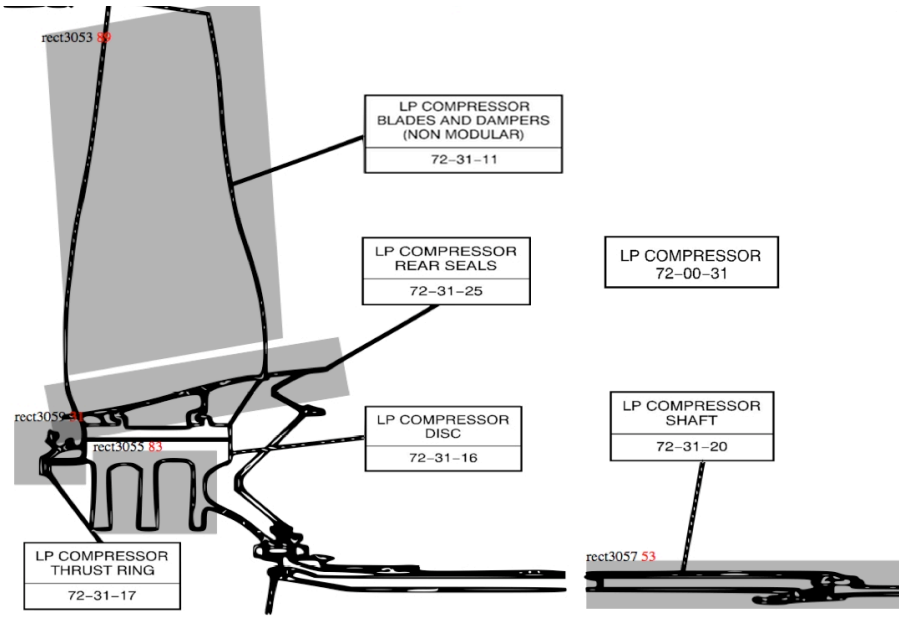

Figure 5 –Detailed view of engine, enriched with knowledge instances

4. Implementation

The system implementation is segmented into a knowledge acquisition and a knowledge exploration system. The knowledge acquisition system is an off-line process implemented in Java. The knowledge visualisation is a web-based dynamic and real-time application, consisting of a javascript frontend and a php backend that communicate using SPARQL queries over a semantic triplestore. The frontend is in charge of interpreting the user interactions and transforming them into corresponding SPARQL queries. For example, clicking on a section of a pie chart would be interpreted as a SPARQL SELECT query. These queries are then transmitted to the backend, which forwards the queries to triplestores. The results from the triplestores are then received by the backend and converted to JSON objects for visualisation in the interface. The system architecture is described in the Figure 6. The block in the

right side of the figure shows the front end, while the left side shows the backend processes.

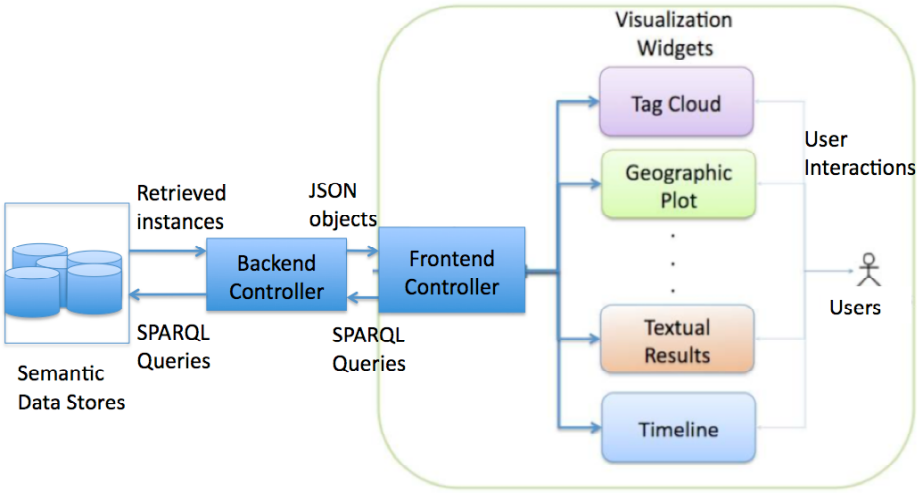

Figure 6 –Visualisation System Architecture

5. Discussion and Conclusions

This paper presented the approach developed during ongoing research work for a project about knowledge management in the manufacturing industry, focusing on how Semantic Web and Information Extraction technologies can be adopted to acquire knowledge from heterogeneous and disparate data whilst providing visualisations to explore, contextualise and aggregate the data, offering multiple perspectives on the knowledge space.

The developed approach is high level and domain independent as it is based on ontologies to structure and visualise knowledge it can be easily applied to a wider context than the manufacturing one. For example it could be applied to any business unit inside a large organisation (i.e. design, service and manufacturing). Expanding the domain will enable organisations to create a large integrated knowledge space available for sharing and reuse.

Future work will concentrate on extending our methodology to different corpora and in enriching the visualisation techniques to better match the user needs. As the project adopts a participatory design paradigm, real users are constantly providing feedbacks on mock-ups and vision demonstrators, to make sure the final prototype will be meeting their needs. This will be complemented by a comparative study of the developed prototype and the current software search systems being used by engineers. Moreover a final user evaluation will be carried out in a real-life scenario to assert the user satisfaction and acceptance of the new technology and a separate in-vitro

evaluation will be conducted to test the efficiency and efficacy of the Adaptable IE framework in terms of precision, recall and F-Measure.

Acknowledgments. This research has been supported by the SAMULET project, co-funded by Technology Strategy Board and Rolls-Royce plc. For privacy and security reasons the dataset under analysis is private and all images and data samples have been anonymised.

References

1. Wenger, E. Communities of Practice: Learning, Meaning, and Identity. Cambridge University Press, 1998.
2. Bhagdev, R., Chakravarthy, A., Chapman, S., Ciravegna, F., Lanfranchi, V. Creating and Using Organisational Semantic Webs in Large Networked Organisations, In 7th International Semantic Web Conference, Karlsruhe, 2008.
3. Harding, J. A., Shahbaz, M., Srinivas, Kusiak, A. Datamining in manufacturing: A review. American Society of Mechanical Engineers (ASME). Journal of Manufacturing Science and Engineering, 128(4), 969–976, 2006.
4. Wang K. Applying data mining to manufacturing: the nature and implications. Journal of Intelligent Manufacturing of Intelligent Manufacturing. 2007.
5. Choudhary, A., Harding, J., Tiwari, M. Data mining in manufacturing: a review based on the kind of knowledge. Journal of Intelligent Manufacturing, 20(5):501–521–521, 2009.
6. Guh, R. S. Real time pattern recognition in statistical process control: A hybrid neural network/decision tree-based approach. Proceedings of the Institution of Mechanical Engineers. Journal of Engineering Manufacture, 2005.
7. Kwak, C., Yih, Y. Data mining approach to production control in the computer integrated testing cell. IEEE Transactions on Robotics and Automation, 2004.
8. Crespo, F., Webere, R. A methodology for dynamic datamining based on fuzzy clustering. Fuzzy Sets and Systems, 2005.
9. Cunha, D., Agard, B., Kusiak, A. Data mining for improvement of product quality. International Journal of Production Research, 2006.
10. Shahbaz, M., Srinivas, Harding, J. A., Turner, M. Product design and manufacturing process improvement using association rules. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 2006.
11. Pan, S. J., Yang, Q. A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, 22(10):1345–1359, 2010.
12. Jing, J.. A literature survey on domain adaptation of statistical classifiers. Technical report, 2008.
13. Ciravegna, F., Dingli, A., Petrelli, D., and Wilks, Y. Timely and nonintrusive active document annotation via adaptive information extraction. In Proc. Workshop Semantic Authoring Annotation and Knowledge Management, 2002.
14. Rohrer, R. Visualization and its importance in manufacturing simulation, Industrial Management. 1996.
15. Rohrer, M. W.: Seeing is believing: the importance of visualization in manufacturing simulation. Proceedings of the 2000 Winter Simulation Conference, 2000.
16. Kamath, R. S, Kamat, R. K. Development of cost effective 3D stereo visualization software suite for manufacturing industries. Indian Journal of Science and Technology, 2010.
17. Agrusa, R.; Mazza, V.G.; Penso, R. Advanced 3D Visualization for Manufacturing and Facility Controls., Human System Interactions, 2009.

18.Edgar, G. W. Visualization for non-linear engineering FEM analysis in manufacturing., Proceedings of the 1st conference on Visualization '90**,** 1990**.**
19.Gausemeier, J., Ebbesmeyer, P., Grafe, M.,Bohuszewicz, O. v. Cyberbikes - Interactive Visualization of Manufacturing Processes in a Virtual Environment. Proceedings of the Tenth International IFIP WG5.2/WG5.3 Conference on Globalization of Manufacturing in the Digital Communications Era of the 21st Century: Innovation, Agility, and the Virtual Enterprise, 1999.
20. Greif, M. The visual factory: building participation through shared information, 1989.
21.Zhong, Y., Shirinzadeh, B. Virtual factory for manufacturing process visualization, Complexity International, 2008.
22.Stowasser, S. Hybrid Visualization of Manufacturing Management Information for the Shop Floor, Human-computer interaction: theory and practice (Part 2), Volume 2, 2008.
23.Few, S**.** Information Dashboard Design: The Effective Visual Communication of Data, number 3900693099, O'Reilly Media, 2006.
24.Shneiderman, B. The eyes have it: A task by data type taxonomy of information visualization. In: Bederson, B., Shneiderman, B. (eds.) The craft of information visualization. Morgan Kaufman, San Francisco, 2003.
25.Thorsten Joachims. Estimating the generalization performance of a SVM efficiently. In Proceedings of International Conference on Machine Learning, 2000.
26.Butters, J. and Ciravegna, F.: Authoring Technical Documents for Effective Retrieval. In Proceeding of Knowledge Engineering and Knowledge Management by the Masses (EKAW), 2010.
27.Ackoff, R. L. From Data to Wisdom. Journal of Applied Systems Analysis, 16, 1989.
28.Blei, D., Ng, A. and Jordan, M. Latent Dirichlet Allocation. Journal of Machine Learning Research, 2003.
29.Honglei Guo, Huijia Zhu, Zhili Guo, Xiaoxun Zhang, Xian Wu, and Zhong Su. Domain adaptation with latent semantic association for named entity recognition. In Proc. HTL-NAACL, pages 281–289, June 2009.