GRACE Ontology Integrating Process and Quality Control

Paulo Leitão[1,2], Nelson Rodrigues[1], Claudio Turrin[3], Arnaldo Pagani[3], Pierluigi Petrali[3]

[1] Polytechnic Institute of Bragança, Campus Sta Apolónia, Apartado 1134, 5301-857 Bragança, Portugal
{pleitao, nrodrigues}@ipb.pt
[2] LIACC - Artificial Intelligence and Computer Science Laboratory, R. Campo Alegre 102, 4169-007 Porto, Portugal
[3] Whirlpool Europe, Cassinetta di Biandronno, Italy, {Claudio_Turrin; Arnaldo_Pagani; Pierluigi_Petrali}@whirlpool.com}

***Abstract*- Multi-agent systems paradigm is a suitable approach to implement distributed manufacturing systems addressing the emergent requirements of flexibility, robustness and responsiveness. In such systems, an ontology is a crucial piece to provide a common understanding on the vocabulary used by the intelligent, distributed agents during the exchange of shared knowledge. This paper describes the design of an ontology to define the structure of the knowledge that is used within a multi-agent system integrating process and quality control in production lines for home appliances, which is being developed within the EU FP7 GRACE (inteGration of pRocess and quAlity Control using multi-agEnt technology) project. The ontology schema is validated by instantiating for a case study derived from a washing machines production line.**

I. INTRODUCTION

A collaborative network, enterprise or production system comprises a set of interacting and heterogeneous hardware and software applications. In such collaborative distributed environments, a common understanding of the shared knowledge is required to guarantee their interoperability. In a similar way, in a multi-agent system (MAS) [1], characterized to be a distributed and heterogeneous system, each agent representing a factory, cell, device or application, has its own knowledge and needs to communicate in order to achieve a pre-defined goal or solve a problem. The interaction between distributed agents requires the understanding of the messages that are used to exchange the shared knowledge. This issue becomes difficult if each agent has its own knowledge structure, in analogy with a meeting with attendances coming from different countries and speaking different native languages.

The representation and organization of the shared knowledge is not an easy task, as pointed out by the study elaborated by the National Institute of Standards and Technology (NIST) that refers that the automotive sector in United States spends one billion dollars per year to solve interoperability problems [2]. In fact, the knowledge sharing may present several problems, namely due to:

- The lack of a common view related to conceptual and terminological terms, leading to confusion and reduced understanding.
- The inter-operability, reuse and sharing of the knowledge for a particular domain.

The solution is to use proper mechanisms or techniques that guarantee the common understanding and data semantics among distributed entities, as well the capability to reuse and share the knowledge. The concept of ontologies addresses this challenge.

The term ontology is vague and not precise. In spite of the several different definitions of ontology that can be found in the literature, e.g. see [3-7], it is consensual that an ontology creates shared understanding, enabling the exchange of knowledge and the capability to reuse that knowledge. In other words, an ontology defines the vocabulary and the semantics that are used in the communication between distributed entities, and the knowledge relating to these terms. An ontology together with a set of individual instances of classes constitutes a knowledge base.

The objective of this paper is the design of an ontology to support the knowledge representation that will be used within the multi-agent system integrating process and quality control in production lines, which is being developed within the EU FP7 GRACE (inteGration of pRocess and quAlity Control using multi-agEnt technology) project [8]. The GRACE ontology will provide the data structure to organize the knowledge that is shared and exchanged between the agents and enable the interoperability between them. In particular, the GRACE ontology formalizes the structure of the knowledge related to:

- The resources available in the production line.
- The product and process models that describe how to produce the catalogue of products.
- The description of the production history, including the results from the inspection tests and supporting the traceability process.

The rest of the paper is organized as follows: section 2 overviews the related work, section 3 presents the proposed ontology and section 4 discusses the validation of the ontology by instantiating for a case study. Section 5 discusses the integration of the designed ontology within the GRACE multi-agent system. At last, section 6 rounds up the paper with the conclusions.

978-1-4673-2420-5/12/$31.00 ©2012 IEEE

II. Related Work

An ontology is designed taking into consideration the domain particularities. The domain discussed in this work is the manufacturing field, and particularly the production lines for home appliances. In the literature, several ontologies addressing the manufacturing domain were proposed by the research community during the last years.

The EU FP6 PABADIS'PROMISE (Plant Automation based on Distributed System Product Oriented Manufacturing Systems for Re-Configurable Enterprises) project proposed a reference meta-ontology for manufacturing [9]. This ontology is generic, with each definition trying to be abstract covering a bigger domain. ADACOR (ADAptive holonic COntrol aRchitecture for distributed manufacturing systems) [10] defines an ontology for manufacturing control domain, which was formalized with the DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) language [11]. The FP6 EUPASS project developed ontologies both to structure the knowledge in assembly systems domain and lightweight versions of those ontologies to be used in runtime [12]. MASON (Manufacturing's Semantics Ontology) introduces an ontology with the same objectives, but is expressed with the OWL (Web Ontology Language) language in order to unify the ontologies using cognitive architectures, leading to an implementation of a generic manufacturing ontology [13].

Other attempts to establish generic manufacturing ontologies are the NIST's description of shop data model [14], the Automation Objects [15], the OOONEIDA proposal focusing on the infrastructure of automation components by applying the semantic web technologies [16], and TOVE (Toronto Virtual Enterprise Ontology) that describes an ontology for virtual enterprise modelling [17]. The ISO 15926 standard [18] aims to support the integration of industrial automation systems, being supported by an ontology taking into account diverse variables, including the space and time.

Other ontologies addressing more specific domains in the manufacturing field were proposed, such as the design of ontologies for flexible manufacturing systems [19], for transport systems [20], for assembly lines control [21], for agent-based reconfiguration of production processes [22], for rent-a-car business [23] and for supply chain and logistic planning [24]. FRISCO is a manufacturing ontology reference that supports the organization of knowledge in automotive supply chains [25].

The problem here is to find an ontology that perfectly fits on the pre-requisites established for the GRACE production lines domain, since some described ontologies are generic and others focusing specific application domains. The idea is to take the insights of several manufacturing ontologies, and particularly from PABADIS'PROMISE and ADACOR, and design an ontology for the agent-based system integrating process and quality control in production lines, that will be generic enough within the boundaries of the problem specifics.

III. GRACE Ontology

The proposed GRACE ontology aims to represent the knowledge associated to the washing machines production lines domain, which will be used in a MAS application to integrate the production and quality control processes.

The design of an ontology requires the definition of the vocabulary used by distributed entities, formalizing the concepts, the predicates (relation between the concepts), the terms (attributes of each concept), and the meaning of each term (type of each attribute). For this purpose, the ontology schema was edited using the Protégé framework (http://protege.stanford.edu/), which is a free, open source ontology editor and knowledge-base framework.

For an easy understanding, the GRACE ontology schema has been initially built using a schema similar to the UML (Unified Modelling Language) class diagram format, as illustrated in Fig. 1.

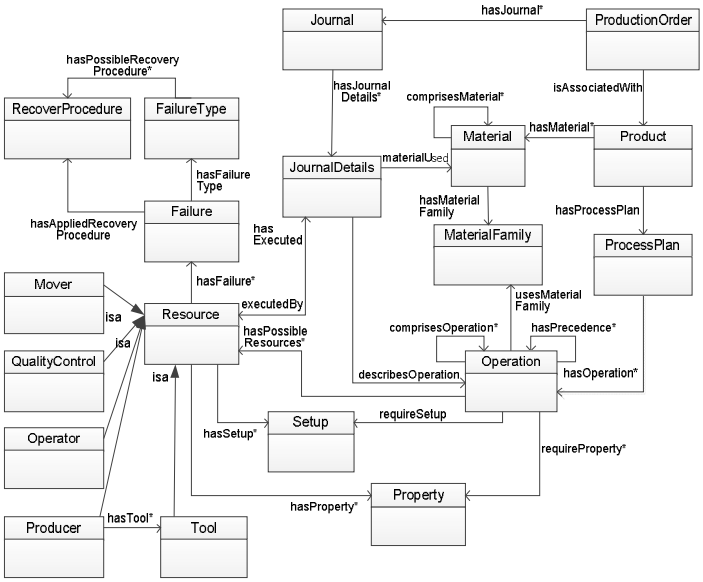

Fig. 1. GRACE ontology schema

In the next sections, the several ontological components will be analyzed.

A. Concepts

Concepts are expressions that indicate domain entities with a complex structure that can be defined in terms of classes or objects. The main concepts defined in the GRACE ontology are informally described as follows:

- *FailureType*: unexpected event type, like machine failure or delay, which degrades the execution of a production plan.
- *Material*: entity used during the production process, e.g. tubs, blocks of steel, bearings, nuts and bolts, according to the Bill of Materials (BOM).
- *MaterialFamily*: family of the material used during the production process, e.g. bearing or tub. (as an example the material bearing SKF123 can belong to the material family ball bearings).

- *ProcessPlan*: the manufacturing process to produce a product, i.e. the description of a sequence of operations (for producing a product) with temporal constraints.
- *Product*: entity (finished or semi-finished) that is produced by the enterprise in a value-adding process.
- *ProductionOrder*: entity obtained by aggregating customer orders for the production of products.
- *Property*: an attribute that characterizes a resource (i.e. a skill) or that a resource should satisfy to execute an operation (i.e. a requirement). It includes a mathematical operator associated to the property value, e.g. the speed is equal to 2000 r.p.m..
- *RecoveryProcedure*: entity that describes the procedure to recover from the occurrence of a failure.
- *Resource*: entity that can execute a certain range of operations as long as its capacity is not exceeded, e.g. a welding robot or a milling machine. Producer, quality controller, transporter, operator and tool are specializations of resource and inherit its characteristics.
- *Setup*: configuration that a resource should has to be able to execute a range of operations.

Events or processes are actions that can be performed by some agents. The proposed ontology comprises the following process entities:

- *Failure*: description of an occurred disturbance.
- *Operation*: a job executed by one resource like drilling, welding, assembly, inspection and maintenance, that may add value to the product or may measure the value of the product, e.g. the quality control.

Descriptions are a special kind of concepts, used as if they form a separate class, avoiding possible confusions between concepts and their description. In the proposed ontology, there are the following descriptions:

- *Journal*: description of the production of a product instance belonging to a production order executed in the production line.
- *JournalDetails*: description of the execution of an operation, including the time, participants and results.

B. Predicates

Relations or predicates establish the relationships among the concepts. The main predicates established in the GRACE ontology are:

- *comprisesMaterial(x, y)*: material *x* uses material *y*.
- *comprisesOperation(x,y)*: operation *x* contains operation *y*.
- *describesOperation(x , y)*: journal details *x* describes the execution of the operation *y*.
- *executedBy(x, y)*: operation described in the journal details *x* was executed by the resource *y*.
- *hasAppliedRecoveryProcedure(x,y)*: recovery procedure *y* was applied to solve the failure *x*.
- *hasExecuted(x, y)*: resource *x* has executed the operation described in the journal details *y*.
- *hasFailure(x, y, t)*: failure *y* occurred in resource *x* at time *t*.
- *hasFailureType(x, y)*: failure *x* belongs to the failure type *y*.
- *hasJournal(x, y)*: production order *x* comprises the production of several product items, each one described by the journal *y*.
- *hasJournalDetails(x,y)*: journal *x* comprises the description of the several operations to execute a product item, each one described by journal details *y*.
- *hasMaterial(x, y)*: product *x* has the material *y*.
- *hasMaterialFamily(x , y)*: material *x* is from the material family *y*.
- *hasOperation (x, y)*: process plan *x* contains operation *y*.
- *hasOperationPrecedence(x, y)*: execution of operation *x* requires the previous execution of operation *y*.
- *hasPossibleRecoveryProcedures(x, y)*: failure type *x* can be solved by applying the recovery procedure *y*.
- *hasPossibleResource(x , y)*: resource *y* is a candidate for the execution of the operation *x*.
- *hasProcessPlan(x, y)*: the production of product *x* requires the process plan *y*.
- *hasProperty(x, y)*: resource *x* has the property (skill) *y*.
- *hasSetup(x , y)*: resource *x* has the setup *y*.
- *hasTool(x, y, t)*: producer *x* has the tool *y* available in its internal magazine at time *t*.
- *isAssociatedWithProduct(x, y)*: production order *x* is associated to the product *y*.
- *materialUsed(x, y)*: journal details *x* describes that the material *y* was used to execute the operation.
- *requiresProperty(x, y)*: operation *x* requires the property *y* to be executed.
- *requiresSetup(x, y)*: operation *x* needs the setup *y* to be executed.
- *usesMaterialFamily(x, y)*: operation *x* uses material family *y*.

C. Attributes

Attributes are values relative to properties of concepts. The following examples are attributes of the *Product* concept:

- *productID*: a non-negative integer number that provides the unique identification of the product.
- *name*: the designation of the product.
- *description*: a statement describing the product.

D. Restrictions

Restrictions are conditions that should be satisfied when instantiating a class. The restrictions can be applied to the predicates, defining the range, domain and cardinality of the classes involved in the relation, and to the attributes of one class, defining the range and domain.

In the presented work, several restrictions were established for predicates and attributes. As an example, the predicate *hasJournalDetails*, between the classes Journal and JournalDetails, has the following specific restriction in terms

of cardinality: $\forall x(A(x) \rightarrow |\{y \mid R(x,y)\}| \geq 1)$, i.e. the cardinality is more than 1 (note: following the OWL formalism described in [26]). Other predicates have different restrictions in terms of cardinality, for example the predicate *isAssociatedtWithProduct* between the *ProductionOrder* and *Product* classes establishes the cardinality equal to 1.

In terms of restrictions for the attributes, it was established several restrictions for domain and range. As an example, the *journalID* attribute has the following restrictions:

- Domain: $\forall x \exists y\big(R(x,y) \rightarrow C\ (x)\big)$, where x is Journal.
- Range: $\forall x, y\big(R(x,y) \rightarrow C\ (y)\big)$, where y is Integer.

The fulfilment of the identified restrictions is crucial to preserve the consistency of the ontology. Since the OWL is a more expressive language than the Resource Description Framework (RDF), the GRACE ontology is described in OWL.

IV. Validation by Instantiating for a Case Study

At this stage, the ontology for production lines integrating quality and process control, was designed (and edited in the Protégé framework). An important step before its implementation and usage is the verification of its correctness and the adjustment of some ontological entities.

Several verifications can be performed, namely by using the Java framework for building Semantic Web (JENA) that provides several reasoning tools, like Pellet (http://pellet.owldl.com), to check the consistency and characteristics of the ontology, and by submitting the ontology to an OWL Validator to check the ontology compliance with the W3C standard. The GRACE ontology has passed with success the set of checking tests.

A manual validation can be performed by instantiating the ontology concepts for a particular case study, to support the verification of the ontology correctness and the detection of missing or misunderstanding ontological components. In fact, it allows illustrating the relevance of the proposed concepts, relations, and to improve, add, modify or remove some of the proposed concepts, relations, attributes or restrictions.

This section describes the manual verification performed by instantiation for a case study derived from a washing machine production line. Aiming a better understanding, the instantiation will be presented by analysing separately two different fragments of the ontology model.

Fig. 2 illustrates the validation of the fragment of the ontology comprising the "ProcessPlan", "Operation" and "Resource" concepts. Here, it is possible to verify that the process plan "FrontLoader", that defines the process to execute the product "_859201049010_0000", comprises the execution of three operations:

- "BearingInsertion-Program1", which uses components from the "ABearing", "BBearing" and "RearTub" material families.
- "SealInsertion-Program1", which should only be executed after the execution of the operation "BearingInsertion-Program1", and uses components from the "RearTub" and "ShaftSeal" material families.
- "Marriage-RearTub-Drum-Program1", which should only be executed after the execution of the operation "SealInsertion-Program1", and uses components from the "ABearing", "BBearing" and "CrossPiece" material families.

Also in this fragment, it is possible the possible resources to execute each operation. In this way:

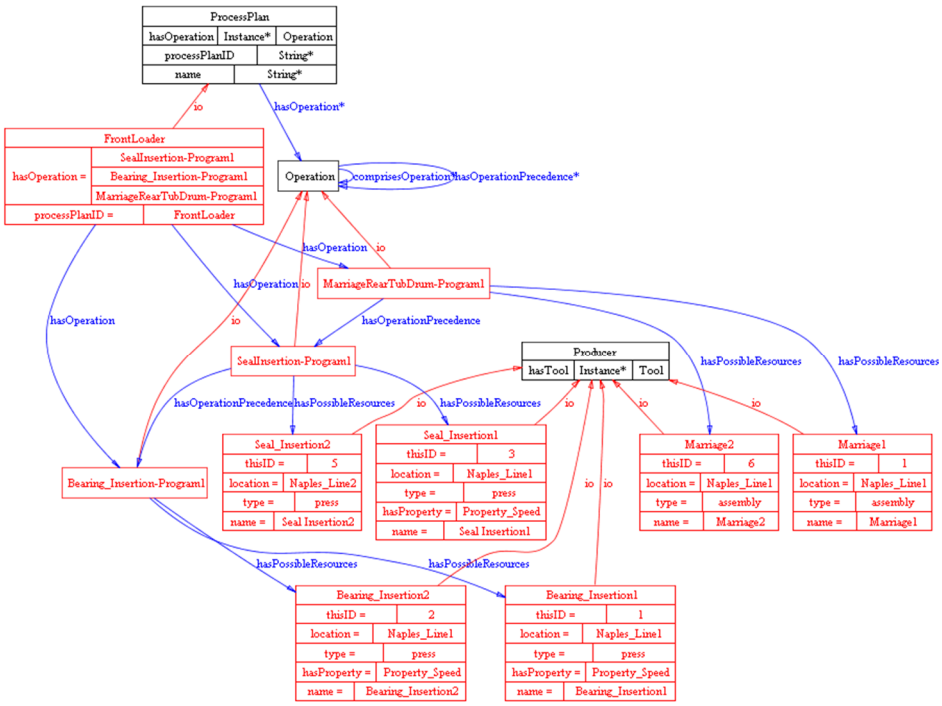

Fig. 2. Fragment for the "ProcessPlan", "Operation" and "Resource" classes and their instances

- The operation "BearingInsertion-Program1" can be executed by the resources "Bearing_Insertion1" and "Bearing_Insertion2".
- The operation "SealInsertion-Program1" can be executed by the resources "Seal_Insertion1" and "Seal_Insertion2".
- The operation "Marriage-RearTub-Drum-Program1" can be executed by the resources "Marriage1" and "Marriage2".

Another perspective of the ontology is related to the classes that describe the dynamic data related to the execution of production orders in the production line, i.e. the classes "ProductionOrder", "Journal" and "JournalDetails", Fig. 3.

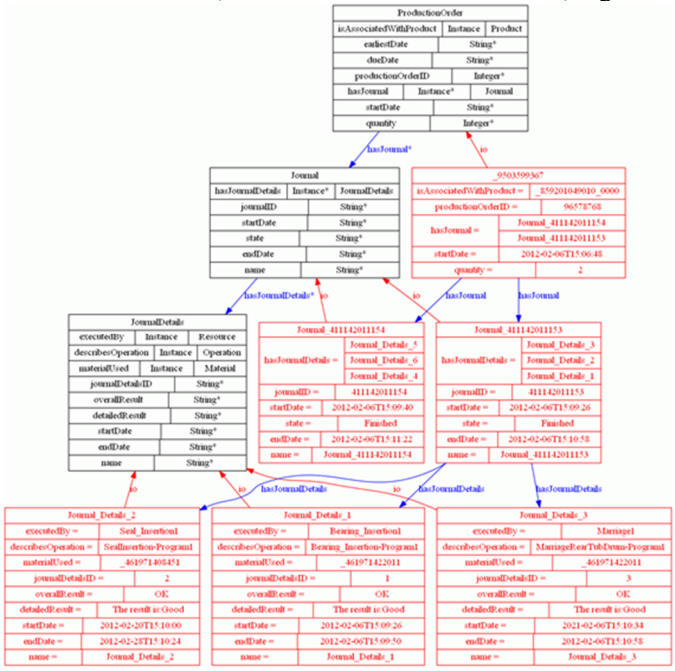

Fig. 3. Fragment for the "ProductionOrder", "Journal" and "JournalDetails" classes and their instances

Here, it is considered a production order to produce a batch of 2 items of the product "_859201049010_0000". The order leads to the production of the two appliances described by the instances "Journal_411142011153" and "Journal_411142011154" from the *Journal* class. Since the production of this product model requires the execution of three operations, as described in the process plan, each one of the two referred instances has three instances of the *JournalDetails* class, related to the description of the execution of each operation. For example, for the appliance described with the "Journal_411142011153", the details are:

- "Journal_Details1": the operation "BearingInsertion-Program1" was performed by the resource "Bearing_Insertion1" with an overall result of OK.
- "Journal_Details2": the operation "SealInsertion-Program1" was performed by the resource "Seal_Insertion1" with an overall result of OK.
- "Journal_Details3": the operation "Marriage-RearTub-Drum-Program1" was performed by the resource "Marriage1" with an overall result of OK.

The manual validation of the ontology allowed a better understanding of the domain and the correction of some misunderstanding issues in the design of the ontological concepts, predicates, attributes and restrictions.

At this stage, the designed ontology is ready to be used, i.e. integrated within the GRACE multi-agent system.

V. Implementation of the GRACE Ontology

The designed ontology plays a crucial role in the GRACE multi-agent system to enable a common understanding among the agents when they are communicating, namely to understand the message at the syntactic level (to extract the content correctly) and at the semantic level (to acquire the exchanged knowledge).

Since the GRACE multi-agent system is being developed using the Java Agent Development Framework (JADE) [27], which uses Java, a pertinent question is how to translate the ontology edited in Protégé to be used by the agents developed in JADE. Several options can be considered for this purpose.

The first option is to express the ontology in an OWL file. For this purpose, agents should be able to read OWL files and extract knowledge, communicating with small pieces of the ontology through FIPA (Foundation for Intelligent Physical Agents) protocols, e.g. using the Jena Framework [28] and Protégé OWL.

The second option is to use the Protégé plug-in, *OntologyBeanGenerator,* which allows generating Java files representing an ontology that can be used with the JADE framework. In this way, the ontology edited and validated in Protégé is exported to Java classes, being the knowledge represented by the instances of each class. The agents communicate by using java objects (classes) to share the knowledge between them. The major disadvantage of this solution is the loss of flexibility when the agents want to reason new facts and new rules to be included in the ontology (applying learning mechanisms). Additionally, the extraction process provided by this plug-in presents a malfunction that requires fixing the errors in the generated classes by hand.

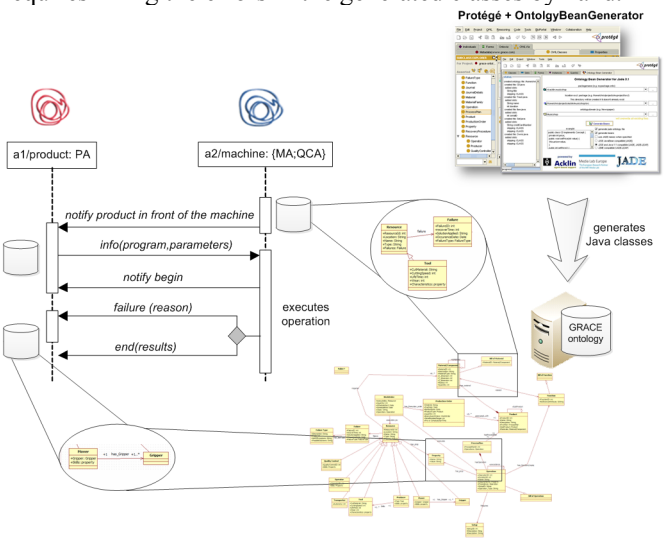

Fig. 4. Agents using ontologies to exchange knowledge

In this work, the integration of the ontology within the GRACE MAS system adopts the second approach, i.e. the use of the Protégé plug-in to generate the Java classes used by agents. Fig. 4 illustrates the use of the ontology (generated from the ontology schema edited in Protégé using the plug-in) to support the interaction among distributed agents, where the agents use the same ontology (but different fragments of the ontology) to express the shared knowledge that is exchanged.

The use of Java classes by the agents closes the several phases of the development of the GRACE ontology, started with the conceptualization, passing by the specification of the ontology schema and followed by its validation.

VI. Conclusions

The GRACE project intends to develop a collaborative multi-agent system which operates at all stages of a production line, integrating process control with quality control at local and global level. Ontologies play a crucial role in the development of such multi-agent system to provide the representation of the shared knowledge.

This paper describes the ontology designed to be used by the multi-agent system integrating process and quality control, and the validation of the ontology by instantiating for a case study derived from a washing machine production. It also discusses how the proposed ontology, edited and validated in Protégé, can be integrated in the multi-agent system that is being developed using the JADE agent development framework.

As future work, some effort will be dedicated to the implementation of the ontology schema within the multi-agent system.

Acknowledgment

This work has been partly financed by the EU Commission, within the research contract GRACE coordinated by Univ. Politecnica delle Marche and having partners SINTEF, AEA srl, Instituto Politécnico de Bragança, Whirlpool Europe srl, Siemens AG.

References

[1] Wooldridge, M., "An Introduction to Multi-Agent Systems", John Wiley & Sons, 2002.

[2] Szykman, S., Fenves, S.J., Keirouz, W., Shooter, S.B., "A Foundation for Interoperability in Next-Generation Product Development Systems", Computer-Aided Design, vol. 33, n. 7, pp. 545-559, 2011.

[3] Neches, R., Fikes, R.E., Finin, T., Gruber, T.R., Senator, T. and Swartout, W.R., "Enabling Technology for Knowledge Sharing", AI Magazine, vol. 12, n. 3, pp. 36-56, 1991.

[4] Gruber, T., "Toward Principles for the Design of Ontologies Used for Knowledge Sharing" International Journal of Human and Computer Studies, vol. 43, n. 5/6, pp. 907-928, 1995.

[5] Guarino, N., "Formal Ontology and Information Systems", Proceedings of the First International Conference on Formal Ontologies in Information Systems, Trento, Italy, pp. 3-15, 1998.

[6] Fensel, D., "Ontologies: a Silver Bullet for Knowledge Management and Electronic Commerce", Springer-Verlag, Berlin 2004.

[7] Lai, L.F., "A Knowledge Engineering Approach to Knowledge Management", Journal of Information Sciences, vol. 177, n. 19, pp. 4072-4094, 2007.

[8] Leitão P., Rodrigues, N., "Multi-Agent System for On-demand Production Integrating Production and Quality Control", V. Maník, P. Vrba, and P. Leitão (eds.): HoloMAS 2011, LNAI 6867, Springer, Heidelberg, pp. 84-93, 2011.

[9] Ferrarini, L., Veber, C., Luder, A., Peschke, J., Kalogeras, A., Gialelis, J., Rode, J., Wunsch, D., Chapurlat, V., "Control Architecture for Reconfigurable Manufacturing Systems: the PABADIS'PROMISE Approach", Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation (ETFA'06), pp.545-552, 2006.

[10] Leitão, P., Restivo, F., "ADACOR: A Holonic Architecture for Agile and Adaptive Manufacturing Control", Computers in Industry, vol. 57, nº 2, pp. 121-130, Elsevier, 2006.

[11] Borgo, S., Leitão, P., "The Role of Foundational Ontologies in Manufacturing Domain Applications", On the Move to Meaningful Internet Systems: 2004: CoopIS, DOA and ODBASE, R. Meersman and Z. Tari (eds.), Lecture Notes in Computer Science, vol. 3290, Springer-Verlag, pp. 670-688, 2004.

[12] Lohse, N., "Towards an Ontology Framework for the Integrated Design of Modular Assembly Systems", PhD thesis, University of Nottingham, 2006.

[13] Lemaignan, S., Siadat, A., Dantan, J.-Y., Semenenko, A., "MASON: A Proposal for an Ontology of Manufacturing Domain", Proc. of the IEEE Workshop on Distributed Intelligent Systems, pp. 195-200, 2006.

[14] McLean, C., Lee, Y., Shao, G., Riddick, F., "Shop Data Model and Interface Specification", NISTIR 7198, 2005.

[15] Lopez, O., Martinez Lastra, J.L., "Using Semantic Web Technologies to Describe Automation Objects", International Journal of Manufacturing Research, vol. 1, n. 4, pp. 482-503, 2006.

[16] Vyatkin, V., Christensen, J., Lastra, J., "OOONEIDA: An Open, Object-Oriented Knowledge Economy for Intelligent Industrial Automation", IEEE Transactions on Industrial Informatics, vol. 1, n. 1, pp. 4-17, 2005.

[17] Fox, M.S., "The TOVE Project: Towards A Common-sense Model of the Enterprise", Enterprise Integration Laboratory Technical Report, 1992.

[18] Batres, R., West, M., Leal, D., Priced, D. and Nakaa, Y., "An Upper Ontology based on ISO 15926", Computers & Chemical Engineering, vol. 31, n. 5-6, pp. 519-534, 2007.

[19] Vrba, P., Rakovic, M., Obitko, M., Maník, V., "Semantic Technologies: Latest Advances in Agent-based Manufacturing Control Systems", International Journal of Production Research, vol. 49, n. 5, pp. 1483-1496, 2011.

[20] Merdan, M., Koppensteiner, G., Hegny, I., Favre-Bulle, B., "Application of an Ontology in a Transport Domain", Proc. of the IEEE International Conference on Industrial Technology, pp. 1-6, 2008.

[21] Cândido, G., Barata, J., "A Mutliagent Control System for Shop Floor Assembly", V. Maník, V. Vyatkin and A.W. Colombo (eds.), HoloMAS'2007, LNAI 4659, Springer-Verlag, pp. 293-302, 2007.

[22] Al-Safi, Y. and Vyatkin, V., "An Ontology-based Reconfiguration Agent for Intelligent Mechatronic System", V. Maník, V. Vyatkin and A.W. Colombo (eds.), HoloMAS 2007, LNAI 4659, Springer-Verlag Berlin/Heidelberg, pp. 114-126, 2007.

[23] Andreev, M., Rzevski, G., Shviekin, P., Skobelev, P., Yankov, I., "A Multi-agent Scheduler for Rent-a-Car Companies", Maník, V. Strasser, T., and Zoitl, A (eds.), HoloMAS'2009, LNAI 5696, Springer-Verlag, Berlin Heidelberg, pp. 305-314, 2009.

[24] Andreev, M., Rzevski, G., Skobelev, P., Shveykin, P., Tsarev, A. and Tugashev, A., "Adaptive Planning for Supply Chain Networks", V. Maník, V. Vyatkin and A.W. Colombo (eds.), HoloMAS'2007, LNAI 4659, SpringerVerlag, pp. 215-224, 2007.

[25] Hellingrath, B., Witthaut, M., Böhle, C., Brügger, S., "An Organizational Knowledge for Automotive Supply Chains", Maník, V., Strasser, T. and Zoitl, A (eds.), HoloMAS 2009, LNAI 5696, Springer-Verlag Berlin Heidelberg, pp. 37-46, 2009.

[26] Kiko, K., Atkinson, C., "A Detailed Comparison of UML and OWL", Technical Report TR-2008-004, Department for Mathematics and Computer Science, University of Mannheim, 2008.

[27] Bellifemine, F., Caire, G., Greenwood, D., "Developing Multi-Agent Systems with JADE", Wiley, 2007.

[28] Grobe,M., "RDF, Jena, SparQL and the 'Semantic Web'", Proceedings of the 37th annual ACM SIGUCCS Fall Conference (SIGUCCS '09), ACM, New York, NY, USA, pp. 131-138, 2009.