An ontology for the definition and validation of assembly processes for evolvable assembly systems

An Ontology for the Definition and Validation of Assembly Processes for Evolvable Assembly Systems

Niels Lohse[†*] Hitendra Hirani[†] Svetan Ratchev[†] Michele Turitto[†]

[†]University of Nottingham
School of M3
Precision Manufacturing Group
University Park, Nottingham, NG7 2RD
[*]epxnl2@nottingham.ac.uk

Abstract

Assembly forms eighty per cent of the cost of manufacturing a product and this is where the greatest competitive advantage can be gained as assembly has not been fully understood in the research community. This paper reports on the development of a specification ontology to describe assembly processes at a level where they can be mapped to individual assembly modules. The ontology is an extension of the Process Specification Language (PSL) defined by NIST. The developed ontology is applied to an existing case to demonstrate the capabilities. It is believed that the development of this ontology will lead to the opening of further channels for research in modular control and modular assembly processes, which are building blocks for Evolvable Assembly Systems.

Keywords: *Evolvable Assembly Systems, Assembly Actions, Domain Ontology, Process Decomposition*

1 Introduction

Current practice in industry today centres around delivering customised solutions for the assembly of specific products. Once these products have been discontinued the assembly system has to follow as the system is designed to assemble that product only. Flexible assembly involves the use of flexible automation equipment to counteract this. However, the technology is too expensive for small and medium enterprises (SMEs) to adopt as it contains functionality that is not needed. The challenge is in delivering customisable flexible assembly solutions that can adapt to changing needs. This is part of the vision for the Evolvable Assembly System [1], where interchangeable modules can be deployed and used for specific assembly tasks supported by intelligent tools for the rapid configuration and deployment. Understanding and modeling assembly processes at a detailed level is a key aspect to making this possible. Also any such model needs to be suitable for the use in a distributed environment integrating all the stakeholders in the specification of the assembly system [2].

Various approaches have been published that contribute to assembly planning. Mostly these are dedicated to solving task distribution and line balancing problems [3]. QFD has been used to assign assembly processes to perform certain assembly operations [4], but there is limited work that focuses on the specification of the individual assembly actions that constitute the basic building blocks of any assembly process.

Rampersad [5] has formalised the link between the product, assembly process and the assembly system in his approach to robotic assembly system design. This has been reinforced with work by Stadzisz and Henrioud [6] and Vos [7]. Vos has used a set of classes to structure assembly processes this is at a higher level of abstraction. The existing approaches are not sufficient for specification and configuration on the level of modularity inherent to Evolvable Assembly Systems [8].

Hence, a new assembly process specification ontology is needed to link products, assembly processes and assembly equipment especially where modularity and future reconfiguration are concerned. The challenge for the specification of Evolvable Assembly Systems is the added complexity of understanding and propagating the change between the new and old system configuration.

The National Institute of Standards and Technology (NIST) has defined a generic Process Specification Language (PSL) [9]. PSL is mainly designed for the exchange of process models between different applications and it does not include any assembly specific concepts and constraints and does not cater for the needs of evolvable systems. The proposed ontology is an extension of PSL providing the missing detail.

This paper reports on an approach where assembly process requirements specifications are derived from user requirements specifications, which consist of the overall project requirements and the product model. An assembly process ontology is presented to describe and guide the assembly process specification for both new assembly system configuration as well as reconfigurations of existing assembly systems. This is demonstrated using an example of an automotive part. The paper is concluded with further discussion of the advantages and limitations of the proposed process specification ontology.

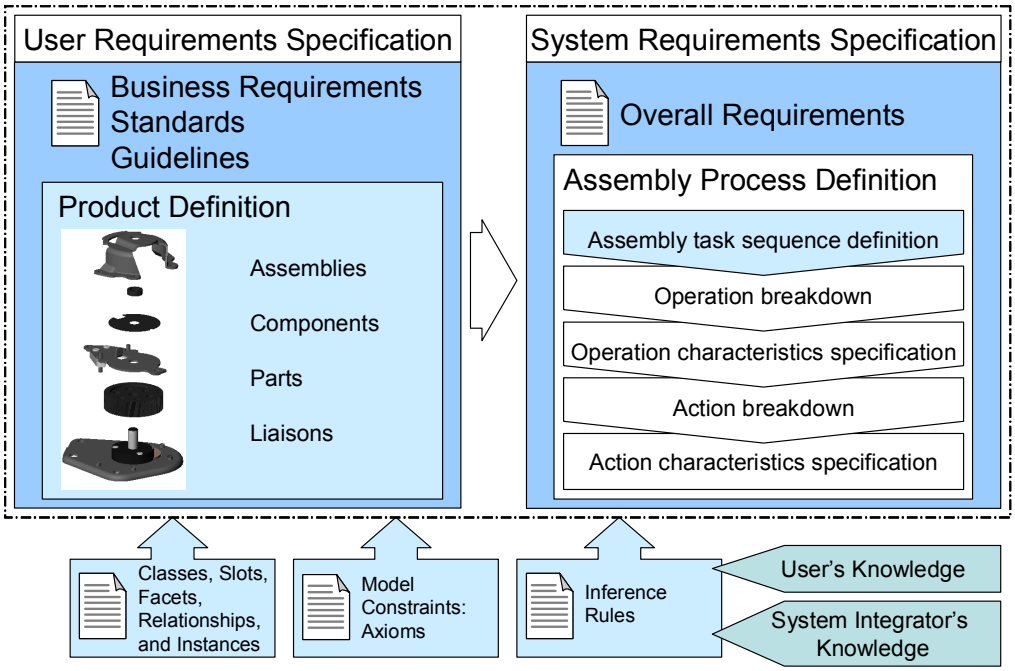

Figure 1: Overview of the Requirements Specification Methodology

2 Assembly Process Requirements Specification

A knowledge enriched requirements specification environment has been developed at the University of Nottingham [10]. This is based on extracting user requirements from the system user and converting these into assembly task requirements as per Figure 1.

User requirements specification consists of specification of business requirements, product definition, part definitions, part liaison definitions and specification of other constraints. This is underpinned by the system user's knowledge about the product and the ability of the user to facilitate an assembly system.

User requirements are converted into system requirements guided by inference rules. The result of system requirements specification is an assembly task specification and assembly module requirements. Task specifications include how the different parts are assembled to form the product. Assembly module requirements define the characteristics of the assembly modules required to assemble the product. This is underpinned by the system integrator's knowledge about how assembly systems are produced, including aspects of assembly processes.

The structure of an assembly process can generally be separated into three levels: task, operation, and action level. Tasks define the highest level of the process by defining the sequence in which the components are being assembled to form the final product. Operations define on an intermediate level the steps required to put the components together including feeding, handling, assembly, etc. Actions on the lowest level define the individual motions and other more hardware and control related activities.

Since the focus of this paper is on the definition of assembly operations and actions it is assumed that a complete product specification and at least one defined assembly task sequence to assemble this product exists and has already been defined. The product model encompasses the definition of its parts and their relations (liaisons). The assembly task sequence defines the order in which the liaisons of the product have to be established (see top of Figure 1).

During the first iteration of the assembly process definition the result is purely based on the user requirements and therefore mainly the product model. In later iterations the assembly process definition will also be influenced by constraints that existing equipment imposes and the assembly process would have to incorporate additional information as the understanding of the required equipment increases. This paper is focused on the first iteration, however, the proposed ontology is defined in a manner that caters for the needs further iterations and feedback from the equipment selection.

The proposed assembly process definition method guides the user through the specification by stepwise increases in the level of detail. Starting with the assembly task sequence, each assembly task is broken down into the required generic operations. Based on the characteristics of the product model, the specific type of operation is defined. The same process is repeated for the actions.

The product as well as the whole assembly process is defined based on a newly developed assembly ontology. The consistency of the assembly process definition is continuously checked during the specification using

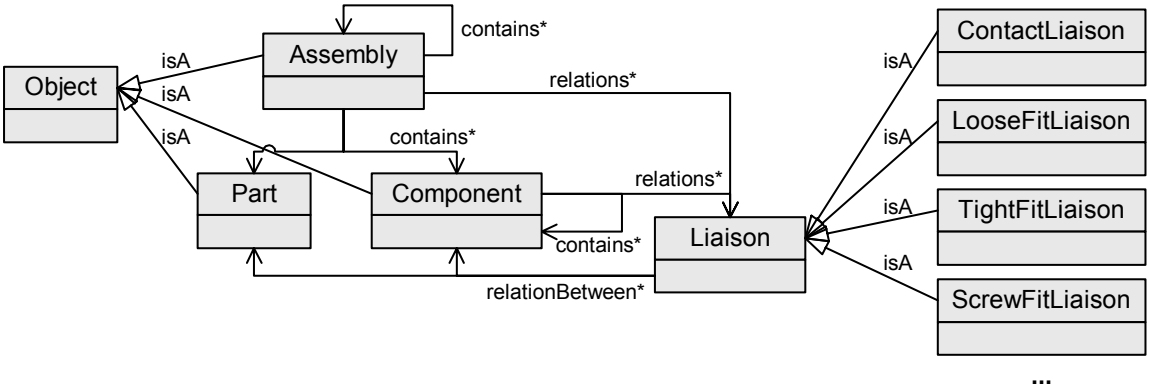

Figure 2: Product Domain Ontology Overview

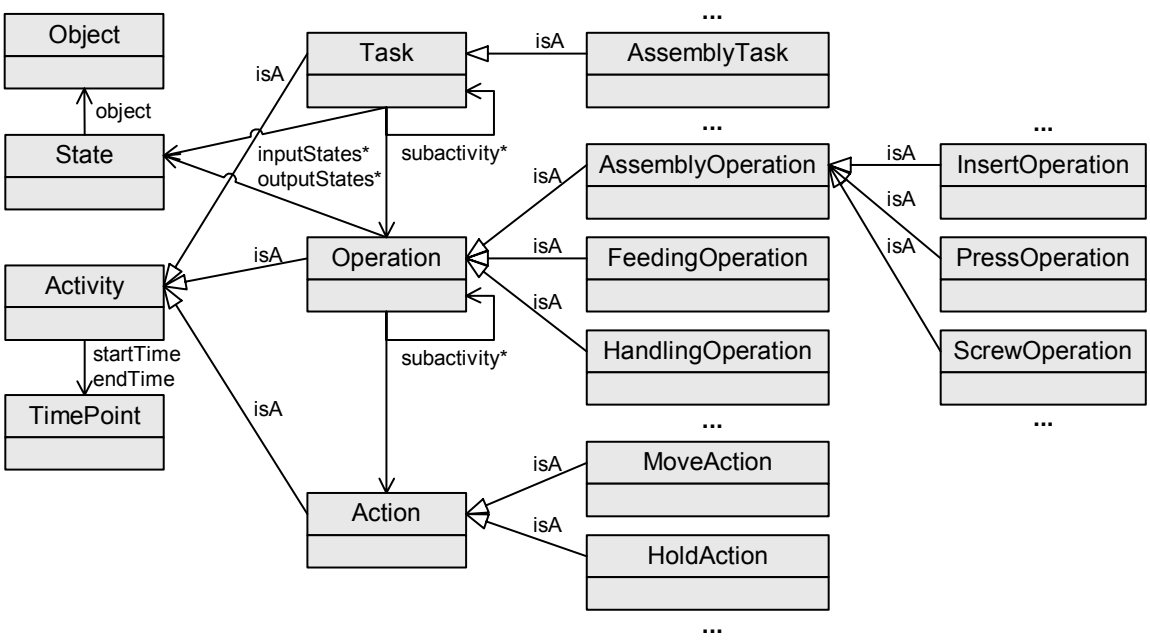

Figure 3: Assembly Process Domain Ontology Overview

detailed axioms. The definition of the process characteristics is guided by inference rules. The inference rules specify the relationships between product attributes and required assembly process characteristics.

3 Assembly Process Ontology

A frame based product and assembly process domain ontology has been defined to model all the concepts involved in the assembly process definition. The ontology defines the classes, slots, and relationships of the product and assembly process domain. The classes can be instantiated to define a specific product and its assembly process. The domain ontology has been extended to include more specific constraints which are expressed as axioms. An example of such constraints would be that each assembly task needs to contain at least one assembly operation and a feeding operation for every part that is supplied for assembly during this task. The assembly operations and actions are specified through stepwise increase of detail guided by inference rules that translate the user requirements into system requirements at the right level of detail. For example the choice of appropriate assembly operation type is based on the types of part liaisons that need to be established during this operation.

The product is defined as a hierarchy of assemblies, components and parts linked together by liaisons (see Figure 2). Assemblies are sets of subassemblies, components, and parts. Parts are defined as entities that do not have any internal degree of freedom. Components are all those entities provided by third parties that have at least one internal degree of freedom like for example electric motors or valves. All three extend the object concept which provides the link to the assembly processes as will be shown later.

The assembly liaisons define how the different parts and components of the product are related to each other. The ontology provides subclasses of the liaison class

specifying the different types of liaisons and the specific attributes (slots).

The assembly process ontology extends the concept of an activity and classifies it to have three generic levels: tasks; operations; and actions. Each activity has a start and end time which define its duration but can also be used to define time relations between activities. Following the definition from Homem de Mello and Sanderson [11] the sequential relation between activities on both task and operation level are defined by mapping input and output states to the activities. States represent objects in a certain condition that has to stay the same for as long as the state is given. The states establish the link from the product specification to the assembly process specification. On the action level this approach becomes ambiguous and their relations are expressed through their relative start and end times.

All activities are defined as individual concepts that have attributes and constraints. They are all subtypes of the activity concept. The attributes and constraints become more specific with increasing level of specialisation. For example an assembly operation can contain a wide variety of different actions whereas an insertion operation has to specifically contain at least one move action as well as hold and release actions.

The assembly process is structured hierarchically as well as sequentially. The hierarchy of the assembly process definition has one task at the top level that represents the assembly of the whole product. This task can contain several levels of subtasks that specify in more detail how the assembly is taking place. On the lowest task level are assembly tasks that define how the individual part liaisons are being established. Each assembly task in turn has one level of operations that define all the necessary steps that need to be taken to establish the liaisons. Finally on the very lowest level are the actions required to facilitate the individual operations. This level is the closest to the actual real process and can later be used to define control algorithms and to select suitable hardware.

The assembly process ontology is defined in a manner that accommodates a step by step increase of detail as information becomes available during the specification process. It also allows for activities to be added at a later point. For example if during the equipment selection it becomes necessary to have additional handling operations, they can be added in at the appropriate place in the model. The same works for checking or additional machining tasks that might be required.

4 Prototype Implementation

A frame based product and assembly process domain ontology has been defined using Protégé 2.0.1 [12]. Protégé is a domain ontology definition tool, which defines classes, slots, relationships, facets, and instances based on the Open Knowledge Base Connectivity (OKBC) [13] protocol. The constraints are expressed as axioms defined in the Protégé Axiom Language (PAL) plug-in [14]. The syntax of PAL is a variant of the Knowledge Interchange Format (KIF) [15]. PAL constraints are defined by a variable range and a logical statement defining the condition that needs to hold for the defined variable range. A variable range is defined as a sentence with the following form:

(defrange [variable] [type] [type specific information]) (1)

A constraint statement is defined as a sequence of sentences linked by logic operators. PAL supports a number of predicates and functions that can be used to define constraints. All the slots defined in the knowledge model can directly be used as either constraints or functions. For example to express that an assembly task has to contain at least one assembly operation is expressed as the following:

```
Constraint1:

(defrange ?aTask :FRAME AssemblyTask)
(defrange ?aOp :FRAME AssemblyOperation)

(forall ?aTask
  (exists ?aOp (subactivity ?aTask ?aOp))
)
```

The constraint reads: *for every assembly task has to exist an assembly operation that is a subactivity of the assembly task.*

The inference rules have been defined using the definitions of the Java Expert System Shell (JESS) [16]. Each rule has the form of if-then statements that match existing facts to new facts to be asserted or actions to be executed. For example the relation between the assembly liaison types and required assembly operation types is expressed as:

```
(defrule assemblyContactLiaisonRule01
  (object(is-a ContactLiaison)(name ?aOp))
  (object(is-a AssemblyOperation)(establishes ?aOp))
  =>
  (assert (object (is-a PickNPlaceOperation)
                  (establishes ?aOp)))
)
```

The rules establish what type of assembly liaisons are being established in an assembly operation and define the more specific type of assembly operation. In the above example a <contact> liaison triggers the specialisation of the assembly operation to <pick and place>. A customised conflict resolution strategy is used to resolve cases in which several liaisons are being established during the same assembly operation based on their complexity. For example a <tight fit> liaison and a <contact> liaison will result in an <press> assembly operation not a <pick and place> operation.

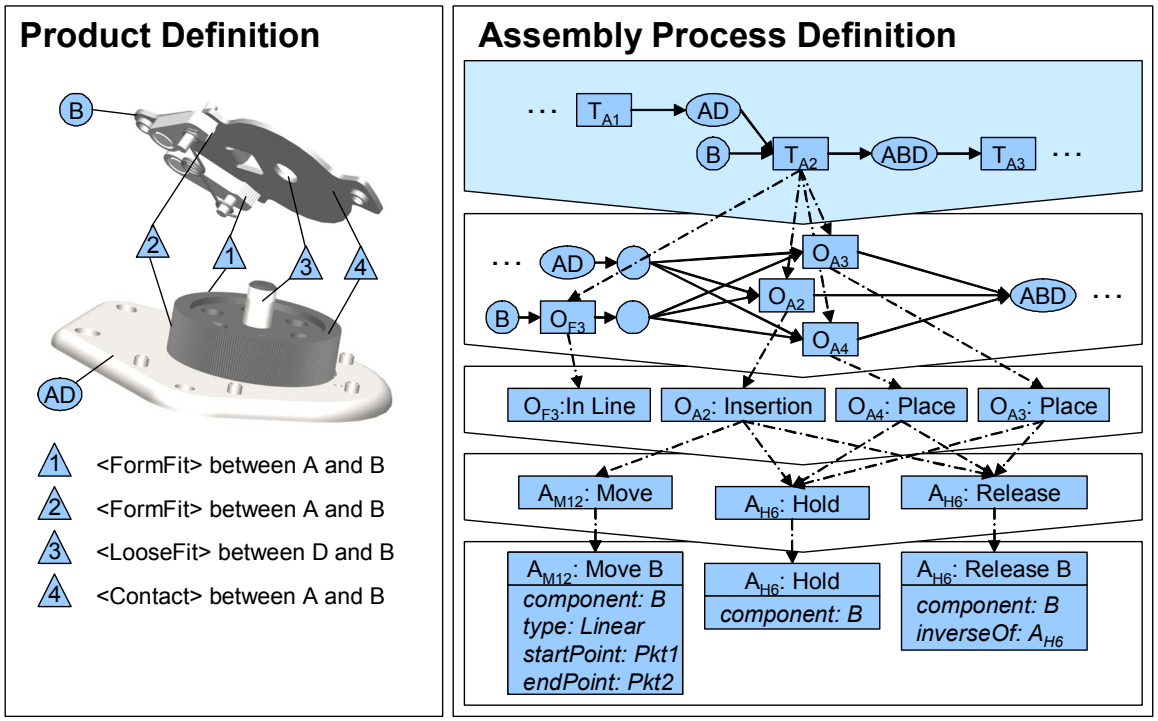

Figure 4: Illustration of a Representative Assembly Task

5 Illustrative Example

The assembly process specification methodology has been tested for initial verification using a set of industrial cases. To illustrate the approach a case of a project developed by a system integrator TQC Ltd. is being considered. The case is based on a seat recliner mechanism (Figure 4) from Lander Automotive Ltd. designed for use in Toyota cars. One of the more complex assembly tasks has been chosen to illustrate the capability of the model.

The connection between component B and assembly AD is defined through 4 part liaisons (see Figure 4). The complexity of the assembly tasks T_{A2} arises from the fact that the two parts of component B that are involved in liaison 1 and 2 are both spring loaded and need to be opened while liaison 3 and 4 are being established.

The assembly task for the two components has to contain three different assembly operations which take place simultaneously. This information is being obtained based on the approach direction of the four liaisons. It also needs to contain a feeding operation for the component that is being introduced during the assembly task.

Liaisons 3 and 4 can be established through the same assembly operation O_{A2}. The assembly operation should be an insertion since both liaisons have the same approach direction and liaison 3 needs a more restricted motion. Both the approach directions of Liaison 1 and 2 establish a form fit and need only a place type assembly operation. However, since the two parts of B involved in O_{A3} and O_{A4} are both spring-loaded, they would need to first be opened before the liaisons can be established. The reasoning required to automatically derive this kind of constraints would need to be based on kinematics as well as a dynamic model of the components. Currently this has not been implemented and the validation of a human expert is required to integrate this kind of constraints. The decision for the required type of feeding operation is based on how the component in question is supplied to the task. In this case B is assembled in a parallel assembly process and is therefore fed in-line.

Once the operation types have been specified the required actions are defined based on the constraints imposed by the operation types. The detailed definition of the action characteristics is derived from the relevant parameters of the product model. For example in the case of O_{A2}, liaisons 3 and 4 define the direction and length of the required move action.

6 Discussion and Conclusions

The proposed assembly process ontology provides clear definitions of all concepts relevant for the specification of an assembly process including a detailed product and process model. A clear link between the product model and the assembly process model allows for a comprehensive propagation of changes in the user requirements based on the product description. This helps to overcome one of the challenges during the reconfiguration of Evolvable Assembly Systems.

The proposed approach also has the advantage that it is suitable for use within distributed design applications

using, for example agent based architectures. The use of assembly specific axioms to define the constraints of the model enables both the sender and receiver of a model to verify that it is correctly specified using standard constraint-checking engines. This reduces the effort of model verification significantly and also avoids time wasted by trying to use a faulty model without checking it first.

One of the disadvantages with more detailed ontologies is that the definition becomes significantly more complex and therefore difficult. However, the use of inference rules combined with customisable user interfaces to guide the user through the specification process helps to deal with the increased complexity.

Initial tests have shown that the specification process is applicable for medium complex products as demonstrated in the given example. Further work will be dedicated to the validation of more complex products and product families.

Acknowledgments

The reported work is partially funded by the Department of Trade and Industry in the United Kingdom as part of the EUREKA Factory E!2851 E-RACE project the support of which is gratefully acknowledged.

References

[1] M. Onori; J. Barata; J. Lastra; M. Tichem, 2003, *"European Precision Assembly Roadmap 2012"*, Assembly-Net Internal Publication, www.Assembly-Net.org

[2] Lohse, Niels, Ratchev, Svetan, Valtchanov, George 2004, *"Towards Web-Enabled Design of Modular Assembly Systems"*, Assembly Automation, Emerald, Vol. 24, No. 3, pp. 270-279, ISSN 0144-5154

[3] H. Hirani, 2004, *"Knowledge Enriched Requirements Specification for Reconfigurable Assembly Systems"*, PhD Thesis, University of Nottingham, Nottingham, UK

[4] S. B. Han; M. S. Sodhi, 2001, *"A Conceptual QFD Planning Model"*, International Journal of Reliability and Quality Management, 18:8, pp796-812

[5] H. K. Rampersad, 1993, *"Integrated and Simultaneous Design for Robotic Assembly"*, John Wiley and Sons, Chichester, West Sussex, UK

[6] P. C. Stadzisz, J. M. Henrioud, 1998, *"An Integrated Approach for the Design of Multi-Product Assembly Systems"*, Computers in Industry, 36:1, pp21-29

[7] J. A. W. M. Vos, 2000, *"Design of A Flexible Industrial Assembly System"*, PhD Thesis, Laboratory for Production Engineering and Industrial Organisation, Delft University of Technology, Delft, The Netherlands

[8] M. Onori, 2002, *"Evolvable Assembly Systems – A New Paradigm?"*, Proceedings of the 31st International Symposium on Robotics, Stockholm, Sweden, October 2002

[9] C. Schlenoff, M. Gruninger, F. Tissot, J. Valois, J. Lubell, J. Lee, , *"The Process Specification Language (PSL) – Overview and Version 1.0 Specification"*, National Institute of Standards and Technology, USA, available at: www.mel.nist.gov/psl/

[10] H. Hirani, S. Ratchev, 2004, *"Knowledge-Based Requirements Engineering for Reconfigurable Precision Assembly Systems"*, Proceedings of the 6th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services

[11] L. S. Homem de Mello, A. C. Sanderson, 1991, *"Representations for assembly sequences"*, Computer-Aided Mechanical Assembly Planning, edited by L. S. Homem de Mello and S. Lee, Kluwer Academic Publishers, Massachusetts, USA, pp. 129-162

[12] Protégé, 2004, Stanford University, Stanford University School of Medicine, Stanford Medical Informatics, California, USA, available at: protege.stanford.edu

[13] V. K. Chaudhri, A. Farquhar, R. Fikes, P. D. Karp, J. P. Rice, 1998, "Open Knowldge Base Connectivity 2.0.3", SRI International, California, USA, available at: www.ai.sri.com/~okbc/

[14] W. Grosso, 2004, *"PAL Constraints and Queries Tabs"*, available at: protege.stanford.edu/plugins/paltabs/PAL_tabs.html

[15] M. R. Genesereth, R. E. Fikes, 1992, *"Knowledge Interchange Format Version 3.0 Reference Manual"*, Computer Science Department, Stanford University, Stanford, California, USA

[16] E. Friedman-Hill, 2004, "Java Expert System Shell", Sandia National Laboratories, Livermore, California, USA, available at: herzberg.ca.sandia.gov/jess/