ISO 18629 PSL : A STANDARDISED LANGUAGE FOR SPECIFYING AND EXCHANGING PROCESS INFORMATION

L.C. Pouchard (+), A.F. Cutting-Decelle (*), J.J. Michel (**), M. Grüninger (++)

(*) University of Evry, IUT-Dept OGP, F,
(**) Idpiconseil, F
(+) Oak Ridge National Laboratory (ORNL), USA,
(++) National Institute of Standards and Technology (NIST), USA

Abstract: As enterprise integration increases, developers face increasingly complex problems related to interoperability. When enterprises collaborate, a common frame of reference or at least a common terminology is necessary for human-to-human, human-to-machine, and machine-to-machine communication. Ontology engineering offers a direction towards solving the inter-operability problems brought about by semantic obstacles related to the definitions of business terms and software classes. Ontology engineering is a set of tasks related to the development of ontologies for a particular domain. This paper is aimed at presenting the approach of ISO 18629, i.e. the Process Specification Language (PSL), to this problem. In the first part, the architecture of the standard is described, with the main features of the language. Then, the problems of the interoperability with PSL and the conformance to the standard are presented. The paper ends with an example showing the use of the standard for interoperability.
Keywords: standards, manufacturing processes, ontologies .

1. INTRODUCTION

As enterprise integration increases, developers face increasingly complex problems related to interoperability (Pouchard, et al. 2000, 2002), (Ray, et al. 2003). Independent contractors and suppliers who collaborate on demand within virtual supply chains to bring to market new products must share product-related data. Legacy vendor applications that are not designed to inter-operate must now share processes. When enterprises collaborate, a common frame of reference or at least a common terminology is necessary for human-to-human, human-to-machine, and machine-to-machine communication. Similarly, within a core enterprise where distributed collaboration between remote sites and production units take place, a common understanding of business and manufacturing-related terms is indispensable. However, this common understanding of terms is often at best implicit in the business transactions and software applications and may not even be always present. Misunderstandings between humans conducting business-related tasks in teams, and ad-hoc translations of software applications contribute to the rising costs of interoperability in manufacturing.

Ontology engineering offers a direction towards solving the inter-operability problems brought about by semantic obstacles, i.e. the obstacles related to the definitions of business terms and software classes. Ontology engineering is a set of tasks related to the development of ontologies for a particular domain. An ontology is a taxonomy of concepts and their definitions supported by a logical theory (such as first-order predicate calculus). Ontologies have been defined as an explicit specification of a conceptualization (Gruber, 1993). Ontology engineering aims at making explicit the knowledge contained within software applications, and within

enterprises and business procedures for a particular domain. An ontology expresses, for a particular domain, the set of terms, entities, objects, classes and the relationships between them, and provides formal definitions and axioms that constrain the interpretation of these terms (Gomez-Perez, 1998). An ontology permits a rich variety of structural and nonstructural relationships, such as generalization, inheritance, aggregation, and instantiation and can supply a precise domain model for software applications (Huhns and Singh 1997). For instance, an ontology can provide the object schema of object-oriented systems and class definitions for conventional software (Fikes, et al., 1999). Ontological definitions, written in a human readable form, can be translated into a variety of logical languages. They can also serve to automatically infer translation engines for software applications. By making explicit the implicit definitions and relations of classes, objects, and entities, ontology engineering contributes to knowledge sharing and re-use (Gomez-Perez 1998).

ISO 18629 is the newest in the family of standards aimed at facilitating interoperability for industrial data integration (of products and processes) in industrial applications in TC 184. Standardized within a joint committee, ISO TC 184 SC4/SC5, PSL provides a generic language for process specifications applicable to a broad range of specific process representations in manufacturing and other applications. PSL is an ontology for discrete processes written in the Knowledge Interchange Format (KIF) (Genesereth and Fikes 1992) itself an ISO candidate in (ISO/JTC1 1999), (Common Logic 2004). Each concept in the PSL ontology is specified with a set of definitions, relations, and axioms all formally expressed in KIF. Relations specify types of links between definitions or elements of definitions; axioms constrain the use of these elements. In addition, the PSL ontology is based on set theory, first order logic, and situation calculus (Etchemendy 1992). Because of this reliance on theories, every element in the PSL language can be proven for consistency and completeness (Gruninger 2003). At the time of this writing, approximately half of the PSL definitions, relations and axioms have been proven to be consistent with the base theories.

The ISO TC184 is one of the one two hundred committees managed by the ISO (International Standardization Organization, Geneva, CH), its scope is : "*Standardization in the field of industrial automation and integration concerning discrete part manufacturing and encompassing the applications of multiple technologies, i.e. information systems, machines and equipments and telecommunications*". This means that the standards developed are applicable to manufacturing and process industries, applicable to all sizes of business, applicable to extending exchanges across the globe through e-business.

PSL is an international standard for providing semantics to the computer-interpretable exchange of information related to manufacturing and other discrete processes. Taken together, all the parts contained in PSL provide a language for describing processes throughout the entire production within the same industrial company or across several industrial sectors or companies, independently from any particular representation model. The nature of this language makes it suitable for sharing process information during all the stages of production. The process representations used by engineering and business software applications are influenced by the specific needs and objectives of the applications. The use of these representation models varies from one application to another, and are often implicit in the implementation of a particular application. One of the manufacturing models on which the PSL ontology is built is provided by the information models of the ISO 15531 MANDATE standard (standardization of manufacturing management information) (Cutting-Decelle et al., 2000-1), particularly for resource management.

A major purpose of PSL is to enable the interoperability of processes between software applications that utilize different process models and process representations. As a result of implementing process interoperability, economies of scale are made in the integration of manufacturing applications.

Table 1: Organization of ISO 18629.

ISO Numbers	Names
ISO IS 18629-1	Overview and Basic Principles
ISO IS 18629-11	PSL-Core
ISO IS 18629-12	Outer Core
ISO CD 18629-13	Duration and ordering Theories
ISO CD18629-14	Resource Theories
ISO WD 18629-15	Actor and agent Theories
ISO 18629-2x	Mappings to EXPRESS, UML, XML
ISO DIS 18629-41	Activity extensions
ISO DIS 18629-42	Temporal and state extensions
ISO CD 18629-43	Activity ordering and duration extensions
ISO CD 18629-44	Resource extensions
ISO WD 18629-45	Process intent extensions

All parts in ISO 18629 are independent of any specific process representation or model used in a given application. Collectively, they provide a structural framework for interoperability. PSL describes what elements should constitute interoperable systems, but not how a specific application implements these elements. The purpose is not to enforce uniformity in process representations. As objectives and design of software applications vary the implementation of interoperability in a application must necessarily be influenced by the particular objectives and processes of each specific application.

PSL currently aims at specifying technical processes for achieving interoperability among various software tool representations used throughout industrial companies. Although mappings to EXPRESS, XML and UML are planned in the future, PSL is not fundamentally based nor fundamentally makes use of Web services. As such, the approach followed, and the use of the language is different from the use of PSLX (PSLX, 2005) and BPEL4WS (BPEL4WS, 2005).

2. ARCHITECTURE AND CONTENT OF ISO18629

PSL (ISO 18629-1, 2004) is organized in a series of parts using a numbering system consistent with that adopted for the other standards developed within ISO TC184/SC4. PSL contains Core theories (Parts 1x), External Mappings (Parts 2x), and definitional extensions (Parts 4x). This discussion focuses on Parts 1x and 4x ; these parts contain the bulk of ISO 18629, including formal theories and the extensions that model concepts found in applications. Parts 1x are the foundation of the ontology, Parts 4x contain the concepts useful for modeling applications and their implementation. Table 1 presents the organization of ISO 18629. Table 2 presents primitive concepts found in PSL Core. Except noted otherwise, PSL version 2.2 is presented.

Table 2: Concepts in PSL Core (ISO IS 18629-11).

PSL Core Primitives	Type	Informal definitions and axioms
activity	relation	Everything is either an activity, an activity occurrence, a timepoint, or an object. Objects, activities, activity occurrences, and timepoints are all distinct kinds of things (disjoint classes).
activity_occurrence	relation	An activity occurrence is associated with a unique activity. But there are activities without occurrences.
timepoint	relation	Given any timepoint t other than inf-, there is a timepoint between inf- and t. Given any timepoint t other than inf+, there is a timepoint between t and inf+.
object	relation	An object participates in an activity at a given timepoint and only at those timepoints when both the object exists and the activity is occurring.
before	relation	The before relation only holds between timepoints. It is a total ordering, irreflexive, and transitive relation.
occurrence_of	relation	Every activity occurrence is the occurrence of some activity and associated with a unique activity.
participates_in	relation	The participates_in relation only holds between objects, activities, and timepoints, respectively.
beginof	function	The beginning of an activity occurrence or of an object are timepoints.
endof	function	The ending of an activity occurrence or of an object are timepoints.
inf+	constant	Every other timepoint is before inf+.
inf-	constant	The timepoint inf- is before all other timepoints.

Core theories (Parts 12 - 15):
Core theories include PSL Core, PSL Outer-Core, Duration and Ordering theories, Resource theories, and Actor and Agent theories. Actor and Agent Theories have not been dealt with at present, thus will not be studied here. The core theories are based on first-order logic. They model basic entities necessary for building the PSL extensions. The PSL Core and Core Theories pose primitive concepts (those with no definition), function symbols, individual constants, and a set of axioms written in the language of PSL. These primitives and all the definitions in PSL are written in KIF for computer interoperability. For the sake of readability, KIF definitions are not reproduced here; the reader is referred to the PSL Ontology Web site (PSL Ontology, 2005) for more details.

Core theories are required to formally prove that extensions are consistent with each other, and with the core theories. The core theories are at the root of the PSL ontology against which every item that claims to be PSL compliant must be tested for consistency.

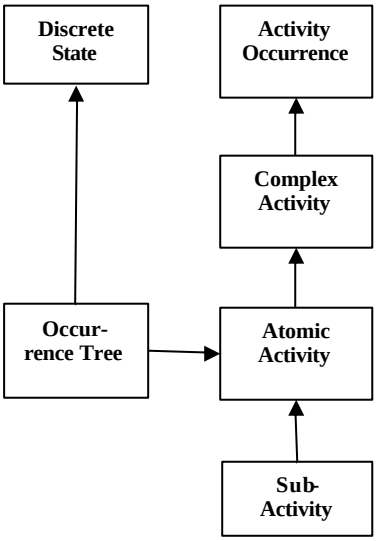

Figure 1a: PSL Outer Core.

Core Theories are a unique feature of PSL as no other standard in SC4 lends itself to formal, logic-based proof. Figure 1a illustrates concepts in the PSL Outer Core. Figure 1b shows Duration and Ordering. Figure 1c shows Resource Theories.

Domain-specific Definitional Extensions (Parts 41 - 45): The extensions to the Core and Outer-core are the constructs used in PSL to represent processes in an application. The Core and Outer Core alone are not sufficient to meaningfully represent the semantics of applications for the purpose of interoperability. They are necessary but have little expressivity by themselves. All terms in the extensions are given definitions using concepts specified and axiomatized

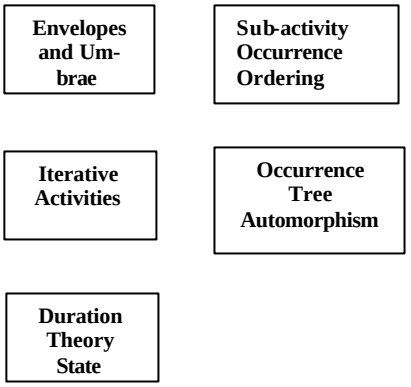

Figure 1b: Duration and Ordering.

in the Core theories. This ensures that definitional extensions are conformant to PSL. A software application will typically use the concepts defined in the extensions. Concepts in the Core and Outer core.

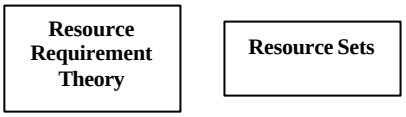

Figure 1c: Resource Theory.

Table 3 presents definitional extensions, and indicates upon which Core theories an extension relies. One must note that the assignment of an individual concept to a specific extension is relatively flexible; it is there for readability and ease of use of the standard. A concept may be moved from one extension to another without affecting the definitions of concepts and the PSL ontology. In other words, extensions do not need to belong to one rather than another of the categories in the left column. However, each concept must conform to the Core Theories in the middle column. Definitional extensions may use concepts defined in other extensions. Figure 2 illustrates how a concept X in a definitional extension is expressed using PSL Core and Core Theories. Concepts such as X help expressing processes used by applications with the semantics of PSL.

As an example, here is Definition 1 (English) for expressing for the concept of Resource Path (PSL version 0.5) about the activity occurrence *occ2*:
"An *activity occurrence occ2 is the next processor subactivity occurrence after occ1 in an activity ?a if and only if the output material of occ1 is the input material of occ2, and there is no other processor subactivity of ? a which consumes the output material from occ1, and which occurs between occ1 and occ2."*

3. INTEROPERABILITY WITH PSL AND CONFORMANCE TO THE STANDARD

Table 3: Examples of PSL concepts defined in extensions.

Names of Definitional Extensions	Core Theories being depended upon	Examples of concepts
Activity Extensions (ISO DIS 18629-41)	Complex Activities	Deterministic and non-deterministic activities Concurrent activities Spectrum of activities
Temporal and State Extensions (ISO DIS 18629-42)	Complex Activities, Discrete States	Preconditions, Effects Conditional activities Triggered activities
Activity Ordering and Duration Extensions (ISO CD 18629-43)	Sub-activity Occurrence Ordering, Iterated Occurrence Ordering, Duration	Complex sequences and branching Iterated activities Duration-based constraints
Resource Extensions (ISO CD 18629-44)	Resource Requirements Resource set theory Sub-activity Occurrence Ordering Resource Requirements	Reusable, consumable, renewable, and deteriorating resources, substitutable resources resource pools, Resource paths Processor activities

language for exchanging processes between software applications such as CAD, and project design software.

As a specification language, PSL can be considered as a specification tool of the information and knowledge related to manufacturing management, as modeled by the MANDATE standard (ISO IS 15531-1, 2002).

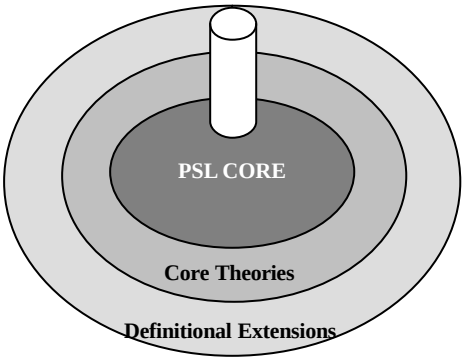

Figure 2: Illustration of the specification of Concept X.(shown as a cylinder).

3.1 The challenges of interoperability

The obstacles to the interoperability of software applications are common, and usually dealt with parsers. Obstacles due to semantic problems, i.e. problems about the "meaning" of a software object or entity, are less visible than those due to syntactic incompatibilities. The lack of formal specifications even if syntax mapping is correct.

The semantic conflict presented in Figure 3 is the example of a "resource" in two software applications A and B: "resource" represents physical resources in: application A and human resources in

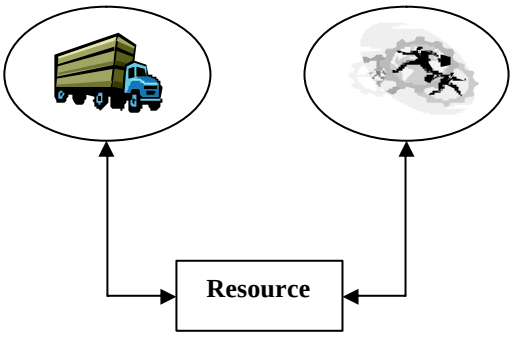

Figure 3: Semantic conflict for resource.

application B. Without a formal specification of processes in a common formal language, poorly designed parsers for semantic mapping can lead to semantic errors that remain undetected. Syntactic interoperability does not resolve these conflicts. Another type of semantic conflict exists when constraints on a concept in applications A and B are different: For instance, an occurrence of an activity in application A is always possible (no constraints), whereas the occurrence of the activity for applica-

tion B is possible only after another activity occurrence. The task of initiating a process that may or may not require user permissions is a good example.

A benefit of specifying processes using PSL is to formally encode each application's concept in a rigorous representation language that is machine readable and not leave semantic reconciliation to subjective, sometimes rushed decisions.

Another benefit in a domain where interoperability between applications has becoming ubiquitous is to reduce the burden of developing parsers for each translation between two applications. The PSL approach reduces the number of translators from O (n^2) to O (n) by requiring that an application maps its concepts to PSL concepts only, rather than mapping to all the other applications.

3.2 Interoperability and conformance

From the point of view of ISO 18629, two applications can interoperate if they are conformant with the same set of ISO 18629 extension. Software applications that claim conformance to PSL will:

Specify application entities into the KIF language. (Entities are the "things" used by applications to refer to processes, relations among these processes, functions, etc...),

provide translation definitions between their processes represented in KIF and PSL definitions,

implement syntactic translators between their applications and PSL process descriptions,

Write a grammar using the using the Backur Naus notation (BNF). The PSL grammar written with BNF can be found on the PSL Ontology Web site (PSL Ontology, 2005).

3.3 User defined extensions

User defined extensions of PSL are extensions that introduce new concepts. Typically, current extensions are sufficiently rich to express processes in existing software applications. However, the case where an application concept is not represented may arise. In this case, PSL can be extended to include a new extension by expressing new concepts using ISO 18629 parts 11-14 (PSL Core and the Core theories). Any new extension must also satisfy the constraints and axioms of ISO 18629. User-defined extensions and new definitions may be needed if an application contains a concept not included in PSL or for new domains. This is what "the PSL ontology is extensible" means.

4. USE OF THE STANDARD FOR INTEROPERABILITY.

In practice, ISO 18629 Parts 4X are what software applications will utilize to specify and exchange their processes using PSL. In order to facilitate specification, the National Institute of Standards and Technology has implemented the 20 Question Wizard (PSL 20 Question Wizard, 2005), a utility that points to the appropriate definitions. A user specifies a process in details by answering questions and checking boxes for their process. The wizard returns the PSL definition for this process written in the KIF syntax.

4.1. Steps to follow in using PSL

The use of PSL for developing a high level translation between the source and target applications (A and B) is now explored. Two applications do not necessarily exchange all their processes for interoperability. Only one or a set of processes may be translated. After identifying the concepts to be exchanged, the translation is performed in three steps and outlined in Figure 4 (Pouchard, 2000) :

a. **Syntactic translation** : the native syntax of an application is parsed to PSL syntax (KIF). This parser keeps the terminology of the application.

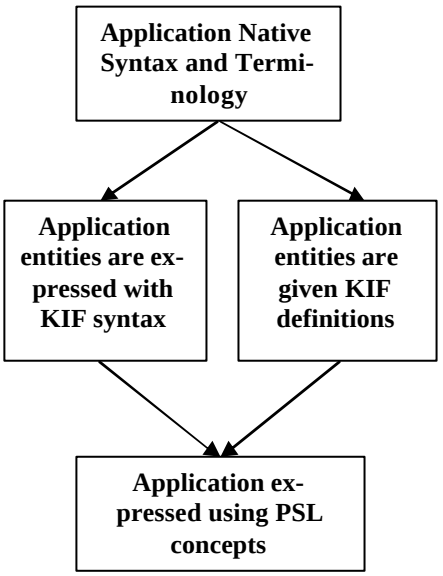

Figure 4: Exchange of processes.

b. **Semantic translation to PSL** : keeping the KIF syntax for the terminology of the application of interest, KIF definitions are written for that application using PSL definitions. These definitions are found within the concepts of the PSL extensions. The question wizard facilitates the attribution of definitions to the terminology and concepts of an application to PSL definitions. Translation definitions between an application and PSL ontology can be derived from the onto-

logical definitions and axioms provided in the different parts of ISO 18629.

c. **Semantic translation from one application to another** : At this point, the processes of the source and target applications have been expressed using PSL terms and KIF syntax. Each should have a one-to-one correspondence between each process definition and a PSL definition. The concepts of the source application are mapped to concepts of the target application using PSL as the intermediate language. On this basis, data for the relevant process can be exchanged.

Following this procedure is not sufficient to claim conformance to PSL according to ISO 18629, but it is sufficient for process exchange with another application. (In order to claim full conformance to the standard, the grammar of the processes used in the application must be written using the BNF notation as seen in section 3.2.)

4.2. Translation definitions

Step a: A resource for A, {resourceA: inject_mold (r)}, written in KIF, is shown in Figure 5:

```
(forall (?r)
      (=> (inject_mold)
          (resourceA ?r))))
```

Figure 5: Resource concept for A in KIF.

Step b: The entities and structural relationships (the ontology of A) is then specified (Figure 6). Technical documentation supplemented by interviews are often used. The ontology of A specifies what kind

```
forall (?r ?a
(<=> (resourceA ?r)
      (exists (?a)
      (reusable ?r ?a)))
```

Figure 6: Ontology of A

of resource is implied by *resourceA*. Here *resourceA* (a machine-tool) is a resource that can be used by a process after another process that also requires resourceA completes its occurrence..In Figure 6, ?r is a resource variable.. Once a definition of resource has been provided for A using A terminology, the next step is to look for possible mappings between *resourceA* and PSL concepts. Three cases occur: 1) a one-to-one mapping exists, 2)a one-to-one mapping is possible under certain conditions, 3)PSL does not contain the concept in question and needs to be extended to accommodate A. Examples for Case 1 and 2 are given here. This mapping is to be done for every concept and relation contained in A. The question wizard is of help for this task.

We must note here that the concept of "resource", as dealt with by the PSL language is taken with a meaning different from its usual interpretation in production management. However, a "resource" for PSL is close to the analogous concept defined within the ISO/IEC 62264 standard (ISO/IEC 62264-1, 2003).

Case 1) PSL contains a concept of resource defined

```
(defrelation resource (?r) :=
      (and  (object ?r)
            (exists (?a)
            (requires ?a ?r))))
```

Figure 7: PSL definition for a resource.

as -- a resource is any object that is required by some activity -- where "activity" ?a and "requires" are defined elsewhere in PSL (Figure 7).

PSL also specifies complex resources and defines

```
(defrelation reusable (?r ?a1) :=

(forall (?a2 ?occ)
   (=>(and (common ?a1 ?a2 ?r)
           (occurs_over ?a1 ?
           occ))

(forall (?b)
   (=> (forall (?s3)
     (=> (and
      (legal_interval ?b)
      (situation_during ?s3 ?b)
      (occurs_during ?occ
(legal ?s3))
           )
       (legal_activity ?a2?s3)
                )
                )))))
```

Figure 8. PSL definition for a reusable resource

the concept of a reusable resource -- a resource ?r is reusable by an activity ?a if any other activity that also requires ?r is still possible to perform after ?a completes its occurrence, in every possible future (Figure 8). The PSL concepts, 'common,' 'occurs_over,' 'legal_interval,' 'legal,' and 'legal_activity' in Figure 8 are defined elsewhere in PSL. In our example, a mapping appears to exist, but more information about A is needed before deciding for a one-to-one mapping.

This information must be obtained from the documentation for Application A, and other sources (vendors). PSL specifies that a reusable resource is such that, as soon as one activity occurs, it is always possible to perform the next activity. An example of reusable resource according to PSL is a machine that

does not require setup between activities. If the resource in application A satisfies this condition, we have a one-to-one mapping between the Application A concept, *resourceA*, and PSL definitions of *resource* and *reusable* (Figure 9). :

```
(forall (?r)

(<=> (resourceA ?r)
      (and (resource ?r)
            (reusable ?r)
```

Figure 9: Direct mapping.

Similar steps are followed in the translation of Application B's concepts into PSL. An index provides an inverse table where the relevant PSL concepts are mapped to the concepts of B. A and B are now interoperable. A model for a manufacturing task using A represented with PSL may be imported into Application B.

Case 2) A conditional mapping between a PSL and an application concept is required. The prototype implementation of ILOG ™, a scheduler, contained the concept *ilcActivity*. *ilcActivity* is narrower than a PSL *activity*. How much narrower must be defined for *ilcActivity* using PSL constraints. *IlcActivity* maps to a PSL non-deterministic activity applied to a set of resources. A PSL activity is non-deterministic if and only if -- it is a nondeterministic selection activity with respect to some resource set. The conditional mapping of A to PSL is shown (Figure 10).

```
(forall (?a)
  (=> (and
  (nondet_res_activity ?a)
  (primitive ?a))

      (<=> (ilcActivity ?a)
            (activity ?a))))
```

Figure 10: Conditional mapping for ilcActivity.

Other research work has been done or is currently on-going, showing examples of interoperability among software tools using PSL, notably at the University of Stanford (CIFE) (Law 2001,) (Cheng et al., 2003), and at the University of Loughborough (Cutting-Decelle et al., 2000), (Cutting-Decelle et al., 2002), (Cutting-Decelle et al., 2004), (Tesfagaber et al., 2002).

5. SUMMARY

This paper was aimed at showing to what extent standards based approaches can be helpful to facilitate information sharing and interoperability among software applications commonly used in manufacturing, and in manufacturing management. Most of the time, technical terms handled by those applications look similar, or, even worse, are exactly the same – however their meaning is different. This is particularly true of applications for which PSL was designed, "built" more or less on the same "manufacturing-flavoured" vocabulary, but with very different and multiple interpretations of the same terms. Given its properties, and its structure, the ISO 18629 standard can be considered as a powerful interoperability "tool" for the information systems of the enterprises.

ACKNOWLEDGEMENTS

The contribution of Line Pouchard, PhD., to this submitted manuscript is sponsored by the US Department of Energy, Office of Science of the Oak Ridge National Laboratory, managed for the U. S. DOE by UT-Battelle, LLC, under contract No. DE-AC05-00OR22725. The work of Michael Gruninger was supported in part by the Manufacturing Engineering Laboratory at NIST, Grant Number 60NANB1D0058. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

REFERENCES

BPEL4WS : Available from http://www-128.ibm.com/developerworks/library/specification/ws-bpel/ .

Cheng J., Gruninger M., Sriram R.D., Law, K.H. (2003). Process specification language for project scheduling information exchange. *International Journal of IT in AEC*, **1:4**.

Common Logic Web site. Available from http://philebus.tamu.edu/cl/.

Cutting-Decelle A.F., C.J. Anumba, A. N. Baldwin, N.M. Bouchlaghem, G. Tesfagaber (2003). Exchanges of process information between software tools in construction : the PSL language, International Conference *ECPPM02*, Portoroz, Slovenia, September.

Cutting-Decelle A.F., R.I.M Young, B.P. Das, C.J. Anumba, A.N. Baldwin, N. M, Bouchlaghem. (2004). A multi-disciplinary representation of the supply chain information in construction : an innovative approach to project management. *TMCE Conference*.

Cutting-Decelle A.F., J.J Michel, C. Schlenoff. (2000). Manufacturing and construction common process representation : the PSL approach. *CE2000*, Lyon, France, October.

Cutting-Decelle A.F., J.J Michel, C. Schlenoff. (2000). Integration of industrial management information and interoperability : MANDATE +

PSL a combined approach. *MICAD2000*.

Etchemendy J. (1992) The language of first-order logic. *CSLI Lecture Notes*, **34.**

Fikes, R. and A. Farquahr. (1999). Distributed Repositories of Highly Expressive Reusable Ontologies. *IEEE Intelligent Systems and their Applications*, **14:2.**

Genesereth, M. and R.E. Fikes. (1992). Knowledge Interchange Format, Version 3.0. Reference Manual. Technical Report Logic-92-1. Computer Science Department, Stanford University, Stanford, CA.

Gomez-Perez, A. (1998). In: *The Handbook of Applied Expert Systems* (J. Liebowitz, Ed.), Knowledge Sharing and Re-Use. Chapter 10, pp.. 1-36. Boca Raton, Florida.

Gruber, T. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisitions* **5**, 199-220.

Gruninger, M. (2003) In: *Hand-book on Ontologies in Information Systems (* R. Studer and S. Staab Eds.), A guide to the ontology of the Process Specification Language. Springer-Verlag, Frankfurt.

Hunhs, M. N. and M. P. Singh. (1997). Ontologies for Agents. *IEEE-Internet Computing* **1: 6.**

ISO IS 10303-1. (1994) Industrial automation systems and integration – product data representation and exchange – part 1: overview and fundamental principles.

ISO IS 15531-1. (2002). Industrial automation systems and integration – Industrial manufacturing management data – part 1 : general overview.

ISO IS 18629-1. (2004). Industrial automation systems and integration – Process specification language – Part 1: Overview and basic principles.

ISO IS 18629-11. (2004). Industrial automation systems and integration – Process specification language – Part 11: PSL-Core.

ISO IS 18629-12 (2004). Industrial automation systems and integration – Process specification language – Part 12: Outer Core.

ISO CD 18629-13 (2004). Industrial automation systems and integration – Process specification language – Part 13: Duration and ordering theories.

ISO CD 18629-14 (2004). Industrial automation systems and integration – Process specification language – Part 14: Resource theories.

ISO DIS 18629-41 (2004). Industrial automation systems and integration – Process specification language – Definitional extensions: Part 41 : Activity extensions.

ISO DIS 18629-42 (2004). Industrial automation systems and integration – Process specification language – Definitional extensions: Part 42 : Temporal and state extensions.

ISO CD 18629-43 (2004) Industrial automation systems and integration – Process specification language – Definitional extensions: Part 43 : Activity ordering and duration extensions.

ISO CD 18629-44 (2004). Industrial automation systems and integration – Process specification language – Definitional extensions: Part 44 : Resource extensions.

ISO/IEC FDIS 62264-1 (2003). Enterprise-control system integration – Part 1 : Models and terminology.

ISO TC 184 / SC4 Web site. (2005). Available from http://www.tc184-sc4.org.

ISO Web site. (2005). Available from http://www.iso.ch.

Knowledge Interchange Format (1999). Part 1 : KIF-Core, ISO/JTC1/SC32/WG2, WD.

Law, K.H. (2001). Process specification and simulation. PSL quarterly progress report (CIFE). Stanford, California..

Pouchard, L., N. Ivezic, C. Schlenoff (2000). Ontology engineering for distributed collaboration in manufacturing. AIS, Tucson, Arizona.

Pouchard, L., O. Rana. (2002). The Role of Ontologies in Agent-oriented Systems. *Sixth Joint Conference on Information Sciences, Computational Semiotics Workshop*. Research Triangle Park, North Carolina.

PSL Twenty Questions Wizard. (2005). Available from http://www.mel.nist.gov/psl/20questions.html.

PSL Ontology Web site. Available from http://www.mel.nist.gov/psl/ontology, 2005.

PSL Web site. (2005). Available from http://www.mwel.nist.gov.

PSLX Consortium (2005). Available from : http://www.pslx.org/en/.

Ray S. R. and A. T. Jones. (2003). Manufacturing interoperability, Concurrent Engineering: Enhanced Interoperable System. *Proceedings of the tenth ISPE International Conference*, pp.535-540.

Tesfagaber G., A. F. Cutting-Decelle, C.J. Anumba, A. N. Baldwin, N.M. Bouchlaghem. (2002). Semantic process modelling for applications integration in AEC. International workshop on information technology in civil Engineering, ASCE.