INTERNATIONAL STANDARD

ISO 18629-1

First edition
2004-11-15

Industrial automation systems and integration — Process specification language —

Part 1: **Overview and basic principles**

Systèmes d'automatisation industrielle et intégration — Langage de spécification de procédé —

Partie 1: Vue d'ensemble et principes de base

Reference number
ISO 18629-1:2004(E)

Contents

Figures :

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 18629-1 was prepared by ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC 4, *Industrial data*.

A complete list of parts of ISO 18629 is available from the Internet.

http://www.tc184-sc4.org/titles

Introduction

As the use of information technology in manufacturing has matured, the necessity for software applications to inter-operate has become crucial to the conduct of business and operations in organisations. To be competitive and maintain good economic performance, manufacturing organisations need to employ increasingly effective and efficient systems. Such systems should result in the seamless integration of manufacturing applications and exchange of manufacturing processes between applications. Organisations should also be able to preserve and retrieve on demand the knowledge contained in their business and operational processes, regardless of the applications used to produce and handle these processes.

Many manufacturing engineering and business software applications use process information, including manufacturing simulation, production scheduling, manufacturing process planning, workflow, business process reengineering, product realisation process modelling, and project management. However, each of these applications utilises process information in a different way, and each representation of process information inherent to these applications is also different. Thus interoperability is difficult to achieve. Consequently, these concerns have led to the development of a process specification language (PSL) that complements the process representations utilised in manufacturing engineering and business software applications. ISO 18629 provides a generic language for process specifications applicable to a broad range of specific process representations in manufacturing applications.

ISO 18629 provides semantics to the computer-interpretable exchange of information related to manufacturing processes. Taken together, all the parts contained in ISO 18629 a language for describing a manufacturing process throughout the entire production process 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 related to manufacturing during all the stages of a production process.

The process representations used by engineering and business software applications are influenced by the specific needs and objectives of the applications. Therefore, the use of the process specification language also varies from one application to another. A major purpose of the Process Specification Language is to enable the interoperability of manufacturing processes between software applications that utilise different process models and process representations. As a result of implementing process interoperability, economies of scale are made in the integration of manufacturing applications.

This part and all other parts of ISO 18629 are independent of any specific process representation or model used in a given application. Collectively, they provide a structural framework for interoperability.

ISO 18629 describes what elements inter-operable systems should encompass, but not how a specific application implements these elements. It is not the purpose of ISO 18629 to enforce uniformity in manufacturing process representations. Objectives and design of software applications vary. Therefore the implementation of an interoperable application must necessarily be influenced by the particular objectives and processes of each specific application. This part of ISO 18629 provides an overview of the principal concepts contained in ISO 18629, and guidance on selection and use of its parts.

Industrial automation systems and integration — Process specification language — Part 1: Overview and basic principles

1. Scope

The scope of this part of ISO 18629 is the provision of an overview of the whole ISO 18629 and of the main underlying principles of the Process Specification Language. This part of ISO 18629 also specifies the characteristics of the various series of parts in ISO 18629 and the relationships among them.

The following are within the scope of this part of ISO 18629:

— general overview of ISO 18629 and of the main principles used;

— structure of ISO 18629 and relationships between the series of parts of which this standard is composed;

— definitions of terms used throughout ISO 18629;

— conformance criteria for process-related applications;

— conformance criteria for other ontologies;

— conformance criteria for parts of ISO 18629.

The scope of this part of ISO 18629 includes providing explanations addressing the following items:

— Annex B: Background to the development of ISO 18629;

— Annex C: Need for semantics;

— Annex D: Interoperability;

— Annex E: Architecture of PSL.

2. Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

- ISO/IEC 8824-1: *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- ISO 10303-1: *Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles.*
- ISO 10303-11: *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual.*
- ISO 15531-1: *Industrial automation systems and integration — Industrial manufacturing management data — Part 1: General overview.*
- ISO 15531-31: *Industrial automation systems and integration — Industrial manufacturing management data — Part 31: Resource information model.*
- ISO 15531-32: —[1] *Industrial automation systems and integration — Industrial manufacturing management data — Part 32: Conceptual information model for resources usage management data.*
- ISO 15531-42:—[2] *Industrial automation systems and integration — Industrial manufacturing management data — Part 42: Manufacturing flow management data — Time model.*

3. Terms, definitions and abbreviations

3.1. Terms and definitions

For the purposes of this document, the following terms and definitions apply:

3.1.1
axiom
well-formed formula in a formal language that provides constraints on the interpretation of symbols in the lexicon of a language

3.1.2
conservative definition
definition that specifies necessary and sufficient conditions that a term shall satisfy and that does not allow new inferences to be drawn from the theory

3.1.3
core theory
set of axioms for relation and function symbols that denote primitive concepts

[1] To be published
[2] To be published

3.1.4
data
a representation of information in a formal manner suitable for communication, interpretation, or processing by human beings or computers

[ISO 10303-1]

3.1.5
defined lexicon
set of symbols in the non-logical lexicon which denote defined concepts

NOTE Defined lexicon is divided into constant, function and relation symbols.

EXAMPLE terms with conservative definitions.

3.1.6
definitional extension
extension of PSL-Core that introduces new linguistic items which can be completely defined in terms of the PSL-Core

NOTE: Definitional extensions add no new expressive power to PSL-Core but are used to specify the semantics and terminology in the domain application.

3.1.7
discrete manufacturing
production of discrete items

EXAMPLE Cars, appliances or computer.

[ISO 15531-1]

3.1.8
duration
length of a period of time, measured using a given unit of time

EXAMPLE 1 the 24 hours between Monday 1.00 p.m. and Tuesday 12.00 a.m.

EXAMPLE 2 every Monday of every week between January and July.

NOTE interval of time measures the distance between two points in time. In that case it is the length of the time domain that is bounded by the two points in time under consideration.

[ISO 15531-42]

3.1.9
extension
augmentation of PSL-Core containing additional axioms

NOTE 1 The PSL-Core is a relatively simple set of axioms that is adequate for expressing a wide range of basic processes. However, more complex processes require expressive resources that exceed those of the PSL-Core. Rather than clutter the PSL-Core itself with every conceivable concept that might prove useful in describing one process or another, a variety of separate, modular extensions need to be developed and added to the PSL-Core as necessary. In this way a user can tailor the language precisely to suit his or her expressive needs.

NOTE 2 All extensions are core theories or definitional extensions.

3.1.10
grammar
specification of how logical symbols and lexical terms can be combined to make well-formed formulae

3.1.11
information
facts, concepts, or instructions

3.1.12
language
combination of a lexicon and a grammar

3.1.13
lexicon
set of symbols and terms

NOTE The lexicon consists of logical symbols (such as Boolean connectives and quantifiers) and non-logical symbols. For ISO 18629, the non logical part of the lexicon consists of expressions (constants, function symbols, and relation symbols) chosen to represent the basic concepts of the ontology.

3.1.14
manufacturing
function or act of converting or transforming material from raw material or semi-finished state to a state of further completion

[ISO 15531-1]

3.1.15
manufacturing process
structured set of activities or operations performed upon material to convert it from the raw material or a semifinished state to a state of further completion

NOTE Manufacturing processes may be arranged in process layout, product layout, cellular layout or fixed position layout. Manufacturing processes may be planned to support make-to-stock, make-to-order, assemble-to-order, etc., based on strategic use and placements of inventories.

[ISO 15531-1]

3.1.16
model
combination of a set of elements and a truth assignment that satisfies all well-formed formulae in a theory

NOTE 1 The word "model" is used, in logic, in a way that differs from the way it is used in most scientific and everyday contexts [7] : if a sentence is true in a certain interpretation, it is possible to say that the interpretation is a model of the sentence. The kind of semantics presented here is often called model-theoretical semantics.

NOTE 2 A model is typically represented as a set with some additional structure (partial ordering, lattice, or vector space). The model then defines meanings for the terminology and a notion of truth for sentences of the language in terms of this model. Given a model, the underlying set of axioms of the mathematical structures used in the set of axioms then becomes available as a basis for reasoning about the concepts intended by the terms of the language and their logical relationships, so that the set of models constitutes the formal semantics of the ontology.

3.1.17
ontology
a lexicon of specialised terminology along with some specification of the meaning of terms in the lexicon

NOTE 1: structured set of related terms given with a specification of the meaning of the terms in a formal language. The specification of meaning explains why and how the terms are related and conditions how the set is partitioned and structured.

NOTE 2: The primary component of a process specification language such as ISO 18629 is an ontology The primitive concepts is the ontology according to ISO 18629 are adequate for describing basic manufacturing, engineering, and business processes.

NOTE 3: The focus of an ontology is not only on terms, but also on their meaning. An arbitrary set of terms is included in the ontology, but these terms can only be shared if there is an agreement about their meaning. It is the intended semantics of the terms that is being shared, not simply the terms.

NOTE 4: Any term used without an explicit definition is a possible source of ambiguity and confusion. The challenge for an ontology is that a framework is needed for making explicit the meaning of the terms within it. For the ISO 18629 ontology, it is necessary to provide a rigorous mathematical characterisation of process information as well as a precise expression of the basic logical properties of that information in the ISO 18629 language.

3.1.18
Outer Core
set of core theories that are extensions of PSL-Core and that are so generic and pervasive in their applicability that they have been put apart

NOTE In practice, extensions incorporate the axioms of the Outer Core.

3.1.19
primitive concept
lexical term that has no conservative definition

3.1.20
primitive lexicon
set of symbols in the non-logical lexicon which denote primitive concepts

NOTE Primitive lexicon is divided into constant, function and relation symbols.

3.1.21
process
structured set of activities involving various enterprise entities, that is designed and organised for a given purpose

NOTE The definition provided here is very close to that given in ISO 10303-49. Nevertheless ISO 15531 needs the notion of structured set of activities, without any predefined reference to the time or steps. In addition, from the point of view of flow management, some empty processes may be needed for a synchronisation purpose although they are not actually doing anything (ghost task).

[ISO 15531-1]

3.1.22
process planning
analysis and design of the sequences of processes, resources requirements, needed to produce goods and services

NOTE This definition applies to discrete part manufacturing and continuous processes.

[ISO 15531-1]

3.1.23
product
a thing or substance produced by a natural or artificial process

[ISO 10303-1]

3.1.24
product information
facts, concepts, or instructions about a product

[ISO 10303-1]

3.1.25
proof theory
set of theories and lexical elements necessary for the interpretation of the semantics of the language

NOTE It consists of three components: the PSL-Core, the Outer Core and the extensions.

3.1.26
PSL-Core
set of axioms for the concepts of activity, activity-occurrence, time-point, and object

NOTE The motivation for PSL-Core is any two process-related applications shall share these axioms in order to exchange process information, and hence is adequate for describing the fundamental concepts of manufacturing processes. Consequently, this characterisation of basic processes makes few assumptions about their nature beyond what is needed for describing those processes, and the PSL-Core is therefore rather weak in terms of logical expressiveness. In particular, PSL-Core is not strong enough to provide definitions of the many auxiliary notions that become necessary to describe all intuitions about manufacturing processes.

3.1.27
resource
any device, tool and means, excepted raw material and final product components, at the disposal of the enterprise to produce goods or services

NOTE 1 Resources as they are defined here include human resources considered as specific means with a given capability and a given capacity. Those means are considered as being able to be involved in the manufacturing process through assigned tasks. That does not include any modelling of an individual or common behaviour of human resource excepted in their capability to perform a given task in the manufacturing process (e.g.: transformation of raw material or component, provision of logistic services). That means that human resources are only considered, as the other, from the point of view of their functions, their capabilities and their status (e.g.: idle, busy). That excludes any modelling or representation of any aspect of individual or common «social» behaviour.

NOTE 2 This definition includes ISO 10303-49 definition but is included in the definition that applies for ISO 18629-14 and ISO 18629-44 that includes raw materials and consumables.

[ISO 15531-1]

3.1.28
satisfiable
a set of sentences is satisfiable if there exists a model for the sentences

3.1.29
scheduling
act, function or result of planning occurrences of manufacturing activities

[ISO 15531-1]

3.1.30
structure
combination of a set of elements, a set of functions, and sets of tuples for each relation

3.1.31
theory
set of axioms and definitions that pertain to a given concept or set of concepts

NOTE this definition reflects the approach of artificial intelligence in which a theory is the set of assumptions on which the meaning of the related concept is based.

3.1.32
translation definition
a KIF sentence of the form (iff P Q) in which P is a term in the application's non-logical lexicon and Q uses only terminology from extensions in the ISO 18629 standard

NOTE iff is a KIF reserved word.

3.2. Abbreviations

BNF	Backus-Naur Formalism
CEN	Comité Européen de Normalisation (European Committee for Standardisation)
EDI	Electronic Data Interchange
ENV	European Pre-standard
IDEF3	ICAM DEFinition language 3 Process description capture method
KIF	Knowledge Interchange Format
JTC 1	Joint Technical Committee between ISO and IEC
MANDATE	MANufacturing management DATa Exchange
MMS	Manufacturing Message Services
MRP	Material Requirement Planning
MRP II	Manufacturing Resources Planning
P-LIB	Parts Library
PSL	Process Specification Language
STEP	STandard for the Exchange of Product model data
UML	Unified Modelling Language

NOTE For further information, see [2]

XML	EXtensible Mark-up Language

4. Overview of ISO 18629

4.1. ISO 18629 general

ISO 18629 specifies a language and ontology for the specification of processes, that is focused on, but not limited to the realm of discrete processes related to manufacturing, including all processes in the design and manufacturing life cycle. Business processes and manufacturing engineering processes are included in this work both to as certain common aspects for process specification and to acknowledge the current and future integration of business and engineering functions.

ISO 18629 addresses information dealing with manufacturing processes (see ISO 15531-1), and makes use of product and component descriptions (See ISO 10303-1 and ISO 13584-1). Process information as specified in ISO 18629 are expected to be also consistent with process management data

(ISO 15531). This requires that all the parts of ISO 18629 to be consistent with the related parts of ISO 15531.

The purpose of this standard is to create a process specification language, not a process characterisation language. The process specification language is composed of a lexicon, an ontology, and a grammar for process descriptions, as further described. A process specification language provides names, definitions and axioms for processes independently of the behaviours and capabilities of the processes. A process characterization language describes the set of possible behaviours and capabilities for processes.

NOTE 1 A process specification language is a language needed to specify a process or a flow of processes, including supporting parameters and settings. This may be done for prescriptive or descriptive purposes. This is different from a process characterisation language (also called process modelling language), which can be defined as a language describing the behaviours and capabilities of a process

NOTE 2 PSL is a language for specifying manufacturing processes based on a mathematically well defined vocabulary and grammar. As such, it is different from the schemas and product representations provided in the standards ISO 10303, ISO 13584 and ISO 15926, it is also different from the representation provided by ISO 15531, but strongly related to it and complementary . In the context of an exchange of information between two processes, PSL specifies each process independently of its behaviour. For example, an object viewed as a resource within one process can be recognised as the same object even though it is viewed as a product within a second process.

NOTE 3 PSL is based on first order logic; as such, it follows significantly different methods for specifying semantics than those used in ISO 10303. The meaning of the concepts defined within PSL follows from sets of axioms and definitions provided within each extension of PSL-Core. A set of supporting notes and examples are provided within each part to aid the understanding of the language.

The following are within the scope of ISO 18629:

— the specification of process information related to discrete manufacturing processes;

EXAMPLE : activity decomposition, process flow, duration

— the exchange and sharing of process information within one industrial sector or through several industrial sectors.

The following are outside the scope of ISO 18629:

— language, architecture and methodologies for the specification of an enterprise as a whole;

— specification and exchange of product information;

— specification and exchange of computer-interpretable parts library information;

— technical maintenance information.

EXAMPLE technical information included in devices repair, operation and maintenance manuals.

4.2. The Process Specification Language (PSL) fundamental principles

The primary component of PSL is its terminology for classes of processes and relations for processes and resources, along with definitions of these classes and relations. Such a lexicon of terminology along with some specification of the meaning of terms in the lexicon constitutes what is known as an ontology. Within the ISO 18629 standard, the ontology is the PSL ontology for processes.

NOTE 1 for further information, see [6]

NOTE 2 See note 3 of clause 3.3.13. The challenge is that some framework is needed for making explicit in definitions the meaning of the terminology for ontologies . All intuitions that are implicit are a possible source of ambiguity and confusion.

ISO 18629 provides a rigorous mathematical characterisation of process information as well as a precise expression of the basic logical properties of that information in the PSL language. In providing the ontology, three notions are specified :

— language;

— specification of models;

— proof theory (axioms and definitions)

A language is a lexicon (a set of symbols) and a grammar (a specification of how these symbols can be combined to make well-formed formulae). The lexicon consists of logical symbols (such as connectives, variables, and quantifiers) and non logical symbols. For PSL, the non logical part of the lexicon consists of expressions (constants, function symbols, and predicates) that refer to everything needed to describe processes.

The underlying language used for PSL is KIF (Knowledge Interchange Format).

NOTE 3 KIF is a formal language based on first-order logic developed for the exchange of knowledge among different computer programs with disparate representations. KIF (see Annex A) provides the level of rigor necessary to define concepts in the ontology unambiguously, a necessary characteristic to exchange manufacturing process information using the PSL Ontology.

The specification of models of PSL provides a rigorous mathematical characterisation of the semantics of the terminology of PSL.

NOTE 4 A model-theoretic semantics has two parts :

— first, there is a representation of the events, properties and relations that make up the situation being modelled. The model provides a description of the denotations of all the basic expressions in the object language. This reflects the fact that human beings make statements in order to convey information about some state-of-affairs. Although it is possible to know the meaning of a sentence without knowing what specific situation is being talked about (by knowing its truth-conditions), the sentence conveys no information unless it is associated with particular individuals and the relations that hold between them ;

— the second part of the model provides the rules for interpreting expressions in the object language with respect to any arbitrary domain. In other words, the interpretation provides a specification of the truth-conditions of the sentences in the object language. The truth-conditions as specified by the interpretation hold independently of particular models, but the interpretation of particular sentences may be carried out only with respect to some model or other.

NOTE 5 In a correspondence theory of truth, a statement in some language (whether a natural language learnt and spoken by human beings or an artificial language devised for specific purposes) is true if and only if it corresponds to some state-of-affairs.

The proof theory of PSL provides a set of axioms (sentences in first-order logic) for the interpretation of concepts in the ontology. There are two types of sentences in this set: core theories and definitions. A core theory is a set of predicates, function symbols, and individual constants representing the primitive concepts of the ontology, together with some set of axioms. The intended interpretations of primitive concepts are specified using the axioms in the core theories.

NOTE 6 This approach to primitive concepts follows the methodology of mathematical logic.

All other terms in the ontology outside the lexicon of the core theories are given definitions using the set of primitive concepts with axioms in the core theories. The set of models for defined concepts are specified using the models of the core theories. The classes of models that have already been specified for the core theories characterise the semantics of the definitions.

4.3. Requirements for PSL extensions

A goal of ISO 18629 is to facilitate application interoperability by means of the development of translators between the native formats of those applications and ISO 18629. To support this goal, each Part of the standard will be composed of one or more extensions to PSL-Core. For each extension, the standard will include the following:

— non-logical lexicon

— specification of models

— set of axioms

— theorems for the verification of extension

— grammar for process descriptions that use the terminology of the non-logical lexicon.

4.3.1. Non-logical Lexicon

The non-logical lexicon is the terminology of the standard that corresponds to the concepts and relationships related to manufacturing processes. All terms in the non-logical lexicon of the standard are either constant, function, or relation symbols in KIF.

Each extension specifies a unique non-logical lexicon. Any term in the standard shall belong to the non-logical lexicon of a unique extension.

4.3.2. Specification of Models

The specification of model for ISO 18629 provides a rigorous abstract mathematical characterisation of the semantics of the terminology of ISO 18629. This characterisation defines the meanings of terms with respect to some mathematical structures together with a notion of truth with respect to those structures for sentences of the language.

4.3.3. Axioms of the extensions

The axioms of ISO 18629 is the set of KIF sentences that constrain the interpretation of the terminology in the non-logical lexicon of ISO 18629.

The axioms of ISO 18629 are organised into PSL-Core and a partially ordered set of extensions to PSL-Core. An ISO 18629 extension provides the logical expressiveness to express information involving concepts that are not explicitly specified in PSL-Core.

All extensions within ISO 18629 shall be consistent extensions of PSL-Core, and may be consistent extensions of other ISO 18629 extensions. However, not all extensions within ISO 18629 need be mutually consistent.

EXAMPLE There may be an extension turning into axioms a discrete timeline and another extension turning into axioms a dense timeline; although these are mutually inconsistent, they are both individually consistent with PSL-Core.

4.3.4. Grammar for process descriptions

The underlying grammar used for ISO 18629 is that of KIF (Knowledge Interchange Format).

NOTE KIF is a formal language based on first-order logic developed for the exchange of knowledge among different computer programs with disparate representations. KIF provides the level of rigour necessary to unambiguously define concepts in the ontology.

Process descriptions shall be sentences in KIF that use the non-logical lexicon of ISO 18629. In particular, process descriptions shall be restricted to sentences that are satisfied by elements in a model of the axioms of ISO 18629. Process descriptions are not arbitrary sentences.

EXAMPLE the process descriptions for deterministic activities will not contain disjunctive sentences over sub-activity occurrences.

Each extension shall have an associated BNF grammar for process descriptions, which extends the BNF grammar associated with PSL-Core.

4.3.5. Format for extensions

All extensions within ISO 18629 will have the following header information:

— Extension name: Core theory names shall have a suffix of the form .th, and definitional extension names shall have a suffix of the form .def;

— Primitive lexicon;

— Defined lexicon;

— Core theories required by the extension: This is a list of PSL-Core theories, each of which is an extension of PSL-Core. The given extension is an extension of the set of axioms which is the union of all theories in the list, together with the axioms in the extension;

— Definitional extensions required by the extension: This is a list of definitional extensions, each of which is an extension of PSL-Core.

The content of any ISO 18629 extension is a set of KIF sentences. For core theories, there is a set of arbitrary KIF sentences for the primitive lexicon of the set of axioms. For definitional extensions and terms in the defined lexicon of a Core theory, each term shall have a conservative definition . In addition, each KIF sentence shall have corresponding text in English that summarises the key intuitions captured by the sentence.

4.4. Organisation of the ISO 18629 Standard

The components of ISO 18629 are grouped into the following parts:

a) Part 1: Overview and basic principles

There are two types of extensions within ISO 18629 core theories and definitional extensions.

b) Part 1x series: Core theories

The current contents of these series of parts addresses:

— Part 11: PSL-Core;

— Part 12: Outer Core;

— Part 13: Time and ordering theories;

— Part 14: Resource theories;

— Part 15: Activity performance theories.

Any new core theory shall be included in the Part 1x series.

c) Part 2x series: External mappings.

The current expected content of this series of part includes:

— Part 21: EXPRESS;

— Part 22: XML;

— Part 23: UML.

This set of mappings may evolve according to industry needs and technology changes.

d) Part 4x series: Definitional extensions

In addition to the core theories, ISO 18629 provides a series of definitional extensions that shall be used to capture the semantics of process terminology in different applications. All definitions in these extensions use the terminology of the core theories. The series currently includes:

— Part 41: Activities;

— Part 42: Time and state;

— Part 43: Ordering;

— Part 44: Resource roles;

— Part 45: Kinds of resource sets;

— Part 46: Processor activities;

— Part 47: Process intent.

Additional extensions are to be developed later according to industry needs by any standardisation committee. Any new extension that is a definitional extension of PSL-Core shall be included in the Part 4x series.

e)Part 2xx series: Translator Implementation Guidelines

Parts of this series shall be developed according to industry needs and technology changes.

NOTE The numbering system for the parts of this standard has been made consistent with the system adopted for the other standards developed within ISO TC184/SC4.

4.5. ISO 18629-1x series Core theories

4.5.1. ISO 18629-11 PSL-Core

The PSL-Core is based upon a precise, mathematical, first-order theory, that is a formal language, a precise mathematical semantics for the language, and a set of axioms that express the semantics in the language.

The basic elements of the language are four primitive classes, two primitive functions, and seven primitive relations of the ontology of the PSL-Core.

The primitive classes are activity, activity_occurrence , timepoint, and object.

The two functions are beginof and endof.

The seven relations are before, occurrence_of, between, before-eq, between-eq, is-occurring-at, participates-in, exist-at.

4.5.2. ISO 18629-12 Outer Core

This set of extensions of PSL-Core is used in defining those extensions that shall be used in practice for specifying manufacturing processes that belong to individual applications. The set of extensions in ISO 18629-12 augment PSL-core but they are not expressive enough to specify the complex items found in practice. The extensions in ISO 18629-12 are more generic and pervasive in their applicability than the rest of the extensions except PSL-Core in ISO 18629. The extensions in ISO 18629-12 are less generic than the PSL-core.

NOTE 1: The main difference is that PSL-Outer Core requires PSL-Core for its specifications whereas PSL-Core does not require any other set of axioms for its specifications.

NOTE 2 : the extensions of outer-core address discrete manufacturing processes whereas another set of extensions to PSL-core may be built to support continuous manufacturing processes.

These extensions are:

— Occurrence Trees;

— Discrete States;

— Subactivity;

— Atomic Activity;

— Complex Activity;

— Activity Occurrences.

The Occurrence Tree extension introduces a tree structure over the set of possible activity occurrences; branches in the tree correspond to different sequences of primitive activity occurrences.

The Discrete State extension specifies the basic concepts for states and their relationships to activity occurrences. In particular, all discrete states are changed by activity occurrences, but they do not change during an activity occurrence.

The Subactivity extension describes how activities can be aggregated and decomposed.

The Atomic Activity extension introduces the class of concurrent activities.

The Complex Activity extension specifies the relationship between occurrences of the subactivities of an activity and occurrences of the activity itself.

The Activity Occurrence extension defines relations that allow the description of how activity-occurrences relate to one another with respect to the time at which they start and end.

These extensions, together with PSL-Core, provide much of the infrastructure for specifying the definitions of terminology within ISO 18629. Every extension within the current version of ISO 18629 is an extension of one or more of these theories.

The following sets of theories define other extensions of the Outer Core, required by some, though not all, definitional extensions within ISO 18629.

4.5.3. ISO 18628-13 Time and ordering theories

The extensions related to time and ordering theories are:

— Duration;

— Subactivity occurrence ordering.

The duration extension introduces the concept of duration as a relationship among timepoint. This allows the introduction of quantitative concepts related to time, as well as providing the basis for defining the duration of activities, activity occurrences, and objects. This duration extension is the specification that is directly supported by the model of duration (also called interval of time in ISO 15531-42) defined by the time model of ISO 15531. ISO 15531-42 clause 3.1.10 provides the definition of duration while ISO 15531-42 clause 5.5.1 provides its EXPRESS model (See ISO 10303-11 for EXPRESS specifications).

NOTE intuitively, duration are the "difference" between two timepoint in the timeline (see the NOTE of clause 3.1.9 of ISO 15531-42).

The subactivity occurrence ordering extension specifies the relations required to represent various kinds of partially ordered sets of activities. This includes sequences, parallelism, AND splits/junctions, and OR splits/junctions.

4.5.4. ISO 18629-14 Resource theories

The extensions related to resource theories are:

— Resource requirements;

— Resource sets.

The resource requirements extension turn into axioms the concept of resource as any object which is required by an activity. That includes the resources as they are defined in ISO 15531-1 (See ISO 15531-clauses 3.6.43 and 4.3) and modelled in ISO 15531-31 and ISO 15531-32 as well as other means at the disposal of the enterprise to achieve its goals.

In particular, resources are defined with respect to the possible interactions among activities, whereas ISO 15531 resources management data model is independent of any kind of activity, assuming that the relations between resources and activities are set up thru the application software or thru the process specification using PSL (See ISO 15531-31 and ISO 15531-32 and especially ISO 15531-31 annex D).

The resource set extension turn into axioms the concept of a set of resources which as a whole also satisfy the axioms for resources. Different kinds of resource sets are defined in Part 44, and they include such concepts as resource pools (sets of machines) and buffers (sets of inventory resources).

4.5.5. ISO 18629-15 Activity performance theories

The extension related to activity performance theories is:

— Activity performance.

The Activity performance extension turn into axioms the relationship that holds between an activity and the actor (such as a human or machine) who performs the activity.

4.6. ISO 18629-2x series External mappings

In addition to specifying the grammar for process descriptions, ISO 18629 also specifies mappings between this grammar and languages used by other manufacturing standards. In particular, there shall be mappings between the grammar of ISO 18629 and EXPRESS as specified in ISO 10303-11, to facilitate interoperability with applications using ISO 10303, as well as mappings to XML and UML.

These other languages are used as alternative ways of representing particular process descriptions, rather than in the specification of the semantics of the terminology of ISO 18629.

4.7. ISO 18629-4x series Definitional Extensions

4.7.1. ISO 18629-41: Activities

The extensions in this part are defined with respect to the Outer Core extensions (Part 12). They include:

— Non-deterministic activities.

The non-deterministic activities extension defines different kinds of activities with respect to constraints on the occurrence of subactivities. An activity is deterministic if all subactivities occur, although they may occur in different orderings, whereas an activity is non-deterministic if not every subactivity occurs when the activity itself occurs.

4.7.2. ISO 18629-42: Time and State

The extensions in this part are defined with respect to the Part 12 Outer Core and the Part 13 Time and ordering theories. They include:

— Activity and occurrence duration;

— Interval activities;

— State constraints.

The activity and occurrence duration extension defines the concept of duration for activities and activity occurrences.

The interval activities extension considers kinds of activities that can be defined using the relations in the state constraints extension. In particular, it considers activities that have the property of being interruptible or non-interruptible.

The state constraints extension defines relations between states and activity occurrences. These include concepts such as preconditions and effects, as well as the notions of an activity achieving or falsifying various states.

4.7.3. ISO 18629-43: Ordering

The extensions in this part are defined with respect to the Part 12 Outer Core and the Part 13 Time and ordering theories. They include:

— Complex sequence ordering relations;

— Ordering relations over activities;

— Temporal ordering constraints.

The complex sequence ordering relations extension is restricted to non-deterministic activities. It supports the specification of activities in which there is branching and conditional occurrence of sub-activities.

The ordering relations over activities extension is restricted to deterministic activities. It specifies different kinds of activities in which there is a partial ordering over sub-activity occurrences.

The temporal ordering constraints extension specifies relations among the occurrences of activities with respect to the times at which they occur. These relations are particularly used within scheduling applications.

4.7.4. ISO 18629-44 Resource roles

The extensions in this part are defined with respect to the Part 12 Outer Core and the Part 14 Resource theories. They include:

— Capacity-based concurrency;

— Resource divisibility;

— Resource role;

— Resource usage.

The capacity-based concurrency extension defines different kinds of activities and resources using concurrency constraints.

The resource divisibility extension considers the different ways in which resources can be shared by multiple activities.

The resource roles extension defines various roles that resources play with respect to activities; these include reusable, consumable, and renewable.

The resource usage extension characterises constraints on resources over the intervals in which activities occur. That is also strongly related to resources usage management data as specified in ISO 15531-1 (See clauses 3.6.43 and 4.3) and modelled in ISO 15531-31 and ISO 15531-32 with the restriction pointed out in clause 4.5.4 of this document.

4.7.5. ISO 18629-45 Kinds of resource sets

The extensions in this part are defined with respect to the Part 12 Outer Core and the Part 14 Resource theories. They include:

— Homogeneous resource sets;

— Inventory resource sets;

— Resource pools;

— Resource set-based activities;

— Substitutable resources.

The primary purpose of these extensions is the set of axioms for discrete capacity resources for which the discreteness of the resource arises from the fact that it is actually composed of a set of resources, and any activity requires or provides some subset of resources in this set.

— Homogeneous sets define different kinds of substitutable resources.

— Inventory resource sets are related to buffers.

— Resource pools are equivalent to discrete capacity resources.

— Resource set-based activities define kinds of activities which use resource sets.

— Substitutable resources make the distinction between sets of arbitrary resources and sets of resources that can be substituted for others in an activity.

4.7.6. ISO 18629-46 Processor Activities

The extensions in this part consider a particular kind of activities that are defined with respect to the roles of the resources required by the activities. This includes:

— Processor actions;

— Resource paths.

Processor actions are actions that use some set of resources, consume some set of material resources, and produce or modify some other set of material resources.

Resource paths are partially ordered sets of processor actions in which the output material resource of one processor action is the input material of the next processor action.

4.7.7. ISO 18629-47 Process intent

This number has been reserved for the development of appropriate extensions.

4.8. ISO 18629-2xx series Translator implementation guidelines

The guidelines in the 200 series of ISO 18629 identify the different extensions within the ontology that are necessary for developing translators among applications in different manufacturing domains.

EXAMPLE: process modelling, process planning, production planning, project management, scheduling, simulation, process execution.

5. Conformance testing methodology and framework

5.1. Conformance of Applications with ISO 18629

A manufacturing process application is conformant to ISO 18629 if and only if the following conditions hold:

— The non-logical lexicon of the manufacturing process application is specified;

NOTE 1 This is the set of terms used by the application that refer either to elements in the domain or relations among these elements.

— There exists a BNF grammar for the process descriptions of the application;

— There exists an implemented bi-directional syntactic translator between ISO 18629 process descriptions and the application process descriptions;

— There exist translation definitions for each term in the application's non-logical lexicon, and these definitions are consistent with the axioms of PSL-Core and a subset of the set of extensions in ISO 18629;

NOTE 2: An application need not be conformant with all extensions of ISO 18629, but it shall be conformant with PSL-Core.

NOTE 3: From the point of view of ISO 18629 two applications can interoperate if they are conformant with the same set of ISO 18629 extensions.

5.2. Conformance of Ontologies with ISO 18629

5.2.1. Conformance of user-defined extensions

An extension consisting of a user-specific set of concepts that are used in conjunction with the ISO 18629 standard shall be conformant with a subset of extensions in ISO 18629 if it uses only the terminology in the non-logical lexicon of those extensions in ISO 18629.

The axioms in any extension that introduces new primitives shall be consistent with the axioms of PSL-Core.

5.2.2. Conformance of external ontologies

An external ontology that includes its own set of axioms of process-related terminology is conformant with a subset of extensions in ISO 18629 if it is consistent with the axioms in those extensions.

EXAMPLE The axioms in any external ontology of time shall be consistent with the axioms of PSL-Core.

5.3. Conformance of future extensions

This clause specifies the conditions that shall be satisfied by any future extensions to the ontology of ISO 18629.

5.3.1. Specification of Models

For each extension, there shall be a technical document that includes a specification of the models of the axioms within the extension. This specification shall contain the following:

— definition of a set of mathematical structures, including the underlying set of elements, the functions and relations on this set of elements, and any distinguished elements of the set;

— for each constant, function and relation symbol in the lexicon, a specification of which substructures are isomorphic to the extension of the constant, function, or relation that is denoted by the symbol;

— a classification of the structures up to isomorphism, and a proof that shows this classification is correct with respect to the definition of the class of structures.

5.3.2. Verification of the extensions

This is a demonstration that the extensions are correct with respect to the specifications of models in the technical document precribed by clause 5.3.1.

For each extension of ISO 18629, the accompanying technical document shall contain proofs of the following two theorems:

— every structure in the set of structures associated with the extension is a model of the axioms in the extension;

— every model of the axioms in the extension is isomorphic to some structure in the set of structures associated with the extension.

Annex A
(normative)

ASN.1 Identifier of ISO 18629-1

To provide for unambiguous identification of an information object in an open system, the object identifier

```
iso standard 18629 part 1 version 1
```

is assigned to this part of ISO 18629. The meaning of this value is defined in ISO/IEC 8824-1 and is described in ISO 18628-1.

Annex B
(informative)

Background to the development of ISO 18629

ISO 18629 is one outcome of the PSL project at the National Institute of Standards and Technology. The approach in developing the language involved five phases: requirements gathering, existing process representation analysis, language creation, pilot implementation and validation, and submission as a candidate standard. The completion of the first phase resulted in a comprehensive set of requirements for specifying manufacturing processes [11]. In the second phase, twenty-six process representations were identified as candidates for analysis by the development team and analysed with respect to the phase one requirements [8]. Nearly all of the representations studied focused on the syntax of process specification rather than the meaning of terms, the semantics. While this is sufficient for exchanging information between applications of the same type, such as process planning, different types of applications associate different meanings with similar or identical terms. As a result of this, a large focus of the third phase involved the development of a formal semantic layer (an ontology [6], [13], [14]) for ISO 18629 based on the Knowledge Interchange Format (KIF) specification [7].

By using this ontology to define explicitly and clearly the concepts intrinsic to manufacturing process information, the language was used to integrate multiple existing manufacturing process applications in the fourth phase of the project. Four families of manufacturing process applications were used in the pilot implementations: process modelling, process planning, scheduling, and manufacturing simulation. Representative applications from each family were used in the demonstration of interoperability. In each case, the terminology of the application was identified and the implicit semantics of this terminology was specified. Based on this semantics, the terminology of the application was mapped to the semantically equivalent terminology within the developing standard. If there were no concepts within the existing ontology that preserved the semantics of the application, an extension was made to the terminology and semantics of the developing standard. For further information about the background of the development of the standard, see [9].

Annex C (informative)

The Need for Semantics

Existing approaches to process modelling lack an adequate specification of the semantics of the process terminology, which leads to inconsistent interpretations and uses of information. Analysis is hindered because models tend to be unique to their applications and are rarely reused. Obstacles to interoperability arise from the fact that the systems that support the functions in many enterprises were created independently, and do not share the same semantics for the terminology of their process models.

The ideal case that we are striving for is guaranteed *complete semantic integration.* This means that software applications can *automatically* exchange information among themselves, and we can provide guarantees that the applications completely share their semantics.

For software applications to be semantically integrated, it is required that all of the terms that one application uses have the same meaning as terms used by the other system. From the receiving application's perspective, the actual meaning of the terms is exactly what is specified by the axioms in the sending application's ontology. It cannot know what was in the mind of the human who designed the application's ontology. For successful semantic integration to take place, this *actual* meaning of a term has to be the same as the meaning that was *intended* by the designer of the application's ontology. For convenience instead of the cumbersome phrase: 'the meaning that was *intended* by the designer of the application's ontology', we will simply say 'the intended semantics of the application's ontology'. Thus, to achieve complete semantic integration, we need to enable the automated exchange of information between software applications in such a way that the intended semantics of the applications' ontologies are preserved by the receiving application.

The key idea is that we must somehow guarantee that the intended semantics of an ontology is the same as the actual semantics of the ontology. The actual semantics is defined by the axioms of the ontology in conjunction with the semantics of the ontology representation language. We must therefore explore the ramifications of explicit formally specified semantics and the requirements that this approach imposes on both the ontology designer as well as the application designer. We need to characterize the relationship between the intended semantics of an application's terminology and the actual semantics of the application's ontology that must hold to support complete semantic integration.

Two software applications can be completely integrated only if they share the semantics of the terminology. One approach to identify semantics is to examine the behavior of the application. The behavior of the application is completely dependent on the inferences it makes, which in turn is completely dependent on the semantics of its axioms. In effect, an application's behavior is constrained by the semantics of its ontology, because any inferences made by the application must conform to the semantics of its internal knowledge together with any information acquired by the application from the environment or other applications. If an application does not do the right thing, or retrieve the right information, or make correct inferences, then it does not share its semantics with other application.

In this operational characterization of semantics, we must make the link between inference and the semantics of the application's ontology. An application's behavior can be characterized by the inferences that it makes, that is, the sentences that can be deduced from its knowledge or the sentences that are consistent with its knowledge.

We now will make the idea of complete semantic integration more precise. Most often, the languages for formal ontologies are closely related to mathematical logic, in which the semantics are based on the notion of an *interpretation*. An interpretation consists of three parts:

— a set of elements (known as the domain or universe of discourse);

— a meaning function that associates symbols in the language with individual elements and sets of elements in the domain (intuitively this specifies what the symbols mean);

— a truth function that associates truth values with sentences in the language.

NOTE For an excellent introduction to logic see [1].

If a sentence is true in the interpretation, we say that the sentence is satisfied by the interpretation. If every axiom in the ontology is satisfied by the interpretation then the interpretation is a called a model of the ontology. With a formal ontology, the application's knowledge is specified as a theory, so that a sentence is consistent if there exists a model of the theory that satisfies the sentence and that a sentence can be deduced if it is satisfied by all models of the theory. Therefore, the application's behavior (and hence the semantics of the application's terminology) can be characterized by this implicit set of models, which we will call the set of *intended models*.

Given this framework, we can say that two applications completely share semantics if their sets of intended models are equivalent. However, applications cannot exchange the models themselves – they can only exchange sentences in the formal language that they use to represent their knowledge. We must be able to guarantee that the inferences made with sentences exchanged in this way are equivalent to the inferences made with respect to the application's intended models – given some input the application uses these intended models to infer the correct output.

We can use this characterization to evaluate the adequacy of the application's ontology with respect to these intended models. We will say that an ontology is verified if and only if the set of models of the axioms of the ontology is equal to the set of intended models for the application's terminology. In a language such as first-order logic, this is equivalent to saying that every sentence that is provable from the axioms of the ontology is also an inference that the application makes, and conversely, every inference that the application makes is provable from the axioms of the ontology.

These properties of an ontology allow us to make the claim that any inferences drawn by the application using the ontology are faithful to the application's semantics. If an ontology is not verified, then it is possible to find sentences that the application infers based on its intended models, but which are not provable from the axioms of the ontology. For example, automated inference cannot be used to determine that two applications can in fact be completely semantically integrated. In the next section, we will consider in some detail the challenges that must be addressed in achieving semantic integration with unverified ontologies.

There are several analogies that we can use to illustrate the relationship between the ontology, the intended models, and the domain in which the application is operating. Physicists use various classes of differential equations to model different phenomena. However, they do not use ordinary linear differential equations to model heat diffusion, and they do not use second-order partial differential equations to model the kinematics of springs. If physicists wish to model some phenomena using a class of differential equations, they can use the equations to predict behaviour of the physical system;

if the predictions are falsified by observations, then we have an inappropriate set of equations. Similarly, in our case, we can use some class of intended models to predict the inferences that an application makes; if there is no physical scenario in the domain that corresponds to these inferences, then we intuitively have an inappropriate set of intended models.

Another analogy is with software engineering. The ontology corresponds to the program and the set of intended models intuitively corresponds to the requirements specification for the program. Verification of the program consists in demonstrating that the output of the program satisfies all of the requirements in the specification; a verified program is then analogous to a verified ontology. Of course, it may be the case that the requirements specification does not reflect all of the conditions necessary for some domain; analogously, the set of intended models for an ontology may not be appropriate for some domain of application, but in such a case, one may argue that the ontology itself is not appropriate.

In summary, we have defined the following important concepts:

Complete Semantic Integration – Complete semantic integration means that applications can automatically exchange information among themselves, and we can provide guarantees that the applications completely understand each other. Formally, complete semantic integration can occur only when the intended models of both applications' ontologies are the same.

Verified Ontology – A verified ontology is one that has the following property: what actually *is represented* in the ontology is *exactly* what the ontology designer *intended to represent*. Formally, this means that the actual models of an ontology are exactly the same as the intended models.

Verified ontologies are required to guarantee complete semantic integration.

The upshot of this is that we have in effect, reduced the problem of complete semantic integration, to the problem of requiring that applications share the semantics of their ontologies.

Annex D
(informative)

Interoperability

A semantics-preserving exchange of information means that there are mappings between logically equivalent concepts in each ontology. The challenge of semantic integration is therefore equivalent to the problem of generating such mappings, determining that they are correct, and providing a vehicle for executing the mappings, thus translating terms from one ontology into another.

We are considering the applications to be operating in an open environment. For simplicity, we will consider just two applications (Alice and Bob). They are attempting to communicate with each other, but have never interacted before. This is a different and more general case than a more closed scenario in which a fixed group of partners (such as a consortium or divisions within an enterprise) attempts to establish interoperability among their software, *a priori.*

There are a variety of architectures that may be used to achieve semantic integration. The differences depend on the origins of the semantic mappings, whether there is a mediating ontology, and the degree of agreement that exists among the anticipated community of interacting applications. Different architectures can be distinguished and compared to one another by considering the following questions:

1. Who is generating and testing the semantic mapping?
    a. application designer
    b. ontology designer, applications are reusing them
    c. applications themselves, dynamically at application-interaction time
2. When are the mappings created that make the link between one application's ontology and the other one's ontology.
    a. Mappings are pre-defined; the applications execute them to achieve translation between their ontologies.
    b. Mappings do not exist *a* priori, they are dynamically generated and executed to achieve translation;
3. What is the topology of the architecture?
    a. Mapping is done point-to-point between the applications;
    b. Mapping is mediated by a third ontology (or set of ontologies).
4. What is the degree of agreement between the applications?
    a. Single agreed ontologies within a community, possible merged from existing ones
    b. Alignment – could be loose or strong.
    c. No a priori agreement

In the case of manual mapping, the human application designers must specify the semantic mappings between Alice's and Bob's ontologies prior to their interaction; Alice and Bob themselves do not generate the mapping. Further, these mappings are generated point-to-point between the applications. There is no a priori agreement about semantics between the applications.

In the Interlingua architecture, Alice's designer specifies the semantic mapping between Alice's ontology and a standard interchange ontology, and Bob's designer specifies the semantic mapping between Bob's ontology and the same interchange ontology. When Alice and Bob first interact, they use these previously specified mappings to automatically generate the semantic mappings between each other's ontologies. In this case, the interlingua ontology mediates the mapping between the application ontologies. The applications that wish to participate in this architecture, must agree *a priori* to use the interlingua ontology.

The interlingua architecture is a generalization of the Manual Mapping architecture. Its distinguishing feature is the existence of a mediating ontology that is independent of the applications' ontologies, and which is used as a neutral interchange ontology [3]. The semantic mappings between application and interlingua ontology are manually generated and verified prior to application interaction time [10]. This process of creating the mapping between the application ontology and the interlingua ontology is identical to the process of creating a mapping directly between two application ontologies, which is done in the manual mapping approach. As such, we can consider the application ontologies to be integrated with the interlingua ontology.

This architecture is much more powerful than manual mapping, particularly when there many applications and the creation and maintenance of mappings becomes unmanageable. For example, we only need to specify one mapping for each application ontology, whereas the manual mapping architecture requires a mapping for each pair of application ontologies. The existence of the pre-defined mappings between the application ontologies and the interlingua ontology enables the automatic generation of the point-to-point mapping between the applications' ontologies. If one application's ontology changes, then only one mapping need be affected, rather than one for each application. New applications can subscribe to the community of applications using this interlingua merely by creating a mapping to and from the interlingua. With no changes to their own mappings, all other applications now can translate between their ontology, and the new application's ontology. This was not possible in the manual mapping case because every point to point mapping has to be pre-specified.

Semantic mappings can be expressed as formal definitions or rules between application ontologies and interlingua ontologies. In the interlingua approach, there are two steps in translation: the execution of the mapping from the application ontology to the interlingua, and from the interlingua to the other application's ontology. The translation can be accomplished by applying deduction to the axioms of the interlingua ontology and the formal mapping rules. If these rules have already been verified to preserve semantics between the application and interlingua ontologies, we are guaranteed that translation between the applications also preserves semantics. In effect, a direct mapping rule from one applications ontology to the other applications ontology is inferred from the two separate rules. If run-time translation efficiency is important, then the point to point mapping rules could be cached as explicit rules; otherwise they need only be implicit. When they are implicit, then the otherwise distinct processes of creating and executing a mapping is conflated – it happens at the same time, rather than being two separate steps. See [14] for a more detailed discussion of the tradeoffs between the point to point and interlingua approaches.

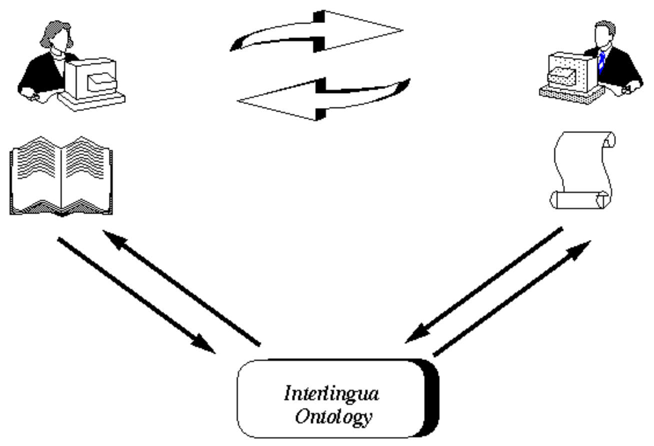

Figure D1: Interlingua Architecture –

The thin arrows represent manually generated mappings created by the application designers prior to application integration. The thick arrows represent the [possibly implicit] application to application mapping that is automatically generated.

Achieving complete semantic integration with this architecture makes several rather strong assumptions. First, the interlingua ontology must be verified. If it is not, then there is no guarantee that the mapping rules are faithfully capturing the intended models of the application ontologies. Although is it logically possible to preserve semantics directly between the application ontologies, it will not be guaranteed by the mapping.

The interesting aspect of this architecture is that complete semantic integration does not assume that the application ontologies themselves are verified; however, it does require that models of the application ontology together with the semantic mappings are equivalent to the intended models of the application. In other words, even though there exist unintended models of the application ontology's axioms, if the interlingua ontology is powerful enough, then it can be used to augment the application ontology so that it is verified.

If on the other hand, the application ontologies *are* verified, this is not enough to guarantee complete semantic integration. We must in addition, assume that they are conformant with the interlingua ontology. That is, every model of an application ontology can be interpreted within a model of the interlingua ontology. Intuitively, this is related to the intuition that the application and interlingua ontologies cover the same set of concepts.

Very importantly, the interlingua should *not be considered to be a global ontology* that all applications are required to use directly. Rather, it should be considered to be a mediating ontology, as in Infomaster [4], in which all applications use their native ontology for their own reasoning, and only use the interlingua ontology to communicate with each other. This perspective is particularly relevant if we consider the applications to be using heterogeneous information resources. Both the users' terminology and the native terminology of the information sources are mapped to a mediating ontology that can be thought of as a reference model for the domain.

Finally, the Interlingua architecture is designed to work best in the context where there are a variety of different kinds of information sources, all pertaining to a common domain, rather than being an overall architecture for a multiplicity of domains. In practice, applications will use ontologies in multiple domains (e.g. process, product, resource, services), so that the architecture must be generalised to be "multi-hub", in which there may be a different interlingua ontology for each domain. In this case, an application directly linked to one hub could be integrated with an application in another hub if point to point mappings between the hubs were provided. The critical challenge for such an approach is to ensure consistency among the set of overlapping concepts between each domain, a problem that motivates the Community integration architecture.

Translation with PSL is done in two steps: syntactic and semantic. Each application maps its ontology into PSL; these mappings are referred to as translation definitions. Semantic translation is the application of these mappings to a process description.

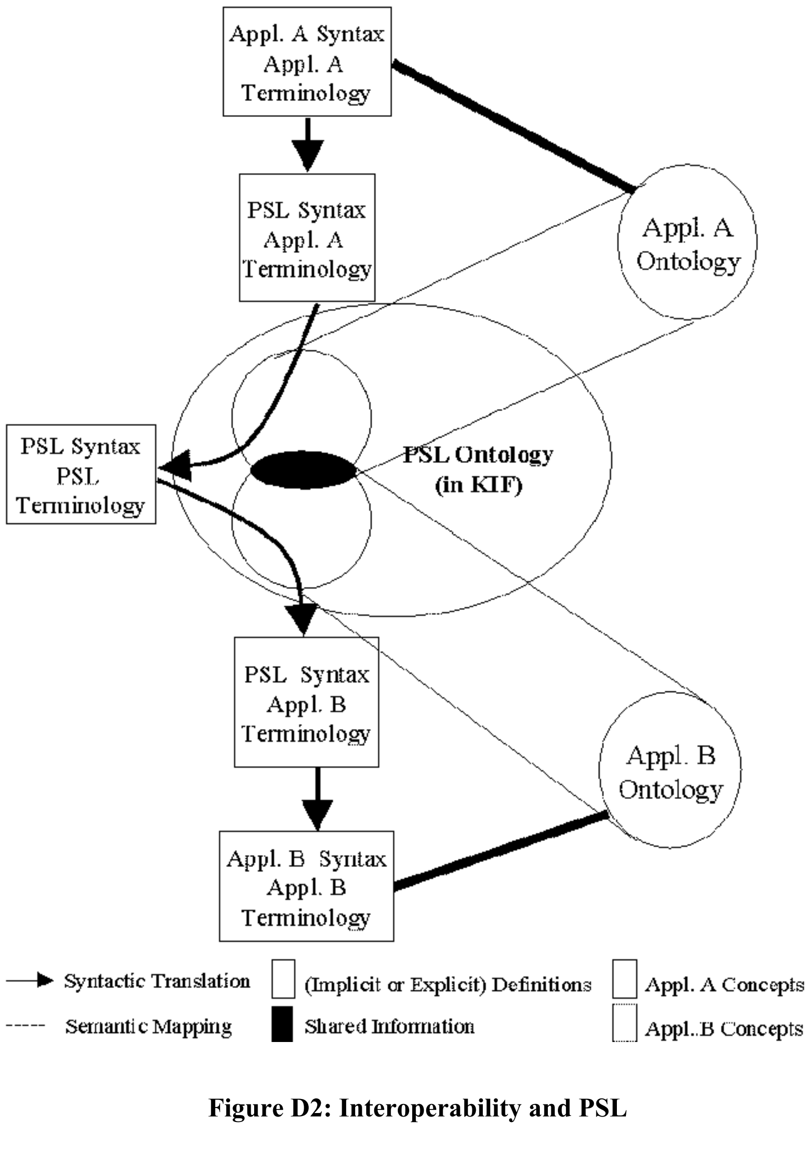

Figure D2: Interoperability and PSL

Annex E (informative)

Architecture of PSL

The PSL Ontology is organized into PSL-Core and a partially ordered set of extensions. All axioms are first-order sentences, and are written in KIF (the Knowledge Interchange Format).

There are two types of extensions within PSL : core theories and definitional extensions. Core theories introduce and provide axioms new relations and functions that are primitive. All terminology introduced in a definitional extension have conservative definitions using the terminology of the core theories. Thus, definitional extensions add no new expressive power to PSL-Core.

E1. PSL Core

The purpose of PSL-Core is to provide axioms for a set of intuitive semantic primitives that is adequate for describing the fundamental concepts of manufacturing processes (refs). Consequently, this characterization of basic processes makes few assumptions about their nature beyond what is needed for describing those processes, and the Core is therefore rather weak in terms of logical expressiveness. In particular, PSL-Core is not strong enough to provide definitions of the many auxiliary notions that become necessary to describe all intuitions about manufacturing processes.

To supplement the concepts of PSL-Core, the ontology includes a set of extensions that introduce new terminology. Any PSL extension provides the logical expressiveness to provide axioms related to intuitions involving concepts that are not explicitly specified in PSL-Core. All extensions within PSL are consistent extensions of PSL-Core, and may be consistent extensions of other PSL extensions. However, not all extensions within PSL need be mutually consistent. Also, the core theories need not be conservative extensions of other core theories.

The PSL-Core can be used to support interoperability by ensuring that concepts that are common across applications can be exchanged correctly, while also identifying those concepts on which the applications disagree and hence cannot share. By organising all concepts within ISO 18629 as a partially ordered set of extensions, applications can explicitly declare which extensions provide the semantics for their application's terminology.

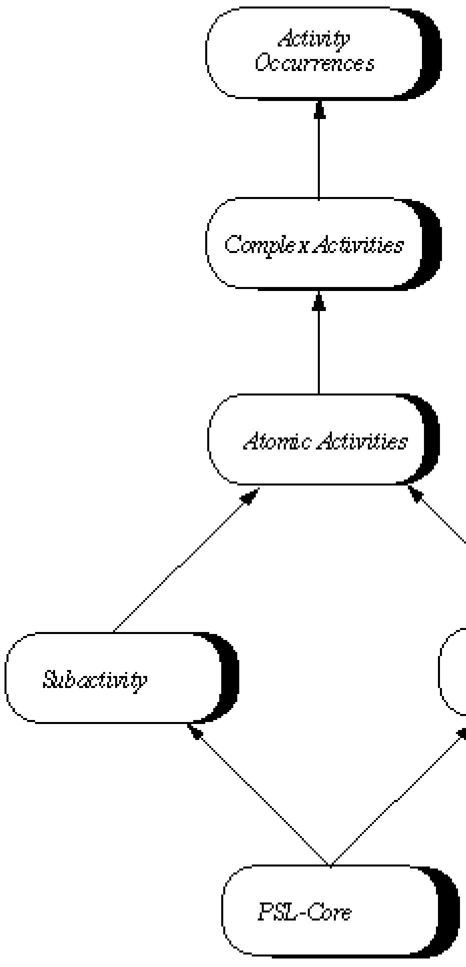

Figure E1: The theories in the Outer Core of PSL.

E1.1 Occurrence Trees

An occurrence tree is the set of all discrete sequences of activity occurrences. They are isomorphic to substructures of the situation tree from situation calculus (refs), the primary difference being that rather than a unique initial situation, each occurrence tree has a unique initial activity occurrence. As in the situation calculus, the *poss* relation is introduced to allow the statement of constraints on activity occurrences within the occurrence tree. Since the occurrence trees include sequences that modellers of a domain will consider impossible, the *poss* relation "prunes" away branches from the occurrences tree that correspond to such impossible activity occurrences.

It should be noted that the occurrence tree is not the structure that represents the occurrences of subactivities of an activity. The occurrence tree is not representing a particular occurrence of an activity, but rather all possible occurrences of all activities in the domain.

E1.2 Discrete States

The Discrete States core theory introduces the notion of state (fluents). Fluents are changed only by the occurrence of activities, and fluents do not change during the occurrence of primitive activities. In addition, activities have preconditions (fluents that must hold before an occurrence) and effects (fluents that always hold after an occurrence).

E1.3 Subactivities

This core theory provides axioms for intuitions about subactivities. The only constraint imposed within this theory is that the subactivity relation is isomorphic to a discrete partial ordering. Other core theories impose additional constraints.

E1.4 Atomic Activities

The core theory of Atomic Activities provides axioms for intuitions about the concurrent aggregation of primitive activities. Such concurrent aggregation is represented by the occurrence of concurrent activities rather than concurrent activity occurrences.

E1.5 Complex Activities

This core theory provides the foundation for representing and reasoning about complex activities and the relationship between occurrences of an activity and occurrences of its subactivities. Within models of the Complex Activities theory, occurrences of complex activities correspond to subtrees of the occurrence tree. An activity may have subactivities that do not occur; the only constraint is that any subactivity occurrence must correspond to a subtree of the activity tree that characterises the occurrence of the activity. Not every occurrence of a subactivity is a subactivity occurrence. There may be other external activities that occur during an occurrence of an activity. Different subactivities may occur on different branches of the activity tree, so that different occurrences of an activity may have different subactivity occurrences.

E1.6 Activity Occurrences

The Complex Activities only provides axioms for constraints on atomic subactivity occurrences. The Activity Occurrences theory generalises these intuitions to arbitrary complex subactivities.

E1.7 Additional Core Theories

The remaining core theories in the PSL Ontology include Subactivity Occurrence Ordering (providing axioms for different partial orderings over subactivity occurrence), Iterated Occurrence Ordering (axioms necessary for defining iterated activities), Duration (augmenting PSL-Core with a metric over the timeline), and Resource Requirements (which specifies the conditions that must be satisfied by any object that is a resource for an activity).

E2. Definitional Extensions

The definitional extensions are grouped into parts according to the core theories that are required for their definitions. Figure 3 gives an overview of these groups together with example concepts that are defined in the extensions. The definitional extensions in a group contain definitions that are conservative with respect to the specified core theories; for example, all concepts in the Temporal and State Extensions have conservative definitions with respect to both the Complex Activities and Discrete States theories.

Definitional Extensions	Core Theories	Example Concepts
Activity Extensions	Complex Activities	deterministic / nondeterministic activities concurrent activities partially ordered activities
Temporal and State Extensions	Complex Activities, Discrete States	preconditions effects conditional activities triggered activities
Activity Ordering and Duration Extensions	Subactivity Occurrence Ordering, Iterated Occurrence Ordering, Duration	complex sequences and branching iterated activities duration-based constraints
Resource Role Extensions	Resource Requirements	reusable, consumable, renewable, deteriorating resources

Figure E2: Definitional extensions of PSL.

Bibliography

[1] Barwise, J., Etchemendy, J., Allwein, G., Barker-Plummer, D. *Language, Proof, and Logic*. Seven Bridges Press, 2000

[2] - Booch G., Rumbaugh J., Jacobson I., *The Unified Modelling Language User Guide*, Addison-Wesley, 1999

[3] Ciocoiu, M., Gruninger M., and Nau, D. (2001) Ontologies for integrating engineering applications*, Journal of Computing and Information Science in Engineering,*1:45-60, 2001.

[4] Oliver M. Duschka and Michael R. Genesereth. *"Infomaster - an information integration tool"*. In Proceedings of the International Workshop on Intelligent Information Integration, Freiburg, Germany, September 1997

[5] – Extensible Markup Language (XML) 1.0, *W3C Recommendation 10-February-1998*, http://www.w3.org/TR/1998/REC-xml-19980210.

[6] – Fox, M., et al, *An Organisation Ontology for Enterprise Modelling: Preliminary Concepts"*, Computers in Industry, 1996, Vol. 19, pp. 123-134.

[7] – Genesereth, M., Fikes, R.: *Knowledge Interchange Format (Version 3.0)* - Reference Manual, 1992, Computer Science Dept., Stanford University, Stanford, CA.

[8] – Knutilla, A., et al., *Process Specification Language: An Analysis of Existing Representations*, NISTIR 6133, 1998, National Institute of Standards and Technology, Gaithersburg, MD.

[9] – Lee, J., et al, *"The PIF Process Interchange Format and Framework"*, The Knowledge Engineering Review, Vol. 13(1), pp. 91-120, Special Issue on "Putting Ontologies to Use" (eds. Uschold, M. and Tate, A.), Cambridge University Press.

[10] Schlenoff, C., Gruninger, M., Ciocoiu, M. *The Essence of the Process Specification Language*, Transactions of the Society for Computer Simulation *vol.16 no.4 (December 1999) pages 204-216.*

[11] – Schlenoff, C., Knutilla, A., Ray, S., *Unified Process Specification Language: Requirements for Modelling Processes:* NISTIR 5910, 1996, National Institute of Standards and Technology, Gaithersburg, MD.

[12] – Uschold, M., et. al., *"The Enterprise Ontology",* The Knowledge Engineering Review, Vol. 13(1), pp. 31-89, Special Issue on "Putting Ontologies to Use" (eds. Uschold, M. and Tate, A.), Cambridge University Press.

[13] – Uschold, M. and Gruninger M*., Ontologies: Principles, Methods, and Applications,* Knowledge Engineering Review, 1996, Vol. 11, pp. 96-137.

[14] M. Uschold, R. Jasper, and P. Clark. *Three Approaches for Knowledge Sharing: A Comparative Analysis*. Proc 12th Workshop on Knowledge Acquisition, Modeling, and Management (KAW'99), 1999.

Index

ICS 25.040.40

Price based on 37 pages