INTERNATIONAL STANDARD

ISO 18629-13

First edition
2006-08-15

Industrial automation systems and integration — Process specification language —

Part 13: **Duration and ordering theories**

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

Partie 13: Théories de classement et de durée

Reference number
ISO 18629-13:2006(E)

Contents

Page

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 part of ISO 18629 may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 18629-13 was prepared by Technical Committee 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

ISO 18629 is an International Standard for the computer-interpretable exchange of information related to manufacturing processes. Taken together, all the parts contained in the ISO 18629 Standard provide a generic 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.

This part of ISO 18629 provides a description of the core elements of the language defined within ISO 18629.

All parts of ISO 18629 are independent of any specific process representation or model proposed in a software application in the domain of manufacturing management. Collectively, they provide a structural framework for improving the interoperability of these applications.

Industrial automation systems and integration — Process specification language —

Part 13: **Duration and ordering theories**

1 Scope

This part of ISO 18629 provides a representation of the primitive concepts related to ordering and durations constraints for activities.

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

- subactivity occurrence orderings;
- duration;
- iterated occurrence orderings;
- occurrence tree endomorphisms;
- activity envelopes.

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) — Part 1: Specification of basic notation.*

ISO 15531-1: *Industrial automation systems and integration — Industrial manufacturing management data — Part 1: General overview.*

ISO 18629-1: *Industrial automation systems and integration — Process specification language — Part 1: Overview and basic principles.*

ISO 18629-11: 2005, *Industrial automation systems and integration — Process specification language — Part 11: PSL-core.*

ISO 18629-12: *Industrial automation systems and integration — Process specification language — Part 12: Outer core*.

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
automorphism
one-to-one mapping of elements on a set that preserves the relations and functions in some model

3.1.2
axiom
well-formed formula in a formal language that provides constraints on the interpretation of symbols in the lexicon of a language

[ISO 18629-1]

3.1.3
commutative group
algebraic structure with an internal binary operation (OP) with respect to which : a OP b = b OP a

3.1.4
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

[ISO 18629-1]

3.1.5
core theory
set of axioms for relation and function symbols that denote primitive concepts

[ISO 18629-1]

3.1.6
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.

[ISO 18629-1]

3.1.7
endomorphism
mapping from a set onto a subset that preserves the relations and functions in some model

3.1.8
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.

[ISO 18629-1]

3.1.9
grammar
specification of how logical symbols and lexical terms can be combined to make well-formed formulae

[ISO 18629-1]

3.1.10
homomorphism
mapping between sets that preserves some relation on the elements of the set

3.1.11
interpretation
universe of discourse and assignment of truth values (TRUE or FALSE) to all sentences in a theory

NOTE See annex B for an example of an interpretation.

[ISO 18629-11]

3.1.12
language
combination of a lexicon and a grammar

[ISO 18629-1]

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.

[ISO 18629-1]

3.1.14
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: 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.

[ISO 18629-1]

3.1.15
monomorphism
one to one mapping that preserves some relation on the elements of the set

3.1.16
ontology
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 in 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.

[ISO 18629-1]

3.1.17
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.

[ISO 18629-1]

3.1.18
primitive concept
lexical term that has no conservative definition

[ISO 18629-1]

3.1.19
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.

[ISO 18629-1]

3.1.20
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]

3.1.21
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.

[ISO 18629-1]

3.1.22
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.

[ISO 18629-1]

3.1.23
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.

[ISO 18629-1]

3.1.24
universe of discourse
the collection of concrete or abstract things that belong to an area of the real world, selected according to its interest for the system to be modelled and for its corresponding environment

[ISO 15531-1]

3.2 Abbreviations

For the purpose of this part of ISO 18629, the following abbreviations apply:

— **FOL** First-Order Logic:

— **BNF** Backus-Naur form;

— **KIF** Knowledge Interchange Format;

— **PSL** Process Specification Language.

4 General information on ISO 18629

As stated in ISO 18629-1, ISO 18629 as a whole specifies a language for the representation of process information, which is a process specification language named PSL. It is composed of a lexicon, an ontology, and a grammar for process descriptions that are distributed in part 11 to 19 and part 41 to 49 of ISO 18629[1].

[1] Certain parts are under development

NOTE 1 PSL is a language for specifying manufacturing processes based on a mathematically well defined vocabulary and grammar. As such, it is different from other languages such as EXPRESS (defined in ISO 10303-11) and used for example in ISO 10303-41, ISO 10303-42, ISO 10303-49, ISO 13584, ISO 15531 and ISO 15926, that are modelling languages. 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. PSL is based on Mathematical Set Theory and Situation Calculus (see ISO 18629-11: 2005, Annex D).

Part 11 to 19 of ISO 18629 specify core theories needed to give precise definitions and the corresponding axioms of the primitive concepts of ISO 18629, thus enabling precise semantic translations between different schemes.

According to that parts 11 to 19 of ISO 18629 provide:

— the representation of the basic elements of the language;

— standardized sets of axioms that correspond to intuitive semantic primitive concepts adequate to describe basic processes;

— the set of rules to develop, in compliance with PSL-Core, other core theories or extensions such as the extensions that are provided in parts 41 to 49.

5 Organization of this part of ISO 18629

This clause specifies the fundamental theories from which this part of ISO 18629 is composed.

The core theories that constitute this part of ISO 18629 are:

— Subactivity occurrence ordering theory (soo.th);

— Duration theory (duration.th);

— Occurrence tree automorphism theory (preserve.th);

— Activity envelope theory (envelope.th);

All theories in this part of ISO 18629 are extensions of theories found within ISO 18629-12

6 Subactivity occurrence ordering core Theory

The Subactivity occurrence ordering theory introduces the concepts necessary to represent intuitions about process flow and partial orderings over the subactivity occurrences within a complex activity.

6.1 Primitive Relations of the Subactivity occurrence ordering theory

The nonlogical lexicon of the Subactivity occurrence ordering theory contains two primitive relation symbols:

— soo;

— soo_precedes.

The nonlogical lexicon of the Subactivity occurrence ordering theory contains one primitive function symbol:

— soomap.

6.2 Defined Relations of the Subactivity occurrence ordering theory

The nonlogical lexicon of the Subactivity Theory contains the following defined relation symbols:

— root_soo;

— leaf_soo;

— next_subactivity.

6.3 Relationship to other sets of axioms

The Subactivity occurrence ordering theory requires the following core theories

— psl_core.th;

— occtree.th;

— atomic.th;

— complex.th;

— actocc.th.

No definitional extensions are required by the Subactivity occurrence ordering theory.

6.4 Informal Semantics of the Subactivity occurrence ordering theory

6.4.1 soo

KIF notation for soo:

(soo ?s ?a)

Informal semantics for soo:

(soo ?s ?a) is TRUE in an interpretation of Subactivity occurrence ordering theory if and only if the activity occurrence ?s is an element of the Subactivity occurrence ordering for activity ?a..

6.4.2 soo_precedes

KIF notation for primitive:
(soo_precedes ?s1 ?s2 ?a)

Informal semantics for soo_precedes:

(soo_precedes ?s1 ?s2 ?a) is TRUE in an interpretation of Subactivity occurrence ordering theory if and only if the activity occurrence ?s1 precedes activity occurrence ?s2 in the subactivity occurrence ordering for the activity ?a. This relation defines a partial ordering over the subactivity occurrences of a complex activity.

6.4.3 soomap

KIF notation for soomap:
(soomap ?s)

Informal semantics for soomap:

(= ?s1 (soomap ?s)) is TRUE in an interpretation of Subactivity occurrence ordering theory if and only if the activity occurrence ?s1 is the element of the subactivity occurrence ordering that corresponds to ?s.

6.5 Definitions in the Subactivity occurrence ordering theory

6.5.1 Definition 1 (related to root_soo)

An activity occurrence is a root of a subactivity occurrence ordering for an activity ?a if and only if it is an element of the ordering and there does not exist an activity occurrence that precedes it in the ordering.

```
(forall (?s ?a) (iff (root_soo ?s ?a)
(and (soo ?s ?a)
    (not (exists (?s1)
      (soo_precedes ?s1 ?s ?a)))))
```

6.5.2 Definition 2 (related to leaf_soo)

An activity occurrence is a leaf of a subactivity occurrence ordering for an activity ?a if and only if it is an element of the ordering and there does not exist an activity occurrence that follows it in the ordering

```
(forall (?s ?a) (iff (leaf_soo ?s ?a)
              (and (soo ?s ?a)
              (not (exists (?s1)
                    (soo_precedes ?s ?s1 ?a)))))
```

6.5.3 Definition 3 (related to next_soo)

An activity occurrence ?s2 is the next subactivity occurrence after ?s1 in a subactivity occurrence ordering for ?a if and only if ?s1 precedes ?s2 in the ordering and there does not exist a subactivity occurrence that is between them in the ordering.

```
(forall (?s1 ?s2 ?a) (iff (next_soo ?s1 ?s2 ?a)
(and    (soo_precedes ?s1 ?s2 ?a)
    (not (exists (?s3)
      (and    (soo_precedes ?s1 ?s3 ?a)
              (soo_precedes ?s3 ?s2 ?a)))))))
```

6.6 Axioms of the Subactivity occurrence ordering theory

6.6.1 Axiom 1

The soo_precedes relation orders elements of the subactivity occurrence ordering.

```
(forall (?s1 ?s2 ?a)
    (implies  (soo_precedes ?s1 ?s2 ?a)
        (and (soo ?s1 ?a)
              (soo ?s2 ?a))))
```

6.6.2 Axiom 2

Elements of the subactivity occurrence ordering are elements of the activity tree.

```
(forall (?a ?s)
    (implies  (soo ?s ?a)
        (or   (root ?s ?a)
              (exists (?s1)
                    (min_precedes ?s1 ?s ?a)))))
```

6.6.3 Axiom 3

The function soomap maps the activity tree to the subactivity occurrence ordering.

```
(forall (?s ?a)
    (soo (soomap ?s) ?a))
```

6.6.4 Axiom 4

The elements of the subactivity occurrence ordering are fixed by soomap.

```
(forall (?s ?a)
    (implies  (soo ?s ?a)
        (= ?s (soomap ?s))))
```

6.6.5 Axiom 5

The function soomap is a branch monomorphism.

```
(forall (?s ?a)
    (or       (mono (?s (soomap ?s) ?a)
      (= ?s (soomap ?s))))
```

6.6.6 Axiom 6

The activity tree is order-homomorphic to the subactivity occurrence ordering.

```
(forall (?a ?s1 ?s2)
    (implies  (min_precedes ?s1 ?s2 ?a)
        (iff   (soo_precedes (soomap ?s1) (soomap ?s2) ?a)
               (not (exists (?s3 ?s4)
                      (and     (min_precedes ?s4 ?s3 ?a)
                         (= (soomap ?s3) (soomap ?s1))
                         (= (soomap ?s4) (soomap ?s2))))))))
```

6.6.7 Axiom 7

soo_precedes is not symmetric.

```
(forall (?a ?s1 ?s2)
    (implies  (soo_precedes ?s1 ?s2 ?a)
        (not (soo_precedes ?s2 ?s1 ?a))))
```

6.6.8 Axiom 8

soo_precedes is transitive.

```
(forall (?a ?s1 ?s2 ?s3)
    (implies  (and   (soo_precedes ?s1 ?s2 ?a)
                (soo_precedes ?s2 ?s3 ?a))
        (soo_precedes ?s1 ?s3 ?a)))
```

7 Duration theory

The Duration theory introduces the concept of duration. It adds a metric to the timeline by mapping every pair of points to a new subclass of objects called timeduration that satisfy the axioms of vector spaces.

7.1 Primitive relations in the Duration theory

The nonlogical lexicon of the Duration theory contains two primitive relation symbols:

— timeduration;

— lesser.

7.2 Primitive Functions and Constants

The nonlogical lexicon of the Duration theory contains three function symbols:;

— duration;

— add;

— mult.

The nonlogical lexicon of the Duration theory contains four constant symbols:

— zero;

— one;

— max+;

— max-.

7.3 Defined Relations of the Duration theory

The nonlogical lexicon of the Duration theory contains one defined relation symbol:

— time_add.

7.4 Relationship to other sets of axioms

The Duration theory requires :

— psl_core.th.

No definitional extensions are required by the Duration theory.

7.5 Informal Semantics of the Duration theory

7.5.1 timeduration

KIF notation for timeduration :
(timeduration ?d)

Informal semantics for timeduration :

(timeduration ?d) is TRUE in an interpretation of the Duration theory if and only if ?f is a member of the set of timedurations in the universe of discourse of the interpretation. Timedurations are a subcategory of object.

7.5.2 lesser

KIF notation for lesser :
(lesser ?d1 ?d2)

Informal semantics for lesser:

All timedurations are linearly ordered.
(lesser ?d1 ?d2) is TRUE in an interpretation of the Duration theory if and only if the value of the timeduration ?d1 is less that the value of the timeduration ?d2.

7.5.3 duration

KIF notation for duration:
(duration ?t1 ?t2)

Informal semantics for duration:

(= (duration ?t1 ?t2) ?d) is TRUE in an interpretation of the Duration theory if and only if ?d denotes the timeduration whose value is the distance between the timepoints ?t1 and ?t2 on the timeline.

7.5.4 time_add

KIF notation for time_add :
(time_add ?t ?d)

Informal semantics for time_add:

(= (time_add ?t ?d) ?t2) is TRUE in an interpretation of the Duration theory if and only if ?t2 denotes the timepoint which is distance ?d from the timepoints ?t on the timeline.

7.5.5 add

KIF notation for add :

(add ?d1 ?d2)

Informal semantics for add:
(= (add ?d1 ?d2) ?d3) is TRUE in an interpretation of the Duration theory if and only if ?d3 denotes the timeduration whose value is sum of the values of the timedurations ?d1 and ?d2.

7.5.6 mult

KIF notation for mult :
(mult ?x ?d)

Informal semantics for mult:
(= (mult ?x ?d) ?d2) is TRUE in an interpretation of the Duration theory if and only if ?d2 denotes the timeduration whose value is product of the value of the timedurations ?d and ?x, where ?x is an element of a commutative group.

7.5.7 zero

Informal semantics for zero:
zero denotes the timeduration constant that is the additive identity for the add function.

7.5.8 one

Informal semantics for one:
one denotes the constant that is the multiplicative identity for the mult function.

7.5.9 max+

Informal semantics for max+:
max+ is the maximum timeduration.

7.5.10 max-

Informal semantics for max-:
max- is the minimum timeduration.

7.6 Definitions of Duration theory

7.6.1 Definition 1

The time_add function maps a timepoint ?t1 and a timeduration ?d to the timepoint ?t2 such that the duration between ?t1 and ?t2 is ?d.


```
(forall (?t1 ?t2 ?d)
(iff(= ?t2 (time_add ?t1 ?d))
    (and    (timepoint ?t1)
      (timepoint ?t2)
      (timeduration ?d)
      (= ?d (duration ?t2 ?t1)))))
```

7.7 Axioms for the Duration theory

The set of axioms in the Duration theory is as follows:

7.7.1 Axiom 1

zero, max+, and max- are all timedurations.

```
(and (timeduration zero)
    (timeduration max+)
    (timeduration max-))
```

7.7.2 Axiom 2

The result of adding two timedurations is a timeduration.

```
(forall (?d1 ?d2)
(implies  (and    (timeduration ?d1)
      (timeduration ?d2))
     (timeduration (add ?d1 ?d2))))
```

7.7.3 Axiom 3

The add function is associative.

```
(forall (?d1 ?d2 ?d3)
(implies  (and    (timeduration ?d1)
      (timeduration ?d2)
      (timeduration ?d3))
       (= (add (add ?d1 ?d2) ?d3) (add ?d1 (add ?d2 ?d3))))
```

7.7.4 Axiom 4

zero is the additive identity.

(forall (?d)

(implies (timeduration ?d)

(= (add ?d zero) ?d)))

7.7.5 Axiom 5

The opposite of a timeduration is a timeduration.

(forall (?d1)

(implies (timeduration ?d1)

(exists (?d2)

(and (timeduration ?d2)

(= (add ?d1 ?d2) zero))))

7.7.6 Axiom 6

The add function is commutative.

(forall (?d1 ?d2)

(implies (and (timeduration ?d1)

(timeduration ?d2))

(= (add ?d1 ?d2) (add ?d2 ?d1))))

7.7.7 Axiom 7

The result of multiplying a timeduration by a scalar value is a timeduration.

(forall (?d ?r)

(implies (timeduration ?d)

(timeduration (mult ?r ?d))))

7.7.8 Axiom 8

Timedurations can be multiplied by scalars:

(forall (?d1 ?d2 ?r)

(= (mult ?r (add ?d1 ?d2)) (add (mult ?r ?d1) (mult ?r ?d2))))

7.7.9 Axiom 9

Multiplication of timedurations by scalars is distributive:

(forall (?d ?r ?s)
    (= (mult (add ?r ?s) ?d) (add (mult ?r ?d) (mult ?s ?d))))

7.7.10 Axiom 10

Multiplication of timedurations by scalars is associative.

(forall (?d ?r ?s)
    (= (mult (mult ?r ?s) ?d) (mult ?r (mult ?s ?d))))

7.7.11 Axiom 11

one is the multiplicative identity.

(forall (?d)
    (= ?d (mult one ?d)))

7.7.12 Axiom 12

The add function on timedurations preserves the lesser ordering over timedurations.

(forall (?d1 ?d2 ?d3)
  (and (timeduration ?d1)
    (timeduration ?d2)
    (timeduration ?d3))
    (iff (lesser ?d1 ?d2)
        (lesser (add ?d1 ?d3) (add ?d2 ?d3))))

7.7.13 Axiom 13

If two timedurations are equal, adding them to another timeduration will also be equal.

(forall (?d1 ?y ?z)
  (and (timeduration ?d1)
    (timeduration ?d2)
    (timeduration ?d3))
    (iff (= ?d1 ?d2)
        (= (add ?d1 ?d3) (add ?d2 ?d3)))))

7.7.14 Axiom 14

max- is lesser than any timeduration, and max+ is greater than any timeduration.

```
(forall (?d)
    (implies  (timeduration ?d)
        (and (lesser ?d max+)
              (lesser max- ?d))))
```

7.7.15 Axiom 15

The result of adding any duration other than max+ to max- is max-, and vice versa, the "sum" of max- and max+ is zero.

```
(forall (?d)
(implies  (timeduration ?d)
     (and    (implies  (not (= ?d max-)) (= max+ (add ?d max+)))
      (implies  (not (= ?d max+)) (= max- (add ?d max-)))
      (= zero (add max+ max-)))))
```

7.7.16 Axiom 16

The duration function assigns a timeduration to every pair of timepoints.

```
(forall (?t1 ?t2)
    (implies  (and    (timepoint ?t1)
                (timepoint ?t2))
        (timeduration (duration ?t1 ?t2))))
```

7.7.17 Axiom 17

Every timeduration is equal to the value of the duration function for some pair of timepoints.

```
(forall (?d)
    (implies  (timeduration ?d)
                 (exists (?t1 ?t2)
                (and (timepoint ?t1)
                     (timepoint ?t2)
                     (= ?d (duration ?t1 ?t2))))))
```

7.7.18 Axiom 18

The value of the duration function is zero if and only if the two timepoints are equal.

```
(forall (?t1 ?t2)
(implies  (and    (timepoint ?t1)
        (timepoint ?t2))
       (iff      (= zero (duration ?t1 ?t2))
        (= ?t1 ?t2)))
```

7.7.19 Axiom 19

The value of the duration function between ?t1 and ?t2 is the additive inverse of the value of the duration function between ?t2 and ?t1.

```
(forall (?t1 ?t2)
(implies  (and    (timepoint ?t1)
        (timepoint ?t2))
       (= zero (add (duration ?t1 ?t2) (duration ?t2 ?t1)))))
```

7.7.20 Axiom 20

Given a timepoint ?t1 other than inf- or inf+, the duration from ?t1to any timepoint is unique.

```
(forall (?t1 ?t2 ?t3)
     (implies  (and    (timepoint ?t1)
              (timepoint ?t2)
              (timepoint ?t3)
              (not (= ?t1 inf-))
              (not (= ?t2 inf+))
              (= (duration ?t1 ?t2) (duration ?t1 ?t3)))
         (= ?t3 ?t2)))
```

7.7.21 Axiom 21

The duration from any point other than inf- to inf- is max- and from any point other than inf+ to inf+ is max+.

```
(forall (?t)
(and          (implies  (and          (timepoint ?t)
```

```
            (not (= ?t inf-)))
        (= max+ (duration inf- ?t)))
    (implies  (and    (timepoint ?t)
            (not (= ?t inf+)))
        (= max- (duration inf+ ?t)))
```

7.7.22 Axiom 22

The duration from inf- to any point other than inf- is max+ and from inf+ to any point other than inf+ is max-.

```
(forall (?t)
(and        (implies  (and          (timepoint ?t)
            (not (= ?t inf-)))
        (= max- (duration ?t inf-)))
    (implies  (and    (timepoint ?t)
            (not (= ?t inf+)))
        (= max+ (duration ?t inf+)))
```

8 Occurrence tree automorphisms

The Occurrence tree automorphism core theory axiomatizes the intuitions about occurrence constraints in which the legal occurrences of one activity depend on the occurrences of other activities.

8.1 Primitive Relations in the Occurrence tree automorphism theory

The nonlogical lexicon of Occurrence tree automorphisms contains one primitive relation symbol

— ubiquitous.

8.2 Defined Relations in the Occurrence tree automorphism theory

The nonlogical lexicon of Occurrence tree automorphisms contains three defined relation symbols:

— end_iso;

— legal_map;

— tree_map.

8.3 Relationship to other sets of axioms

The core theories required are :

— pslcore.th;

— occtree.th;

— atomic.th;

— complex.th;

— actocc.th.

The definitional extensions required is:

— occ_precond.def.

8.4 Informal semantics of the Occurrence tree automorphism theory

8.4.1 ubiquitous

KIF notation for ubiquitous :

(ubiquitous ??a1 ?a2)

Informal semantics for ubiquitous:
(ubiquitous ?a1 ?a2) is TRUE in an interpretation of the Occurrence tree automorphism theory if and only if the activity occurrences that are preserved by occurrence tree automorphisms for ?a1 are subactivity occurrences of occurrences of ?a2.

8.4.2 end_iso

KIF notation for end_iso :

(end_iso ?s1 ?s2 ?s3 ?s4)

Informal semantics for end_iso :

(end_iso ?s1 ?s2 ?s3 ?s4) is TRUE in an interpretation of the Occurrence tree automorphism theory if and only if the same occurrence tree automorphism that maps ?s1 to ?s2 also maps ?s3 to ?s4.

8.4.3 legal_map

KIF notation for legal_map:

(legal_map ?s3 ?s4 ?a)

Informal semantics for legal_map :

(legal_map ?s3 ?s4 ?a) is TRUE in an interpretation of the Occurrence tree automorphism theory if and only if the occurrence tree automorphism that maps ?s3 to ?s4 preserves legal occurrences of ?a.

8.4.4 tree_map

KIF notation for tree_map :

(tree_map ?s1 ?s2 ?a1 ?a2)

Informal semantics for tree_map :

(end_iso ?s1 ?s2 ?a1 ?a2) is TRUE in an interpretation of the Occurrence tree automorphism theory if and only if any mapping on the legal occurrence tree that preserves legal occurrences of ?a1 and that maps ?s1 to ?s2 also preserves activity trees for ?a2.

8.5 Definitions in the Occurrence tree automorphism theory

The definitions introduced by the Occurrence tree automorphism theory are:

8.5.1 Definition 1

The same occurrence tree automorphism that maps ?s1 to ?s2 also maps ?s3 to ?s4.

```
(forall (?s1 ?s2 ?s3 ?s4) (iff (end_iso ?s1 ?s2 ?s3 ?s4)
(exists (?s5 ?s6)
    (and    (precedes ?s5 ?s1)
            (precedes ?s5 ?s3)
            (precedes ?s6 ?s2)
            (precedes ?s6 ?s4)
            (tree_equiv ?s1 ?s2)
            (tree_equiv ?s3 ?s4)
            (tree_equiv ?s5 ?s6))))
```

8.5.2 Definition 2

The occurrence tree automorphism that maps ?s3 to ?s4 preserves legal occurrences of ?a.

```
(forall (?s ?s3 ?s4) (iff (legal_map ?s3 ?s4 ?a)
```

```
(and    (occurrence_of ?s3 ?a)
        (occurrence_of ?s4 ?a)
        (tree_equiv ?s3 ?s4)
        (forall (?s1)
                (implies  (occurrence_of ?s1 ?a)
                        (exists (?s2)
                                (and    (occurrence_of ?s2 ?a)
                                        (legal ?s2)
                                        (end_iso ?s1 ?s2 ?s3 ?s4))))))))
```

8.5.3 Definition 3

The legal occurrence tree automorphism that preserves legal occurrences of ?a1 and that maps ?s1 to ?s2 also preserves activity trees for ?a2.

```
(forall (?a1 ?a2 ?s1 ?s2) (iff (tree_map ?s1 ?s2 ?a1 ?a2)
(and    (legal_map ?s1 ?s2 ?a1)
        (forall (?s3 ?s4)
                (implies  (min_precedes ?s3 ?s4 ?a)
                        (exists (?s5 ?s6)
                                (and    (min_precedes ?s5 ?s6 ?a2)
                                        (end_iso ?s3 ?s5 ?s1 ?s2)
                                        (end_iso ?s4 ?s6 ?s1 ?s2))))))))
```

8.6 Axioms for the Occurrence tree automorphism theory

The set of axioms in the Occurrence tree automorphism theory is as follows:

8.6.1 Axiom 1

If ?a2 is a ubiquitous activity for ?a1, then the subactivity occurrences in activity trees for ?a2 are preserved by occurrence tree automorphisms that preserve legal occurrences of ?a1.

```
(forall (?a1 ?a2 ?s1 ?s2)
    (implies  (and    (ubiquitous ?a1 ?a2)
                      (min_precedes ?s1 ?s2 ?a2))
        (exists (?s3 ?s4 ?s5 ?s6)
```

```
        (and (tree_equiv ?s1 ?s2)
             (tree_equiv ?s3 ?s4)
             (occurrence ?s5 ?a1)
             (occurrence ?s6 ?a1)
             (legal_equiv ?s5 ?s6)
             (end_iso ?s3 ?s4 ?s5 ?s6)))))
```

8.6.2 Axiom 2

If ?a2 is a ubiquitous activity for ?a1, then any occurrence tree automorphisms that preserve subactivity occurrences in activity trees for ?a2 also preserve legal occurrences of ?a1..

```
(forall (?a1 ?a2 ?s1 ?s2)
   (implies (and (ubiquitous ?a1 ?a2)
                 (min_precedes ?s1 ?s2 ?a2))
      (not (exists (?s3 ?s4)
           (and (occurrence ?s3 ?a1)
                (occcurrence ?s4 ?a1)
                (legal_equiv ?s3 ?s4)
                (end_iso ?s1 ?s2 ?s3 ?s4))))))
```

8.6.3 Axiom 3

Every activity occurrence preserved by occurrence tree automorphisms that preserve legal occurrences of ?a1 are subactivity occurrences of a ubiquitous activity..

```
(forall ?a1 ?a2 ?s1 ?s2 ?s3 ?s4)
   (implies (and (ubiquitous ?a1 ?a2)
                 (occurrence ?s3 ?a1)
                 (occurrence ?s4 ?a1)
                 (legal_equiv ?s3 ?s4)
                 (end_iso ?s1 ?s2 ?s3 ?s4))
      (exists (?o1 ?o2)
           (and (occurrence ?o1 ?a1)
                (occurrence ?o2 ?a1)
                (subactivity_occurrence ?s1 ?o1)
                (subactivity_occurrence ?s2 ?o1)))))
```


9 Activity envelope theory

The core theory of Activity envelopes provides axioms for intuitions about the interaction between occurrences of an activity and the occurrences of external activities. In particular, there may be external activities that must necessarily occur during legal occurrences of an activity, and there may be external activities that are forbidden to occur during legal occurrences of an activity.

9.1 Primitive Relations in the Activity envelope theory

The nonlogical lexicon of the Activity envelope theory contains two primitive relation symbols:

— envelope;

— umbra.

9.2 Defined relation in Activity envelope theory

None

9.3 Relationship to other sets of axioms

The core theories required by the Activity envelope theory are:

— pslcore.th;

— occtree.th;

— subactivity.th;

— atomic.th;

— complex.th;

— actocc.th.

The definitional extensions required by the Activity envelope theory are:

— embedding.def.

9.4 Informal semantics of the Activity envelope theory

9.4.1 envelope

KIF notation for envelope :

(envelope ?a1 ?a2 ?s)

Informal semantics for envelope :

(envelope ?a1 ?a2) is TRUE in an interpretation of the Activity envelope theory if and only if the activity tree for ?a2 with root ?s contains all of the external activity occurrences that must necessarily occur for legal occurrences of ?a1.

9.4.2 umbra

KIF notation for umbra :

(umbra ?a1 ?a2)

Informal semantics for umbra :

(umbra ?a1 ?a2) is TRUE in an interpretation of the Activity envelope theory if and only if the activity tree for ?a2 with root ?s contains all of the external activity occurrences that are forbidden to occur in legal occurrences of ?a1.

9.5 Definitions in activity envelop theory

None

9.6 Axioms of the Activity envelope theory

The set of axioms in the Activity envelope theory is as follows:

9.6.1 Axiom 1

Every activity tree has an envelope activity.

```
(forall (?a2 ?s)
    (implies  (root ?s ?a2)
        (exists (?a1)
              (envelope ?a1 ?a2 ?s))))
```

9.6.2 Axiom 2

Every activity tree has an umbra activity.

```
(forall (?a2 ?s)
    (implies  (root ?s ?a2)
        (exists (?a1)
              (umbra ?a1 ?a2 ?s))))
```


9.6.3 Axiom 3

Any branch that is a live branch of the envelope is not a dead branch of the original activity.

```
(forall (?a1 ?a2 ?s ?s1)
    (implies  (and   (envelope ?a1 ?a2 ?s)
                     (live_branch ?s1 ?s ?a1))
        (not (dead_branch ?s1 ?s ?a2))))
```

9.6.4 Axiom 4

Any branch that is a live branch of the umbra is not a live branch of the original activity.

```
(forall (?a1 ?a2 ?s ?s1)
    (implies  (and   (umbra ?a1 ?a2 ?s)
                     (live_branch ?s1 ?s ?a1))
        (not (live_branch ?s1 ?s ?a2))))
```

9.6.5 Axiom 5

Any live branch of an activity is a live branch of the envelope.

```
(forall (?a1 ?a2 ?s ?s1)
    (implies  (and   (envelope ?a1 ?a2 ?s)
                     (live_branch ?s1 ?s ?a2))
        (live_branch ?s1 ?s ?a1)))
```

9.6.6 Axiom 6

Any dead branch of an activity is a live branch of the umbra.

```
(forall (?a1 ?a2  ?s ?s1)
    (implies  (and   (umbra ?a1 ?a2 ?s)
                     (dead_branch ?s1 ?s ?a2))
        (live_branch ?s1 ?s ?a1)))
```

9.6.7 Axiom 7

The envelope for ?a1 is unconstrained on the live branches for ?a1.

```
(forall (?a1 ?a2 ?s ?o)
    (implies  (and   (envelope ?a1 ?a2 ?s)
              (occurrence ?o ?a1)
              (root_occ ?s ?o))
        (unconstrained ?o)))
```

9.6.8 Axiom 8

The umbra for ?a1 is unconstrained on the dead branches for ?a1.

```
(forall (?a1 ?a2 ?s ?o)
    (implies  (and   (umbra ?a1 ?a2 ?s)
              (occurrence ?o ?a1)
              (root_occ ?s ?o))
        (unconstrained ?o)))
```

Annex A
(normative)
Use of ASN.1 Identifiers in SC4 standards

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

```
iso standard 18629 part 13 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 18629-1

Annex B
(**informative**)
Example of process description using ISO 18629-13

The purpose of this annex is to provide a detailed scenario in which the ISO 18629 PSL is used in a knowledge-sharing effort which involves multiple manufacturing functions.

This scenario is an "interoperability" manufacturing scenario. This means that its goal is to show how PSL can be used to facilitate the communication of process knowledge in a manufacturing environment. Specifically, this scenario is centred around the exchange of knowledge from a process planner to a job shop scheduler.

This annex extends the test case introduced in ISO 18629-11: 2005, Annex C to illustrate the application of some duration and ordering concepts in the specification of the manufacturing process of a product named GT-350.

B.1 GT-350 Manufacturing Processes

This section unites the various departmental processes into a high-level collection of activities which are enacted to create a GT-350 product. As described in the GT-350 product structure (see ISO 18629-11: 2005, table C.1), subcomponents of this product are either purchased, sub-contracted, or made internally. These process descriptions address the activities performed to manufacture the internal subcomponents. This top-down view of the manufacturing process provides an overall picture from an abstract, “make GT350” activity which is expanded down to the detailed departmental levels.

As the Figure B.1 below shows, the GT-350 manufacturing process is divided into 6 main areas of work. The first five: make interior, make drive, make trim, make engine and make chassis are all unordered with respect to each other but they must all be completed before final assembly takes place.

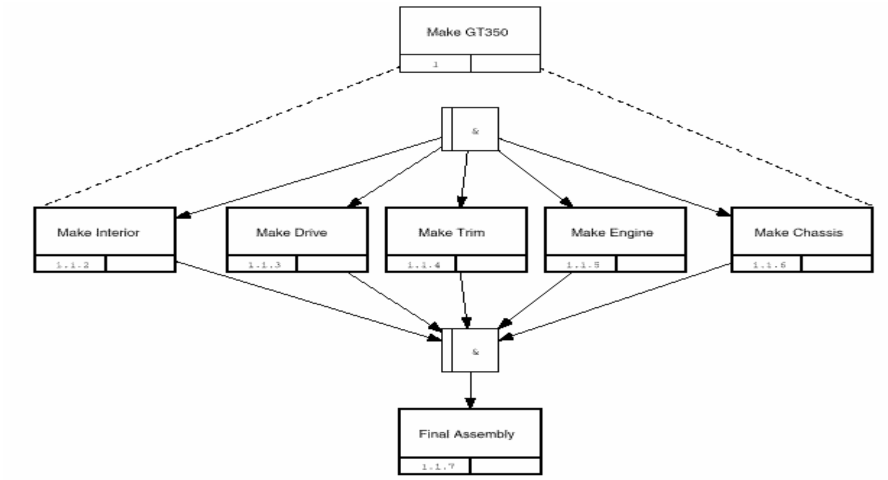

Figure B1: TOP level process for manufacturing a GT350 [8]

The PSL-duration and ordering-based representation of the top level process extends the outercore-based representation in the following way :

(subactivity make-chassis make_gt350)

(subactivity make-interior make_gt350)

(subactivity make-drive make_gt350)

(subactivity make-trim make_gt350)

(subactivity make-engine make_gt350)

(subactivity final-assembly make_gt350)

(forall (?occ)

(⇔ (occurrence_of ?occ make_gt350)

(exists (?occ1 ?occ2 ?occ3 ?occ4 ?occ5 ?occ6)

(and (occurrence_of ?occ1 make_chassis)

(occurrence_of ?occ2 make_interior)

(occurrence_of ?occ3 make_drive)

(occurrence_of ?occ4 make_trim)

(occurrence_of ?occ5 make_engine)

(occurrence_of ?occ6 final_assembly)

(subactivity_occurrence ?occ1 ?occ)

(subactivity_occurrence ?occ2 ?occ)
(subactivity_occurrence ?occ3 ?occ)
(subactivity_occurrence ?occ4 ?occ)
(subactivity_occurrence ?occ5 ?occ)
(subactivity_occurrence ?occ6 ?occ)
(soo_precedes (soomap ?occ1) (soomap ?occ6) make_gt350)
(soo_precedes (soomap ?occ2) (soomap ?occ6) make_gt350)
(soo_precedes (soomap ?occ3) (soomap ?occ6) make_gt350)
(soo_precedes (soomap ?occ4) (soomap ?occ6) make_gt350)
(soo_precedes (soomap ?occ5) (soomap ?occ6) make_gt350)))))))

In this representation, the soo_precedes relation is used to specify the ordering constraints among the subactivity occurrences of make_chassis, make_interior, make_drive, make_engine, and final_assembly. Informally, each arrow in Figure B.1 corresponds to a soo_precedes formula. The soomap function is used to distinguish among any possible multiple occurrences of the subactivities.

Each of these abstract activities can be further detailed, however for the example proposed in this annex, we will not develop all of them.

On the basis of the IDEF3 representation (in terms of process representation) of the abstract activities met during the different stages of the manufacturing process, we will extract some examples of process descriptions using the PSL-duration and ordering presented in this part of ISO 18629.

B.1.1 The "make-engine" abstract activity

The 350-Engine is assembled from work performed in several CMW departments. The manufacturing process is shown in Figure B.2. The part is made up of an engine block, a harness, and wiring. The sub-processes are detailed in the sub-sections below. The 350-Engine is assembled at the A004 assembly bench and takes 5 minutes per piece.

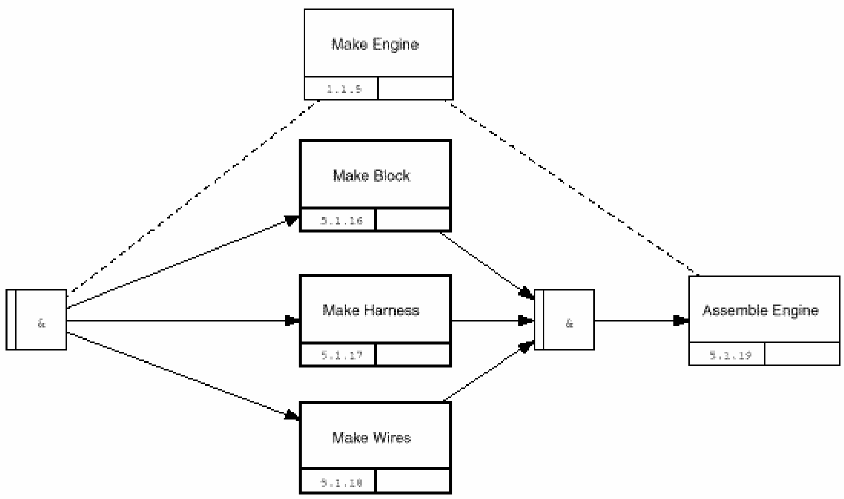

Figure B.2: PROCESS for manufacturing the 350–Engine [8]

The PSL-duration and ordering-based representation of some activities and of the process related information at the make-engine stage is :

(subactivity make_block make_engine)

(subactivity make-harness make_engine)

(subactivity make-wires make_engine)

(subactivity assemble_engine make_engine)

(forall (?occ)

(⇔(occurrence_of ?occ make_engine)

(exists (?occ1 ?occ2 ?occ3 ?occ4)

(and (occurrence_of ?occ1 make_block)

(occurrence_of ?occ2 make_harness)

(occurrence_of ?occ3 make_wires)

(occurrence_of ?occ4 assemble_engine)

(= (duration (beginof ?occ1) (endof ?occ1)) 10)

(= (duration (beginof ?occ2) (endof ?occ2)) 5)

(= (duration (beginof ?occ3) (endof ?occ3)) 12)

(= (beginof ?occ4) (time_add (endof ?occ3) 10))

(subactivity_occurrence ?occ1 ?occ)

(subactivity_occurrence ?occ2 ?occ)

(subactivity_occurrence ?occ3 ?occ)

(subactivity_occurrence ?occ4 ?occ)

(soo_precedes (soomap ?occ1) (soomap ?occ4) make_gt350)

(soo_precedes (soomap ?occ2) (soomap ?occ4) make_gt350)

(soo_precedes (soomap ?occ3) (soomap ?occ4) make_gt350)))))

In this representation, the soo_precedes relation is used to specify the ordering constraints among the subactivity occurrences of make_block, make_harness, make_wires, and make_engine. Informally, each arrow in Figure B.2 corresponds to a soo_precedes formula. The soomap function is used to distinguish among any possible multiple occurrences of the subactivities.

In addition, the representation specifies that the duration of the occurrence of make_block is 10 units, the duration of the occurrence of make_harness is 5 units, and the duration of make_wires is 12 units,

Finally, the representation uses the time_add function to specify that the occurrence of assemble_engine begins 10 time units after the occurrence of make_wires.

B.1.2 Make Block

The 350-Block is manufactured as part of the 350–Engine sub-assembly. This involves an integration of work from the foundry and machine shop, as shown in the Figure B.3.

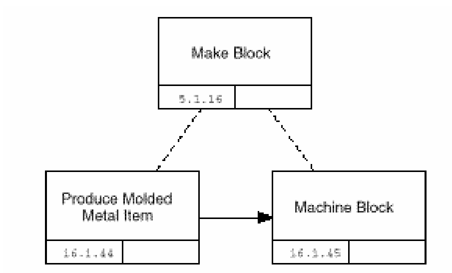

Figure B.3: PROCESS for manufacturing the 350–Block [8]

The PSL-duration and ordering-based representation of some activities and of the process related information is :

(subactivity produce_molded_metal make_block)

(subactivity machine_block make_block)

(primitive machine_block)

(primitive produce_molded_metal)

(forall (?occ)
(⇔ (occurrence_of ?occ make_block)
(exists (?occ1 ?occ2)
(and (occurrence_of ?occ1 produce_molded_metal)
(occurrence_of ?occ2 machine_block)
(= (beginof ?occ2) (time_add (endof ?occ1) 12))
(soo_precedes (soomap ?occ1) (soomap ?occ2) make_block)))))

This representation uses the time_add function to specify that the occurrence of the machine_block subactivity begins 12 time units after the end of the occurrence of the produce_molded_metal subactivity.

B.1.3 Make Harness

The 350-Harness (Figure B.4) is manufactured as part of the 350–Engine sub-assembly. This involves work performed at the wire and cable department. Figure B.5 expands the harness wire production process. The 350-Harness is assembled by a bench worker at a wire and cable bench. It takes 10 minutes per set.

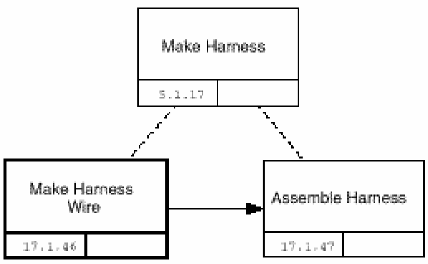

Figure B.4: PROCESS for manufacturing the 350–Harness [8]

The PSL-duration and ordering-based representation of some activities and of the process related information is :

(subactivity make_harness_wire make_harness)

(subactivity assemble_harness make_harness)

(primitive assemble_harness)

(forall (?occ)

(⇔ (occurrence_of ?occ make_harness)

(exists (?occ1 ?occ2 ?occ3)

(and (occurrence_of ?occ1 make_harness_wire)

(occurrence_of ?occ2 assemble_harness)

(leaf_occ ?occ3 ?occ1)

(soo_precedes (soomap ?occ3) (soomap ?occ2) make_harness)))))

This representation formalizes the specification of the process in Figure B.5. In this representation, the soo_precedes relation is used to specify the ordering constraints among the subactivity occurrences of make_harness_wires and assemlbe_wires. Informally, each arrow in Figure B.5 corresponds to a soo_precedes formula. The soomap function is used to distinguish among any possible multiple occurrences of the subactivities.

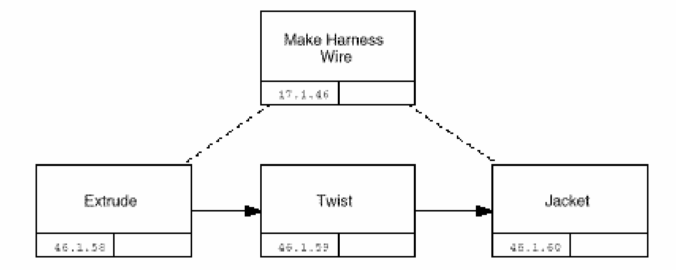

Figure B.5: PROCESS for manufacturing the harness wire [8]

B.1.4 Make Harness Wires

The 350-Wire-Set is manufactured as part of the 350–Engine sub-assembly. This involves work performed at the wire and cable department.

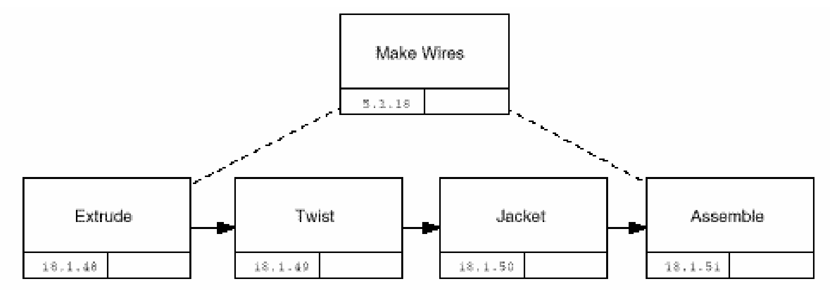

Figure B.6 : Process for manufacturing the 350-Wire [8]

The PSL-duration and ordering-based representation of some activities and of the process related information is :

(subactivity extrude make_harness_wire)

(subactivity twist make_harness_wire)

(subactivity jacket make_harness_wire)

(primitive extrude)

(primitive twist)

(primitive jacket)

```
(forall (?occ)
    (⇔      (occurrence_of ?occ make_harness_wire)
            (exists (?occ1 ?occ2 ?occ3 ?occ4)
                  (and    (occurrence_of ?occ1 extrude)
                          (occurrence_of ?occ2 twist)
                          (occurrence_of ?occ3 jacket)
                          (occurrence_of ?occ4 assemble)
                          (soo_precedes (soomap ?occ1) (soomap ?occ2) make_harness_wire)
                          (soo_precedes (soomap ?occ3) (soomap ?occ4) make_harness_wire)
                          (soo_precedes (soomap ?occ2) (soomap ?occ3) make_harness_wire))))
```

This representation formalizes the specification of the process in Figure B.6. In this representation, the soo_precedes relation is used to specify the ordering constraints among the subactivity occurrences of extrude, twist, jacket, and assemble. Informally, each arrow in Figure B.6 corresponds to a soo_precedes formula. The soomap function is used to distinguish among any possible multiple occurrences of the subactivities.

Bibliography

[1] ISO 10303-11, *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual*

[2] ISO 10303-41, *Industrial automation systems and integration — Product data representation and exchange — Part 41: Integrated generic resource: Fundamentals of product description and support*

[3] ISO 10303-42, *Industrial automation systems and integration — Product data representation and exchange— Part 42: Integrated generic resource: Geometric and topological representation*

[4] ISO 10303-49, *Industrial automation systems and integration — Product data representation and exchange— Part 49 : Integrated generic resource: Process structure and properties*

[5] ISO 13584 (all parts), *Industrial automation systems and integration — Parts library*

[6] ISO 15531 (all parts), *Industrial automation systems and integration — Industrial manufacturing management data*

[7] ISO 15926 (all parts), *Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities*

[8] *Federal Information Processing Standards Publication 184, Integration Definition for Information Modeling (IDEF3)*, FIPS PUB 184, National Institute of Standards and Technology, December 1993. IDEF3. Available from the Internet: <http://www.idef.com >

Index

ICS 25.040.40

Price based on 39 pages