ISO/DIS PRF 19450:2022(E)

Date: 2022-10

N/A

ISO/TC 184/SC 5/WG 1

Secretariat: ANSI

Date: 2023-10-10

Automation systems and integration — Object-Process Methodology

Systèmes d'automatisation et intégration — Méthodologie du processus-objet — Object-Process Methodology

Copyright notice

ThisFDIS stage

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Field Code Changed

Attention is drawnISO draws attention to the possibility that some of the elementsimplementation of this document may beinvolve the subjectuse of (a) patent(s). ISO takes no position concerning the evidence, validity or applicability of any claimed patent rights. in respect thereof. As of the date of publication of this document, ISO had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.htmlwww.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 184, *Automation systems and integration*, Subcommittee SC 5, *Interoperability, integration, and architectures for enterprise systems and automation applications*.

This first edition cancels and replaces ISO/PAS 19450:2015, which has been technically revised.

The main changes are as follows:

— document designation from PAS to International Standard (this document);

— clarified several defined terms and addadded term cross references;

— Addadded introduction statement for all figures and tables;

— Clarifyclarified use of "may" or "can" as appropriate;

— Correctcorrected identified errors in figures and tables;.

— Update to revisable figures and images where possible.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Field Code Changed

Introduction

Object-Process Methodology (OPM) is a compact conceptual approach, language, and methodology for modelling and knowledge representation of automation systems. The application of OPM ranges from simple assemblies of elemental components to complex, multidisciplinary, dynamic systems. OPM is suitable for implementation and support by tools using information and computer technology. This document specifies both the language and methodology aspects of OPM in order to establish a common basis for system architects, designers, and OPM-compliant tool developers to model all kinds of systems.

OPM provides two semantically equivalent modalities of representation for the same model: graphical and textual. A set of hierarchically structured, interrelated Object-Process-Diagrams (OPDs) constitutes the graphical model, and a set of automatically generated sentences in a subset of the English language constitutes the textual model expressed in the Object-Process Language (OPL). In a graphical-visual model, each OPD consists of OPM elements, depicted as graphical symbols, sometimes with label annotation. The OPD syntax specifies the consistent and correct ways to manage the arrangement of those graphical elements. Using OPL, OPM generates the corresponding textual model for each OPD in a manner that retains the constraints of the graphical model. Since OPL's syntax and semantics are a subset of English natural language, domain experts easily understand the textual model.

OPM notation supports the conceptual modelling of systems with formal syntax and semantics. This formality serves as the basis for model-based systems engineering in general, including systems architecting, engineering, development, life cycle support, communication, and evolution. Furthermore, the domain-independent nature of OPM opens system modelling to the entire scientific, commercial and industrial community for developing, investigating and analysing manufacturing and other industrial and business systems inside their specific application domains~~;~~, thereby enabling companies to merge and provide for interoperability of different skills and competencies into a common intuitive yet formal framework.

OPM facilitates a common view of the system under construction, test, integration, and daily maintenance, providing for working in a multidisciplinary environment. Moreover, using OPM, companies can improve their overall, big-picture view of the system's functionality, flexibility in assignment of personnel to tasks, and managing exceptions and error recovery. System specification is extensible for any necessary detail, encompassing the functional, structural and behavioural aspects of a system.

One particular application of OPM is in the drafting and authoring of technical standards. OPM helps sketch the implementation of a standard and identify weaknesses in the standard to reduce, thereby significantly improving the quality of successive drafts. With OPM, even as the model-based text of a system expands to include more details, the underlying model keeps maintaining its high degree of formality and consistency.

This document provides a baseline for system architects and designers, who can use it to model systems concisely and effectively. OPM tool vendors can utilise this document as a formal standard specification for creating software tools to enhance conceptual modelling.

This document provides a presentation of the normative text that follows the Extended Bachus Naur form (EBNF) specification of the language syntax. All elements are presented in ~~Clause 6 through 13~~Clause 6 to 13 with only minimal reference to methodological aspects, ~~Clause 14~~Clause 14 presents the context management mechanisms related to in-zooming and unfolding.

NOTE OPM is an established modelling paradigm with a 15-year history of use in international commerce. As such, several conventions for its use and presentation already exist in the literature and practice.

This document uses the presentation conventions for the expression of OPM related constructs found in the originating and current literature for OPM. Using a different set of conventions, or simply applying the ~~general~~

ISO/IEC Directives, Part 2 drafting guidelines for these terms and presentations, creates a discontinuity between this document and the supporting references and practice, and can cause confusion in application of this document to existing and future practice.

This document applies the following conventions for the presentation of OPM elements and terminology:

— The phrases and associated abbreviations "Object-Process Methodology (OPM)", "Object-Process Diagram (OPD)" and "Object-Process Language (OPL)" are terms of art associated with the OPM paradigm and appear as specified with the hyphen and capitalization of each word in the phrase.

— In OPD and OPL text, the object name and process name appear in Cambria bold font text with capitalization of each word to distinguish the object and process from the other OPD and OPL text. The same convention for object and process name bold capitalization is carried into the text of this document as well to indicate that in the text, the word or phrase corresponds to an OPM object or process.

— In OPD and OPL text, the object state label and attribute value label appear in Cambria bold font text in lowercase, and somewhat smaller font in OPD figures, to distinguish the object state label and attribute value label from the other OPD and OPL text. The same convention for object state label and attribute value label in bold font lowercase is carried into the text of this document as well to indicate that in the text, the word or phrase corresponds to an OPM object state label or attribute value label.

— In OPD and OPL text, link tags that are not user-specified appear in Cambria lowercase, and somewhat smaller font in OPD. Link tags that are user-specified appear as entered by the user in Cambria bold font text, and somewhat smaller font in OPD.

— In OPL, the first letter of the first word of a sentence is capitalized.

— Some of these conventions are repeated in the text as appropriate to remind the reader of the distinctions. Some OPD figures contain colorcolour to help distinguish OPM modelling element type distinctions.

Most figures contain both a graphical image, the OPD portion, and a textual equivalent, the OPL portion of the figure. Because this is a language specification, the precise use of term definitions is essential and several terms in common use have particular meaning when using OPM. In addition to those listed above as OPM presentation conventions, Annex BAnnex B explains other conventions for the use of OPM.

Annex AAnnex A presents the formal syntax for OPL, in EBNF form.

Annex BAnnex B presents conventions and patterns commonly used in OPM applications.

Annex CAnnex C presents aspects of OPM as OPM models.

Annex DAnnex D summarizes the dynamic and simulation capabilities of OPM.

The International Organization for Standardization (ISO) draws attention to the fact that it is claimed that compliance with this document may involve the use of a patent.

ISO takes no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured ISO IEC that he/she is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with ISO IEC. Information may be obtained from the patent database available at www.iso.org/patents.

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

Automation systems and integration — Object-Process Methodology

1 Scope

This document specifies Object-Process Methodology (OPM) with detail sufficient for enabling practitioners to utilise the concepts, semantics, and syntax of OPM as a modelling paradigm and language for producing conceptual models at various extents of detail, and for enabling tool vendors to provide application modelling products to aid those practitioners.

While this document presents some examples for the use of OPM to improve clarity, it does not attempt to provide a complete reference for all the possible applications of OPM.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

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

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ~~—~~ISO Online browsing platform: available at ~~https://www.iso.org/obp~~https://www.iso.org/obp

— ~~—~~IEC Electropedia: available at ~~https://www.electropedia.org/~~https://www.electropedia.org/

Note 1 to entry ~~:~~To facilitate a term search, terms are in alphabetical sequence.

3.1
abstraction,noun
outcome of an ~~*abstraction process* (3.2)~~*abstractionprocess* (3.2)

3.2
abstraction process
decreasing the extent of detail and system model *completeness* ~~(3.9)~~(3.9) in order to achieve better comprehension

3.3
affectee
transformee ~~(3.79)~~(3.79) that is affected by a *process* ~~(3.59)~~(3.59) occurrence, i.e. its *state* ~~(3.69)~~(3.69) changes

Note 1 to entry: An affectee can only be a stateful object. A stateless object can only be created or consumed, but not affected.

3.4
agent
enabler ~~(3.18)~~(3.18) that is a human or a group of humans

3.5
attribute
object ~~(3.40)~~(3.40) that characterizes a *thing* ~~(3.77)~~(3.77) other than itself

3.6
behaviour
transformation ~~(3.78)~~(3.78) of *objects* ~~(3.40)~~(3.40) resulting from the execution of an *Object-Process Methodology (OPM)* ~~(3.44)~~(3.44) model comprising a collection of *processes* ~~(3.59)~~(3.59) and *links* ~~(3.37)~~(3.37) to *objects* in the model

3.7
beneficiary
<system> stakeholder who gains functional *value* ~~(3.83)~~(3.83) from the system's *operation* ~~(3.47)~~(3.47)

3.8
class
collection of *things* ~~(3.77)~~(3.77) with the same *perseverance* ~~*(3.51)*,~~(3.51), essence, and affiliation valuation, and the same *feature* ~~(3.22)~~(3.22) and *state* ~~(3.69)~~(3.69) set

Note 1 to entry: Perseverance, essence and affiliation are properties of things ~~–~~ (see ~~7.3.3.~~ 7.3.3).

3.9
completeness
<system model> extent to which all the details of a system are specified in a model

3.10
condition link
procedural link ~~(3.57)~~(3.57) from an *object* ~~(3.40)~~(3.40) or object *state* ~~(3.69)~~(3.69) to a *process* ~~(3.59),~~(3.59), denoting a procedural constraint

3.11
consumee
transformee ~~(3.79)~~(3.79) that a *process* ~~(3.59)~~(3.59) occurrence consumes or eliminates

3.12
context
<model> portion of an *Object-Process Methodology (OPM)* ~~(3.44)~~(3.44) model represented by an *Object-Process Diagram (OPD)* ~~(3.42)~~(3.42) and corresponding *Object-Process Language (OPL)* ~~(3.43)~~(3.43) text

3.13
control link
procedural link ~~(3,57)~~(3.57) with additional control semantics

3.14
control modifier
symbol embellishing a *link* ~~(3.37)~~(3.37) to add control semantics to the *link*, making it a *control link* ~~(3.13)~~(3.13)

Note 1 to entry: The control modifiers are the symbols 'e' for event and 'c' for condition.

3.15
discriminating attribute
attribute ~~(3.5)~~(3.5) whose different *values* ~~(3.82)~~(3.82) identify corresponding specialization relations

3.16
effect
change in the *state* ~~(3.69)~~(3.69) of an *object* ~~(3.40)~~(3.40) or the *value* ~~(3.82)~~(3.82) of an *attribute* ~~(3.5)~~(3.5)

Note 1 to entry: An effect only applies to a stateful object.

3.17
element
thing ~~(3.77)~~(3.77) or *link* ~~(3.37)~~(3.37)

3.18
enabler
<process> *object* ~~(3.40)~~(3.40) that enables a *process* ~~(3.59)~~(3.59) but which the *process* does not transform

3.19
event
<OPM> point in time of creation (or appearance) of an *object* ~~(3.40),~~(3.40), or entrance of an *object* to a particular *state* ~~(3.69),~~(3.69), initiating an evaluation of the *precondition* ~~(3.54)~~(3.54)

3.20
event link
control link ~~(3.13)~~(3.13) denoting an *event* ~~(3.19)~~(3.19) originating from an *object* ~~(3.40)~~(3.40) or object *state* ~~(3.69)~~(3.69) to a *process* ~~(3.59)~~(3.59)

3.21
exhibitor
thing ~~(3.77)~~(3.77) that exhibits (is characterized by) a *feature* ~~(3.22)~~(3.22) by means of the exhibition-characterization relation

3.22
feature
attribute ~~(3.5)~~(3.5) or *operation* ~~(3.47)~~(3.47)

3.23
folding
abstraction ~~(3.1)~~(3.1) achieved by hiding the *refineables* ~~(3.62)~~(3.62) of a *refinee* ~~(3.63)~~(3.63) to which *unfolding* ~~(3.81)~~(3.81) applies

Note 1 to entry: The four kinds of folded refineables are parts (part folding), features (feature folding), specializations (specialization folding), and instances (instance folding).

Note 2 to entry: Folding is primarily applied to objects. When applied to a process, its subprocesses are unordered, which is adequate for modelling asynchronous systems, in which processes' temporal order is undefined.

Note 3 to entry: The opposite of folding is unfolding.

3.24
function
process ~~(3.59)~~(3.59) that provides functional *value* ~~(3.83)~~(3.83) to a *beneficiary* ~~(3.7)~~(3.7)

3.25
general,noun
<OPM> *refineable* ~~(3.62)~~(3.62) with specializations

3.26
informatical,adj.
of, or pertaining to informatics, e.g~~.,~~., data, information, knowledge

3.27
inheritance
assignment of *Object-Process Methodology (OPM)* ~~(3.44)~~ (3.44)*elements* ~~(3.17)~~(3.17) of a *general* ~~(3.25)~~(3.25) to its specializations

3.28
input link
link ~~(3.37)~~(3.37) from *object* ~~(3.40)~~(3.40) source (input) *state* ~~(3.69)~~(3.69) to the transforming *process* ~~(3.59)~~(3.59)

3.29
instance
<model> modelled *thing* ~~(3.77)~~(3.77) that is a *refinee* ~~(3.63)~~(3.63) in a classification-instantiation relation

3.30
instance
<operational> actual, uniquely identifiable *thing* ~~(3.77)~~(3.77) that exists during model execution, e.g. during simulation or runtime implementation

Note 1 to entry: A process instance is identifiable by the operational instances of the involved object set during process occurrence and the process start and end time stamps of the occurrence.

3.31
instrument
non-human *enabler* ~~(3.18)~~(3.18)

3.32
invocation
<process> initiating of a *process* ~~(3.59)~~(3.59) by a *process*

3.33
involved object set
union of *preprocess object set* ~~(3.55)~~(3.55) and *postprocess object set* ~~(3.53)~~(3.53)

3.34
in-zoom context
things ~~(3.77)~~(3.77) and *links* ~~(3.37)~~(3.37) within the boundary of the *thing* to which *object* ~~(3.40)~~ (3.40)*in-zooming* ~~(3.35)~~(3.35) or *process* ~~(3.59)~~ (3.59)*in-zooming* ~~(3.36)~~(3.36) applies

3.35
in-zooming
<object> *object* ~~(3.40)~~(3.40) part *unfolding* ~~(3.81)~~(3.81) that indicates spatial ordering of the constituent *objects*

3.36
in-zooming
<process> *process* ~~(3.59)~~(3.59) part *unfolding* ~~(3.81)~~(3.81) that indicates temporal partial ordering of the constituent *processes*

3.37
link
graphical expression of a ~~*structural relation* (3.74)~~*structuralrelation* (3.74) or a *procedural relation* ~~(3.58)~~(3.58) between two *Object-Process Methodology (OPM)* ~~(3.44)~~ (3.44)*things* ~~(3.77)~~(3.77)

3.38
metamodel
model of a modelling language or part of a modelling language

3.39
model fact
relation between two *Object-Process Methodology (OPM)* ~~(3.44) *things* (3.77) or *states* (3.69)~~(3.44)*things* (3.77) or *states* (3.69) in the *OPM* model

3.40
object
<OPM> model *element* ~~(3.17)~~(3.17) representing something that does or can exist physically or *informatically* ~~(3.26)~~(3.26)

3.41
object class
pattern for *objects* ~~(3.40)~~(3.40) that have the same *structure* ~~(3.75)~~(3.75) and pattern of *transformation* ~~(3.78)~~(3.78)

3.42
Object-Process Diagram
OPD
Object-Process Methodology (OPM) ~~*(3.44)*~~(3.44) graphic representation of an *OPM* model or part of a model, in which *objects* ~~(3.40)~~(3.40) and *processes* ~~(3.59)~~(3.59) in the universe of interest appear together with the *structural links* ~~(3.73)~~(3.73) and *procedural links* ~~(3.57)~~(3.57) among them

3.43
Object-Process Language
OPL
subset of English natural language that represents textually the *Object-Process Methodology(OPM)* ~~*(3.44)*~~(3.44) model that the *Object-Process Diagram(OPD)* ~~(3.42)~~(3.42) represents graphically

3.44
Object-Process Methodology
OPM
formal language and method for specifying complex, multidisciplinary systems in a single function-structure-behaviour unifying model that uses a bimodal graphic-text representation of *objects* ~~(3.40)~~(3.40) in the system and their *transformation* ~~(3.78)~~(3.78) or use by *processes* ~~(3.59)~~(3.59)

3.45
Object-Process Diagram object tree
OPD object tree
tree graph, whose root is an *object* ~~(3.40),~~(3.40), depicting elaboration of the *object* through *refinement* ~~(3.64)~~(3.64)

3.46
Object-Process Diagram process tree
OPD process tree
tree graph whose root is the *System Diagram (SD)* ~~(3.76)~~(3.76) and each node is an *Object-Process Diagram (OPD)* ~~(3.42)~~ (3.42) obtained by *in-zooming* ~~(3.36)~~(3.36) of a *process* ~~(3.59)~~(3.59) in its ancestor *OPD* (or the *SD*) and for which each directed edge connected to the ancestor OPD *process* points to the same *process* in the child *OPD*

Note 1 to entry: OPM model elaboration usually occurs by process decomposition through in-zooming, therefore the OPD process tree is the primary way to navigate an OPM model.

3.47
operation
process ~~(3.59)~~(3.59) that a *thing* ~~(3.77)~~(3.77) performs, which characterizes the *thing* other than itself

3.48
output link
link ~~(3.37)~~(3.37) from the transforming *process* ~~(3.59)~~(3.59) to the output (destination) *state* ~~(3.69)~~(3.69) of an *object* ~~(3.40)~~(3.40)

3.49
out-zooming
<object> inverse of *object* ~~(3.40)~~ (3.40)*in-zooming* ~~(3.35)~~(3.35)

3.50
out-zooming
<process> inverse of *process* ~~(3.59)~~ (3.59)*in-zooming* ~~(3.36)~~(3.36)

3.51
perseverance
property ~~(3.61)~~(3.61) of *thing* ~~(3.77)~~(3.77) which can be static, defining an *object* ~~(3.40),~~(3.40), or dynamic, defining a *process* ~~(3.59)~~(3.59)

3.52
postcondition
<process> condition that is the outcome of successful *process* ~~(3.59)~~(3.59) completion

