PUBLICLY AVAILABLE SPECIFICATION

ISO/PAS 19450

First edition
2015-12-15

Automation systems and integration — Object-Process Methodology

Systèmes d'automatisation et intégration — Object-Process Methodology

Reference number
ISO/PAS 19450:2015(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

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).

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. 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 on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is Technical Committee ISO/TC 184, *Automation systems and integration*, Subcommittee SC 5, *Interoperability, integration, and architectures for enterprise systems and automation applications*.

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 Publicly Available Specification 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 graphic symbols, sometimes with label annotation. The OPD syntax specifies the consistent and correct ways to manage the arrangement of those graphically elements. Using OPL, OPM generates the corresponding textual model for each OPD in a manner that retains the constraints of the graphical model. Since the syntax and semantics of OPL 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 Publicly Available Specification provides a baseline for system architects and designers, who can use it to model systems concisely and effectively. OPM tool vendors can utilise the PAS as a formal standard specification for creating software tools to enhance conceptual modelling.

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

This specification utilizes several conventions for the presentation of OPM. Specifically, Arial bold font in text and Arial bold italic font in figure captions, table captions and headings distinguish label names for OPM objects, processes, states, and link tags. OPL reserved words are in Arial regular font with commas and periods in Arial bold font. Most figures contain both a graphic image, the OPD portion, and a textual equivalent, the OPL portion. 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. Clause B.6 explains other conventions for the use of OPM.

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

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

Annex C presents aspects of OPM as OPM models.

Annex 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 concerning OPM as a modelling system given in Clauses 6 to 14.

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

The holder of this patent right has assured the ISO that he/she is willing to negotiate licences either free of charge or 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. Information may be obtained from:

Prof. Dov Dori

Technion Israel Institute of Technology

Technion City

Haifa 32000, Israel

dori@ie.technion.ac.il

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

ISO (www.iso.org/patents) and IEC (http://patents.iec.ch) maintain on-line databases of patents relevant to their standards. Users are encouraged to consult the databases for the most up to date information concerning patents.

Automation systems and integration — Object-Process Methodology

1 Scope

This Publicly Available Specification specifies Object-Process Methodology (OPM) with detail sufficient for enabling practitioners to utilise the concepts, semantics, and syntax of Object-Process Methodology 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 Publicly Available Specification presents some examples for the use of Object-Process Methodology to improve clarity, it does not attempt to provide a complete reference for all the possible applications of Object-Process Methodology.

2 Normative references

There are no normative references.

3 Terms and definitions

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

3.1
abstraction
decreasing the extent of detail and system model *completeness* (3.8) in order to achieve better comprehension

3.2
affectee
transformee (3.78) that is affected by a *process* (3.58) occurrence, i.e. its *state* (3.69) changes

Note 1 to entry: An affectee can only be a *stateful object* (3.66). A *stateless object* (3.67) can only be created or consumed, but not affected.

3.3
agent
enabler (3.17) that is a human or a group of humans

3.4
attribute
object (3.39) that characterizes a *thing* (3.76) other than itself

3.5
behaviour
transformation (3.77) of *objects* (3.39) resulting from the execution of an *Object-Process Methodology* (3.43) model comprising a collection of *things* (3.76) and *links* (3.36) to objects in the model

3.6
beneficiary
<system> *stakeholder* (3.65) who gains functional *value* (3.82) from the system's *operation* (3.46)

3.7
class
collection of *things* (3.76) with the same *perseverance* (3.50), essence, and affiliation values, and the same *feature* (3.21) and *state* (3.69) set

3.8
completeness
<system model> extent to which all the details of a system are specified in a model

3.9
condition link
procedural link (3.56) from an *object* (3.39) or object *state* (3.69) to a *process* (3.58), denoting a procedural constraint

3.10
consumee
transformee (3.78) that a *process* (3.58) occurrence consumes or eliminates

3.11
context
<model> portion of an *Object-Process Methodology* (3.43) model represented by an *Object-Process Diagram* (3.41) and corresponding *Object-Process Language* (3.42) text

3.12
control link
procedural link (3.56) with additional control semantics

3.13
control modifier
symbol embellishing a *link* (3.36) to add control semantics to it, making it a *control link* (3.12)

Note 1 to entry: The control modifiers are the symbols ‘e’ for *event* (3.18) and ‘c’ for condition.

3.14
discriminating attribute
attribute (3.4) whose different *values* (3.81) identify corresponding specialization relations

3.15
effect
change in the *state* (3.69) of an *object* (3.39) or an *attribute* (3.4) *value* (3.81)

Note 1 to entry: An effect only applies to a *stateful object* (3.66).

3.16
element
thing (3.76) or *link* (3.36)

3.17
enabler
<process> *object* (3.39) that enables a *process* (3.58) but which the process does not *transform*

3.18
event
<OPM> point in time of creation (or appearance) of an object, or entrance of an object (3.39) to a particular *state* (3.69), either of which may initiate an evaluation of the *process* (3.58) *precondition* (3.53)

3.19
event link
control link (3.12) denoting an *event* (3.18) originating from an *object* (3.39) or object *state* (3.69) to a *process* (3.58)

3.20
exhibitor
thing (3.76) that exhibits (is characterized by) a *feature* (3.21) by means of the exhibition-characterization relation

3.21
feature
attribute (3.4) or *operation* (3.46)

3.22
folding
mechanism of *abstraction* (3.1) achieved by hiding the *refineables* (3.61) of an unfolded *refinee* (3.62)

Note 1 to entry: The four kinds of folded refineables are parts (part folding), *features* (3.21) (feature folding), specializations (specialization folding), and *instances* (3.28) (instance folding).

Note 2 to entry: Folding is primarily applied to *objects* (3.39). 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* (3.80).

3.23
function
process (3.58) that provides functional *value* (3.82) to a *beneficiary* (3.6)

3.24
general
<OPM> *refineable* (3.61) with specializations

3.25
informatical
of, or pertaining to informatics, e.g. data, information, knowledge

3.26
inheritance
assignment of *Object-Process Methodology* (3.43) *elements* (3.16) of a *general* (3.24) to its specializations

3.27
input link
link (3.36) from *object* (3.39) source (input) *state* (3.69) to the transforming *process* (3.58)

3.28
instance
<model> *object* (3.39) instance or *process* (3.58) instance that is a *refinee* (3.62) in a classification-instantiation relation

3.29
instance
<*operational*> *object* (3.39) instance or *process* (3.58) instance that is an actual, uniquely identifiable *thing* (3.76) that exists during model *operation* (3.46), 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* (3.32) during process occurrence and the process start and end time stamps of the occurrence.

3.30
instrument
non-human *enabler* (3.17)

3.31
invocation
<process> initiating of a *process* (3.58) by a process

3.32
involved object set
union of *preprocess object set* (3.54) and *postprocess object set* (3.52)

3.33
in-zoom context
things (3.76) and *links* (3.36) within the boundary of the thing being *in-zoomed*

3.34
in-zooming
<object> *object* (3.39) part *unfolding* (3.80) that indicates spatial ordering of the constituent objects

3.35
in-zooming
<process> *process* (3.58) part *unfolding* (3.80) that indicates temporal partial ordering of the constituent processes

3.36
link
graphical expression of a *structural relation* (3.73) or a *procedural relation* (3.57) between two *Object-Process Methodology* (3.43) *things* (3.76)

3.37
metamodel
model of a modelling language or part of a modelling language

3.38
model fact
relation between two *Object-Process Methodology* (3.43) *things* (3.76) or *states* (3.69) in the Object-Process Methodology model

3.39
object
<OPM> model *element* (3.16) representing a *thing* (3.76) that does or might exist physically or *informatically* (3.25)

3.40
object class
pattern for *objects* (3.39) that have the same *structure* (3.74) and pattern of *transformation* (3.77)

3.41
Object-Process Diagram
OPD
Object-Process Methodology (3.43) graphic representation of an Object-Process Methodology model or part of a model, in which *objects* (3.39) and *processes* (3.58) in the universe of interest appear together with the *structural links* (3.72) and *procedural links* (3.56) among them

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

3.43
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* (3.39) in the system and their *transformation* (3.77) or use by *processes* (3.58)

3.44
OPD object tree
tree graph, whose root is an *object* (3.39), depicting elaboration of the object through *refinement* (3.63)

3.45
OPD process tree
tree graph whose root is the *System Diagram* (3.75) and each node is an *Object-Process Diagram* (3.42) obtained by in-*zooming* (3.35) of a *process* (3.58) in its ancestor Object-Process Diagram (or the System Diagram) and each directed edge points from the refined process at the parent Object-Process Diagram to the same process in the child Object-Process Diagram

Note 1 to entry: *Object-Process Methodology* (3.43) model elaboration usually occurs by process decomposition through in-zooming, therefore the OPD process tree is the primary way to navigate an Object-Process Methodology model.

3.46
operation
process (3.58) that a *thing* (3.76) performs, which characterizes the thing other than itself

3.47
output link
link (3.36) from the transforming *process* (3.58) to the output (destination) *state* (3.69) of an *object* (3.39)

3.48
out-zooming
<object> inverse of *object* (3.39) *in-zooming* (3.34)

3.49
out-zooming
<process> inverse of *process* (3.58) *in-zooming* (3.35)

3.50
perseverance
property (3.60) of *thing* (3.76) which can be static, defining an *object* (3.39), or dynamic, defining a *process* (3.58)

3.51
postcondition
<process> condition that is the outcome of successful *process* (3.58) completion

3.52
postprocess object set
collection of *objects* (3.39) remaining or resulting from *process* (3.58) completion

Note 1 to entry: The postprocess object set may include *stateful objects* (3.66), for which specific *states* (3.69) result from process performance.

3.53
precondition
<process> condition for starting a *process* (3.58)

3.54
preprocess object set
collection of *objects* (3.39) to evaluate prior to starting a *process* (3.58)

Note 1 to entry: The collection of the objects may include *stateful objects* (3.66) for which specific *states* (3.69) are necessary for process performance.

3.55
primary essence
<system> *essence* of the majority of *things* (3.76) in a system, which can be either *informatical* (3.25) or physical

3.56
procedural link
graphical notation of *procedural relation* (3.57) in *Object-Process Methodology* (3.43)

3.57
procedural relation
connection or association between an *object* (3.39) or object *state* (3.69) and a *process* (3.58)

Note 1 to entry: Procedural relations specify how the system operates to attain its *function* (3.23), designating time-dependent or conditional initiating of processes that transform objects.

Note 2 to entry: An *invocation* (3.31) or exception *link* (3.36) signifies a transient object in the flow of execution control between two processes.

3.58
process
transformation (3.77) of one or more *objects* (3.39) in the system

3.59
process class
pattern for *processes* (3.58) that perform the same *object* (3.39) *transformation* (3.77) pattern

3.60
property
modelling annotation common to all *elements* (3.16) of a specific kind that serve to distinguish that *element*

Note 1 to entry: Cardinality constraints, path labels, and *structural link* (3.72) tags are frequent property annotations.

Note 2 to entry: Unlike an *attribute* (3.4), the value of a property may not change during model simulation or operational implementation. Each kind of element has its own set of properties.

Note 3 to entry: Property is an attribute of an element in the *Object-Process Methodology* (3.43) *metamodel* (3.37).

3.61
refineable
<OPM> *thing* (3.76) amenable to *refinement* (3.63), which can be a *whole* (3.83), an *exhibitor* (3.20), a *general* (3.24), or a *class* (3.7)

3.62
refinee
thing (3.76) that refines a *refineable* (3.61), which can be a part, a *feature* (3.21), a specialization, or an instance (3.29)

Note 1 to entry: Each of the four kinds of refinees has a corresponding refineable (part-whole, feature-exhibitor, specialization-generalization, instance-class).

3.63
refinement
<model> elaboration that increases the extent of detail and the consequent model *completeness* (3.8)

3.64
resultee
transformee (3.78) that a *process* (3.58) occurrence creates

3.65
stakeholder
<OPM> individual, organization, or group of people that has an interest in, or might be affected by the system being contemplated, developed, or deployed

3.66
stateful object
object (3.39) with specified *states* (3.69)

3.67
stateless object
object (3.39) lacking specified *states* (3.69)

3.68
state
<object> possible situation or position of an *object* (3.39)

Note 1 to entry: In *Object-Process Methodology* (3.43) there is no concept of *process* (3.58) *state*, such as "started", "in process", or "finished" within a model. Instead, Object-Process Methodology represents and models subprocesses, such as starting, processing, or finishing. See also discussion of Object-Process Methodology process metamodel in Annex C.

3.69
state
<system> snapshot of the system model taken at a certain point in time, which shows all the existing *object* (3.39) instances, current states of each *stateful object* (3.66) instance, and the *process* (3.58) instances, with their elapsed times, executing at the time the snapshot occurs

3.70
state expression
refinement (3.63) involving the revealing of any proper subset of an *object's* (3.39) set of *states* (3.69)

3.71
state suppression
abstraction (3.1) involving the hiding of any proper subset of an *object's* (3.39) set of *states* (3.69)

3.72
structural link
graphic notation of *structural relation* (3.73) in *Object-Process Methodology* (3.43)

3.73
structural relation
operationally invariant connection or association between things

Note 1 to entry: Structural relations persist in the system for at least some interval of time. They provide the structural aspect of the system, and are not contingent upon conditions that are time-dependent.

3.74
structure
<OPM> collection of *objects* (3.39) in an *Object-Process Methodology* (3.43) model and the non-transient relations or associations among them

3.75
System Diagram
SD
Object-Process Diagram (3.41) with one systemic *process* (3.58) indicating the system *function* (3.23) and the *objects* (3.39) connecting with that function to depict the overall *context* (3.11) for and top-level view of the system

Note 1 to entry: System Diagram is the root of the *OPD process tree* (3.45) and has no extent of detail beyond the overall context depicted, i.e. no in-zoomed *refinee* (3.62) is present. Any Object-Process Diagram other than System Diagram is a node in the OPD process tree resulting from *refinement* (3.63).

3.76
thing
<OPM> *object* (3.39) or *process* (3.58)