INTERNATIONAL STANDARD ISO/IEC 19793:2008
TECHNICAL CORRIGENDUM 1

Published 2010-09-15

Information technology — Open Distributed Processing — Use of UML for ODP system specifications

TECHNICAL CORRIGENDUM 1

Technologies de l'information — Traitement réparti ouvert — Utilisation de l'UML pour les spécifications de système ODP

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to ISO/IEC 19793:2008 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*, in collaboration with ITU-T. The identical text is published as ITU-T Rec. X.906 (2007)/Cor.1 (10/2009).

ICS 35.080 **Ref. No. ISO/IEC 19793:2008/Cor.1:2010(E)**

INTERNATIONAL STANDARD
RECOMMENDATION ITU-T

Information technology – Open distributed processing – Use of UML for ODP system specifications

Technical Corrigendum 1

1) Scope

1.1) Defect 1

This Technical Corrigendum changes the use of UML comments to the use of UML constraints. A UML constraint is a packageable element (and therefore has a name, and can be traced and managed, since it can be directly owned by a package), which declares some of the semantics of one or more elements. UML superstructure 2.1.1, section 7.3.10, justifies the representation of rules with constraints, and using constraints leaves room to specify such rules using more powerful languages, such as those specific for policies and rules (the UML spec explicitly mentions that "A user-defined constraint is described using a specified language, whose syntax and interpretation is a tool responsibility. One predefined language for writing constraints is OCL. In some situations, a programming language such as Java may be appropriate for expressing a constraint. In other situations natural language may be used"). A constraint is associated with an ordered set of elements to which the constraint applies. In this way, we can trace these elements.

1.2) Defect 2

This Technical Corrigendum changes the use of UML note to UML comment. According to the UML Superstructure, a note is precisely different from a comment (metaclass Comment) because a note is just considered as a notational element (it cannot be instantiated from any UML element and, in consequence, it cannot be stereotyped).

However, an UML comment is instantiated from its given metaclass and can be stereotyped, as really stated by the standard.

1.3) Defect 3

This Technical Corrigendum changes the cardinality between PolicyA and PolicyValue from 1..0 to 0..1. This was a typographical error.

2) Changes

2.1) Defect 1

Replace subclauses 7.2.15 to 7.2.18 with:

7.2.15 Obligation

If required, the fact that some behaviour places or fulfils an obligation may be stated in a constraint stereotyped as «EV_Obligation» on that behaviour.

7.2.16 Authorization

If required, the fact that some behaviour requires or creates an authorization may be stated in a constraint stereotyped as «EV_Authorization» on that behaviour.

7.2.17 Permission

If required, the fact that some behaviour requires or creates a permission may be stated in a constraint stereotyped as «EV_Permission» on that behaviour.

7.2.18 Prohibition

If required, the fact that some behaviour requires or creates a prohibition may be stated in a constraint stereotyped as «EV_Prohibition» on that behaviour.

2.2) Defect 2

Use UML ***comment*** instead of UML ***note*** in all the following references:

Subclause 7.2.14: *replace* "note" *by* "comment" *so that the sentence now reads (changes marked for easy recognition)*:

The *policy* envelope is expressed by a `class` stereotyped as «EV_PolicyEnvelope», with a **comment** stereotyped as «description» which explains the *policy* and its rules in natural language.

Subclause 11.3: *replace* "note" *by* "comment" *in the two following sentences (changes marked for easy recognition)*:

– a taxonomy of such specifications, which may be provided with name(s) of *implementable standards* described in stereotyped **comments** attached to a deployment diagram including a component instance diagram.

– information required from implementers to support testing, which may be specified with a stereotyped **comment** describing IXIT.

Subclause 3.3: *add* "comment" *and remove* "note" *in the list of terms (changes marked for easy recognition)*:

abstract class; action; activity; activity diagram; aggregate; aggregation; association; association class; association end; attribute; behaviour; behaviour diagram; binary association; binding; call; class; classifier; classification; class diagram; client; collaboration; collaboration occurrence; communication diagram; **comment**; component; component diagram; composite; composite structure diagram; composition; concrete class; connector; constraint; container; context; delegation; dependency; deployment diagram; derived element; diagram; distribution unit; dynamic classification; element; entry action; enumeration; event; exception; execution occurrence; exit action; export; expression; extend; extension; feature; final state; fire; generalizable element; generalization; guard condition; implementation; implementation class; implementation inheritance; import; include; inheritance; initial state; instance; interaction; interaction diagram; interaction overview diagram; interface; internal transition; lifeline; link; link end; message; metaclass; metamodel; method; multiple classification; multiplicity; n-ary association; name; namespace; node; object; object diagram; object flow state; object lifeline; operation; package; parameter; parent; part; partition; pattern; persistent object; pin; port; postcondition; precondition; primitive type; profile; property; pseudo-state; realization; receive [a message]; receiver; reception; refinement; relationship; role; scenario; send [a message]; sender; sequence diagram; signal; signature; slot; state; state machine diagram; state machine; static classification; stereotype; stimulus; structural feature; structure diagram; subactivity state; subclass; submachine state; substate; subpackage; subsystem; subtype; superclass; supertype; supplier; tagged value; time event; time expression; timing diagram; trace; transition; type; usage; use case; use case diagram; value; visibility.