NORME INTERNATIONALE

ISO/CEI 10165-4

Première édition
1992-09-01

AMENDEMENT 3
1998-12-15

Technologies de l'information — Interconnexion de systèmes ouverts — Structure des informations de gestion: Directives pour la définition des objets gérés

AMENDEMENT 3: Directives pour l'utilisation du langage Z dans la formalisation du comportement des objets gérés

Information technology — Open Systems Interconnection — Structure of management information: Guidelines for the definition of managed objects

AMENDMENT 3: Guidelines for the use of Z in formalizing the behaviour of managed objects

Numéro de référence
ISO/CEI 10165-4:1992/Amd.3:1998(F)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax. + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Version française parue en 2000

Imprimé en Suisse

Sommaire

Avant-propos

L'ISO (Organisation internationale de normalisation) et la CEI (Commission électrotechnique internationale) forment le système spécialisé de la normalisation mondiale. Les organismes nationaux membres de l'ISO ou de la CEI participent au développement de Normes internationales par l'intermédiaire des comités techniques créés par l'organisation concernée afin de s'occuper des domaines particuliers de l'activité technique. Les comités techniques de l'ISO et de la CEI collaborent dans des domaines d'intérêt commun. D'autres organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'ISO et la CEI participent également aux travaux.

Dans le domaine des technologies de l'information, l'ISO et la CEI ont créé un comité technique mixte, l'ISO/CEI JTC 1. Les projets de Normes internationales adoptés par le comité technique mixte sont soumis aux organismes nationaux pour vote. Leur publication comme Normes internationales requiert l'approbation de 75 % au moins des organismes nationaux votants.

L'Amendement 3 à l'ISO/CEI 10165-4:1992 a été élaboré par le comité technique mixte ISO/CEI JTC 1, *Technologies de l'information*, sous-comité SC 33, *Services d'applications distribuées*, en collaboration avec l'UIT-T. Le texte identique est publié en tant que Rec. UIT-T X.722/Amd.3.

NORME INTERNATIONALE

RECOMMANDATION UIT-T

TECHNOLOGIES DE L'INFORMATION – INTERCONNEXION DE SYSTÈMES OUVERTS – STRUCTURE DES INFORMATIONS DE GESTION: DIRECTIVES POUR LA DÉFINITION DES OBJETS GÉRÉS

AMENDEMENT 3
Directives pour l'utilisation du langage Z dans la formalisation du comportement des objets gérés

1) Table des matières

Ajouter à la table des matières la référence suivante:

Annexe B – Directives pour l'utilisation du langage Z dans la formalisation du comportement des objets gérés

2) Paragraphe 2.1

Ajouter la référence suivante:

– Recommandation X.731 du CCITT (1992) | ISO/CEI 10164-2:1992, *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion-systèmes: fonction de gestion d'états*.

3) Nouveau paragraphe 2.3

Ajouter un nouveau paragraphe comme suit:

2.3 Autres Références

– ISO/CEI 13568:[1)], *Technologies de l'information – Langage de spécification Z.*

[1)] Actuellement à l'état de projet.

4) Nouvelle Annexe B

Ajouter une nouvelle Annexe B comme suit:

Annexe B

Directives pour l'utilisation du langage Z dans la formalisation du comportement des objets gérés

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

B.1 Introduction

La présente annexe est un guide technique d'application du langage Z dans la définition du comportement des objets gérés utilisés pour l'interfonctionnement de la gestion OSI. De nature descriptive, elle n'est pas normative. Elle ne requiert pas de techniques de définition formelles (FDT, *formal definition techniques*) pour spécifier le comportement des objets gérés. Si les techniques FDT doivent être employées, il n'est pas nécessaire d'utiliser le langage Z; d'autres langages, tels que le SDL, conviennent également. Même s'il est indiqué d'utiliser le langage Z, on peut spécifier le comportement des objets gérés par d'autres moyens.

La spécification formelle du comportement des objets gérés a une utilité évidente car elle est claire et univoque. Comme elle oblige à examiner de près le détail des comportements, elle peut aussi servir d'outil pour déceler des imprécisions qui auraient pu passer inaperçues dans une spécification faisant uniquement usage du langage naturel, et pour y remédier. C'est pourquoi elle permet d'améliorer la spécification des comportements.

La présente annexe contient, à titre indicatif, un exemple montrant la meilleure manière de procéder actuelle. Elle vise à établir une base et une compréhension communes de cette approche formelle particulière qui contribuera à l'uniformité dans les réalisations similaires. Elle sera un point de départ utile pour les utilisateurs des directives GDMO qui souhaitent utiliser le langage Z pour améliorer la spécification des comportements.

Elle est destinée aux utilisateurs qui sont familiarisés avec les concepts de base de la spécification des objets gérés au moyen des modèles GDMO et du langage Z.

Dans la suite de la présente annexe, les expressions «objet géré» et «MO» seront utilisées pour désigner une définition de classe d'objets gérés faite au moyen des modèles GDMO.

B.2 Questions de langage

Le langage Z est une notation de spécification formelle fondée sur la théorie des ensembles et des calculs fonctionnels. Sa capacité d'expression est suffisante pour décrire des classes uniques d'objets gérés.

Toutefois, elle ne contient aucune notion d'encapsulage. Une spécification en langage Z est généralement constituée d'un modèle d'un certain état et d'une suite d'opérations pour modifier cet état. Le langage Z n'incorpore aucune méthode permettant de réunir l'état et ses opérations dans un module unique pouvant être réutilisé dans une autre spécification. La conséquence de cela apparaît quand il faut décrire des objets gérés qui ont des variables et un comportement hérités d'autres définitions de classe d'objets gérés.

Cet effet d'héritage peut être obtenu par la technique de l'inclusion de schémas, au prix d'un peu de clarté. A tout autre égard, le langage Z convient pour exprimer des classes particulières d'objets gérés.

B.3 Ce qu'il y a lieu de traduire

Les définitions d'un comportement ou de parties de celui-ci doivent être traduites en langage Z à partir de leur description informelle. La mesure dans laquelle les parties restantes des modèles GDMO doivent être formalisées dépend dans une large mesure des besoins du spécificateur.

Les modèles GDMO contiennent une définition semi-formelle des types de données en ASN.1. Il est possible d'écrire une spécification en langage Z en utilisant ces définitions en ASN.1 comme base pour les types utilisés dans la spécification en langage Z, ce qui permet d'économiser beaucoup de travail.

Toutefois, si une spécification est écrite de cette manière, le spécificateur aura davantage de travail pour s'assurer qu'elle est syntaxiquement correcte. Sans spécification en langage Z des définitions en ASN.1, on ne peut pas utiliser les outils en Z existants, qui permettent de contrôler la syntaxe et la sémantique statique d'une spécification en Z.

En résumé, on peut améliorer les définitions de comportement en utilisant le langage Z sans récrire les types de données ASN.1, mais on peut obtenir un avantage significatif en traduisant entièrement en langage Z les types de données en ASN.1. Le paragraphe B.7.1 donne des exemples de la manière de convertir en langage Z les types de base écrits en ASN.1.

B.3.1 Des modèles GDMO au langage Z

Le présent paragraphe contient les directives générales sur la manière de traduire un objet géré en langage Z à partir de sa description informelle, telle qu'elle est donnée dans la présente Recommandation | Norme internationale. Il convient de souligner d'emblée qu'une telle traduction ne peut être faite qu'informellement étant donné qu'une traduction formelle nécessiterait, pour le moins, que les langages source et cible soient tous deux formels.

De plus, comme c'est invariablement le cas lorsque l'on met en correspondance deux langages distincts, il y a inévitablement une certaine discordance entre leurs structures. Le problème s'aggrave lorsque l'un des langages est informel ou qu'il comporte des composantes informelles.

Le présent paragraphe contient la liste de quelques-unes des principales caractéristiques des modèles définis dans la présente Recommandation | Norme internationale, avec les points sur lesquels ils diffèrent des structures en langage Z ou y correspondent. Elles sont accompagnées de méthodes générales pour résoudre les discordances ou de conseils sur la manière de les traiter individuellement suivant les circonstances.

La présente annexe est orientée sur ce qui est nécessaire pour décrire le comportement d'un objet géré. Des informations supplémentaires sur la manière de convertir des types ASN.1 sont données au paragraphe B.6.

B.3.2 Types de données

La première étape consiste à récrire sous la forme de types en langage Z les types de données de la présente Recommandation | Norme internationale. L'ASN.1 fournit les moyens habituels de typage des données mais ses constructeurs sont orientés vers la description des flux de données communiqués entre les systèmes.

En ASN.1, les constructeurs de types sont définis comme des formes de liste. En Z, les types sont des ensembles. Bien qu'il soit possible de modéliser les constructeurs de types ASN.1 comme des séquences Z, il est parfois plus simple d'envisager les opérations disponibles sur les types ASN.1 et de les mettre en correspondance avec les types Z qui décrivent plus clairement leur structure. Les types séquence ASN.1 et les types ensemble peuvent être représentés par des nuplets Z. Le type séquence-de ASN.1 peut être représenté par une séquence Z. Le type ensemble-de ASN.1 peut être représenté par un ensemble Z.

L'ASN.1 inclut une prise en charge particulière du codage, comme les étiquettes de types et les valeurs par défaut. Elles n'ont pas à être représentées en Z, n'ayant pas d'incidence sur la définition du comportement.

Le paragraphe B.6.2 offre des informations supplémentaires sur la conversion des types ASN.1.

B.3.3 Attributs d'objets gérés

Par définition, les objets gérés ont certains attributs de gestion; ceux-ci ont un type de données défini en ASN.1 et des identificateurs d'objet leur sont attribués. Ils peuvent en outre avoir une propriété de correspondance. Deux manières de modéliser de tels attributs ont été proposées:

- les types d'attribut simples;
- les types d'attribut ayant la forme de schémas.

La solution la plus simple consiste à représenter l'attribut d'objet géré dans l'objet géré sous forme de variable Z avec le type de données approprié. Parallèlement, il faut définir une constante qui représente l'identificateur d'objet de cet attribut. Cette constante sera liée à l'attribut en question par convention seulement. On peut recourir à la propriété de correspondance en question si des opérations de mise en correspondance pour cet attribut ont été définies. Un exemple en est donné au B.6.3.

On peut aussi encapsuler toutes les propriétés de l'attribut dans un type de schéma unique qui sera alors le type de la variable Z modélisant l'attribut d'objet géré. Le schéma inclura alors la valeur de l'attribut aussi bien que l'identificateur d'objet et la propriété de correspondance éventuelle. Un exemple en est donné au B.6.4. Là où d'autres règles de correspondance que l'égalité sont requises, on peut définir le paramètre de correspondance (*matches-for*) comme une relation Z sur le type de la valeur de l'attribut. Cela permet la représentation formelle de règles de correspondance particulières arbitraires, qui peut être importante dans la validité, le filtrage et la sélection d'objets.

Il est difficile de modéliser en langage Z le type ANY de l'ASN.1. Un cas où le problème est fréquent est celui dans lequel il faut donner une liste de valeurs d'attribut. En conséquence, un modèle entièrement formel nécessitera sans doute un type exempt de langage Z réunissant les types d'attribut qui sont déjà définis. On en trouvera un exemple au B.6.1 et au B.6.5.

Les identificateurs d'objet sont formellement modélisés par un ensemble donné.

[OBJECTID]

B.3.4 Autres identificateurs d'objet

Beaucoup d'éléments autres que les attributs ont également un identificateur d'objet. Il est utile de les introduire tous sous la forme de constantes dans des définitions axiomatiques. On leur attribuera par convention le suffixe «Oid». Généralement, de telles constantes seront nécessaires pour les classes, pour les lots de propriétés (paquetages) et pour les notifications.

Exemple:

packagesPackageOid : OBJECTID
allomorphsPackageOid : OBJECTID
topClassOid : OBJECTID

B.3.5 Héritage et compatibilité

On peut utiliser le langage Z pour construire des hiérarchies d'héritage d'objets gérés en introduisant des schémas pour modéliser l'héritage et la spécialisation. Cela modélise correctement le comportement d'une classe d'objets gérés et de ses sous-types mais ne rend pas explicite la forte relation de sous-typage qui est effectivement présente. A cet effet, il faut un langage qui modélise l'héritage de manière explicite.

La langage Z peut donc être utilisé pour définir de manière satisfaisante les objets gérés individuels; mais, pour être en mesure de traiter de l'héritage et de la compatibilité, il faut disposer de la puissance additionnelle d'un langage pouvant modéliser l'héritage de manière explicite.

L'héritage n'est pas pris en compte dans le langage Z. Il peut être modélisé simplement par l'introduction de schémas d'état.

La définition de l'héritage d'objets gérés nécessite que les sous-classes soient compatibles; mais, en langage Z, il n'est pas indispensable que les sous-classes soient des sous-types, ce qui est regrettable. Donc, généralement, un objet géré peut annoncer sa classe réelle. Etant donné que l'attribut de classe réelle signale toujours la classe réelle de l'objet, une sous-classe ne peut signaler la classe d'une superclasse. Pour cette raison, une sous-classe ne peut pas afficher le même comportement que sa superclasse par le renvoi de la valeur de son attribut de classe réelle (c'est-à-dire qu'elle n'est pas substituable), même si son comportement est allomorphique. Pour cette raison, la sous-répartition des classes d'objets gérés n'est pas équivalente aux sous-types du langage Z, dans lequel un sous-type afficherait le même comportement que son supertype. Néanmoins, une sous-classe affiche très peu de comportements «non substituables».

On constate ainsi que l'héritage d'objets gérés, tel qu'il est défini dans la présente Recommandation | Norme internationale, permet à un parent d'avoir un comportement spécifique qui ne concorde pas avec le comportement de ses descendants. Etant donné qu'un tel comportement non substituable est très rare, on peut représenter une classe d'objets gérés par deux spécifications de classe. L'une est le comportement que toute instance ou objet géré étendu doit afficher; l'autre est une spécialisation qui capture le comportement affiché uniquement par des instances de la classe compatible et non par une quelconque extension. C'est cette dernière spécification qui est instanciée pour donner le comportement complet d'une instance d'objet géré réelle.

B.3.6 Lots (de propriétés)

De nombreuses parties de la fonctionnalité d'une classe peuvent être présentes dans certains objets gérés individuels et non dans d'autres. La présente Recommandation | Norme internationale décrit ce processus en groupant les fonctionnalités dans des lots conditionnels. Ensuite, chaque objet géré instancie les lots appropriés. En langage Z, on ne peut pas donner la fonctionnalité de cette manière conditionnelle, mais on peut faire en sorte que le comportement de l'objet géré dépende de la nature des lots instanciés. Cela est logique vu qu'un objet géré doit contenir un lot appelé par un attribut de gestion qui énumère les identificateurs d'objet des lots effectivement instanciés. Donc, en modélisant le comportement dans un lot conditionnel, ce comportement lui-même devient conditionnel par la présence de l'identificateur de lot dans l'attribut de lots.

B.3.7 Classe

Pour définir une classe d'objets gérés, il faut représenter ses attributs et ses opérations. Les attributions deviennent des parties du schéma d'état Z et les opérations deviennent des schémas d'opération Z.

B.3.7.1 Attributs

Les attributs d'un objet géré sont déclarés dans un schéma d'état. On donne à chaque attribut un type, qui peut être un type déclaré dans la partie ASN.1 du modèle GDMO ou un type déclaré en langage Z dans un modèle entièrement formel.

B.3.7.2 L'opération Get (obtenir)

Le gestionnaire peut demander qu'une opération Get soit effectuée sur un objet géré. La définition des éléments CMISE d'une opération M-Get a de nombreux paramètres dont la plupart concernent toutefois la commande d'accès, la sélection d'objet et ainsi de suite. Dans ce cas, l'opération Get pourra être modélisée à la limite de l'objet géré, sans tenir compte de ces questions et en remplaçant l'opération Get unique par une série d'opérations Get<name>, où <name> est un attribut unique.

B.3.7.3 L'opération GetAll

L'opération GetAll, qui est dépourvue d'entrée, est également modélisée. Elle renvoie un ensemble non vide de valeurs d'attribut.

B.3.7.4 Les opérations Replace (remplacer)

L'opération CMISE M-Set demande une opération Set sur un objet géré individuel. Cette spécification modélise les opérations Replace telles qu'elles sont vues à la limite de l'objet géré. Dans cette spécification, les opérations Replace se réfèrent aux attributs «operations set», «set to default», «add» et «remove».

La conséquence de ce qui précède est la spécification d'un schéma Z pour représenter chaque modification.

B.3.7.5 Notifications

Les notifications sont des messages non sollicités qui sont envoyés par un objet géré pour rendre compte d'événements internes. Elles ne sont toutefois pas modélisées comme des opérations mais comme des résultats d'opérations sur l'objet géré. Donc, toute opération (qu'elle soit invoquée par le gestionnaire ou, de manière interne, par la ressource) peut produire un résultat. Si celui-ci entraîne une notification, celle-ci doit faire partie du résultat de l'opération.

Cela signifie que le résultat d'un schéma d'opération Z pouvant entraîner des notifications doit être un *ensemble* de notifications. Dès lors, on peut représenter les cas dans lesquels il n'émet pas de notification en donnant pour résultat un ensemble vide.

Les données contenues dans une notification sont constituées d'un type EventType qui est l'identificateur d'objet de sa définition standard. Celui-ci est suivi de diverses informations s'appliquant à cette notification particulière. L'identificateur d'objet peut généralement être défini comme une constante et les données particulières comme un type de schéma. Le comportement de la notification est inclus dans tout objet qui a la capacité de produire la notification.

B.3.7.6 Actions

Les actions sont des opérations effectuées par le gestionnaire sur l'objet géré. En langage Z, elles sont représentées tout naturellement par des opérations.

B.3.8 Spécification du système d'objets

Le reste de la présente annexe décrit la manière de représenter le comportement d'un objet unique. Lorsqu'on aborde la création ou la suppression d'objets, les corrélations de leur nom, leur inclusion et leur dénomination, il faut décrire l'état du système où réside l'objet. On peut représenter la création ou la suppression d'un objet par un changement d'état de ce système. On peut représenter la corrélation des noms et l'inclusion par une relation avec l'ensemble d'objets. La dénomination peut alors être définie à partir de cette relation.

B.4 Exemple

Dans le présent paragraphe, on trouvera des exemples de définitions des attributs «top» et «State Management» d'une classe d'objets gérés. Etant donné que le principal objectif du présent guide est de modéliser le comportement, la création de types Z à partir de types ASN.1 n'est pas présentée dans le présent paragraphe. Une définition formelle complète est donnée au B.7.

B.4.1 top (sommet)

La première classe qu'il y a lieu de définir est *top*, qui est l'ascendant ultime (le sommet de la hiérarchie d'héritage) de tous les objets gérés.

La classe *top* a quatre attributs de gestion, à savoir *objectClass, packages, allomorphs* et *nameBinding*. L'attribut *objectClass* contient l'identificateur d'objet de la classe, l'attribut *packages* contient les identificateurs d'objet des blocs qu'il instancie, l'attribut *nameBinding* contient l'identificateur d'objet de la correspondance de noms utilisée pour instancier l'objet et l'attribut *allomorphs* contient les identificateurs d'objet des classes pour lesquelles l'objet peut être allomorphique. Etant donné que les attributs de gestion peuvent former des blocs, les attributs présents dans les objets gérés d'une classe donnée peuvent varier. Cela est modélisé par l'inclusion d'un attribut de modélisation additionnel appelé *attributes*, qui contient des identificateurs d'objet des attributs qui sont réellement instanciés dans l'objet géré individuel. On notera que tous les attributs présents dans la classe *top* sont fixes pendant la durée de vie de tout objet géré individuel.

Le langage Z ne modélise pas explicitement les interfaces et pour cette raison il n'est pas possible de définir formellement celles des opérations qui sont invoquées de manière interne et celles qui le sont de manière externe, par le gestionnaire.

TopState

$$
\begin{array}{|l}
\hline
allomorphs: \mathbb{F}\, OBJECTID \\
objectClass: OBJECTID \\
nameBinding: OBJECTID \\
packages: \mathbb{F}\, OBJECTID \\
attributes: \mathbb{F}\ OBJECTID \\
\hline
\{objectClassOid, nameBindingOid\} \subseteq attributes \\
allomorphsPackageOid \in packages \Rightarrow allomorphsOid \in attributes \\
packagesPackageOid \notin packages \\
packages \neq \varnothing \Rightarrow packagesOid \in attributes \\
\hline
\end{array}
$$

L'élément *attributes* n'est pas un attribut d'objet géré mais une nouvelle composante d'état, définie pour des raisons de commodité, qui énumère les attributs d'objet géré contenus dans cet objet géré. Donc l'invariant l'oblige à contenir les identificateurs d'objet des attributs appropriés décrits au B.3.3 (et définis au B.7.4). Les attributs *objectClas*s et *nameBinding* sont obligatoires; l'attribut *packages* est présent si un lot enregistré donné est instancié séparément de *packagesPackage*. Dans ce cas, cela signifie *allomorphsPackage*.

L'opération *TopGetNameBinding* interroge l'objet géré et retourne la valeur de l'attribut *nameBinding* sans changer l'état *TopState*. L'opération *TopGetNameBinding* est invoquée par le gestionnaire.

TopGetNameBinding

$$
\begin{array}{|l}
\hline
\Xi\, TopState \\
result!: OBJECTID \\
\hline
result! = nameBinding \\
\hline
\end{array}
$$

Les opérations *TopGetAllomorphs*, *TopGetObjectClass* et *TopGetPackages* n'ont pas été définies ici. On notera qu'il n'y a pas d'opération pour obtenir *attributes* étant donné que ce n'est pas un attribut réel d'objet géré tel que spécifié dans le modèle GDMO.

L'opération *TopGetAll* obtient toutes les valeurs d'attribut d'un objet. Elle retourne toujours des valeurs pour *objectClass* et *nameBinding*. En cas de présence de lots conditionnels ou d'allomorphismes, il les obtient aussi. L'opération *TopGetAll* est invoquée par le gestionnaire.