Union internationale des télécommunications

UIT-T

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

Z.130

Amendement 1

(06/2006)

SÉRIE Z: LANGAGES ET ASPECTS GÉNÉRAUX LOGICIELS DES SYSTÈMES DE TÉLÉCOMMUNICATION

Techniques de description formelle – Langage étendu de définition d'objets

---

Langage étendu de définition d'objets (eODL): techniques de développement de composants logiciels répartis – Bases conceptuelles, notations et correspondances technologiques

Amendement 1: Nouvelle Annexe E – Mappage du langage eODL vers le langage CIDL

Recommandation UIT-T Z.130 (2003) – Amendement 1

RECOMMANDATIONS UIT-T DE LA SÉRIE Z

LANGAGES ET ASPECTS GÉNÉRAUX LOGICIELS DES SYSTÈMES DE TÉLÉCOMMUNICATION

TECHNIQUES DE DESCRIPTION FORMELLE	
Langage de description et de spécification (SDL)	Z.100–Z.109
Application des techniques de description formelle	Z.110–Z.119
Diagrammes des séquences de messages	Z.120–Z.129
Langage étendu de définition d'objets	**Z.130–Z.139**
Notation de test et de commande de test	Z.140–Z.149
Notation de prescriptions d'utilisateur	Z.150–Z.159
LANGAGES DE PROGRAMMATION	
CHILL: le langage de haut niveau de l'UIT-T	Z.200–Z.209
LANGAGE HOMME-MACHINE	
Principes généraux	Z.300–Z.309
Syntaxe de base et procédures de dialogue	Z.310–Z.319
LHM étendu pour terminaux à écrans de visualisation	Z.320–Z.329
Spécification de l'interface homme-machine	Z.330–Z.349
Interfaces homme-machine orientées données	Z.350–Z.359
Interfaces homme-machine pour la gestion des réseaux de télécommunication	Z.360–Z.379
QUALITÉ	
Qualité des logiciels de télécommunication	Z.400–Z.409
Aspects qualité des Recommandations relatives aux protocoles	Z.450–Z.459
MÉTHODES	
Méthodes de validation et d'essai	Z.500–Z.519
INTERGICIELS	
Environnement de traitement réparti	Z.600–Z.609

Pour plus de détails, voir la Liste des Recommandations de l'UIT-T.

Recommandation UIT-T Z.130

Langage étendu de définition d'objets (eODL): techniques de développement de composants logiciels répartis – Bases conceptuelles, notations et correspondances technologiques

Amendement 1

Nouvelle Annexe E – Mappage du langage eODL vers le langage CIDL

Résumé

Le présent amendement fournit un exemple de mappage du langage ITU eODL ayant des spécifications de composants indépendantes de la technologie vers un langage dont les spécifications de composants dépendent de la technologie considérée, à savoir ici le langage CIDL (langage d'implémentation de composants de l'OMG dans le cadre de CORBA 3.0). Le présent amendement transforme (à l'aide de différents mappages) le concept de composants, pour passer de la phase de conception et d'implémentation (où les modules sont bien connus) à la phase de logiciel binaire. La composition des composants se fait durant le temps d'exécution.

Source

L'Amendement 1 de la Recommandation UIT-T Z.130 (2003) a été approuvé le 13 juin 2006 par la Commission d'études 17 (2005-2008) de l'UIT-T selon la procédure définie dans la Recommandation UIT-T A.8.

AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'UIT. Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

L'Assemblée mondiale de normalisation des télécommunications (AMNT), qui se réunit tous les quatre ans, détermine les thèmes d'étude à traiter par les Commissions d'études de l'UIT-T, lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution 1 de l'AMNT.

Dans certains secteurs des technologies de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI.

NOTE

Dans la présente Recommandation, l'expression "Administration" est utilisée pour désigner de façon abrégée aussi bien une administration de télécommunications qu'une exploitation reconnue.

Le respect de cette Recommandation se fait à titre volontaire. Cependant, il se peut que la Recommandation contienne certaines dispositions obligatoires (pour assurer, par exemple, l'interopérabilité et l'applicabilité) et considère que la Recommandation est respectée lorsque toutes ces dispositions sont observées. Le futur d'obligation et les autres moyens d'expression de l'obligation comme le verbe "devoir" ainsi que leurs formes négatives servent à énoncer des prescriptions. L'utilisation de ces formes ne signifie pas qu'il est obligatoire de respecter la Recommandation.

DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente Recommandation puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des Recommandations.

A la date d'approbation de la présente Recommandation, l'UIT n'avait pas été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux développeurs de consulter la base de données des brevets du TSB sous http://www.itu.int/ITU-T/ipr/.

TABLE DES MATIÈRES

Recommandation UIT-T Z.130

Langage étendu de définition d'objets (eODL): techniques de développement de composants logiciels répartis – Bases conceptuelles notations et correspondances technologiques

Amendement 1

Nouvelle Annexe E – Mappage du langage eODL vers le langage CIDL

1) Remplacer les points suivants du paragraphe "Résumé"

- L'Annexe D donne une référence à un logiciel contenant la représentation en XML [12] du métamodèle eODL conformément au format d'échange de métadonnées XML (XMI) [6]. Cette représentation est contenue dans un fichier distinct afin que les outils UML puissent importer et traiter le métamodèle eODL.
- Le paragraphe 1 présente un aperçu de la manière dont l'eODL est utilisé par les concepteurs, implémenteurs et gestionnaires d'un système réparti. L'Appendice I donne un exemple concret d'utilisation.

par:

- L'Annexe D donne une référence à un logiciel contenant la représentation en XML [12] du métamodèle eODL conformément au format d'échange de métadonnées XML (XMI) [6]. Cette représentation est contenue dans un fichier distinct afin que les outils UML puissent importer et traiter le métamodèle eODL.
- L'Annexe E contient des règles de mappage du langage eODL indépendant de la technologie vers le langage CIDL spécifique de la technologie [7].
- L'Appendice I présente un aperçu général de la manière dont l'eODL est utilisé par les concepteurs, implémenteurs et gestionnaires d'un système réparti. L'Appendice I donne un exemple concret d'utilisation.

2) Mettre à jour comme suit la table des matières

Ajouter à la table des matières les entrées suivantes avant celles relatives à l'Appendice I. Modifier la numérotation en conséquence.

3) Insérer avant l'Appendice I

Annexe E

Mappage du langage eODL vers le langage CIDL

E.1 Introduction

Le développement logiciel fondé sur les composants correspond à une méthode de développement logiciel modulaire établie à partir d'un modèle. Différents mappages permettent de transformer des **modèles de composants** (considérés de différents points de vue tels que la conception et

l'implémentation) en **composants logiciels** binaires. La composition des composants logiciels se fait durant le temps d'exécution.

eODL est un langage qui propose des concepts permettant une **description de modèle indépendante de la technologie** pour les composants au cours de leurs cycles de vie, et ce depuis différents points de vue. Des concepts tels que ceux d'objet de traitement, de composant, d'interface, de module, de signal et de type de données sont essentiels pour les points de vue traitement et implémentation. Il existe en outre d'autres composants pour décrire les environnements d'exécution et le déploiement de composants logiciels.

Le modèle **CCM** (modèle de composants CORBA [7]) est une norme de l'OMG applicable à un cadre dépendant de la plate-forme. Il fournit un métamodèle permettant de décrire les composants CORBA **dépendants de la technologie** ainsi que l'environnement technologique et d'exécution pour les composants développés à l'aide de ce métamodèle. Le modèle CCM est fondé sur des technologies CORBA matures telles que le protocole GIOP et sur des liens linguistiques pour les langages d'implémentation. Le modèle de composants CCM définit deux types d'interaction de composants: une interaction de type RPC avec demande/réponse et une interaction de type signal associée à des événements. Pour chacun de ces types d'interactions, les composants peuvent déclarer une utilisation ou une offre. Le modèle CCM utilise le langage CIDL pour la notation de l'implémentation des composants.

Pour **combler le fossé** entre les modèles de composants logiciels indépendants de la technologie (spécifications eODL) et les modèles dépendants de la technologie (modèles CIDL), il faut des mappages fondant des transformations de modèle automatisées.

Le langage de définition d'implémentation de composants (**CIDL**) de l'OMG est un langage utilisé pour décrire la structure et l'état des implémentations de composants CORBA. Les compileurs activateurs de composants génèrent des squelettes d'implémentation à partir des définitions CIDL. Les développeurs de composants étoffent ces squelettes pour créer des implémentations complètes.

La présente annexe définit les règles pour un mappage du langage eODL vers le langage CIDL. L'application de ces règles est vérifiée grâce à l'implémentation d'un compileur.

E.2 Mappage restreint du langage eODL vers le langage CIDL

La définition du langage eODL est largement fondée sur les concepts définis par la norme CORBA IDL 2.x [5]. Le métamodèle eODL constitue aussi une extension du métamodèle CORBA. Les concepts adoptés sont rattachés au *point de vue traitement* du langage eODL. Le métamodèle CMM ne prend malheureusement pas en charge les concepts eODL relatifs au *point de vue déploiement* et au *point de vue environnement cible*. Ce champ n'est pas encore défini par le métamodèle MOF de CCM. Seuls les types de document XML définis sont nécessaires pour la réalisation finale de l'architecture de déploiement.

Conclusion: les concepts eODL relatifs au *point de vue déploiement* et au *point de vue environnement cible* ne sont pas mappés. Les règles de mappage devraient être étendues dans la mesure où un processus de normalisation OMG correspondant sera finalisé.

E.3 Mappage vers le modèle CCM des concepts eODL qui sont des concepts CORBA

Comme le métamodèle eODL, le métamodèle CCM élargit également les concepts CORBA 2.x. On choisit donc pour les concepts eODL issus de CORBA le mappage le plus simple, à savoir le **mappage à l'identique**. On peut ainsi attribuer les concepts de base relatifs au *point de vue traitement* eODL (tels que **types de données, interfaces, opérations et attributs**) du niveau indépendant de la plate-forme au niveau dépendant de la plate-forme, d'une façon qui soit la plus simple possible pour le développeur.

Si le métamodèle CORBA est utilisé comme source et comme destination du mappage, on peut observer des chevauchements en définissant les règles de transformation. Du fait du mappage à l'identique, cela ne se produit que lorsque les concepts du métamodèle CORBA sont utilisés dans un contexte qui n'est pas issu de ce métamodèle.

E.4 Mappage des concepts relatifs au point de vue traitement

E.4.1 Signaux

Les signaux portent l'information en langage eODL. Ils sont acheminés de l'émetteur vers le récepteur au cours d'une interaction de type signal.

Règle 1: pour chaque élément **SignalDef** du langage eODL, on crée dans le modèle CCM un élément **EventDef** portant le même nom. Les noms et les types de données associés dans un élément **CarryField** du langage eODL sont mappés vers des éléments **ValueMemberDef** du modèle CCM, qui sont contenus dans l'élément **EventDef**. Tous les éléments **ValueMemberDef** créés ont une visibilité publique (isPublicMember==vrai).

Exemple:

```
signal Sig {
  long l;
};
```

est mappé vers CIDL,

```
eventtype Sig {
  public long l;
};
```

E.4.2 Consommer et produire

Les éléments d'interaction consommer et produire du langage eODL sont supposés définir l'interaction de type signal dans une interface. Même s'il existe une interaction de type signal dans le modèle CCM (***EventDef***), celle-ci n'est pas autorisée à faire partie d'une définition de composant. Le modèle CMM ne définit une telle interaction qu'en tant que partie directe d'une définition de composant. Une fois encore, ceci n'est pas autorisé en langage eODL: seuls les attributs le sont. Une interdiction complète des éléments consommer et produire dans le modèle eODL empêcherait toute interaction de type signal. On définit donc une construction de remplacement, qui certes accroît la complexité du mappage mais autorise au moins une interaction de type signal. Dans le modèle CCM, la définition d'un signal (***EventDef***) définit une interface pour l'échange de signaux. Lorsque l'on définit des ports de composant, ceux-ci sont gérés comme des ports distincts. Ainsi, pour chaque élément d'interaction de type signal du langage eODL, un port distinct est défini au niveau du composant.

Règle 2: les éléments du type ***ConsumeDef*** et ***ProduceDef*** du langage eODL ne sont pas mappés en propre mais sont gérés par le biais des règles applicables aux ports.

Exemple:

```
signal Sig;
interface A {
  consumes Sig c;
  produces Sig p;
};
```

est mappé vers CIDL,

*où les éléments d'interaction de type signal issus de l'élément **EnhancedInterfaceDef** sont supprimés. Dans le modèle CCM, seuls les signaux apparaissent en tant qu'éléments **EventDef**.*

```
eventtype Sig {
};
interface A {
};
```

E.4.3 Média, puits et source

Les interactions de types opération et signal sont pris en charge par le modèle CCM. Une interaction de type flux **n'est pas représentée**. On s'efforce d'élargir le modèle CCM pour y intégrer ces concepts, qui n'appartiennent cependant pas à la norme.

Le mappage ne transformera donc pas les éléments de modèle eODL associés à ces concepts en éléments de modèle CCM.

E.4.4 Type de CO, prendre en charge et requérir

Le langage eODL et le modèle CCM sont des extensions de concepts CORBA et introduisent le concept de composant. En eODL, ce concept est appelé type de CO tandis que dans le modèle CCM, on l'appelle composant. Un type de CO est donc mappé vers un composant. Les interactions de type de CO sont autorisées uniquement via des ports car cette variante d'interaction existe également dans le modèle CCM.

Règle 3: pour chaque élément ***COTypeDef*** du langage eODL, on crée dans le modèle CCM un élément ***ComponentDef*** portant le même nom. Si l'élément ***COTypeDef*** B est une spécialisation de l'élément ***COTypeDef*** A en langage eODL, l'élément ***ComponentDef*** B correspondant est alors une spécialisation de l'élément ***ComponentDef*** A dans le modèle CCM. Les relations ***prendre en charge*** et ***requérir*** ne sont pas mappées. L'héritage multiple de l'élément ***COTypeDef*** en langage eODL n'est pas permis.

Exemple:

```
CO A {
};
CO B {
};
```

Les types de CO du langage eODL sont mappés vers d'autres composants du modèle CCM tant que la relation d'héritage est préservée.

```
component A {
};
component B : A {
};
```

E.4.5 Rattachement (HomeDef)

Dans le modèle CCM, il existe un autre concept étroitement lié au concept de composant CCM. Le concept de rattachement est utilisé pour gérer les composants pendant l'exécution. C'est un moyen de créer les instances des composants. Une définition de composant sans mention d'un rattachement est donc incomplète. Le mappage crée donc également, pour chaque type de CO, un *rattachement* dans le modèle CCM.

Règle 4: pour chaque élément ***COTypeDef*** du langage eODL, on crée dans le modèle CCM un élément ***HomeDef*** dont le nom résulte de la concaténation du nom de l'élément ***COTypeDef*** et de "**_Home**". Les éléments ***ComponentDef*** et ***HomeDef*** résultants participent à l'association ***Component_Home***. Si un élément ***COTypeDef*** B est une spécialisation de l'élément ***COTypeDef*** A en langage eODL, l'élément ***HomeDef*** B correspondant est alors une spécialisation de l'élément ***HomeDef*** A dans le langage CCM.

Exemple:

```
CO A {
};
CO B : A {
};
```

Le mappage crée pour chaque type de CO un type de composant dans le modèle CCM ainsi qu'un élément rattachement.

```
component A {
};
home A_Home manages A {};
component B : A {
};
home B_Home : A_Home manages B {};
```

E.4.6 Ports provisionnés et ports utilisés

Les concepts ***ProvidePortDef*** et ***UsedPortDef*** du langage eODL sont utilisés pour définir l'interface entre le type de CO et l'environnement. Dans le modèle CCM, le concept correspondant est celui de port (***ComponentFeature***), qui apparaît pour différentes spécialisations. Il existe un port pour la réalisation et l'utilisation d'une interface dans le contexte CORBA ainsi qu'un port pour une interaction de type signal.

Règle 5: pour chaque élément ***ProvidePortDef*** du langage eODL figurant dans un élément ***COTypeDef***, on crée dans le modèle CCM un élément ***ProvidesDef*** de l'élément correspondant ***ComponentDef*** portant le même nom. L'élément ***InterfaceDef*** référencé par rapport à l'élément ***ProvidePortDef*** appartient à l'association ***provides*** de l'élément ***ProvidesDef***. L'élément ***ProvidesDef*** a le rôle ***facet*** par rapport à l'élément ***ComponentDef*** dans le regroupement ***Component_Facet***. Les ports multiples (*multiple==true*) ne sont pas autorisés.

Exemple:

```
interface A {
};
CO C {
  provides A the_a;
};
```

Les types d'interface provisionnés de type CO sans interaction de type signal sont mappés vers des ports simples.

```
interface A {
};
component C {
  provides A the_a;
};
```

Règle 6: pour chaque élément ***UsedPortDef*** du langage eODL figurant dans un élément ***COTypeDef***, on crée dans le modèle CCM un élément ***UsesDef*** de l'élément correspondant ***ComponentDef*** portant le même nom. L'élément ***InterfaceDef*** référencé par rapport à l'élément ***UsedPortDef*** appartient à l'association ***uses*** de l'élément ***UsesDef***. L'élément ***UsesDef*** a le rôle ***receptacle*** par rapport à l'élément ***ComponentDef*** dans le regroupement ***Component_Receptacle***. Les ports multiples (*multiple==true*) sont mappés vers des ports multiples (*multiple==true*).

Exemple:

```
interface A {
};
CO C {
  uses multiple A the_a;
};
```

*Contrairement au mappage de l'élément **ProvidesDef,** les ports multiples sont autorisés en tant que ports utilisés.*

```
interface A {
};
component C {
  uses multiple A the_a;
};
```

E.4.7 Port produire et consommer

Les éléments d'interaction ayant été supprimés des interfaces en raison du mappage (*Règle 2*), il faut définir des règles additionnelles pour le mappage d'interactions de type signal ayant des ports au niveau des interfaces.

Règle 7: pour chaque élément ***ProduceDef*** du langage eODL figurant dans un élément ***InterfaceDef*** référencé par un élément ***ProvidePortDef*** d'un élément ***COTypeDef,*** on crée un élément ***PublishesDef*** dont le nom résulte de la concaténation du nom de l'élément ***ProvidedPortDef***, de "_" et du nom de l'élément ***ProduceDef***. L'élément ***PublishesDef*** a le rôle ***publishes*** par rapport à l'élément ***ComponentDef*** dans le regroupement ***Component_Publishes***.

Exemple:

```
signal Sig;
interface A {
  produce Sig p;
};
CO C {
  provides A the_a;
};
```

L'utilisation d'interfaces avec des interactions de type signal de port conduit toujours à l'utilisation d'un port distinct car chaque interaction de type signal doit être mappée vers un port distinct.

```
eventtype Sig {
};
interface A {
};
component C {
  provides A the_a;
  publishes Sig the_a_p;
};
```

Règle 8: pour chaque élément ***ConsumeDef*** du langage eODL figurant dans un élément ***InterfaceDef*** référencé par un élément ***ProvidePortDef*** d'un élément ***COTypeDef,*** on crée dans le modèle CCM un élément ***ConsumesDef*** dont le nom résulte de la concaténation du nom de l'élément ***ProvidedPortDef***, de "_" et du nom de l'élément ***ConsumeDef***. L'élément ***ConsumesDef*** a le rôle ***consumes*** par rapport à l'élément ***ComponentDef*** dans le regroupement ***Component_Consumes***.

Exemple:

```
signal Sig;
interface A {
  consume Sig s;
};

CO C {
  provides A the_a;
};
```

Par rapport à l'exemple précédent, le sens de l'interaction de type signal a changé. C'est pourquoi le port d'émission est à présent un port de réception.

```
eventtype Sig {
};
interface A {
  consume Sig c;
};
component C {
  provides A the_a;
  consumes Sig the_a_c;
};
```

Règle 9: pour chaque élément ***ProduceDef*** du langage eODL figurant dans un élément ***InterfaceDef*** référencé par un élément ***UsedPortDef*** d'un élément ***COTypeDef,*** on crée dans le modèle CCM un élément ***ConsumesDef*** dont le nom résulte de la concaténation du nom de l'élément ***ProvidedPortDef***, de "_" et du nom de l'élément ***ConsumeDef***. L'élément ***ConsumesDef*** a le rôle ***consumes*** par rapport à l'élément ***ComponentDef*** dans le regroupement ***Component_Consumes***.

Exemple:

```
signal Sig;
interface A {
  produce Sig p;
};

CO C {
  use A the_a;
};
```

Le mappage est le même que celui de la Règle 7 sauf en ce qui concerne le mappage d'un port utilisé.

```
eventtype Sig {
};
interface A {
};
component C {
   uses A the_a;
  consumes Sig the_a_p;
};
```

Règle 10: pour chaque élément ***ConsumeDef*** du langage eODL figurant dans un élément ***InterfaceDef*** référencé par un élément ***UsedPortDef*** d'un élément ***COTypeDef,*** on crée dans le modèle CCM un élément ***PublishesDef*** dont le nom résulte de la concaténation du nom de l'élément ***UsedPortDef***, de "_" et du nom de l'élément ***ConsumeDef***. L'élément ***ConsumesDef*** a le rôle ***publishes*** par rapport à l'élément ***ComponentDef*** dans le regroupement ***Component_Publishes***.

Exemple:

```
signal Sig;
interface A {
  consume Sig s;
};
CO C {
  use A the_a;
};
```

Par rapport à l'exemple précédent, le sens de l'interaction de type signal a changé. C'est pourquoi le port d'émission est à présent un port de réception.

```
eventtype Sig {
};
interface A {
  consume Sig c;
};
component C {
  uses A the_a;
  publishes Sig the_a_c;
};
```

E.4.8 Attribut

Les types de CO du langage eODL peuvent ne contenir que des éléments ***AttributeDef*** comme éléments d'interaction. Le concept correspondant dans le modèle CCM pouvant aussi contenir l'élément ***AttributeDef***, les éléments ***AttributeDef*** des types de CO sont mappés vers des éléments ***AttributeDef*** de types de composants.

Règle 11: l'élément ***COTypeDef***, en tant que spécialisation de l'élément ***InterfaceDef*** avec les contraintes définies par la Rec. UIT-T Z.130, est autorisé à contenir l'élément ***AttributeDef***. Pour chaque élément ***AttributeDef*** figurant dans un élément ***COTypeDef,*** on crée dans le modèle CCM un élément ***AttributeDef*** dans l'élément ***ComponentDef*** correspondant. Le nom, l'association et les attributs de l'élément ***AttributeDef*** sont conservés conformément au § E.1.

Exemple:

```
CO C {
  readonly attribute long l;
};
```

En fait, le mappage des attributs est un mappage à l'identique mais est utilisé hors du contexte CORBA.

```
Component C {
  readonly attribute long l;
};
```

E.5 Mappage des concepts du point de vue implémentation

Les concepts du point de vue implémentation eODL décrivent la relation entre les éléments d'interaction fournis par les types d'interface de type de CO et les réalisations à travers des artefacts du langage d'implémentation. Les éléments du langage d'implémentation de la technologie des composants sont ainsi modélisés par des artefacts. Dans le contexte des langages d'implémentation orientés objet, il s'agit généralement de classes. Le langage eODL est moins restrictif en ce qui concerne la relation entre les éléments d'interaction et les artefacts. Il n'y a notamment pas de contrainte relative au groupement des éléments d'interaction et des interfaces. Le modèle CCM définit également des concepts permettant de spécifier la structure de la réalisation de composants. Il utilise pour ce faire le concept ***ComponentImplDef***. Chaque composant étant associé à un rattachement défini, il existe un concept ***HomeImplDef*** pour sa réalisation. Le modèle CCM définit également le concept ***SegmentDef*** pour une spécification plus détaillée de la sous-structure. En utilisant ***SegmentDef***, on définit un artefact qui réalise les interfaces provisionnées du composant. Le modèle CCM n'autorise donc que des relations entre les éléments d'interaction et les artefacts sur la base de leurs groupements dans les interfaces (voir la Figure E.1). De plus, l'attribution à l'élément ***SegmentDef*** des interfaces utilisées par les composants est interdite. Connecter des éléments d'interaction de type signal à l'élément ***SegmentDef*** n'est autorisé ni dans le cas d'une émission, ni dans le cas d'une réception. Il faudrait au moins que l'attribution des signaux de réception soit possible dans le modèle CCM.

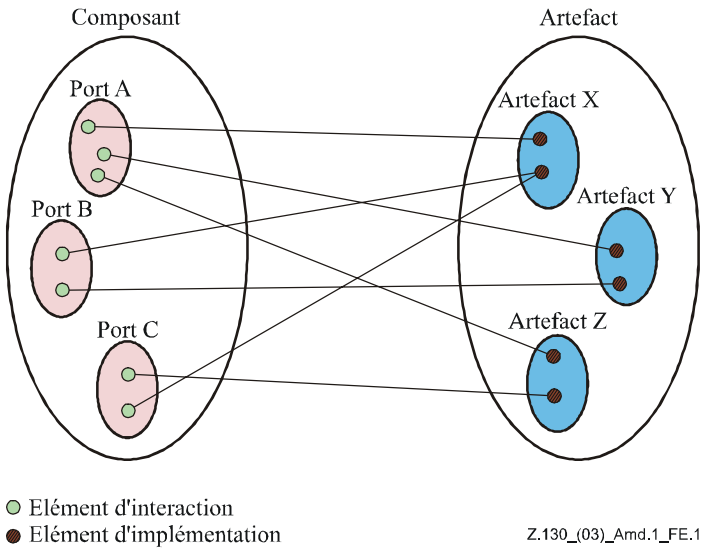

Figure E.1/Z.130 – Attribution d'éléments d'interaction et d'artefacts

E.5.1 Artefact

Le langage eODL utilise les concepts ***ArtifactDef*** et ***ImplementationalElementDef*** pour décrire la structure d'une réalisation de types de CO et l'attribution d'éléments d'interaction aux artefacts. Le modèle CCM spécifie le concept ***SegmentDef*** pour cette attribution. Les artefacts sont donc mappés vers l'élément ***SegmentDef***. Le langage eODL définit pour chaque artefact l'un des schémas d'instanciation suivants

ARTIFACT_PER_REQUEST, ***ARTIFACT_POOL***, ***SINGLETON*** ou ***USER_DEFINED***.

Ils décrivent le moment d'instanciation d'un artefact. Les concepts les plus proches dans le modèle CCM sont ceux de catégories de types de composants ***PROCESS***, ***ENTITY***, ***SESSION*** et ***SERVICE***. D'une part, ils ne font référence qu'à la réalisation complète de composants lorsqu'une séparation pour l'élément ***SegmentDef*** spécifique n'est pas donnée et, d'autre part, on ne peut trouver aucune attribution directe entre le schéma d'instanciation et les catégories de types de composants (voir le § 5.4.1 et la section 4.1.4 sur les composantes CORBA).

Règle 12: pour chaque élément ***ArtifactDef*** et les éléments ***ImplementationalElementDef*** contenus du langage eODL, on crée dans le modèle CCM un élément ***SegmentDef*** portant le même nom. Les éléments ***ImplementationElementDef*** contenus dans l'élément ***ArtifactDef*** sont uniquement autorisés à être reliés à des éléments d'interaction du type ***OperationDef*** et ***AttributeDef*** pour des types d'interface (*Case == supply*) provisionnés via des ports. Tous les éléments ***ImplementationElementDef*** figurant dans un élément ***ArtifactDef*** doivent couvrir tous les éléments d'interaction d'un élément ***InterfaceDef***. L'association ***implementedBy*** dans le modèle CCM contient alors l'élément ***SegmentDef*** et tous les éléments ***ProvidesDef*** correspondant aux différents éléments ***InterfaceDef***. L'association ***implemented_by*** du langage eODL est mappé vers l'association ***segments*** du modèle CCM conformément aux éléments créés correspondants.

Pour tous les éléments ***ArtifactDef*** du langage eODL qui réalisent le même type de CO, on crée dans le modèle CCM un élément ***ComponentImplDef***. Son nom résulte de la concaténation du nom de l'élément ***COTypeDef*** et de "**`Impl`**". La catégorie implémentations de composants est session (*category == SESSION*). L'élément ***ComponentImplDef*** et l'élément ***ComponentDef*** conformes à l'élément ***COTypeDef*** participent à la relation ***implemented_by***.

Pour chaque élément ***HomeDef*** créé conformément à l'élément ***COTypeDef,*** on crée un élément ***HomeImplDef*** dont le nom résulte de la concaténation du nom ***HomeDef*** et de "**`Impl`**". L'élément ***HomeImplDef*** est associé à l'élément ***HomeDef*** par la relation ***implements*** et à l'élément ***ComponentImplDef*** par la relation ***manages***.

Exemple:

```
interface A {
  void op();
};

CO C {
  provides A the_a;
};
artefact AImpl {
  op implements supply A::op;
};
```

Dans le langage eODL, il n'y a pas de concept distinct relatif à une implémentation de type CO. Le modèle CCM nécessite toutefois un élément ComponentImplDef. Il faut également créer un élément HomeImplDef pour l'élément HomeDef créé implicitement.

```
interface A {
  void op();
};
component C {
  provides A the_a;
};

home C_Home manages C {};
composition session CImpl {
  home executor C_HomeImpl {
    implements C_Home;
    manages CSessionImpl {
      segment AImpl {
        provides facet the_a;
      };
    };
  };
};
```

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	Systèmes audiovisuels et multimédias
Série I	Réseau numérique à intégration de services
Série J	Réseaux câblés et transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	Gestion des télécommunications y compris le RGT et maintenance des réseaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	Réseaux de données, communication entre systèmes ouverts et sécurité
Série Y	Infrastructure mondiale de l'information, protocole Internet et réseaux de prochaine génération
Série Z	**Langages et aspects généraux logiciels des systèmes de télécommunication**

Imprimé en Suisse
Genève, 2006