Deuxième conférence maghrébine en génie logiciel et intelligence artificielle, Tunis, 13-16 avril 1992

CONCEPTION ET SPÉCIFICATION PAR OBJETS DU CONTRÔLE CENTRALISÉ D'UN SYSTÈME DE TRANSMISSION

M. Barbeau[1], P. de Saqui-Sannes[2], G. v. Bochmann[3]

RÉSUMÉ

Un des objectifs poursuivis par ce travail est l'introduction des techniques de description formelle dans le processus de conception des contrôleurs de systèmes de transmission. Nous démontrons également qu'une approche de spécification orientée objet est particulièrement bien adaptée à ce genre de problème car elle donne lieu à des représentations très naturelles de la structure des systèmes de transmission. Dans le cadre d'un projet conjoint avec Recherches Bell-Northern, nous avons mis au point une méthode de conception par objets ainsi qu'un langage formel de description par objets. L'objectif de cet article est de relater une expérience avec ces outils conceptuels. Cette expérience a consisté à développer et modéliser l'aspect détection et récupération des pannes, au moyen d'un contrôleur centralisé, d'un site terminal d'un système de transmission.

ABSTRACT

One of the issues considered in this work is the introduction of formal description techniques in the design process of controllers of transmission systems. We also demonstrate that an object-oriented specification approach is particularly well suited to this type of application since it leads to very natural representations of the structure of transmission systems. In the course of a joint project with Bell-Northern Reasearch, we developed both an object-oriented design methodology and an object-oriented formal description technique. The goal of the current paper is to present an experiment, with the above conceptual tools, which consisted of developing and modeling fault detection and recovery, by a central controller, in a terminal site of a transmission system.

"This research was supported by a grant from the Canadian Institute for Telecommunications Research under the NCE program of the Government of Canada"

[1]Département de mathématiques et d'informatique, Université de Sherbrooke, Sherbrooke, Canada J1K 2R1; barbeau@dmi.usherb.ca

[2]LAAS-CNRS, Groupe OLC, 7 avenue du colonel Roche, 31077 Toulouse Cedex, France; pdss@laas.fr

[3]Département d'informatique et de recherche opérationnelle, C.P. 6128, Succursale A, Université de Montréal, Montréal, Canada H3C 3J7; bochmann@iro.umontreal.ca

1. INTRODUCTION

Dans cet article, nous nous intéressons aux problèmes de la conception et de la spécification du contrôle dans les systèmes de transmission. Un des objectifs poursuivis par ce travail est l'introduction des techniques de description formelle dans le processus de conception des contrôleurs. De plus, nous montrons qu'une approche de spécification orientée objet est particulièrement bien adaptée à ce genre de problème car elle donne lieu à des représentations très naturelles de la structure des systèmes de transmission. Dans le cadre d'un projet conjoint avec Recherches Bell-Northern, nous avons mis au point une méthode de conception par objets ainsi qu'un langage formel de description par objets. L'objectif de cet article est de relater une expérience, avec ces outils conceptuels. Cette expérience a consisté à développer et modéliser l'aspect détection et récupération des pannes, dans un site terminal d'un système de transmission, au moyen d'un contrôleur centralisé.

Dans la littérature, les algorithmes distribués et de contrôle sont généralement décrits en utilisant: i) soit des notations semi-formelles (ex. [Gold 88]), ii) soit des notations formelles basées sur des modèles abstraits et idéalisés des machines (ex. [Rama 89]). Des preuves d'exactitude peuvent être développées semi-formellement dans le premier cas et formellement dans le deuxième. Cependant, l'intégration des algorithmes dans de réels environnements distribués est une étape durant laquelle plusieurs nouvelles erreurs peuvent être introduites. Ces erreurs sont causées, par exemple, par une fausse interprétation de certaines parties de la description semi-formelle ou à l'implantation d'aspects non formalisés du modèle abstrait.

Dans ce contexte, la contribution originale de notre travail consiste en l'introduction des techniques de description formelle (TDF) pour la définition de l'équipement contrôlé et de son contrôleur. Les TDF sont utilisées depuis longtemps dans le domaine des protocoles [Boch 90a]. Elles permettent: i) la production de descriptions lisibles et non ambiguës des algorithmes, ii) la vérification en utilisant plusieurs méthodes complémentaires de preuve [Groz 85], iii) la compilation vers une implantation réelle, et iv) la vérification de la cohérence de l'implantation par rapport à la description initiale en utilisant des méthodes systématiques de test.

Nous croyons que l'expérience acquise avec les TDF dans le domaine des protocoles peut être avantageusement utilisée dans le domaine du contrôle. À cette fin, nous utilisons une méthodologie de conception par objets, introduite dans [Mond 90], et un langage de description nommé Mondel [Boch 91]. Mondel intègre les concepts: i) de la programmation orientée objet, ii) de la programmation parallèle, et iii) des bases de données. De plus, la sémantique de Mondel est définie formellement [Barb 90]. Nous avons développé la description formelle du contrôleur d'un système de transmission et la description formelle du système de transmission lui-même. L'exécution simulée de la combinaison de ces deux modèles a permis de vérifier, dès la conception, la conhérence du contrôleur par rapport aux exigences.

À la section 2, nous présentons un aperçu de la méthode de conception utilisée. La conception, suivant cette méthode, et la description en Mondel du contrôle centralisé pour le système de transmission sont discutées à la section 3. Nos conclusions sont données à la section 4.

2. APERÇU DE LA MÉTHODE DE CONCEPTION

Les langages de programmation laissent beaucoup de liberté aux utilisateurs. Des méthodologies sont nécessaires et utiles afin de guider le développement d'une application. L'approche de conception utilisée dans notre travail en est une par objets. Par contraste, les méthodologies traditionnelles sont soit orientées décomposition fonctionnelle, soit orientées conception de structures de données. D'une part, les méthodes basées sur la décomposition fonctionnelle, par exemple l'analyse structurée, mettent l'emphase sur l'identification de fonctions pouvant être combinées ensemble afin de produire des processus complexes de transformation des données. D'autre part, les méthodes orientées conception de structures de données, par exemple la méthode entité-relation, mettent l'accent sur les données, leur contenu et les relations qui les unies. Les méthodologies orientées objet intègrent les orientations "décomposition fonctionnelle" et "conception de structures de données". Le concept élémentaire est celui d'objet. Les diagrammes entité-relation étendus peuvent être utilisés pour formaliser la structure de base des modèles orientés objet alors qu'un processus de décomposition fonctionnelle peut être adoptée lors de la définition des opérations sur les objets.

Il existe actuellement un bon nombre de méthodes de conception orientée objet (ex. [Coad 91a], [Coad 91b] et [Booc 91]). Bien qu'elles puissent différer quant à la dénomination des étapes et des concepts, la plupart sont d'accord pour dire que la conception par objets est un processus itératif basé sur l'intuition et l'expertise humaine et comprenant trois ou quatre grandes étapes principales.

La méthode de conception utilisée dans le présent travail est similaire aux méthodes susmentionnées. Elle a été proposée dans [Mond 90] et comprend quatre étapes principales définies comme suit:

Étape préliminaire - Définition du problème: Cette étape consiste à déterminer les exigences et à identifier les aspects du problème à traiter. Son caractère est plutôt informel.

Étape 1 - Définition du domaine: L'étape 1 consiste à identifier les classes d'objets du domaine et les types de relations qui les associent. Le résultat de cette étape est un modèle conceptuel exprimant la structure du domaine. Le modèle conceptuel est décrit au moyen d'un diagramme entité-relation étendu. Ce diagramme montre les classes d'objets, les attributs et les relations d'héritage entre classes d'objets.

Étape 2 - Identification des opérations: Suivant l'orientation objet, le traitement sur les données est décrit au moyen d'opérations associées aux objets. Durant l'étape deux, nous identifions et associons aux classes d'objets des opérations. Ces dernières sont dérivées à partir des fonctionnalités qui ont été exigées lors de la définition du problème. Les opérations sont distribuées de façon à ce que chaque objet “encapsule” les procédures requises pour modifier ou accéder à sa structure interne.

Étape 3 - Définition des comportements: L'objectif de l'étape trois est de définir la sémantique de chacune des opérations et les contraintes sur l'ordre dans lequel elles peuvent être accomplies. Plusieurs styles de définition peuvent être envisagés. Chaque opération peut être définie par un algorithme. En outre, les diagrammes état-transition conviennent bien pour décrire le comportement d'un objet qui possède un attribut modélisant son état interne et dont les transitions sont provoquées par l'exécution des opérations.

Ce processus de conception est itératif. L'identification et la définition des opérations conduisent en général à la découverte de nouvelles classes d'objets qui doivent être intégrées au modèle de la structure du domaine. L'application de cette méthode à un problème de contrôle de système de transmission est le sujet de la section suivante. Le langage de description orientée objet Mondel sera introduit au moyen de l'exemple.

3. CONCEPTION DU CONTRÔLEUR

La conception du modèle du système de transmission et de son contrôleur, suivant l'approche introduite dans la section 2, et la description en Mondel sont les sujets de la présente section. Dans notre approche, nous developpons à la fois le modèle de l'équipement contrôlé et le modèle de son contrôleur. Le modèle de l'équipement décrit précisément le domaine contrôlé. Par ailleurs, le langage Mondel est exécutable [Will 90]. La composition de la description Mondel de l'équipement avec la description Mondel du contrôleur permet de vérifier par exécution simulée l'exactitude de la conception du contrôleur par rapport aux exigences.

3.1 Étape préliminaire - Définition du problème

Nous considérons le problème de la conception du contrôleur pour le système illustré à la figure 1. Il s’agit d’un site terminal d’un réseau de transmission de données. Sa fonction consiste à recevoir des signaux basse vitesse, au niveau des puits s1, s2 et s3, les décoder, régénérer et ré-encoder; au moyen des syndes; pour retransmission au niveau des puits t1, t2 et t3. Le terme synde signifie *synchronizer/desynchronizer*. Des connexions dynamiques peuvent être établies dans les commutateurs. On peut spécifier par une matrice “A” les connexions permises entre les ports d’entrée et de sortie d’un commutateur (voir figure 2). Chaque élément “A_{ij}” de la matrice indique qu’une connexion peut être activée du port d’entrée “p_i” vers le port de sortie “q_j”. Un port ne peut pas être

relié simultanément dans une même direction à plus d'un autre port. À partir des puits, les signaux sont multiplexés en un signal haute vitesse pour transmission via un canal optique. Le multiplexeur et le canal optique ne sont pas considérés explicitement dans cet article. Nous nous intéressons plutôt à la conception du contrôleur dont le but est d'accepter les messages d'alarme, produits par l'équipement, et déterminer les signaux de contrôle dont l'objectif est de maximiser le temps durant lequel le service de transmission est disponible.

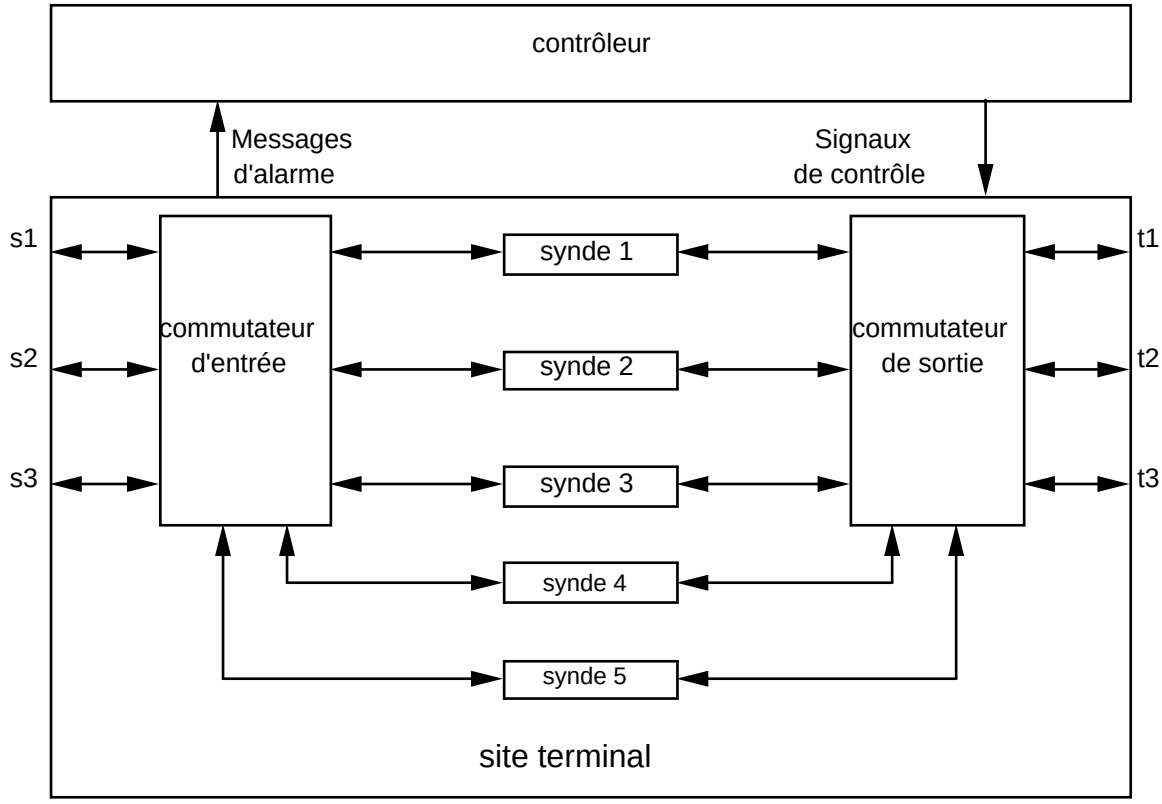
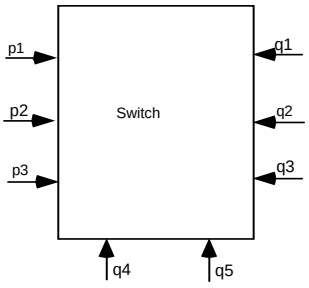


Figure 1: Système de transmission



Matrice A

ports de sortie

		q1	q2	q3	q4	q5
ports	p1	1			1	1
d'	p2		1		1	1
entrée	p3			1	1	1

Figure 2: Connexions dynamiques dans un commutateur

L'aspect de ce système dont nous voulons nous occuper est la récupération des pannes au niveau des syndes et leur impact sur le service de transmission. De plus le modèle de l'équipement devra représenter: i) son aspect logique, ii) son aspect physique, et iii) les relations de soutien existant entre les aspects logique et physique. D'une part, l'aspect logique du système est constitué d'une structure hiérarchique de composantes logiques offrant des services de transmission; d'autre part, l'aspect physique comprend des composantes physiques qui implantent les composantes logiques. Les messages d'alarmes produits par l'équipement concernent les éléments physiques. La représentation des relations de soutien qui existent entre les éléments physiques et logiques permettra de déterminer quels sont les composantes logiques et les services de transmission affectés par les messages d'alarmes.

On distingue les composantes logiques suivantes: le commutateur d'entrée, les trois syndes normaux (1, 2 et 3), les deux syndes de secours (4 et 5) et le commutateur de sortie. Les composantes logiques comprennent les ports logiques (représentés par des flèches). Le site terminal comprend les ports-source et les ports-puits. Les composantes logiques sont reliées entre-elles via leurs ports communs.

La structure logique de la figure 1 est implantée (ou soutenue) par des composantes physiques appelées cirscuits. Il existe un certain nombre de types de circuits, soient les blocs d'alimentation, les commutateurs, les unités synde normales et les unités synde de secours. Une des principales propriétés d'une composante physique est de pouvoir tomber en panne. Si un bloc d'alimentation tombe en panne, tous le circuits qui en dépendent tombent également en panne.

On peut distinguer deux catégories d'événements pouvant survenir, soient les événements incontrôlables et les événements contrôlables. Les événements incontrôlables surviennent dans l'équipement et sont toujours acceptés par le contrôleur alors que les événements contrôlables sont provoqués par le contrôleur. Les événements incontrôlables considérés ici sont les pannes et les réparations au niveau des circuits de type “unité synde”. Les événements contrôlables consistent en l'activation et l'arrêt du fonctionnement des composantes logiques et des connexions dynamiques entre ports d'entrée et de sortie d'un commutateur.

3.2 Étape 1 - Définition du domaine

L'étape un consiste à définir les classes d'objets, leur contenu, l'héritage et les relations entre les membres des classes d'objets. Le résultat de cette étape est exprimé au moyen d'un diagramme entité-relation étendu. Le diagramme entité-relation pour notre exemple est illustré à la figure 3.

Les classes d'objets sont représentées par des rectangles. Notre système de transmission comprend un sous-domaine logique et un sous-domaine physique. Le sous-domaine logique (partie gauche de la figure) consiste en une hiérarchie de composantes logiques. Les composantes logiques sont spécialisées en

processus site terminal, commutateur et synde. Ces relations d'"héritage" sont représentées, sur la figure, par des lignes brisées. Chacune d'elles va d'une classe spécialisée vers une classe plus générale. Les composantes logiques comprennent des ports logiques qui se subdivisent en trois spécialisations représentées par les sous-classes port-source, port-puits et port-intermédiaire. Le sous-domaine physique comprend (partie droite) des circuits qui peuvent contenir des ports physiques (ex. une unité synde) ou non (ex. un bloc d'alimentation).

Dans cet exemple, il y a cinq types de relations entre les objet: i) fait-partie-de: Elle définit la hiérarchie de composition des classes, c'est-à-dire, comment les objets d'une classe entre dans la composition des objets d'une autre classe. ii) connexion: Elle représente le fait qu'il est possible de connecter directement un port logique à un autre. iii) port physique-port logique: Elle établit les correspondances entre les ports physiques et les ports logiques soutenus. iv) composante physique-composante logique: Elle exprime les correspondances entre les composantes physiques et les composantes logiques. v) liaison: Elle associe les ports-source aux ports-puits.

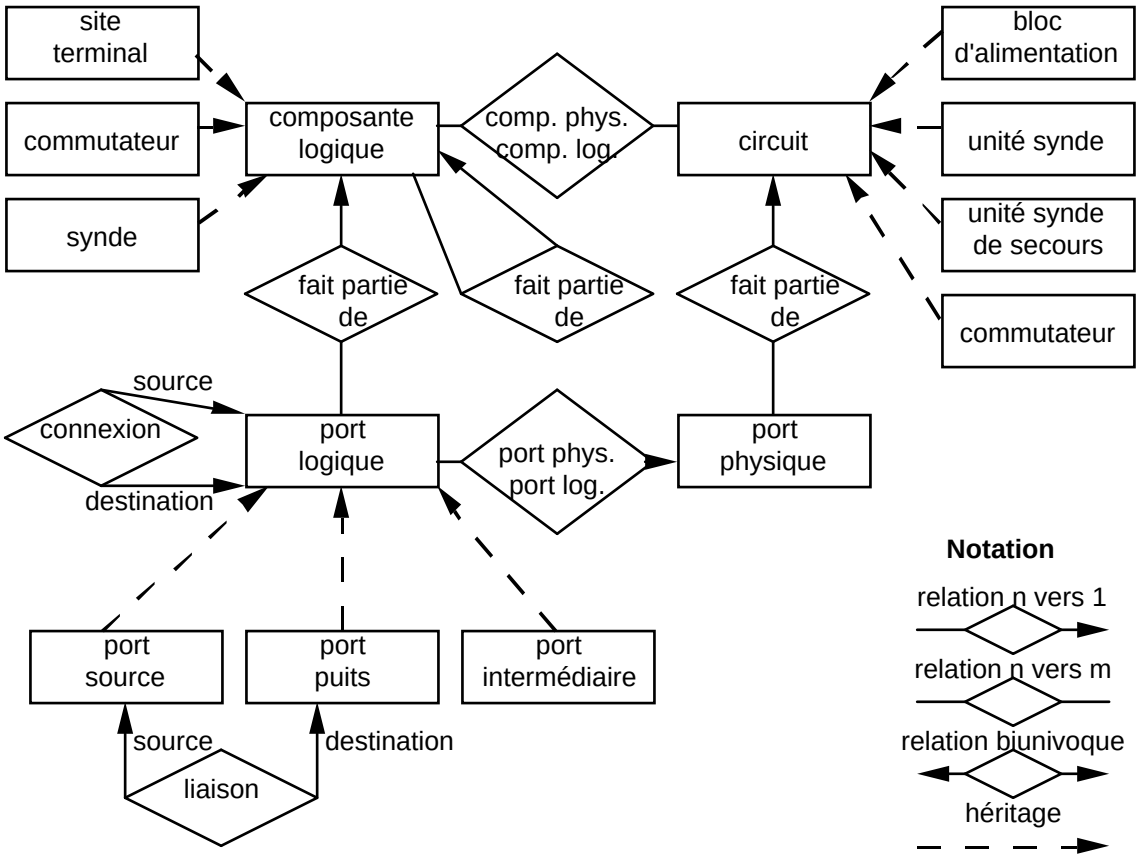


Figure 3: Diagramme entité-relation du domaine contrôlé

En Mondel, les classes d'objets et les classes de relations sont décrites par des types d'objets. Une spécification Mondel possède la structure suivante:

unit spec =
    ...définitions de types d'objets...
behavior
    ...instructions créant la configuration initiale...
endunit

Les relations entre instances d'objet peuvent être décrites de plusieurs façons [Boch 90b]. Une relation n vers 1 peut être représentée par un attribut dans la classe d'objets du côté n de la relation. Par exemple, les classes d'objets “composante logique” et “site terminal” peuvent être définies par les types Mondel suivants:

type composanteLogique = object **with**
    composé : composanteLogique opt;
endtype

type siteTerminal = composanteLogique
endtype

Dans cet exemple, la relation “fait partie de” est représentée par l’attribut appelé composé. Le mot-clé “opt ” veut dire que l'attribut peut prendre la valeur “nil”, c'est-à-dire, pour la composante logique à la racine de la hiérarchie de composition.

Dans une définition de type, le ou les noms des parents dans la hiérarchie d’héritage sont indiqués à la suite du nom du type courant et du symbole “=”. La création d'objets à partir de définitions de types est obtenue en Mondel en utilisant l'instruction “new”.

Les classes de relations n vers m peuvent être représentées par une approche dite “base de données relationnelle”. Dans cette approche, la classe de relations est modélisée par un type d'objets Mondel composé d'attributs représentant les objets reliés. Par exemple, la relation “composante physique-composante logique” peut être traduite de la façon suivante:

type compPhysCompLog = object **with**
    compPhys:composantePhysique; compLog:composanteLogique;
endtype

L'étape suivante de la méthode consiste à déterminer quelles opérations pourront être effectuées sur les objets membres de ces classes.

3.3 Étape 2 - Identification des opérations

Les événements pouvant survenir au niveau du système de transmission peuvent être classés en deux catégories, soient les événements contrôlables et les événements incontrôlables. Les événements incontrôlables sont modélisés par des opérations sur le contrôleur. Inversement, les événements contrôlables sont exprimés par des opérations sur les composantes du site terminal. Dans cet article, nous considérons principalement les événements incontrôlables “panne synde” et “réparation synde”.

Une panne de synde “logique” est causée par la défaillance d'une composante physique la soutenant. Cette défaillance est d'abord propagée de la composante physique vers la composante logique puis indiquée au contrôleur au moyen d'une opération appelée “panneSynde”. La définition de cette opération est le sujet de la sous-section suivante.

3.4 Étape 3 - Définition des comportements

À l’étape trois, nous définissons le comportement des instances de chaque type d’objets, c'est-à-dire, comment elles réagissent aux appels d'opération. Le comportement est d’abord modélisé par des diagrammes état-transition. Il n'est pas commode de décrire tous les aspects du comportement d'un objet; les variables d'état, les paramètres d'opération et les prédicats; avec des diagrammes état-transition. Les diagrammes état-transition sont utilisés pour définir la structure générale des comportements. Les autres aspects sont détaillés durant la traduction en Mondel des diagrammes état-transition.

Le diagramme état-transition du contrôleur est présenté à la figure 4. Les états sont interprétés de la façon suivante: i) protégé: toutes les liaisons sont opérationnelles et un ou deux syndes de secours sont disponibles, ii) non-protégé: toutes les liaisons sont opérationnelles mais aucun synde de secours est disponible (soit ils sont en panne, soit ils ont remplacé des syndes normaux), iii) x panne(s): x connexion(s) est (sont) non opérationnelle(s).

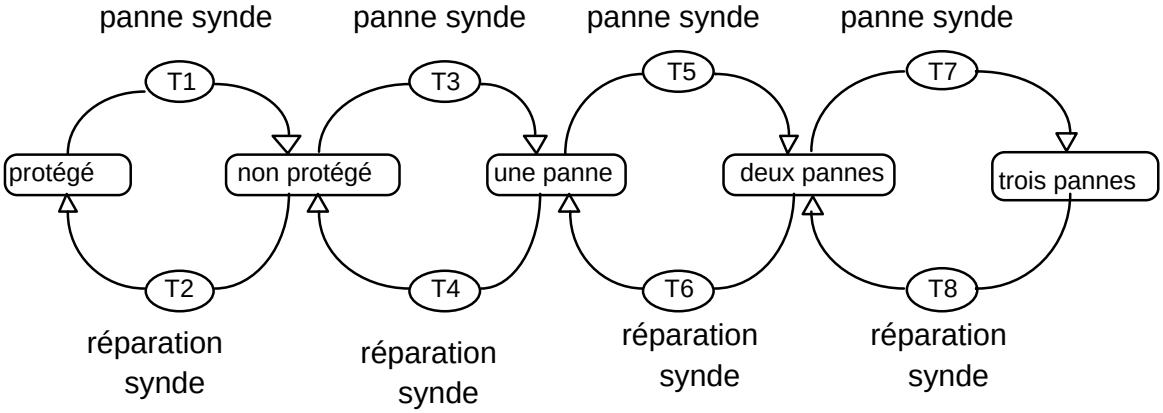


Figure 4: Diagramme état-transition du contrôleur

Nous discutons maintenant de la traduction des diagrammes état-transition dans la syntaxe de Mondel. Chaque état est exprimé au moyen d'une procédure Mondel et chaque transition sortante devient une alternative parmi les actions pouvant être exécutées par la procédure. Une alternative consiste en une instruction (*accept* ou autre), suivie d'un appel de procédure spécifiant l'état successeur. L'appel d'opération est exprimé en Mondel par une instruction de la forme: *Expr!OpName(ParamList)*, où l'évaluation de *Expr* dénote l'objet appelé. La possibilité pour un objet d'accepter l'appel d'une opération est exprimée par l'instruction *accept*. L'exécution de cette instruction provoque un rendez-vous entre l'appelant et l'appelé. L'instruction *accept* doit contenir une instruction *return* dont l'exécution produit un autre rendez-vous avec l'appelant qui s'attend à cette interaction.

L'état opérationnel des liaisons est enregistré dans les instances de la classe de relations “liaison”. Le type “liaison” est défini comme suit:

```
type liaison = object with
        source:portSource; puits:portPuits;
operation
        activer;
        desactiver;
behavior
        inactif;
where
procedure inactif =
        accept activer do return; end;
        actif;
endproc
procedure actif =
        accept desactiver do return; end;
        inactif;
endproc
endtype
```

Nous présentons ici les détails de la procédure modélisant l'état appelé “non protégé” du contrôleur. Selon la figure 4, de l'état non protégé deux alternatives sont possibles, c'est-à-dire, l'acceptation de l'opération “réparation synde” (T2) ou l'acceptation de l'opération “panne synde” (T3). Afin de simplifier la description du comportement, nous dinstinguerons les cas où un synde de secours est réparé (T2') et un synde normal est réparé (T2''). La possibilité de choisir, de l'état protégé, parmi ces trois alternatives est exprimée au moyen de l'instruction Mondel “choice”. La structure générale de la procédure modélisant l'état non protégé est la suivante:

```
procedure nonProtege =
choice
        ...Transition T2': Gérer la réparation d'un synde de secours “s”...
        or
        ...Transition T2'': Gérer la réparation d'un synde normal “s”...
        or
        ...Transition T3: Gérer une panne d'un synde “s”...
end;
endproc
```

L'alternative représentant la transition T2' est exprimée en Mondel comme suit:

```
{ Transition T2': Gérer la réparation d'un synde de secours “s” }
accept reparationSynde (s, est_de_secours)
provided est_de_secours do return; end;
```

protege; { fin de la transition T2' }

Afin de faciliter la description des alternatives T2'' et T3, nous rendons explicite deux relations entre les syndes et les liaisons. Une relation de la classe “soutien” représente le fait qu'un synde soutienne une liaison donnée. En outre, une relation de la classe “peut soutenir” représente le fait qu'un synde puisse soutenir une liaison donnée, selon son état opérationnel. Pour sélectionner une relation des classes “soutien” et “peut soutenir”, nous utilisons l'instruction Mondel “ifexist” qui est similaire aux instructions “select” ou “retrieve” que l'on retrouve dans les bases de données. Le format général du “ifexist” est le suivant: **ifexist** O:T **suchthat** E **then** I. L'exécution de cette instruction a pour but la sélection d'un objet “O” de type “T” tel que l'expression booléenne “E” est satisfaite puis d'accomplir l'instruction “I”. Si un tel objet n'existe pas, le contrôle passe à l'instruction suivant le “ifexist”.

Notez que durant la définition des comportements, il est parfois nécessaire d'ajouter de nouveaux attributs qui représentent les relations d'appel entre objets, si celles-ci ne coïncident pas avec la représentation de d'autres associations déjà exprimées. L'attribut “siteTerminal” est défini dans le contrôleur et les attributs “commutateurEntree” et “commutateurSortie” sont définis dans un site terminal afin que le contrôleur puisse appeler l'opération “commuter” sur les composantes du site terminal. Ces attributs expriment aussi la réciproque de la relation “fait partie de”. L'expression:

“siteTerminal.commutateurEntree”

représente la composante “commutateurEntree” du site terminal. L'alternative représentant la transition T2'' est traduite en Mondel de la façon suivante:

```
{ Transition T2'': Gérer la réparation d'un synde normal “s” }
accept reparationSynde (s, est_de_secours)
provided not est_de_secours do
        { déterminer la liaison qui peut être soutenue par le synde “s"
        (elle est unique car “s” n'est pas un synde de secours) }
        ifexist a:peut_soutenir suchthat a.synde=s then
                { déterminer le synde de secours qui soutien “a.liaison” }
                ifexist b:soutien suchthat b.liaison=a.liaison then
                        { arrêter le fonctionnement du synde de secours }
                        b.synde!arreter;
                        { détruire la relation de soutien }
                        b!delete;
                end;
                { démarrer le fonctionnement du synde “s” }
                s!demarrer;
                { commuter la liaison “a.liaison”
                (“portEntree” et “portSource” sont deux attributs du synde “s”) }
                siteTerminal.commutateurEntree!commuter(
                        a.liaison.source,
```

```
                    s.portEntree);
                siteTerminal.commutateurSortie!commuter(
                    s.portSortie,
                    a.liaison.destination);
                { créer la nouvelle relation de soutien }
                new soutien (s, a.liaison);
        end; { ifexist }
end;
protege; { fin de la transition T2" }
```

L'alternative représentant la transition T3 est traduite en Mondel de la manière suivante:

```
{ Transition T3: Gérer la panne d'un synde “s” }
accept panneSynde(s) do
{ “s” est le synde en panne }
        { désactivation de la relation de liaison }
        ifexist a:soutien suchthat a.synde=s then
                a.liaison!desactiver;
                a!delete; { détruire la relation de soutien }
        end;
end;
unePanne; { fin de la transition T3 }
```

Les classes de relations “soutien” et “peut soutenir” illustrent bien le caractère itératif de la conception par objets. Dans cet exemple, deux nouvelles classes de relations sont découvertes dont l'intégration dans le modèle entité-relation, développé à une étape antérieure, réduit la complexité de la définition des comportements.

4. CONCLUSIONS

Les méthodes proposées jusqu'à maintenant pour la conception de contrôleurs ont été principalement tirées de la théorie du contrôle. Une alternative a été discutée dans cet article. Elle repose sur une méthode de conception par objets [Mond 90] et un langage de description orientée objet [Boch 91]. Le produit initial de la méthodologie est un diagramme entité-relation étendu. Il donne la structure de base à la forme achevée de la conception qui est une description en Mondel de l'équipement contrôlé et de son contrôleur. L'exactitude de la conception est vérifiée par une exécution simulée de la composition des modèles du contrôleur et de l'équipement contrôlé.

Un algorithme de contrôle distribué a également été développé pour le système de transmission traité à la section 3. Alors que notre approche centralisée est plutôt spécifique à l'équipement considéré, l'algorithme distribué s'applique à des systèmes de transmission plus généraux. Un autre aspect important de notre travail est la vérification exhaustive des descriptions formelles. L'algorithme distribué ainsi que la méthode de vérification exhaustive sont discutés dans [Barb

91]. Notre approche intègre donc les aspects conception, spécification et vérification de contrôleurs centralisés ou distribués.

REMERCIEMENTS

Ce travail a été réalisé dans le cadre d'un projet conjoint du Centre de recherche informatique de Montréal et Recherches Bell-Northern. Ce travail a été effectué pendant que le second auteur était chercheur postdoctoral à l'Univesité de Montréal et il remercie l'Institut canadien de recherche en télécommunications pour le soutien financier. Les auteurs remercient M. Angelo E. Bean de Recherches Bell-Northern et M. Anindya Das de l'Université de Montréal pour plusieurs discussion fructueuses.

RÉFÉRENCES

[Barb 90] M. Barbeau, G. v. Bochmann, "Formal Semantics of Mondel", Progress Report for CRIM/BNR Project, June 1990.

[Barb 91] M. Barbeau, P. de Saqui-Sannes, G. v. Bochmann, "Design, Formal Specification and Validation of Centralized and Distributed Control in a Transmission System", Quality Engineering Workshop, Ottawa, 1991.

[Boch 90a] G. v. Bochmann, "Protocol Specification for OSI", Computer Networks and ISDN Systems, Vol. 18, No. 13, April 1990, pp. 167-184.

[Boch 90b] G. v. Bochmann, P. Mondain-Monval, S. Poirier, L. Lecomte, M. Barbeau, N. Williams, "System Specification With Mondel and Relation With Other Formalisms", Progress Report for CRIM/BNR Project, June 1990.

[Boch 91] G. v. Bochmann, M. Barbeau, A. Bean, M. Erradi, L. Lecomte, A. Liu, "The Specification Language Mondel V1", Technical Report from CRIM, April 1991.

[Booc 91] G. Booch, "Object-Oriented Design with Applications", Benjamin/Cummings, 1991.

[Coad 91a] P. Coad, E. Yourdon, "Object-Oriented Analysis", Second Edition, Yourdon Press Computing Series, 1991.

[Coad 91b] P. Coad, E. Yourdon, "Object-Oriented Design", Yourdon Press Computing Series, 1991.

[Gold 88] A. V. Goldberg, R. E. Tarjan, "A New Approach to the Maximal-Flow Problem", J. of ACM, Vol. 35, No. 4, October 1988, pp. 921-940.

[Groz 85] R. Groz, C. Jard, C. Lassudrie, "Attacking a Complex Distributed Algorithm from Different Sides: An Experience with Complementary Validation Tools", in: Y. Yemini, R. Strom, S. Yemini (Eds.), Protocol, Specification, Testing and Verification, Elsevier Science Publishers B. V. (North-Holland), 1985, pp. 3-17.

[Mond 90] P. Mondain-Monval, G. v. Bochmann, "An Object-oriented Software Design Methodology", Progress Report Document No. 7 for CRIM/BNR project, June 1990.

[Rama 89] P. J. G. Ramadge, W. M. Wonham, "The Control of Discrete Event Systems", Proceedings of the IEEE, Vol. 77, No. 1, January 1989, pp. 81-97.

[Will 90] N. Williams, “Un simulateur pour un langage de spécification orientée objet”, Rapport de stage de maîtrise, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1990.