Du test de conformité des systèmes d'administration de réseaux

Paul KABORE, André SCHAFF
CRIN & Université Henri POINCARE Nancy1
BP 239, 54506 Vandoeuvre-les-Nancy Cedex France

Résumé

Pour le test de conformité des protocoles de communication, l'ISO a défini un cadre et une méthodologie de test. En suivant cette méthodologie, des techniques de sélection automatique de tests ont été proposées. Cependant, pour le test des couches hautes du modèle OSI, cette méthodologie et ces techniques ne sont plus applicables. Cela est dû au fait que les protocoles de ces couches ont une architecture spécifique et utilisent des structures de données très complexes. Il devient alors impératif de définir des approches de test appropriées, prenant en compte tous les aspects des applications OSI. Dans ce papier nous discutons des approches et procédures de test pour les couches hautes du modèle OSI et plus particulièrement pour les systèmes d'administration de réseaux.

Mots-clés: OSI, couches hautes, systèmes d'administration de réseaux, test de conformité, ASN.1

1 Introduction

Dans le cadre du test de conformité des protocoles de communication, l'ISO[1] a défini dans la norme ISO 9646 [ISO-9646 91], un cadre et une méthodologie générale de test. Cependant, un certain nombre de problèmes se posent lorsqu'il s'agit de tester les couches hautes du modèle OSI[2]. Ces problèmes sont dûs essentiellement à la spécificité de l'architecture et des fonctionnalités des couches présentation et application qui ne sont pas très bien couvertes par la norme. En effet, cette norme est beaucoup plus orientée vers le test des protocoles des couches basses, et, bien que certains aspects non protocolaires des couches hautes aient été normalisés par l'ISO, aucune approche ni procédure n'ont été proposées pour tester si les opérations des utilisateurs conduisent aux modifications attendues sur les ressources réelles du réseau. Pour remédier à cela, l'ISO a initié un travail sur le test de conformité des couches hautes, en prenant pour référence la norme ISO 9646.

1. International Organization for Standardization
2. Open Systems Interconnection

Nous discutons dans ce papier des approches et procédures de test des couches hautes et plus particulièrement de systèmes d'administration de réseaux. Ces systèmes ont en charge la supervision, le contrôle et la comptabilité de l'activité des ressources physiques et logiques d'un réseau. L'utilisateur d'un tel système attend que les opérations d'administration qu'il effectue soient exécutées de manière correcte sur les ressources. Par exemple si l'utilisateur restreint l'accès à une ressource donnée par l'intermédiaire des services d'administration de réseaux, il attend que la disponibilité de cette ressource soit modifiée comme il le souhaite. Il est donc essentiel que les implantations soient conformes aux spécifications des besoins des utilisateurs. Il y a deux aspects dans l'administration de réseaux. D'une part il y a les protocoles OSI d'administration et d'autre part les autres composantes non protocolaires des applications d'administration. Une méthodologie de test complète doit prendre en compte ces deux aspects.

Dans cet article un modèle général de systèmes d'administration de réseaux a été donné. Sur la base de ce modèle des configurations possibles de test ont été proposées suivant les principes de la norme ISO 9646. Dans la section **2** de ce papier nous présentons la méthodologie de test définie dans la norme ISO 9646. Nous discutons ensuite dans la section **3** des apects qui sont spécifiques aux couches hautes et à l'administration de réseaux dans le cadre du test de conformité et en section 4 de l'applicabilité de la norme ISO 9646 et des méthodes de génération automatiques de tests. Nous présentons ensuite une approche pour tester des systèmes d'administration de réseaux en section **5**. Nous présentons enfin quelques travaux futurs et donnons une conclusion à l'article.

2 La méthodologie de test OSI

Suivant le principe du modèle en couches OSI [ISO 84], deux entités de protocoles homologues communiquent en appliquant un ensemble de règles définies à l'aide d'un automate de protocole. Une spécification formelle de protocole définit de façon rigoureuse ces diverses règles. Pour s'assurer que les implantations de protocoles sont conformes aux spécifications, l'ISO a défini dans la norme ISO 9646 une méthodologie de test de conformité des protocoles OSI [ISO-9646 91]. Nous montrons dans ce qui suit les concepts clés de cette méthodologie.

2.1 Le principe du test de conformité

Pour permettre une communication correcte entre systèmes ouverts, il est nécessaire que les implantations d'un protocole de la couche(N) dans tout système ouvert soient conformes à la norme qui définit ce protocole. Le test de conformité de protocole permet de vérifier que l'implantation d'un protocole satisfait sa spécification.

Dans le test de conformité seul le comportement visible est testé, i.e. les interactions d'une implantation de protocole avec son environnement;

aucune référence n'est faite à la structure interne de l'implantation. Ceci vient du fait d'une part que les normes OSI ne spécifient que les règles d'échange entre les systèmes et non la manière de les réaliser; et d'autre part que pendant l'exécution du test de conformité, la structure interne de l'implantation n'est, la plus part du temps, pas accesssible au testeur. Le système informatique dans lequel se trouve l'implantation sous test (IUT) peut ne pas être accessible. Le testeur peut seulement observer l'IUT en communiquant avec lui.

Le test d'une implantation va donc porter sur son comportement externe, c'est-à-dire sur ses interfaces accessibles ou SAP[3] (cf figure 1 [Rafiq 91]).

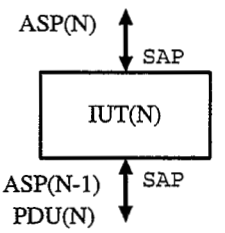


FIG. 1 - *Interface d'une implantation de protocole*

Conceptuellement, le test consiste à effectuer des échanges d'ASP(N)[4] et de PDU(N)[5] dérivés de la spécification avec l'implantation (N) à tester, et à comparer les réponses obtenues avec celles attendues. Un verdict basé sur le résultat de cette comparaison est formulé sur la correction de l'implantation.

2.2 Les méthodes abstraites de test

Comme nous venons de le voir, le test de conformité consiste à vérifier le comportement d'une entité de protocole à ses points d'accès supérieur et inférieur (SAP(N) et SAP(N-1)). Cependant ces points ne sont pas toujours accessibles directement par le testeur. Les points où le testeur observe et contrôle l'IUT sont appelés point d'observation et de contrôle (PCO). Normalement il y a deux PCOs: l'un correspond au SAP(N) et l'autre au SAP(N-1) de l'IUT notés respectivement PCO(N) et PCO(N-1). La partie du testeur qui observe et contrôle le PCO(N) est appelé testeur supérieur (UT) et l'autre testeur inférieur (LT).

Une méthode abstraite de test ou architecture de test définit un modèle d'accès à l'IUT par les PCOs. Dans la norme ISO 9646 des méthodes de test ont été définies dans lesquelles le SAP(N-1) est toujours accessible, le SAP(N) peut ne pas l'être. Nous montrons sur la figure 2 une de ces méthodes appelée méthode de test distribuée ou DS-Method[6] . Cette méthode

3. Service Acces Point
4. Abstract Service Primitive
5. Protocol Data Unit
6. Distributed Single-layer test Method

prévoit le contrôle et l'observation de l'interface supérieure de l'IUT par un UT qui se trouve dans le système sous test. La spécification de chaque test implique la définition du comportement de l'UT et du LT séparément. Les événements de test sont spécifiés sous forme d'ASP(N) au dessus de l'IUT et d'ASP(N-1) et PDU(N) au niveau de l'entité paire distante. L'existence de deux testeurs indépendants et éloignés engendrent des problèmes de coordination concernant l'ordre d'exécution des différents tests et la récupération des résultats. L'ajout de procédures de coordination des tests devient alors nécessaire.

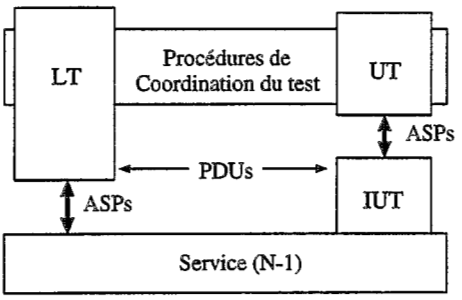


FIG. 2 - *Méthode de test distribuée*

2.3 La spécification des suites de tests

La norme ISO 9646 distingue plusieurs phases dans le processus de spécification des suites de tests.

La `première phase` consiste à spécifier une suite abstraite de tests pour le protocole considéré. Cette phase est connue sous le nom de *génération* ou *dérivation* de tests. Cette suite est abstraite dans le sens qu'elle a été développée indépendamment de toute implantation. La génération des tests consiste à dériver systématiquement des cas de test à partir d'une spécification du protocole. Le cas de test indique la succession d'actions à effectuer pour atteindre un objectif de test donné, sans tenir compte de la méthode de test qui sera utilisée ni de l'environnement dans lequel le test sera réalisé. Des cas de test abstraits sont ensuite dérivés à partir des cas de test génériques, en prenant en compte la méthode de test utilisée et les restrictions imposées par l'environnement de test.

La `seconde phase` consiste à déterminer une suite de tests particulière, exécutable pour une implantation donnée. Ceci s'appelle *sélection* ou *implantation* de tests. Les particularités de l'IUT sont à ce moment prises en compte.

La `dernière phase` est l'*exécution* du test. La suite de tests sélectionnée est exécutée avec l'IUT sur un testeur réel et les réponses correspondantes sont observées. Ceci conduit à un verdict sur la conformité de l'IUT par rapport à la spécification du protocole.

En se basant sur ce processus de spécification des tests, des méthodes de génération et de sélection automatiques de tests ont été proposées. Ces méthodes supposent que la spécification du protocole peut être décrite sous forme d'automates à états finis ([Kohavi 78]) ou de systèmes de transitions étiquetés ([Milner 89]).

2.4 La notation de tests TTCN

Les suites de test sont spécifiées dans une notation appelée TTCN[7] définie dans [ISO-9646-3 91]. TTCN permet de décrire de manière précise, pour chaque cas de test, les séquences d'événements à envoyer à l'IUT et celles attendues i.e., les séquences d'entrées et de sorties qui surviennent au niveau des PCOs. Il décrit en fait le comportement attendu de l'implantation du protocole sous forme d'arbre. Un exemple de spécification en TTCN [Rafiq 91] d'un cas de test du protocole de transport est donné sur la figure 3 et a pour but de vérifier que l'implantation peut accepter une demande de transfert de données. En (1) le testeur envoie par l'UT une demande de connexion de transport T-CONreq à l'IUT. S'il reçoit après un T-DISind de l'IUT (indication de déconnexion), le protocole est refusé (Fail). Par contre si le LT qui contrôle la partie inférieur de l'IUT reçoit un CR (demande de connexion), le processus du test continue en séquence et le LT doit envoyer un CC (confirmation de connexion). En (7), le message reçu par l'UT (T-DISind) ne permet pas de savoir si l'IUT peut accepter une demande d'envoi de données (T-DTreq): après l'envoi de T-DTreq, l'UT reçoit un T-DISind (indication de déconnexion). Mais ce cas est aussi prévu par le protocole (on peut envoyer à tout moment une demande de déconnexion) et non pris en compte par la présente fiche.

3 La couche application et l'administration de réseaux

La couche application OSI a une structure complexe. Contrairement aux couches basses, les entités de protocole de la couche application ne sont pas composées d'une seule entité qui interagit avec d'autres entités via ses interfaces d'interaction supérieure et inférieure. La couche application est composée de plusieurs blocs ayant chacun une fonctionnalité bien définie. Chaque bloc, appelé élément de service d'application ASE[8], a son entité propre qui interagit avec l'entité paire distante en utilisant un protocole spécifique. Un ASE demande un service à un autre ASE en lui envoyant une primitive abstraite de service, ASP. A la réception d'un ASP, l'ASE exécute une action (par exemple construire une unité de donnée à envoyer) et demande à son tour un service à la couche présentation ou à un autre ASE. Ainsi, dans l'opération d'une application, une pile d'ASEs

7. Tree and Tabular Combined Notation
8. Application Service Element

Test Case Dynamic Behaviour

Reference : Protocole de transport / Testeur supérieur / Transfert de données simples
Identifier : PT / UT / TD1
Purpose : Le Testeur Supérieur Lance un Transfert de Données Simple
Defaults Reference : non

Behaviour Description	Label	Constraints References	Verdict	Comments	
TEST-PT [LT, UT]					
UT ! T_CONreq				Requête acceptable	(1)
#					
UT ? T_DISind			Fail		(2)
LT ? CR					(3)
#					
LT ! CC					(4)
#					
UT ? T_CONcnf				Connexion établie	(5)
#					
UT ! T_DTreq					(6)
#					
UT ? T_DISind			Inconclusive		(7)
LT ? DT					(8)

Extended Comments:

FIG. 3 - *Spécification TTCN d'un cas de test*

peut être utilisée. L'ensemble des ASEs pour une application particulière est déterminé par le contexte de cette application, qui est unique pour ce type d'application. Cependant, un certain nombre d'ASEs communs à un grand nombre d'applications ont été identifiés: ACSE[9], ROSE[10], RTSE[11], CCR[12]. Pour les applications d'administration de réseaux, les échanges entre les processus d'administration se font par l'intermédiaire d'entité d'application d'administration système SMAE[13] comportant trois ASEs (CMISE[14], ROSE et ACSE) communs à toutes les applications d'administration. La figure 4 montre le modèle d'un système d'administration de réseaux. Les ressources réelles à gérer sont modélisées par des objets appelés *objets gérés*. Un objet géré représente une vue abstraite de la ressource qu'il modélise et définit à son interface les opérations d'administration exécutables sur la ressource. Il peut émettre des notifications lorsque des événements particuliers surviennent sur la ressource. Tous les objets gérés d'un système ouvert constituent la base d'informations d'administration du système (MIB[15]). CMISE est un élément de service d'application permettant d'effectuer des

9. Association Contrôle
10. Remote Operation
11. Reliable Transfer
12. Commitment, Concurrency and Recovery
13. System Management Application Entity
14. Common Management Information Service
15. Management Information Base

échanges d'informations d'administration sous la forme de demande et/ou de demande-réponse. Il offre aux applications le moyen de réaliser sur les objets gérés des opérations et de rapporter les notifications émises par les objets. Six services ont été normalisés(M-EVENT-REPORT, M-GET, M-SET, M-ACTION, M-CREATE, M-DELETE). Pour réaliser ces échanges, le protocole CMIP qui fournit les services de CMISE utilise les services de l'élément de service réalisant les opérations distantes ROSE. Les opérations que CMIP demande d'effectuer sont spécifiées à l'aide de la RO-notation, i.e. des macros ASN.1. Les services de ROSE sont les suivants: RO-INVOKE, RO-ERROR, RO-REJECT-U, RO-REJECT-P. Toutes les valeurs des paramètres utilisés dans les primitives de service ROSE (exepté RO-REJECT-P) sont définis par l'utilisateur de ROSE. Les entités d'application communiquent grâce à une association d'application définie par négociation dans un contexte d'application. L'élément de service de contrôle d'association ACSE permet d'établir, de maintenir et de libérer une association d'application.

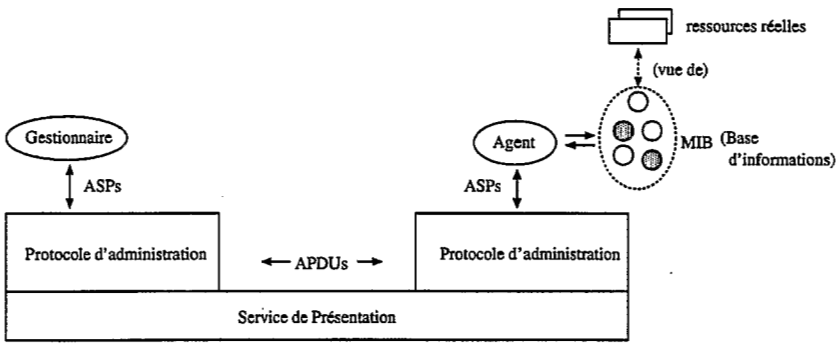


FIG. 4 - *Modèle d'administration de réseaux*

Comme nous venons de le voir, pendant la durée d'une connexion, la mise en oeuvre d'un «protocole de la couche application» et en particulier d'administration de réseaux, est une association dynamique d'ASEs d'une part et de leurs protocoles individuels d'autre part. Pour cela, la couche application n'a pas un protocole unique à l'instar des couches basses. En plus de cela, contrairement aux protocoles des couches basses qui utilisent des structures de données simples, les ASEs ont comparitivement des structures de données très complexes. En revanche, la partie contrôle de ces protocoles est relativement simple.

4 Applicabilité de la méthodologie de test OSI

En définissant la norme ISO 9646, l'intention était de définir une méthodologie de test générale pour toutes les couches OSI. Cependant, force est de constater que les méthodes définies dans cette norme sont plustôt orientées vers le test de protocoles des couches basses. Ceci est surtout re-

marquable dans la définition de la notation de test TTCN. Comme indiqué dans la norme ISO 9646, l'objectif du test de conformité est de déterminer si les «systèmes sont conformes aux normes correspondantes». Les normes des applications OSI ne spécifient pas uniquement des services et des protocoles mais aussi les fonctionnalités de l'application toute entière. Cela fait que tout développeur de tests a le choix entre soit tester uniquement ces protocoles, soit tenir compte également des autres fonctionnalités de l'application. L'objectif principal du test de conformité étant d'assurer une communication correcte entre systèmes informatiques, la première option devrait suffire. Cependant, et principalement pour des applications OSI, le test de conformité doit également donner l'assurance aux utilisateurs que leur application fonctionne comme ils l'auraient voulu [Bekker 91]. Et donc le test des fonctionnalités non protocolaires de l'application doit également être pris en compte. Comme la norme ISO 9646 concerne uniquement le test de protocoles, cet aspect n'est pas pris en charge. Cette norme est spécialement orientée vers le test des couches ayant un protocole complexe mais définissable par des systèmes à transitions. De même, les techniques de génération automatique des suites de tests connues à nos jours ([Bochmann 88]) ne sont applicables que lorsque les spécifications peuvent être modélisées sous forme de systèmes à transitions finis et ne couvrent pas la partie données de ces spécifications.

La couche application peut être vue comme une couche ayant un protocole unique résultant de la combinaison des protocoles des différents ASEs. Cependant, ce protocole ne peut être défini à l'aide d'un système à transitions et la structure de donnée résultante est très complexe. Un test de ce protocole est un test multi-protocoles et multi-buts, puisque c'est un test d'une combinaison des protocoles des différents ASEs. Ce genre de test n'est pas défini dans la norme ISO 9646. Une alternative consiste à dériver des tests pour chaque ASE, en tenant compte des interactions entre ASEs et des interactions avec l'environnement. Des travaux ont été initiés pour définir une méthodologie de test pour les couches hautes du modèle OSI [N5080 90], [N3245 88] et des approches et configurations possibles de test ont été proposées dans [Bekker 91] et [Haven 91]. La section suivante montre comment ces approches peuvent être spécialisées ou étendues pour le test de systèmes d'administration de réseaux.

5 Le test des systèmes d'administration de réseaux

Dans cette section, nous montrons une approche de test des systèmes d'administration de réseaux, en nous basant sur le modèle présenté en section 3.

5.1 Test de la communication entre gestionnaire et agent

La communication entre le gestionnaire et l'agent concerne l'aspect protocole des systèmes d'administration de réseaux. Le test de cette communication est donc un test de protocole et peut être supporté par les méthodes abstraites que nous avons vues en **2.2**. Cependant un problème subsiste dû au fait que nous avons ici une combinaison de plusieurs protocoles à tester. En effet nous avons vu qu'un protocole d'administration de réseaux était composé d'un ensemble d'ASEs ayant chacun un protocole particulier. Une configuration et une suite de tests pour les applications d'administration de réseaux doivent donc prendre en compte les ASEs utilisés et comment ils sont utilisés. Une configuration de test et la spécification des suites de tests pour l'ensemble des ASEs peuvent se faire de deux manières [Bekker 91].

Une première approche consiste à combiner les suites de tests des différents ASEs pour donner une suite de tests multi-ASE comme le montre la figure 5.

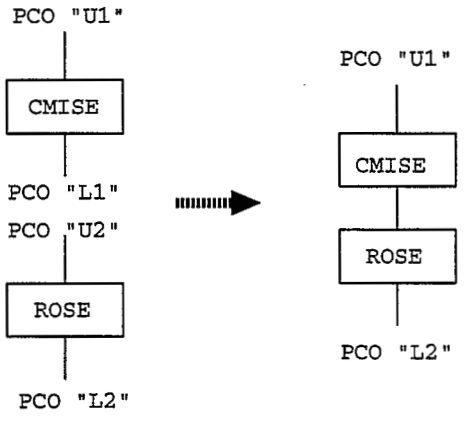
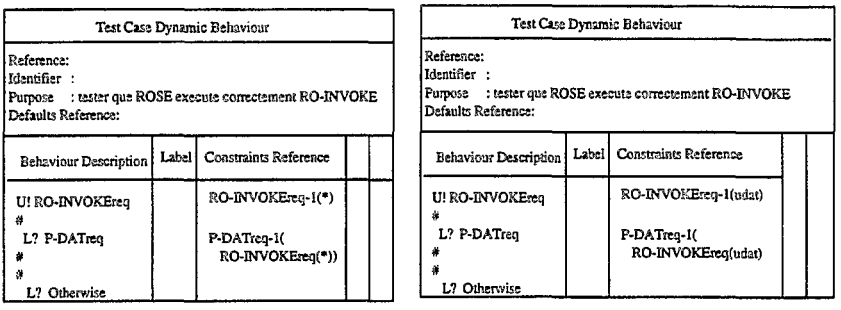


FIG. 5 - *Configuration de test multiple pour CMISE et ROSE*

Le problème avec cette approche est qu'on ne sait pas quel diagnostic donner lorsque le test échoue. Cela est dû au fait qu'on effectue un test multi-but et si le test révèle un comportement incorrect de l'implantation, cela ne veut pas dire que tous les ASEs sont mal implantés. Une deuxième approche consiste à tester individuellement chaque ASE. Elle consiste à spécifier des suites de tests pour chaque ASE indépendamment du contexte d'utilisation puis à combiner ces tests avec des informations dépendant du contexte. Cette combinaison produit une suite abstraite de tests pour l'ASE conforme à l'environnement dans lequel il sera utilisé. Considérons par exemple une suite de tests pour ROSE et supposons qu'on veut spécifier un test pour vérifier que l'opération RO-INVOKE est correctement exécutée. Comme nous l'avons dit plus haut, c'est l'utilisateur de ROSE qui doit donner la spécification des paramètres de l'opération. Un test pour cette opération, sans tenir compte de l'utilisateur, est un test paramétré par la

donnée utilisateur comme l'indique la figure 6 (a) et une instanciation de ce test par un utilisateur particulier est donné sur la figure 6 (b) où udat est le paramètre donné par l'utilisateur.



Test Case Dynamic Behaviour

Reference:
Identifier :
Purpose : tester que ROSE execute correctement RO-INVOKE
Defaults Reference:

Behaviour Description	Label	Constraints Reference		
U! RO-INVOKEreq # L? P-DATreq # # L? Otherwise		RO-INVOKEreq-1(*) P-DATreq-1(RO-INVOKEreq(*))		

(a)

Test Case Dynamic Behaviour

Reference:
Identifier :
Purpose : tester que ROSE execute correctement RO-INVOKE
Defaults Reference:

Behaviour Description	Label	Constraints Reference		
U! RO-INVOKEreq # L? P-DATreq # # L? Otherwise		RO-INVOKEreq-1(udat) P-DATreq-1(RO-INVOKEreq(udat)		

(b)

FIG. 6 - *Test du service RO-INVOKE*

Cette technique consiste donc à substituer dans les actions du test les paramètres des actions par des paramètres formels appropriés. Chaque test a un but unique et un verdict peut être donné à la suite de l'exécution du test. Une dérivation de tests utilisant cette méthode comprend donc deux parties: la dérivation de la suite de tests dépendant uniquement de l'ASE et la partie spécifique au contexte d'utilisation. La partie générale est réutilisable, alors que les tests pour la partie spécifique est reproduite à chaque fois, et pour cette raison elle doit être faite de manière automatique.

5.2 Test de l'agent

L'agent a pour but de convertir les ASPs en opérations sur les objets de la MIB, de retourner les réponses aux requêtes d'administration et transformer les notifications émises par les objets en ASP à émettre. Il doit en plus retourner le message d'erreur correspondant, lorsqu'il reçoit un ASP non valide pour un service confirmé. L'exécution de ces activités est indépendante de la MIB utilisée. Il doit donc être possible de tester l'agent en utilisant une base d'objets de référense, définie pour les besoins du test comme le montre la figure 7. Cette base d'objets peut être incorporée avec l'agent par l'implanteur, de sorte qu'on n'aie plus besoins d'autres contraintes sur le système sous test concernant la relation entre l'agent et les objets. La manière d'accéder aux objets à distance en utilisant le service OSI devrait suffire. Cependant, le contenu et la structure de cette base doivent être acceptés par tous pour que le verdict du test soit reconnu.

5.3 Test de la base d'objets

Pour pouvoir tester une base d'objets quelconque, une méthodologie générale de test doit être définie pour vérifier tous les aspects des ob-

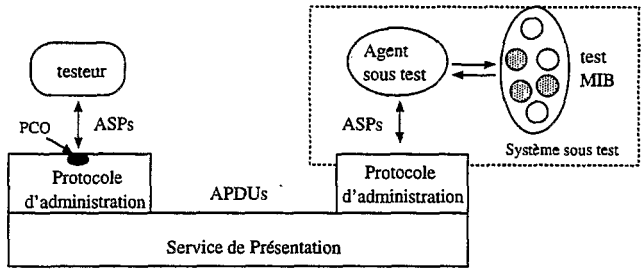


FIG. 7 - *Modèle de test de l'agent*

jets de la base. Les classes d'objets sont spécifiées en utilisant GDMO[16] [ISO-10165.4 92], une notation normalisée par l'ISO et le CCITT. Le test de la base d'objets doit vérifier que tout objet de cette base est correctement implanté par rapport à la spécification de sa classe. L'implantation d'une classe peut ne pas être conforme à sa spécification pour plusieurs raisons: d'une part des erreurs ont pu se glisser dans le code de l'implantation au moment de la programmation, d'autre part une interprétation erronée du comportement de la classe a pu être faite. C'est pourquoi ce comportement doit être défini de manière formelle. A partir de la spécification formelle d'une classe, une suite de tests peut être définie. Tous les aspects pouvant influencer le comportement d'un objet d'administration de réseaux sont résumés sur la figure 8. Une méthode de test doit permettre de tester tous les types de comportement.

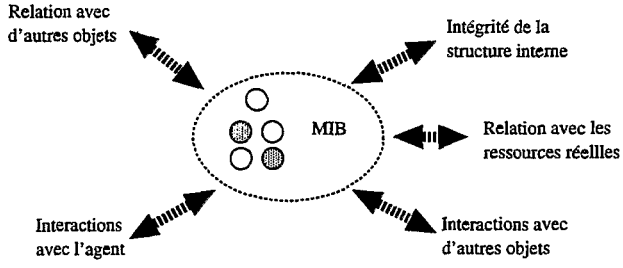


FIG. 8 - *Types d'interaction entre un objet et son environnement*

La configuration de la figure 9 ([Bekker 91] n'est plus valide puisqu'elle permet de tester uniquement le comportement des objets vis-à-vis de l'agent. Un test basé sur le modèle de comportement de la figure 8 peut se ramener à deux types de test ([Kaboré 94]): un test interne et un test externe. Le test externe a pour but de vérifier les interactions entre l'objet et son environnement, i.e. que l'objet accepte les requêtes et fournit les réponses appropriées. Ce comportement peut naturellement être modélisé par un sytème de transitions. Le test interne a pour but de vérifier le comportement

16. Guidelines for the Definition of Managed Ojects

des attributs et l'existence des relations. Les suites de tests doivent couvrir tous les cas de comportement possibles et doivent être basées sur une spécification des propriétés des attributs. Comme les attributs sont définis en ASN.1, cette spécification doit être faite sous forme de contraintes ASN.1. Dans une configuration de ce test, les objets qui sont en relation avec l'objet sous test sont représentés par des objets de test (cf figure 10 (a)). Un problème se pose par contre pour le test des relations avec les ressources réelles. En effet, même si une description de l'interface de ces ressources est disponible, il peut ne pas être possible de tester tous les aspects concernant ces relations pour des raisons par exemple de délais: un événement peut être très long à survenir, ce qui limite la testabilité de l'effet de cet événement. Nous pouvons remédier à ce problème en modélisant également les ressources par des objets de test (cf figure 10 (b)) dont les états peuvent être modifiés interactivement.

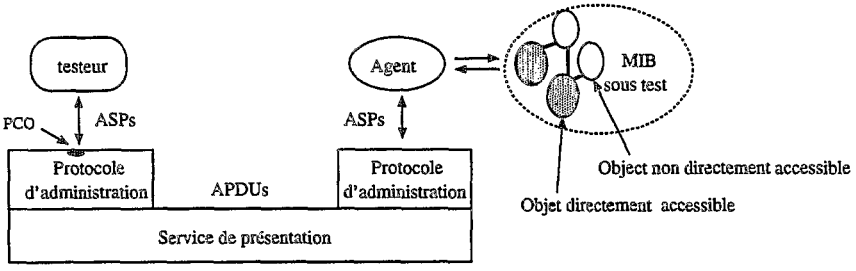


FIG. 9 - *Test de la MIB en utilant le service OSI*

Le test des objets n'est possible que si une interface est définie permettant aux testeurs d'accéder aux objets. Cette interface doit être indépendante de la façon dont les objets ont été implantés et du langage de programmation utilisé. Une telle interface doit être standardisée et offerte par tout implanteur d'une MIB aux testeurs. Une autre alternative consiste à tester les objets en utilisant les services offerts par le réseau (cf figure 9). Ceci ne devrait pas nécessiter d'autres contraintes sur l'accessibilité aux objets. Mais ce type de test aboutit à un test moins efficace car cela suppose que le service de communication offert par le réseau est fiable et correctement implanté.

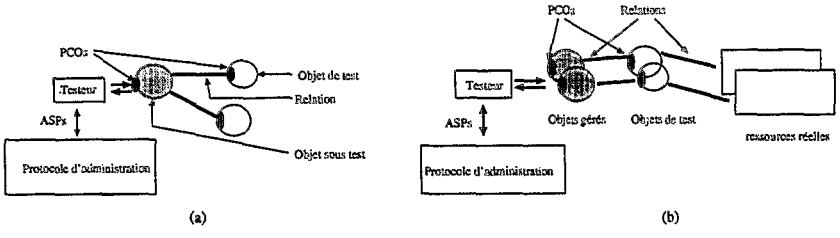


FIG. 10 - *Test des relations d'un objet géré*

6 Travaux futurs

Il est important, dans le domaine de l'administration de réseaux, de développer un environnement général de vérification et de test des objets GDMO et des protocoles utilisés, intégrant des outils de dérivation automatique de suites de tests. Nous avons vu que la spécification des suites de tests pour les différents ASEs et pour les objets GDMO devait tenir compte d'une part des propriétés des données manipulées et d'autre part du contexte d'utilisation. Il est concevable de spécifier les contraintes qu'impose l'environnement et les propriétés des attributs sous forme de contraintes sur des spécifications ASN.1 et dériver des tests à l'aide d'outils appropriés. Nos travaux consistent à chercher un formalisme de spécification de contraintes sur des définitions ASN.1 (peut-être suivant [Varvitsiotis 93]) et à générer des tests à partir de ces spécifications pour les protocoles d'administration et les objets GDMO. Un environnement MODE de validation de spécifications GDMO a déjà été développé. Cet environnement est en train d'être couplé avec un compilateur SNACC étendu pour pendre en compte la nouvelle version d'ASN.1. La prochaine étape consistera à intégrer dans l' environnement des outils de génération automatiques de tests et un environnement d'exécution des tests pour les systèmes d'administration de réseaux.

7 Conclusion

Nous avons présenté dans ce papier des approches de test pour des applications OSI et plus particulièrement pour les systèmes d'administration de réseaux. Nous avons présenté un modèle général de ces systèmes sur la base duquel nous avons montré les insuffisances de la méthodologie de test ISO 9646. Des configurations possibles de test ont été proposées pour tenir compte des spécificités de ce modèle.

En utilisant la méthode de test par ASE, et pour le test des objets GDMO, le travail le plus intéressant est la dérivation automatique de tests adaptés à un contexte d'utilisation particulier et aux propriétés des attributs des objets.

Références

[Bekker 91] M. Bekker et all. *Methodological Aspects of OSI Upper Layer Conformance Testing.* I. Davidson et D.W. Litwack, éditeurs, *Protocol Test Systems III*, pages 97–115. North-Holland, 1991.

[Bochmann 88] G.V. Bochmann, R. Dssouli et B. Sarikaya. Méthodes de test de protocoles: Architectures et sélection de tests. R. Castanet et O. Rafiq, éditeurs, *CFIP'88 - Ingénierie des protocoles*, pages 337–363. Eyrolles, Paris France, 1988.

[Haven 91] van der Haven, M.G. L Kockelmans et E.J. Slotboom. *Real Effects Testing of OSI Applications.* J. Kroon, R.J. Heijink et E. Brinksma, éditeurs, *Protocol Test Systems,* IV, pages 107–117. North-Holland, October 1991.

[ISO-10165.4 92] ISO, *Structure of Management Information - Part 4: Guidelines for the Definition of Managed Objects,* IS, ISO-10165.4, January 1992.

[ISO 84] ISO. *Information Processing System, Open Systems Interconnexion, Basic Reference Model.* International Standard IS-7498, ISO, 1984.

[ISO-9646 91] ISO-9646. Information processing system, open systems interconnexion, osi conformance testing and framework. Rapport no. ISO-IS 9646, ISO, 1991.

[ISO-9646-3 91] ISO-9646-3. Information processing system, open systems interconnexion, osi conformance testing and framework - part 3: The tree and tabular combined notation (ttcn). Rapport no. ISO-IS 9646-3, ISO, 1991.

[Kaboré 94] P. Kaboré et A. Schaff. *Objets de gestion de réseaux: une approche de test de conformité. JISI'94,* Tunis, Tunisie, 26-28 May 1994.

[Kohavi 78] Z. Kohavi. *Switching and Finite Automata Theory.* McGraw-Hill, New-York, 1978.

[Milner 89] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[N3245 88] ISO/IEC JTC1/SC21 N3245. Proposed new question on conformance to objects in the context of osi management. Rapport, ISO, December 1988.

[N5080 90] ISO/IEC JTC1/SC21 N5080. Call for contributions on osi management conformance issues. Rapport, ISO, May 1990.

[Rafiq 91] O. Rafiq. Le test de conformité des protocoles, une introduction. *Réseaux et informatique répartie,* 1(1):101–142, 1991.

[Varvitsiotis 93] A.P. Varvitsiotis et G.I. Stassinopoulos. Extending asn.1 into a full-fledged constraint language in the context of osi protocol conformance testing. *Computer Networks and ISDN Systems,* 25(11):1243–1263, June 1993.