Aide à l'intégration de services par la génération de tests

Gilles Bernot, Hélène Jouve
LaMI - Université d'Évry Val d'Essonne
{bernot, hjouve}@lami.univ-evry.fr

Francis Klay
CNET - France Télécom
Francis.Klay@cnet.francetelecom.fr

Farid Ouabdesselam, Jean-Luc Richier
LSR/IMAG
{Farid.Ouabdesselam, Jean-Luc.Richier}@imag.fr

Résumé

La modularisation des développements est un enjeu majeur du génie logiciel ; dans ce cadre nous présentons une approche originale pour appréhender le problème de l'intégration de services de télécommunication. Par définition, ces services interagissent entre eux, et l'ajout ou la modification d'un service peut introduire des interactions néfastes qui correspondent à un comportement indésirable du réseau.

La résolution des interactions néfastes en phase de spécification est problématique car si la complexité d'un service est humainement appréhendable, il en est tout autrement pour celle d'un ensemble de services. Dans un contexte concurrentiel, les coûts de développement, les délais de mise sur le marché et la qualité du service proposé sont des aspects économiquement essentiels. L'approche proposée prend en compte ces trois aspects conjointement pour aboutir à une méthode fondée sur la manipulation d'approximations, la génération automatique de tests et une indispensable intervention du concepteur qui reste limitée.

1 Introduction

Le réseau intelligent [13] est une architecture générale de réseau de télécommunication dont l'objectif est de concevoir, mettre en œuvre et maintenir des services avec souplesse et efficacité. Les services sont des fonctionnalités qui augmentent les possibilités et modifient les comportements de ce réseau. Ils interagissent entre eux et par conséquent, un ajout ou une modification de service peut introduire des interactions néfastes, c'est-à-dire gênantes, voir désastreuses du point de vue d'un utilisateur. Dans un contexte concurrentiel où les délais de mise sur le marché de produits fiables sont primordiaux, la maîtrise de l'intégration de nouveaux services est un enjeu essentiel [4]. Cependant, l'intégration est difficile du fait de la complexité des combinaisons de services possibles : l'ensemble des comportements qu'elles sont susceptibles d'engendrer est difficilement appréhendable par un humain sans assistance [10].

En matière d'interactions de services, tout comme en génie logiciel classique, les erreurs de conception doivent être détectées au plus tôt, c'est-à-dire dès la phase de spécification. Cette affirmation est encore plus vraie pour les interactions de services, car les interactions néfastes relèvent de la logique même des services plutôt que de leur réalisation. Cependant, on aborde ici un niveau de difficulté supérieur à celui qu'on rencontre en génie logiciel classique, car une interaction néfaste ne relève pas d'une erreur de conception d'un service, hormis quelques cas rares d'interactions parfaitement circonscrites et prévisibles, dans lesquels les méthodes de génie logiciel sont applicables [12, 15]. Pire encore, certains cas d'interactions peuvent être jugés acceptables par divers groupes d'utilisateurs et néfastes par d'autres. L'expérience montre que la notion même d'interaction néfaste est presque subjective et en tout état de cause, impossible à caractériser rigoureusement.

Si à cela on ajoute que les spécifications des services sont souvent incomplètes, on comprend tout l'intérêt d'un outil d'aide à l'intégration. Ce dernier permettrait au concepteur de maîtriser la complexité induite par les différents choix d'intégration et d'estimer leurs impacts sur le fonctionnement global du système. Bien entendu, l'automatisation totale de ce processus n'est pas un objectif raisonnable. C'est pourquoi, nous prenons le parti de traiter le problème difficile de la résolution des interactions néfastes de manière seulement *assistée* et non pas totalement automatisée.

Dans ces conditions, le but est de limiter autant que possible le travail du concepteur. Pour cela nous proposons de construire des approximations successives de la caractérisation des interactions néfastes, en utilisant la génération automatique de tests pour guider leur analyse. Chaque approximation décrit les comportements litigieux pour l'ensemble de services à intégrer et les choix d'intégration potentiels.

Dans la section 2 nous donnons un exemple concret d'intégration de services afin de mettre en évidence la difficulté et la subjectivité du problème. Dans la section 3 nous exposons les principes génériques de la méthode proposée car cette dernière est indépendante des techniques de génération de tests et d'approximation utilisées. Finalement dans la section 4 nous montrons quelques pas de la méthode pour l'exemple introduit dans la section 2.

2 Exemple d'intégration de services

Nous allons illustrer les notions d'intégration de services et d'interaction néfaste sur un exemple d'école mettant en jeu les deux services connus respectivement sous les noms de *liste noire* et *transfert inconditionnel d'appel*. Nous les désignerons par la suite par les sigles [1] TCS et CFU. Ces deux services sont décrits ci-dessous de façon informelle.

- Le service TCS de liste noire a pour fonction d'interdire les appels en provenance d'une liste de postes, cette liste étant précisée par l'abonné. De plus, un utilisateur dont l'appel est en principe rejeté par cette liste noire recevra un message du choix de l'abonné à la place des signaux de la tonalité classique (ligne occupée ou tonalité de sonnerie). Pour fixer les notations, `X : TCS([X1,...,Xn],m)` indiquera que le téléphone `X` est abonné au service TCS de façon à refuser les appels en provenance des n téléphones `X1, ..., Xn` en leur substituant le message `m` en lieu et place d'une communication standard.
- Le service CFU permet de rediriger systématiquement les appels entrants vers un autre poste prédéfini par l'abonné. Toujours pour fixer les notations, `X : CFU(Y)` signifiera que les appels vers le téléphone `X` sont redirigés vers le téléphone `Y`.

Ces deux services interagissent car une ou plusieurs redirections d'appels (appels de CFU) posent la question de l'origine de l'appel : faut-il considérer que l'origine de l'appel est le premier appelant ou bien l'un quelconque, ou tous ou encore le dernier seulement parmi les postes intermédiaires lors des redirections ? Et évidemment, la réponse à cette question a une forte incidence dès lors que le service TCS est invoqué. Or, la description de CFU ne mentionne pas l'évolution éventuelle de la notion d'origine d'appel lors des redirections. La raison en est que la redirection d'appel est systématique et définie indépendamment de la notion de l'origine d'appel.

De façon générale, les services sont décrits d'une part à partir des concepts associés au service téléphonique de base et d'autre part des nouveaux concepts propres [2] à ce service (comme la notion d'origine d'appel pour le service TCS). En effet, il est communément admis que les services

[1] Ces sigles sont issus comme il se doit de la terminologie anglaise "Terminating Call Screening" et "Call Forward Unconditionnal".

[2] Un concept introduit lors de la conception d'un service peut être partagé avec un autre service. Ainsi, le service CND qui affiche le numéro de l'appelant partage naturellement avec le service TCS le concept d'origine d'appel.

sont vus comme de simples options ajoutées et retirées à la demande à partir d'un système de base. Comme tels, ils sont conçus et décrits en des termes aussi minimalistes que possible :

- Ils sont conçus, *a priori* et le plus souvent, les uns indépendamment des autres. Ceci explique que lors de la conception d'un service, on ne tient compte que des concepts propres à ce dernier. *A fortiori*, on ne peut anticiper les besoins des services non encore existants et donc, mentionner à l'avance des concepts introduits postérieurement à la définition du service en cours de conception.
- Les concepts étrangers au service (e.g. l'origine de l'appel pour CFU) sont évacués de la description : l'idée sous-jacente est qu'en dehors de la modification apportée par le service sur le système, *"tout se passe comme dans le schéma d'appel de base"*, en particulier, pour ce qui concerne les concepts étrangers au service.

Plus concrètement, imaginons maintenant un scénario mettant en jeu les services TCS et CFU et faisant intervenir trois téléphones : `A`, `B` et `C` avec les souscriptions de services suivantes :

- `B : CFU(C)` et
- `C : TCS([A],m)`.

Nous envisageons un appel de `A` vers `B`. Naturellement, de par les souscriptions mentionnées, une redirection vers `C` est envisageable, ainsi qu'un refus de connexion par `C` pour `A`. De fait, selon la manière dont sont combinés les deux services, c'est-à dire, selon la notion d'origine d'un appel considérée, l'appel de `A` vers `B` donnera lieu à des observations différentes.

L'origine de l'appel est `A` On aboutit à une configuration dans laquelle une fois que CFU a été invoqué, est transmise au service TCS l'information d'un appel de `A` vers `C`, auquel cas, `A` se voit refuser la communication et envoyer le message `m`.

L'origine de l'appel est `B` On peut observer que `A` et `C` sont en communication. Ceci est plausible sous l'hypothèse qu'il est considéré que l'origine de l'appel est `B`, poste de la redirection. Sous cet angle, le service TCS n'interdit pas la communication.

L'expert, à la vue de ces deux simulations d'appels, peut conclure au bien fondé de la première description et au rejet de la seconde, à savoir que dans le premier cas la spécification des deux services est respectée, alors que dans le second, celle de TCS est clairement violée.

Les différences d'observation à l'issue de la simulation proviennent de deux différents modes d'intégration des services. Intuitivement, dans le premier cas, CFU est transparent face à TCS, et dans le second, CFU masque à TCS l'origine réelle de l'appel.

Cela met en lumière une difficulté supplémentaire de l'intégration de services : le mode de combinaison des services. Une combinaison de services que l'on pourrait juger satisfaisante au vu d'une simulation peut s'avérer problématique, lorsque la simulation se complexifie.

Ainsi, reprenant le premier cas (hypothèse de transparence de CFU face à TCS), considérons de plus que le téléphone C souscrit en plus de l'abonnement à TCS, un abonnement à CFU, avec redirection vers un autre téléphone `D` :

- `C : CFU(D)`.

Même en gardant l'hypothèse de transparence de CFU, il reste à définir la collaboration de TCS et de CFU souscrits par un même abonné (ici `C`). Selon que CFU est prioritaire ou non sur TCS, les deux situations suivantes émergeront :

CFU est prioritaire sur TCS Une communication est établie entre `A` et `D`

TCS est prioritaire sur CFU `A` reçoit le message `m` de refus d'appel de sa part en provenance de `C`.

La pertinence de ces deux scénarios est laissée à l'appréciation de l'expert. En effet, de prime abord, aucun critère mécanique ne permet de privilégier l'un par rapport à l'autre. Dans le premier cas, si l'on imagine que l'utilisateur habituel du téléphone `C` se trouve à cet instant à proximité du téléphone `D` (ce qui est justement l'un des rôles du service CFU), il se retrouvera en communication avec l'utilisateur du téléphone `A` (ce qu'il ne souhaitait en aucun cas, car

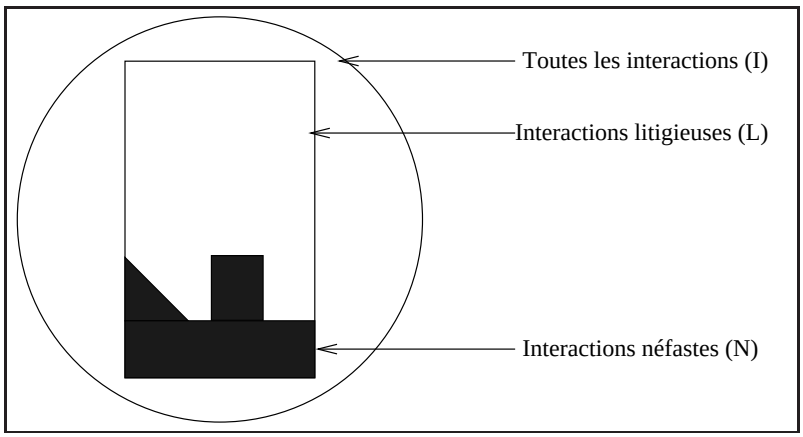

FIG. 1 – Carte des interactions

il avait souscrit `TCS([A],m)`...). Dans le second cas, l'utilisateur du téléphone `A` entendra un message de refus d'appel dont il ne comprendra pas la raison (provenance inconnue : `C` ou refus de communication de la part de `B`).

Ce petit exemple illustre le fait qu'un même type de combinaison de services amène à observer, selon les simulations, des comportements très différents que seul un expert peut classifier respectivement comme néfastes ou souhaitables, sur des critères *de facto* assez subjectifs. Cela montre donc qu'une définition mathématique de la notion d'interaction néfaste serait peu crédible, et justifie une intervention humaine dans le processus d'intégration des services.

3 Une méthode incrémentale d'aide à l'intégration

L'expert est appelé à émettre des verdicts sur des scénarios qui lui sont soumis par l'environnement de création de services. Ces scénarios sont caractéristiques d'interactions litigieuses (i.e. susceptibles d'être néfastes). Le verdict émis par l'expert consiste donc avant tout à décider si l'interaction mise au jour est néfaste ou non.

Le calcul des scénarios présentés à l'utilisateur est fondé sur des techniques de test fonctionnel (à partir de spécifications) adaptées à la caractérisation des interactions litigieuses. On part d'un ensemble L *a priori* infini contenant des interactions considérées comme litigieuses, puis, procédant par approximations successives, de manière incrémentale, on détermine l'ensemble des interactions néfastes N. La démarche est dynamique car le but de test (c'est-à-dire l'ensemble courant des interaction litigieuses restant à explorer) est affiné en permanence. Chaque fois que le but de test est modifié, la génération des scénarios de test continue à partir du point courant.

La figure 1 représente en quelque sorte l'état initial des connaissances de l'expert au démarrage du processus. Les spécifications préliminaires (phases 1 et 2 de la figure 2) comprennent en particulier pour chaque service les propriétés que son comportement doit observer. Ces spécifications préliminaires permettent de déterminer l'ensemble I des scénarios mettant en jeu une interaction. Selon la logique utilisée dans les spécifications, I est caractérisé par un prédicat et/ou par un schéma explicite de scénario. Ceux-ci traduisent la mise en jeu d'au moins deux services ou instances d'un même service. Ils peuvent porter sur une situation à repérer : par exemple, deux services accèdent à une même variable, ou réagissent à une même entrée. Un prédicat peut aussi correspondre à des conditions que doit satisfaire l'exécution conjointe des services : par exemple, la conjonction des propriétés de comportement de ces services (le fait qu'un des comportements est perturbé révèle une interaction).

Démarrer le processus à partir de cet ensemble I serait par trop inefficace. En pratique,

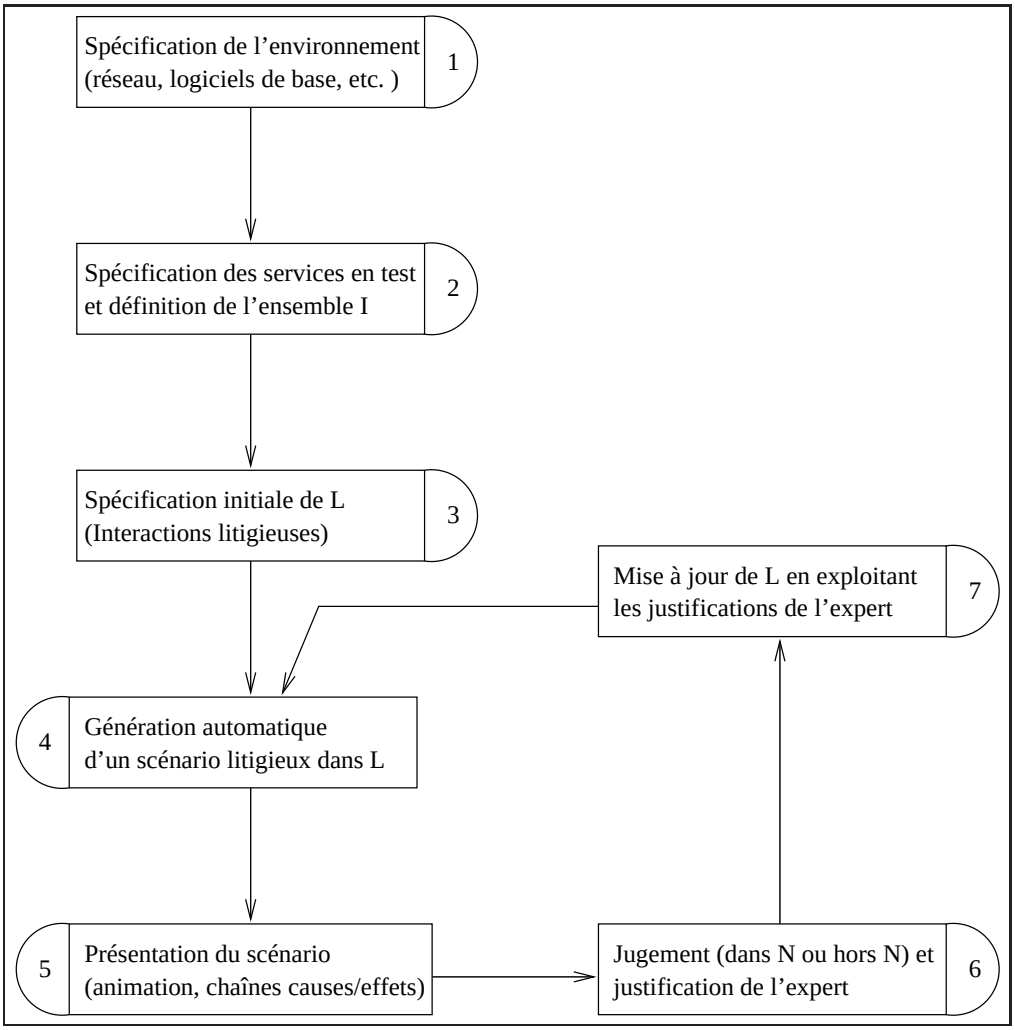

FIG. 2 – Boucle d'analyse

les experts en télécommunications possèdent des connaissances qui permettent de déterminer *a priori* un sous-ensemble de I contenant les interactions potentiellement néfastes que l'on appellera litigieuses. Souvent, cet ensemble L permet de réduire la taille des scénarios considérés, de mieux cibler les enchaînements d'actions des utilisateurs, de réduire le nombre de services mis en jeu dans un scénario, le nombre de téléphones, etc. Un important travail initial sera de récolter certains schémas de modes d'intégration qui sont standardisés dans certains domaines applicatifs. L'ensemble L duquel on part est ainsi considérablement plus petit que l'ensemble I [3]. Les ensembles I et L peuvent être décrits dans divers langages de spécification. Nous n'en privilégions aucun par la suite. Notre objectif, dans cet article, est de définir une *méthode* d'analyse qui peut être appliquée à plusieurs formalismes de spécification.

La méthode que nous proposons se fonde sur les faits exposés ci-dessus et propose un cycle d'analyse assistée des interactions de services schématisé sur la figure 2. Le générateur de tests qui intervient en phase 4 pour produire un scénario se fonde sur les spécifications produites dans les trois premières phases. La spécification de l'espace L des scénarios litigieux (phase 3) donne les buts de test assignés au générateur. Les spécifications de l'environnement (phase 1) et des services

[3]Notez que l'on fait confiance à l'expert pour ce qui concerne la détermination de L et qu'aucun système de preuve ne peut assurer que L contient N tout entier. En fait, l'expert considère même que certaines interactions néfastes peuvent être laissées hors de L car elle seront révélées par d'autres interactions plus simples.

(phase 2) sont utilisées par le générateur de test pour décomposer son but de test (le domaine L) en sous-domaines ayant potentiellement des comportements similaires. Intuitivement, les phases 1 et 2 décrivent des services indépendants, et permettent de cerner les situations d'interactions. À partir des spécifications de I (produites en fin de phase 2), la phase 3 donne des schémas d'interaction à simuler. L'intervention de l'expert débute à la phase 3.

La présentation du scénario engendré par la phase 4 doit être fournie à l'expert sous une forme exploitable pour produire un verdict. Cette phase 5 est assez délicate car elle doit fournir tous les éléments pertinents sans pour autant noyer l'utilisateur sous les informations. Une animation du schéma est souhaitable mais des diagrammes décrivant les enchaînements action-réaction seront sans doute également nécessaires pour asseoir la compréhension du scénario.

Le jugement produit par l'expert en phase 6 , seule intervention humaine dans le cycle, n'est pas réduit à une acceptation ou un refus. L'expert indique également une généralisation du scénario présenté qui couvre un ensemble de scénarios pour lesquels son verdict est valide. Comme on le voit sur la figure, on se propose d'assister le travail de l'expert en lui suggérant une généralisation par défaut : celle utilisée par le générateur de test pour décrire le domaine auquel appartient le test soumis. Ainsi, l'expert pourra se contenter d'élargir ou restreindre la description par défaut, sans devoir produire *ex nihilo* une généralisation en partant uniquement du scénario présenté.

La généralisation fournie par l'expert en phase 6 est alors immédiatement réinjectée via la phase 7 pour réduire l'espace de recherche du générateur de test de manière significative. Intuitivement, cet aspect évite la présentation à l'expert de scénarios sémantiquement comparables. Notons que cette restriction a toujours lieu, que le verdict soit positif ou négatif. Ceci a pour effet de réduire l'espace de recherche du générateur de test et de converger vers une intégration acceptable des services considérés. Il n'en reste pas moins que la phase 7 est importante car c'est elle qui exploite de manière incrémentale les connaissances de l'expert. Ces connaissances sont indispensables, car, comme on l'a vu précédemment, on ne connait pas *a priori* tous les modes d'intégration de service (on doit donc gérer, à tout moment, des spécifications incomplètes).

Naturellement, comme pour tout processus réentrant, on doit évaluer l'état d'avancement à chaque cycle et assurer la terminaison en un temps raisonnable. Comme on l'a vu, le nombre de cycles est égal au nombre de sous-domaines finalement dégagés au travers de chaque passage en phase 4. Cependant l'évaluation de l'état d'avancement ne peut être qu'approximative à chaque cycle car l'utilisateur expert pourra ultérieurement réduire ou agrandir chaque domaine via la phase 7. Pour assurer la terminaison du processus sans perdre une complétude de couverture des scénarios possibles, il suffira à l'utilisateur de ne plus partitioner les domaines, qui sont par défaut proposés en nombre raisonnable par le générateur de tests.

4 Une instance de la méthode proposée

Nous allons montrer par l'exemple comment l'ensemble des interactions néfastes peut être approximé. Reprenant les scénarios et services précédemment utilisés, nous classifions les scénarios initiaux d'appels de `B` par `A` comme suit :

- `Sc1`, la classe de scénarios dans laquelle on a
  - au moins les souscriptions de services suivantes
    - `B : CFU(C)`
    - `C : TCS([A],m)`
  - au moins le mode d'intégration de services suivant :
    - CFU masquant l'origine réelle de l'appel (c'est-à-dire que, pour ces scénarios, l'origine de l'appel redirigé par l'abonnement de `B` sera `B`)
- `Sc2`, la classe de scénarios dans laquelle on a
  - au moins les souscriptions de services suivantes

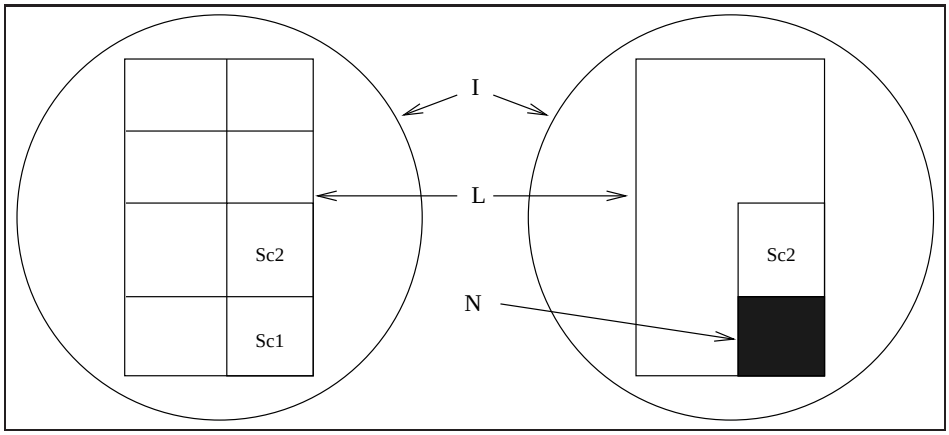

FIG. 3 – Premier cycle

- `B : CFU(C)`
- `C : TCS([A],m)`

– au moins le mode d'intégration de services suivant :
  – transparence de CFU pour ce qui concerne l'origine de l'appel (c'est-à-dire que, pour ces scénarios, l'origine de l'appel, même redirigé par `B`, sera `A`)

Remarquons que les deux classes de scénarios `Sc1` et `Sc2` ne diffèrent que par les modes d'intégration du service CFU. Ces deux classes de services, avant simulation sont considérées comme litigieuses (partie gauche de la figure). Après simulation, `Sc1` a clairement révélé une interaction néfaste, mais le verdict n'est pas aussi évident pour `Sc2`. En effet, hormis le fait que l'utilisateur du téléphone `A` reçoit le message de refus `m`, on peut considérer que, selon la teneur de ce dernier, l'utilisateur pourra ne pas en comprendre la raison. C'est un exemple d'interaction litigieuse. Cependant son caractère néfaste peut, *a priori*, ne pas être considéré comme "grave" : c'est le point de vue que nous adopterons par la suite. L'expert décide donc, au vu des résultats :

– de classer tous les scénarios `Sc1` parmi les scénarios néfastes
– d'affiner le jugement concernant `Sc2`.

La classe de scénarios `Sc2` permet de mettre en évidence les deux sous-classes suivantes :

– `Sc'2`, la classe de scénarios dans laquelle on a
  – au moins les souscriptions de services suivantes :
    - `B : CFU(C)`
    - `C : TCS([A],m)`
    - `C : CFU(D)`
  – au moins les modes d'intégration de services suivant :
    – transparence de CFU pour ce qui concerne l'origine de l'appel
    – TCS prioritaire par rapport à CFU (pour un même abonné)
– `Sc''2`, la classe de scénarios dans laquelle on a
  – au moins les souscriptions de services suivantes :
    - `B : CFU(C)`
    - `C : TCS([A],m)`
    - `C : CFU(D)`
  – au moins les modes d'intégration de services suivant :
    – transparence de CFU pour ce qui concerne l'origine de l'appel
    – CFU prioritaire par rapport à TCS (pour un même abonné)

Après simulation, un scénario de `Sc''2` est reconnu par l'expert en tant que scénario révélateur d'une interaction néfaste. Tout scénario de `Sc''2` serait reconnu pour les mêmes raisons ; l'expert

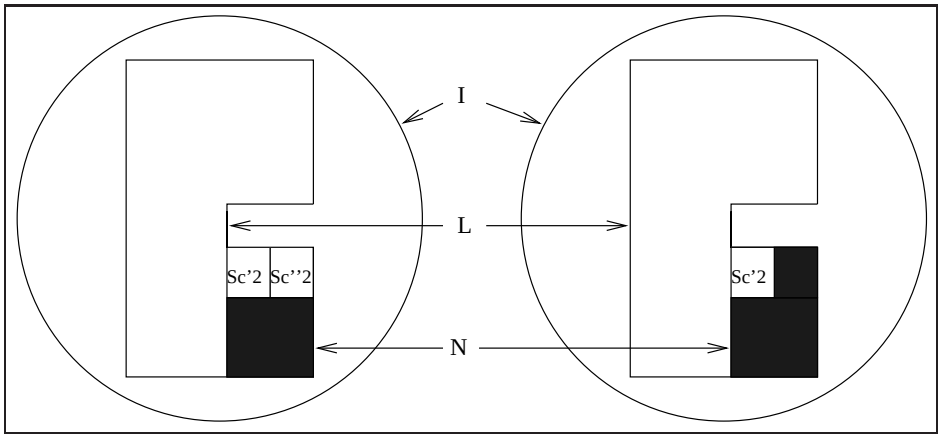

FIG. 4 – Résultats des second et troisième cycles

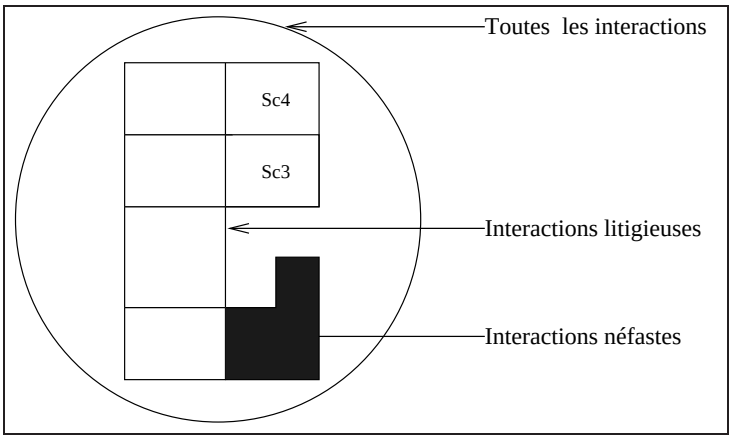

FIG. 5 – Résultat du quatrième cycle

classe donc `Sc''2` en totalité dans l'ensemble des interactions néfastes.

On admettra pour simplifier que `Sc'2` ne contient que des scénarios présentant un comportement similaire au premier scénario sélectionné dans `Sc2` (début de cette section). Ils ne seront donc pas considérés comme néfastes (rappelons que l'utilisateur du téléphone `A` recevra le message `m`, sans peut-être en comprendre la provenance), en conséquence modulo une adaptation du message cette branche pourrait aboutir à une intégration acceptable. On obtient ainsi la situation suivante avant d'aborder l'examen d'autres classes : `Sc3`, `Sc4`, etc.

Évidemment, tant que tout l'espace des scénarios litigieux n'a pas été exploré, rien ne permet d'affirmer définitivement que `Sc'2` conduit à la meilleure solution d'intégration. Cependant, comme dans notre exemple le générateur de tests a proposé par défaut huit classes de scénarios et comme deux d'entre elles (`Sc2` et de `Sc1`) ont été traitées, l'estimation grossière de la proportion de scénarios restant à examiner pourrait être 75%. Cette estimation n'est qu'indicative, car pour certaines classes, l'expert peut effectuer plus de simulations que pour d'autres (c'est ce qui a été fait pour `Sc2` par rapport à `Sc1` et ce qui aurait pu être fait pour `Sc'2` lors d'une analyse plus fine).

5 Conclusion

Cet article porte sur une méthode de détection d'interactions au niveau des spécifications des services. Il s'agit d'un problème largement étudié, dont la plupart des solutions s'appuie sur des modèles et des techniques outillées de vérification-validation. Pour entièrement automatiser la détection, de rares travaux ont consisté soit à créer des langages dédiés pour la création et la validation de services [11, 14], soit à produire une définition formelle de l'interaction [3, 9]. Ces travaux ont abouti à un cadre théorique spécifique dans lequel la notion d'interaction est simplifiée ; en conséquence de nombreuses interactions ne peuvent être découvertes. L'approche la plus répandue revient à appliquer des méthodes formelles associées à diverses logiques temporelles [3, 5, 9], SDL [1] ou Lotos [8].

La méthode que nous avons décrite est une méthode pragmatique motivée par l'expérience. Nous avons en effet constaté que,

- d'une part, les approches entièrement formelles, comme celles décrites ci-dessus, reposant sur des techniques de preuve ou de test, requièrent en réalité une grande expérience des spécifieurs qui adaptent les spécifications au fur et à mesure que la validation avance afin de converger vers une intégration correcte ;
- d'autre part, les approches entièrement fondées sur l'expérience d'un expert deviennent rapidement impraticables lorsque le nombre de services mis en jeu augmente et que l'aide de spécifications formalisées apparaît incontournable.

L'originalité essentielle de notre méthode est de mettre en œuvre à la fois des techniques de test fonctionnel automatisées, maintenant relativement bien connues, et des interventions humaines régulières là où les décisions de succès ou d'échec des tests ne sont pas automatisables.

Par rapport à des techniques classiques de test fonctionnel, on pourrait dire en première approximation que l'expert joue simplement un rôle d"“oracle”. En fait, l'intervention de l'expert dans la méthode que nous proposons est également imbriquée avec la génération de tests puisque les diagnostics fournis ont une rétro-action sur le domaine géré par le générateur de tests. C'est cette dynamicité de la méthode qui devrait la rendre performante.

Plusieurs instances de cette méthode sont en cours de développement dans nos laboratoires. L'une est fondée sur des spécifications en logique du premier ordre et utilisera une extension d'un générateur de tests [2] écrite en Prolog. Une autre repose sur des spécifications exprimées en Lustre (vu comme une logique temporelle) et utilisera l'outil de test Lutess [7] déjà appliqué à la recherche d'interactions [6]. En l'état actuel, cet article doit être considéré comme purement prospectif et nous n'avons encore aucun résultat concret à rapporter ici. Toutefois, en se fondant sur l'expérience de nos trois équipes (CNET-Lannion, LaMI, LSR-IMAG), nous abordons cette recherche avec une certaine confiance en l'avenir. Notre méthode fait l'objet du projet ValiServ qui débute cette année au sein du Réseau National de Recherche en Télécommunications.

Enfin il faut noter que l'approche proposée se plaçant résolument au niveau de la logique des services, est indépendante de l'architecture cible. En conséquence, cette méthode ne se limite pas aux services du réseau intelligent, et devrait être exploitable sans modification pour d'autres architectures comme TINA-CORBA ou Internet. Plus généralement nous pensons qu'une telle approche est applicable au problème de l'ajout de nouvelles fonctions à un système quelconque.

Références

[1] Aggoun (I.) et Combes (P.). – Observers in the SCE and the SEE to detect and resolve services interactions. *In : Feature Interactions in Telecommunications Systems IV*. pp. 198–212. – IOS Press.

[2] Bernot (G.), Gaudel (M-C.) et Marre (B.). – Software testing based on formal specifications : a theory and a tool. *Software Engineering Journal*, vol. 6, 1991, pp. 387–405.

[3] Blom (J.), Jonsson (B.) et Kempe (L.). – Using temporal logic for modular specification of telephone services. *In : Feature Interactions in Telecommunications Systems*, éd. par Bouma (L.G.) et Velthuijsen (H.). pp. 197–216. – IOS Press.

[4] Bowen (T.F.), Dworak (F.S.), Chow (C.-H.), Griffeth (N.D.), G.E. (Herman) et Lin (Y.-L.). – The feature interaction problem in telecommunication systems. *In : Proceedings of the seventh International Conference on Softwre Engineering for Telecommunication Switching Systems*. – Bournemouth, United Kingdom, 1989.

[5] Combes (P.) et Pickin (S.). – Formalization of a user view of network and services for feature interaction detection. *In : Feature Interactions in Telecommunications Systems*. pp. 120–135. – IOS Press.

[6] du Bousquet (L.), Ouabdesselam (F.), Richier (J.-L.) et Zuanon (N.). – Feature interaction detection using synchronous approach and testing. *Computer Networks and ISDN Systems*. – à paraître.

[7] du Bousquet (L.), Ouabdesselam (F.), Richier (J.-L.) et Zuanon (N.). – Lutess : a specification-driven testing environment for synchronous software. *In : 21st International Conference on Software Engineering*. pp. 267–276. – ACM Press.

[8] Faci (M.) et Logrippo (L.). – Specifying features and analyzing their interactions in a lotos environment. *In : Feature Interactions in Telecommunications Systems*, éd. par Bouma (L.G.) et Velthuijsen (H.). pp. 136–151. – IOS Press.

[9] Gammelgaard (A.) et Kristensen (J. E.). – Interaction detection, a logical approach. *In : Feature Interactions in Telecommunications Systems*, éd. par Bouma (L.G.) et Velthuijsen (H.). pp. 178–196. – IOS Press.

[10] Griffeth (N. D.) et Lin (Y.-J.). – Extending telecommunication systems : The feature-interaction problem. *In : IEEE Computer*, pp. 14–18.

[11] Hall (R.J.). – Feature combination and interaction detection via foreground/background models. *In : Feature Interactions in Telecommunications Systems V*, éd. par Kimbler (K.) et Bouma (L.G.). pp. 232–246. – IOS Press.

[12] Kimbler (K.) et Søbirk (D.). – Use case driven analysis of feature interactions. *In : Feature Interactions in Telecommunications Systems*, éd. par Bouma (L.G.) et Velthuijsen (H.). pp. 166–177. – IOS Press.

[13] Kung (R.) et Maitre (X.). – L'architecture du réseau intelligent. *L'écho des recherches, CNET- France Télécom*, no157, 1994.

[14] Turner (K.J.). – An architectural description of intelligent network features and their interactions. *Computer Networks and ISDN Systems*, vol. 30, n15, 1998, pp. 1389–1420.

[15] Utas (G.). – A pattern language of feature interaction. *In : Feature Interactions in Telecommunications Systems V*, éd. par Kimbler (K.) et Bouma (L.G.). pp. 98–114. – IOS Press.