

Détection d'interactions entre services modélisés par des HMSCs

Loïc Hérouët[♦], Ferhat Khendek[■]

[♦] *France Télécom, 2 av Pierre Marzin, 22307 Lannion Cedex, France*
loic.helouet@francetelecom.com

[■] *Electrical and Computer Engineering, Concordia University,
1455 de Maisonneuve Bd, Montréal, Québec, Canada H3G 1M8*
khendek@ece.concordia.ca
<http://www.ece.concordia.ca/~khendek>

RÉSUMÉ. Cet article présente un environnement formel pour la description et la simulation de services téléphoniques au moyen de High-level Message Sequence Charts (HMSCs). Dans ce but, les HMSCs sont étendus avec des variables permettant de décrire les gardes pour le déclenchement et les exigences liées à chaque service. Cet environnement permet de définir formellement deux types d'interactions : les conflits de déclenchement, et la violation des exigences, qui sont détectés grâce à une technique de simulation de HMSCs.

ABSTRACT. This paper introduces a formal framework for the description and simulation of telephone features using High-level Message Sequence Charts (HMSCs). For that purpose HMSCs are extended with variables to describe guards for the triggering conditions and feature requirements. This formal framework takes into consideration and defines formally two types of interactions: triggering conflicts and requirement violations. These types of interactions are detected using a HMSC simulation technique.

MOTS-CLÉS : Interaction de services, MSCs, scénarios, simulation.

KEYWORDS: Feature interaction, MSCs, scenarios, simulation.

1. Ce travail a bénéficié du soutien du Conseil de Recherches en Sciences Naturelles et en Génie du Canada (CRSNG).

1. Introduction

Un *service* téléphonique est défini comme une fonctionnalité supplémentaire ajoutée au comportement initial d'un réseau, fournie à un abonné ou servant à l'administration du réseau. Malheureusement, un service fonctionnant parfaitement en isolation peut se comporter de manière inattendue en présence d'autres services, produisant ce que l'on appelle des *interactions de services*. La détection des interactions de services dans les réseaux téléphoniques est un problème bien étudié [CAM 94, KIM 95, FIW 92, FIW 94, FIW 95, FIW 97, FIW 98, SEF 99]. Un récapitulatif et une classification des approches proposées peut être trouvé dans [KEC 98]. Les techniques de détection off-line sont les plus fréquentes. Elles s'appuient sur une modélisation d'un système dans un langage formel, puis explorent l'espace d'états ainsi défini à la recherche de propriétés indésirables. Lorsque l'on modélise le comportement d'un réseau, la représentation d'une plate-forme peut être considérée comme important, et défini très précisément dans la spécification (on parle alors de vision orientée réseau). Au contraire, le rôle du réseau peut être minimisé, et la plate-forme considérée comme une entité centrale qui fournit des réponses à des requêtes d'utilisateurs (on parle alors de vision orientée utilisateur). La taille du système exploré peut souvent être réduite sous certaines hypothèses, comme celle de communications synchrones. Cette supposition limite la concurrence entre les entités du système, et empêche l'accumulation de messages dans une file, réduisant ainsi la taille des systèmes de transitions à explorer. Malheureusement, l'arrivée de nouvelles technologies telles que la téléphonie par internet nécessite de modéliser explicitement des communications asynchrones, amenant ainsi des spécifications à espace d'états infinis, qui rendent inutilisables la plupart des techniques de recherche exhaustive. Cependant, des techniques de simulation peuvent toujours être utilisées pour détecter des comportements indésirables.

Une approche basée sur les interworkings, des scénarios à communications synchrones, a été définie dans [BER 97]. Nous proposons un environnement asynchrone pour la simulation de réseaux téléphoniques, basé sur un langage de scénarios, les Message Sequence Charts (MSCs). Les MSCs permettent de définir graphiquement et intuitivement le comportement d'un système distribué, tout en autorisant des manipulations formelles telles que la simulation. Le réseau téléphonique est décrit suivant une vision utilisateur, et les spécifications sont données en terme de communications asynchrones entre des composants du système. Chaque vue utilisateur consiste en un comportement normal attendu du réseau, et un ensemble de services. Un service définit un comportement qui doit se substituer au comportement normal et un but (également appelé exigence [KIM 95]), exprimé au moyen de prédicats sur un ensemble de variables. Afin d'exprimer ces buts, la notation standard des MSCs est étendue avec des opérations d'affectation de variables. L'environnement ainsi défini permet de mettre en évidence deux types d'interactions de services.

Cet article est organisé comme suit : le chapitre 2 introduit la notation standard pour les MSCs, et l'étend par des manipulations de variables. Le chapitre 3 décrit le modèle utilisé pour la définition de services, et lui donne une sémantique opérationnelle. Une approche pour la détection des interactions de service est ensuite donnée au chapitre 4, et illustrée par des exemples. Enfin, le chapitre 5 conclut ce travail.

2. Message Sequence Charts

Nous souhaitons modéliser le comportement de réseaux téléphoniques par des scénarios. Un des avantages d'une approche par scénarios est de définir graphiquement et intuitivement un système. Contrairement aux sémantiques d'entrelacement, les scénarios donnent une représentation visuelle de la concurrence, facilitant ainsi la compréhension d'une spécification. Pour chaque utilisateur, nous allons définir par des scénarios une vue du système, c'est à dire un ensemble de comportements possibles. Ce chapitre présente les Message Sequence Charts (ou MSCs), un formalisme graphique standardisé et maintenu par l'ITU (International Telecommunication Union), et une extension de ce formalisme par des affectations de variables.

2.1. bMSCs étendus

Un basic MSC (bMSC) décrit le comportement d'entités communicantes appelées *instances*. Ces instances sont représentées par des axes verticaux, et les échanges de messages par des flèches de l'instance émettrice vers l'instance réceptrice. Un *événement* d'un bMSC peut être une émission de message, une réception, une opération sur une horloge (armement, désarmement, expiration), ou une action interne. L'émission d'un message précède sa réception, et les événements sont ordonnés de haut en bas le long de l'axe d'une instance. On peut donc dire qu'un bMSC définit un *ordre causal* entre des événements. Jusqu'à très récemment, les MSCs ne permettaient pas de manipuler de variables ni de données. Une nouvelle version de la norme appelée MSC'2000 introduit cette possibilité. Dans cet article, nous définissons une approche légèrement différente des MSC'2000 pour la gestion des variables.

Dans notre modèle des réseaux téléphoniques, un service est gardé par un prédicat sur des *variables*. Ces variables contiennent des informations sur l'état du système, et sont mises à jour lors de l'exécution des événements. Un événement peut être paramétré (c'est à dire dépendre de la valeur d'une variable). Enfin, il est souhaitable de pouvoir décrire des comportements génériques, c'est à dire des réponses systématique du réseau ne dépendant pas de l'origine d'un signal (par exemple, renvoyer le signal *Busy* à tout abonné *x* essayant de joindre l'abonné *A* lorsque la ligne de *A* est occupée). Dans ce but, des *variables d'instance* sont également introduites. Elles permettront de définir des événements dont la localité ne sera connue qu'à l'exécution. Pour illustrer nos propos, considérons le MSC de la Figure 1. Les événements e_1 et e_6 dans cette spécification modifient la valeur d'une variable d'état T . L'événement e_4 est exécuté par une instance spécifiée par la variable u . u dépendra de la valeur choisie par l'abonné A lorsque le numéro d'un correspondant sera composé, et sera instanciée au moment de la simulation de la spécification.

L'approche décrite dans cet article est orientée utilisateur, les détails internes de l'architecture du réseau seront donc cachés. Le réseau est modélisé par ses interaction (requêtes / réponses) avec les utilisateurs, qui ne peuvent communiquer que par son intermédiaire. Par conséquent, les spécifications seront construites au moyen de bMSCs contenant des échanges de messages entre un utilisateur et le réseau, et des interactions du réseau avec le monde extérieur .

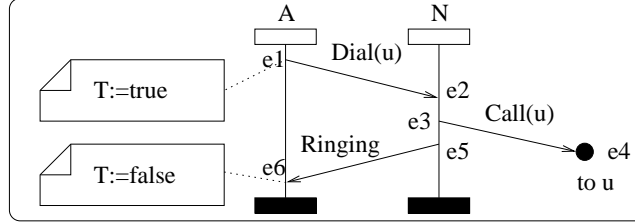


Figure 1 – bMSC comportant une variable d'état et une variable d'instance.

Definition 1: Un bMSC étendu est défini par un nuplet $M = (E, \leq, Var, VI, \alpha, v, A, I)$, où E est un ensemble d'événements, \leq est une relation d'ordre causal entre ces événements, Var est un ensemble de variables globales, et VI un ensemble de variables d'instances, α est une fonction qui associe une instance (localité) et un nom d'action à tout événement. $\alpha = E \rightarrow A \times I \cup VI$, v est une fonction qui associe un ensemble d'affectations de variables à chaque événement, A est un ensemble de noms d'actions I est l'ensemble des instances du bMSC (un utilisateur et le réseau). Soit e un événement. Nous noterons $\phi(e)$ la localité d'un événement, c'est à dire le nom de l'instance qui exécute e ($\phi(e) = i$ si $\alpha(e) = (a, i)$). Afin d'illustrer ces définitions, revenons à l'exemple de la Figure 1. Ce bMSC décrit une communication simple entre un utilisateur A et le réseau N . Notons que la variable T est positionnée à *vrai* par l'événement e_1 , et à *faux* par l'événement e_6 . L'événement e_4 est exécuté par une instance dont l'identité dépend des choix de l'instance A . Ce bMSC peut donc être formalisé comme suit : $E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$, $\leq = \{(e_1, e_2); (e_1, e_3); (e_1, e_4); (e_1, e_5); (e_1, e_6); (e_2, e_3); (e_2, e_4); (e_2, e_5); (e_2, e_6); (e_3, e_4); (e_5, e_6)\}$, $Var = \{T : Boolean\}$, $I = \{A, N\}$ et $VI = \{u\}$
 $\alpha = \{ (e_1, !dial(u), A); (e_2, ?dial(u), N); (e_3, !call(u), N); (e_4, ?call(u), u); (e_5, !ringing, N); (e_6, ?ringing, A) \}$
 $v = \{(e_1, T := true); (e_6, T := false)\}$

2.2. HMSCs

Un bMSC ne peut décrire que des scénarios très simples, une notation de plus haut niveau est donc nécessaire pour définir des comportements plus élaborés. Les HMSCs sont des graphes de haut niveau, qui permettent de composer les bMSCs au moyen d'opérateurs de composition parallèle, d'alternatives, de séquences, ou d'itérations.

Definition 2: Un HMSC peut être représenté par un graphe $H = (N, \rightarrow, M, l, End)$, où N est un ensemble de noeuds, \rightarrow est un ensemble d'arcs, M est un ensemble de bMSCs étendus, l est une fonction associant un nom de bMSC ou un type de noeud (connecteur, noeud de départ, noeud de fin) à tout noeud ($l : N \rightarrow M \cup \{end, start, connect\}$), End est une fonction qui associe une référence à chaque noeud de fin du HMSC. Cette fonction ne fait pas partie des définitions standard : elle servira à indiquer dans quel état le système doit se trouver une fois qu'un service a été effectué. Une description détaillée des HMSCs peut être trouvée dans [GRA 96], et une sémantique détaillée dans [MAUW 97].

2.3. HMSCs et grammaires de graphes

D'une manière intuitive, les grammaires de graphes peuvent être vues comme une manière commode de représenter des modèles infinis. Une sémantique des HMSCs basée sur les grammaires de graphes a été définie dans [HEL 98], et un environnement de simulation basé sur cette sémantique est décrit dans [HEL 99]. Ce chapitre fait une rapide introduction aux grammaires associées aux HMSCs, et présente les notations qui seront utilisées dans la suite de l'article.

Soit H un HMSC. La représentation sous forme de grammaire de graphes de H sera notée \mathcal{G}_H . Cette définition des HMSC permet de représenter un système de transitions infini, et fournit un environnement efficace pour la recherche de événements exécutables dans un état particulier du système. A partir d'une grammaire de graphes \mathcal{G}_H , un système de transitions infini peut être calculé. Chaque état est un graphe représentant la relation d'ordre dans le HMSC initial, et est construit par dépliage partiel de la grammaire et abstraction. Les états ainsi obtenus sont en fait des scénarios d'exécution possibles. L'état initial de ce système de transition sera noté $S_{\mathcal{G}_H}^0$. La représentation d'un état $S_{\mathcal{G}_H}$ sous forme de dépliages partiels d'une grammaire permet une recherche efficace des événements exécutables (que nous noterons $exec(S_{\mathcal{G}_H})$). Après l'exécution d'un événement $e \in exec(S_{\mathcal{G}_H})$, le système passe dans un autre état $S'_{\mathcal{G}_H}$, ce qui sera noté $S_{\mathcal{G}_H} \xrightarrow{e} S'_{\mathcal{G}_H}$. Pour tout HMSC $H = (N, \longrightarrow, M, l, End)$, nous noterons $\mathcal{G}_n(H)$ la grammaire de graphe calculée en prenant le noeud n comme noeud de départ du HMSC. La préservation de l'ordre causal du HMSC dans chaque état permet de définir un opérateur de *composition d'états*. Cette composition sera utilisée par la suite pour recoller un scénario à la suite d'un comportement, produisant ainsi un nouveau scénario d'exécution. La *concaténation locale* de deux états S et S' est un état $S \circ S'$, tel que :

$$exec(S \circ S') = exec(S) \cup \{e \in exec(S') \mid \forall S \xrightarrow{e_1} S_1 \cdots \xrightarrow{e_n} S_n, \forall i \in 1..n, \phi(e) \neq \phi(e_i)\}$$

Plus intuitivement, la concaténation locale empêche tout événement de S' de s'exécuter tant qu'un événement exécutable par la même instance peut être tiré à partir d'un état accessible depuis S . Cet opérateur sera utile lors de la définition d'une sémantique opérationnelle des réseaux de télécommunication.

3. Modélisation de services par des MSCs

Ce chapitre décrit une modélisation en MSCs du Plain Old Telephone System (POTS), quelques services, ainsi qu'une interprétation sémantique de ces modèles. Les MSCs ont déjà été utilisés comme formalisme d'entrée pour une approche axiomatique de la détection des interactions [BLO 97]. Dans cette approche, des bMSCs avec communication synchrones sont utilisés pour définir des systèmes de transitions décrivant des propriétés. L'approche présentée dans cet article utilise des MSCs à communications asynchrones, autorise l'utilisation d'itérations, et décrit les aspects comportementaux mais aussi axiomatiques des services.

3.1. Modélisation du POTS par des MSCs

Nous souhaitons modéliser les comportements d'un réseau téléphonique au moyen de vues utilisateur, qui représentent toutes les interactions possibles entre un utilisateur et le réseau, et les connexions au monde extérieur (voir Figure 2). Un utilisateur ne peut établir de communication avec un autre abonné que par l'intermédiaire du réseau, et suivant un protocole donné.

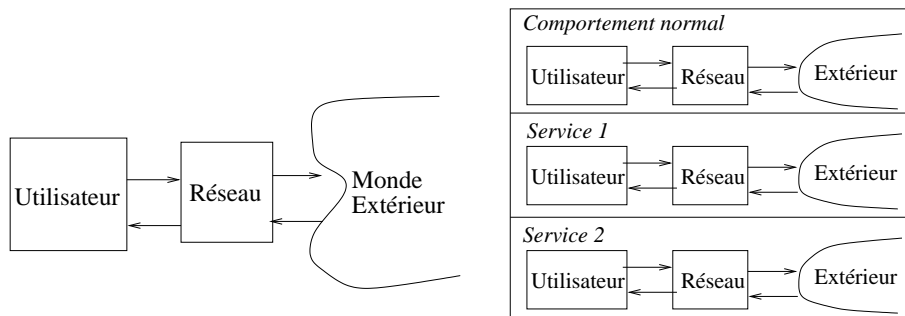


Figure 2 — a) Vue utilisateur du réseau normal et services

b) Détail d'une vue : comportement normal et services

Chaque vue utilisateur décrit les interactions entre utilisateur et réseau lorsqu'aucun service n'est déclenché (ce que l'on appellera par la suite comportement normal), mais aussi le comportement attendu lorsqu'un service est déclenché (voir Figure 2-b). Une vue utilisateur décrit un système de transitions. Les vues utilisateur définissent des comportements qui se superposent : un événement produit dans le monde extérieur d'un abonné *A* peut être un événement interne dans la vue d'un autre abonné *B*. Le comportement global d'une spécification composée de vues utilisateur peut donc être considéré comme un produit de ces vues.

Un réseau téléphonique peut effectuer en même temps plusieurs tâches pour un même utilisateur. Par exemple, lorsqu'un utilisateur *u* compose un numéro, le réseau doit traiter cet appel, mais également envoyer une réponse *Busy* à tout abonné tentant de joindre *u* au même moment. Si la première tâche peut être vue comme un échange normal d'une requête entre un utilisateur et le réseau, la deuxième s'apparente plus à une *réaction* du réseau à un stimulus extérieur (un message qui n'est pas envoyé par *u*), qui dépend de l'état dans lequel *u* se trouve. Notons que la représentation du réseau dans la vision utilisateur de *u* ne recevra de stimulus que s'il existe un envoi de message dans une autre vision utilisateur associée à un utilisateur *u'*. Le réseau complet est ainsi défini au moyen de scénarios dont les comportements se superposent.

Nous distinguons deux modes d'exécution : un *mode normal*, qui correspond à un appel classique dans la spécification du POTS lorsqu'aucun service n'est déclenché, et un *mode service*. Le comportement normal peut être interrompu par le déclenchement d'un service.

Definition 3: Un *Modèle* d'un réseau téléphonique est un ensemble $\mathcal{M} = \{B_u\}_{u \in U \text{ sers}}$, dans lequel chaque B_u décrit un ensemble d'échanges possibles entre un utilisateur u et le réseau téléphonique.

Definition 4: Un *comportement* est une paire $B_u = (N_u, \mathcal{F}_u)$ où N_u est l'ensemble des communications entre un utilisateur u et le réseau lorsqu'aucun service n'est déclenché. \mathcal{F}_u décrit un ensemble de *services*, c'est à dire un ensemble d'interactions qui peuvent se substituer au comportement normal.

Definition 5: Un *comportement normal* est une paire $N_u = (P_u, \mathcal{R}_u)$. P_u est un HMSC appelé *protocole normal*, qui définit les échanges de messages entre un utilisateur u et le réseau. $\mathcal{R}_u = \{(g_h, H_h)\}_{h \in 1..H}$ est un ensemble de MSCs gardés, appelés *réactions* du réseau. La Figure 3 montre un exemple de comportement normal d'un réseau téléphonique aussi appelé POTS (Plain Old Telephone System).

Definition 6: Une *réaction* (g_h, H_h) est la réponse du réseau à un stimulus extérieur en fonction de l'état d'un utilisateur u . g_h est une garde, c'est à dire un prédicat sur un ensemble de variables, qui peut être évaluée à *vrai* ou *faux*. Le HMSC H_h décrit de quelle manière le réseau doit répondre à un message extérieur lorsque la garde g_h est vraie. H_h ne doit pas contenir de message envoyé par u , et ne doit pas contenir d'itération (une réaction est un comportement fini).

Un exemple de réaction est la réponse du réseau lorsque l'abonné appelé est occupé. La réponse *Busy* doit être envoyée sans prévenir l'abonné actuellement en communication (ce comportement peut être modifié par certains services). Supposons qu'une variable A_Busy est positionnée à vrai lorsque l'utilisateur A est occupé. La réaction normale du réseau à un appel extérieur peut être modélisée par une paire $(g_1 = A_Busy, R_1)$, où R_1 est le HMSC de la Figure 4-a. Deux réactions peuvent s'exécuter au même moment (quand deux utilisateurs différents appellent un abonné occupé, le réseau peut leur adresser une réponse simultanément).

Dans notre approche, les services sont séparés en deux catégories distinctes : la première classe de services (appelée classe 1 par la suite) décrit un ensemble de nouveaux comportements qui n'impliquent pas d'exécution d'événements par l'utilisateur. Ce type de service sera modélisé par une paire $f = (\mathcal{R}_f, I_f)$, où : $\mathcal{R}_f = \{(g_h, P_h)\}$ est un ensemble de réactions gardées : chaque P_h est un HMSC sans itération, ni communication entre le réseau et l'utilisateur u . Le premier événement exécutable dans P_h sera appelé événement déclencheur. De plus, toutes les gardes doivent être exclusives. I_f est une propriété appelée exigence du service f . I_f est une propriété invariante qui doit être préservée par chaque exécution.

Le service Call Forward on Busy (abrégé en CFB) appartient à cette classe de services. Un abonné ayant souscrit au CFB peut spécifier un numéro vers lequel ses appels sont redirigés lorsque sa ligne est occupée. Ce service peut être décrit par le nuplet suivant, où $R2$ est le MSC de la Figure 4-b

$$cfb = (\emptyset, \{ (g_1 = whenA_Busy \wedge \neg C_Busy, R1), (g_1 = whenA_Busy \wedge C_Busy, R2) \})$$

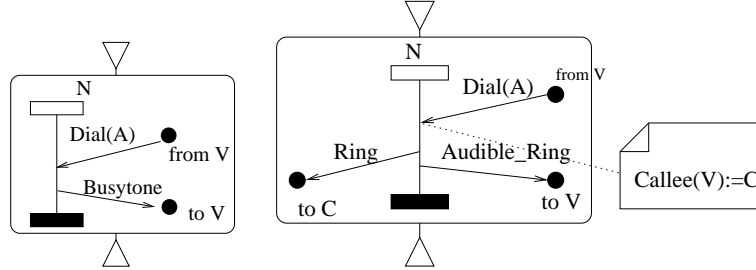


Figure 4 – a)HMSC R_1 b)HMSC R_2 : Service Call Forward on Busy

La deuxième classe de services (appelée classe 2 par la suite) décrit des communications entre le réseau et l'utilisateur. Ce type de service est modélisé par un nuplet $f = (g_f, P_f, \mathcal{R}_f, I_f)$ où :

– g_f est une garde, également appelée *condition de déclenchement*, qui indique à quel moment un service se déclenche,

– P_f est le *protocole* du service, qui remplace le comportement normal lorsque le service est lancé. Le premier événement exécutable dans P_f est appelé *événement déclencheur*. Le service est considéré comme lancé lorsque cet événement a été exécuté,

– \mathcal{R}_f est l'ensemble des réactions qui remplacent les réaction du comportement normal lorsque le service f est lancé,

– I_f a le même sens que précédemment.

Le filtrage d'appels sortants, ou Outgoing Call Screening (OCS) est un service de type 2. Il permet à un utilisateur de définir une liste de numéros qui ne doivent jamais être contactés à partir de son combiné. Ce service contient une partie comportementale, qui prévoit que lorsqu'un numéro filtré est composé, un message indiquant que le numéro ne peut être atteint est délivré par une boîte vocale. Du point de vue de l'utilisateur, ce service peut également être décrit par la propriété : Un numéro de ma liste de filtrage ne peut jamais être contacté en utilisant mon téléphone . Soit $Talking(u, u')$ une variable booléenne positionnée à vrai lorsque les utilisateurs u et u' sont connectés. Soit $OCS_list(u)$ la liste de filtrage de l'utilisateur u . Alors, l'exigence du service OCS pour un utilisateur A qui doit être préservée par toute exécution peut s'exprimer par la propriété : $\nexists u, Talking(A, u) \wedge u \in OCS_list(A)$. Le service OCS peut être défini par un nuplet :

$ocs = (g_{ocs} = u \in OCS_list, H_{ocs}, \mathcal{R}_{OCS}, I_{ocs} = \nexists u \text{ tel que } Talking(A, u) \wedge u \in OCS_list(A))$, où H_{ocs} est le HMSC de la Figure 5, et \mathcal{R}_{OCS} est l'ensemble des réactions normales du POTS.

La raison pour laquelle les services sont classés en deux catégories est que dans le premier cas, un abonné définit de nouvelles réactions du réseau aux stimuli extérieurs, mais ne peut déclencher lui-même ou interagir avec les nouvelles fonctionnalités. La Figure 6-a illustre une exécution de service de type 1 : dans un premier temps, le comportement suit la description donnée dans le mode normal. Lorsqu'un événement déclenche le service de type 1, une réaction du réseau traite cet événement parallèlement à l'exécution, et sans interaction avec l'utilisateur.

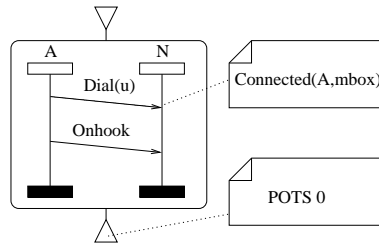


Figure 5 –. HMSC R_3 : Service Outgoing Call Screening.

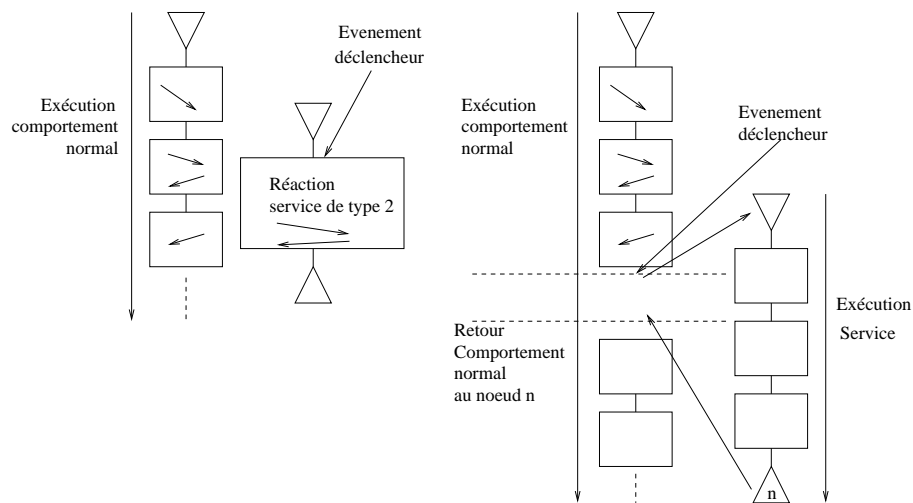


Figure 6 –. Services de type 1. b) Services de type 2.

Dans la deuxième classe de services, un nouvel ensemble de scénarios remplace le comportement normal, fournissant ainsi à l'utilisateur de nouvelles fonctionnalités. Lorsqu'un service de type 2 se termine, l'utilisateur et le réseau reviennent à un comportement normal, dans un état précisé par le noeud de fin du service. La Figure 6-b illustre l'exécution d'un service de type 2 : à partir du comportement normal, un événement déclenche un service. Le comportement de l'utilisateur et du réseau suivent alors la description donnée par ce service. Lorsque celui-ci s'achève, réseau et utilisateur reviennent au comportement normal à partir du noeud n du comportement normal spécifié à la fin du service.

3.2. Sémantique opérationnelle

Un réseau téléphonique réagit suivant son comportement normal tant qu'aucun service n'est déclenché. Lorsqu'un service est activé, le comportement du système est défini par la spécification du service. Une fois que le service est achevé (c'est à dire lorsqu'un noeud de fin du HMSC décrivant le service est atteint), le système revient

au mode normal d'exécution, dans un état particulier. Une exécution d'un service peut être vue comme une interruption du comportement normal. Par la suite, nous supposons qu'un service ne peut être interrompu par un autre service. Un service modifie le comportement possible d'un utilisateur et du réseau, mais également l'ensemble des réactions activables du réseau. A un instant donné, chaque vue du réseau, se trouve dans un état local.

Definition 7: Un *état local* ls_u d'un utilisateur u est un nuplet $ls_u = (S_{P_u}, S_R, mode)$, dans lequel S_{P_u} est un état du système de transitions associé au comportement de u dans le mode courant, S_R est l'ensemble des états de chaque système de transitions associé à chaque réaction en cours, et $mode$ est soit le mode normal (appel de base), ou un nom de service (Cette information est nécessaire pour savoir quelles réactions du réseau doivent être activées par des stimuli extérieurs).

Chaque comportement définit un système de transitions. L'évolution de ce système est donnée par les règles suivantes. Une transition décrit une action, le déclenchement d'un service, une réaction à un stimulus externe, la fin d'un service ou d'une réaction. Pour tout utilisateur u dont le comportement normal est défini par le protocole P , l'état de départ est : $ls_u^0 = (S_{G_P}^0, \emptyset, mode = normal)$. $S_{G_P}^0$ est calculé grâce à la représentation par grammaire de graphes des états présentée au chapitre 2.3. Dans l'état initial, aucune réaction ou service n'a été déclenché. L'ensemble des réactions en cours est donc vide, et la simulation débute en mode normal.

Actions : Une action est l'occurrence d'un événement qui n'affecte pas le mode courant d'un utilisateur, ni le nombre de réactions en cours.

$$\frac{S_{G_H} \xrightarrow{e} S'_{G_H}}{(S_{G_H}, S_R, mode) \xrightarrow{e} (S'_{G_H}, S_R, mode)} \quad \frac{\begin{array}{l} \exists s_r \in S_R, s_r \xrightarrow{e} s'_r, \\ S'_R = S_R - \{s_r\} \cup \{s'_r\} \end{array}}{(S_{G_H}, S_R, mode) \xrightarrow{e} (S_{G_H}, S'_R, mode)}$$

Réactions : Une réaction est la réponse du réseau à un stimulus extérieur. Le déclenchement d'une réaction peut être comparé à la création d'un processus s'exécutant parallèlement au comportement normal ou au service en cours d'exécution.

$$\frac{\exists r = (g_r, H_r) \in R_u, S_{G_{H_r}}^0 \xrightarrow{e} S'_{G_{H_r}}}{(S, S_R, mode = normal) \xrightarrow{e} (S, S_R \cup \{S'_{G_{H_r}}\}, mode = normal)}$$

$$\frac{\exists r = (g_r, H_r) \in R_f, S_{G_{H_r}}^0 \xrightarrow{e} S'_{G_{H_r}}}{(S, S_R, mode = feature(f)) \xrightarrow{e} (S, S_R \cup \{S'_{G_{H_r}}\}, mode = feature(f))}$$

Déclenchement de service : Comme indiqué précédemment, nous considérons qu'un seul service peut être déclenché à la fois. Par conséquent, un service ne peut être déclenché qu'en mode normal. Nous définissons une règle spécifique pour chaque type de service. Déclencher un service de type 1 consiste juste à ajouter une nouvelle réaction aux réactions existantes. Puisque les réactions n'impliquent pas d'échange de message entre l'utilisateur et le réseau, le comportement normal ne sera pas affecté.

$$\frac{\exists f = (R_f, I_f) \in \mathcal{F}, \exists r = (g_r, H_r) \in R_f, S_{G_{H_r}}^0 \xrightarrow{e} S'_{G_{H_r}}}{(S, S_R, mode = normal) \xrightarrow{e} (S, S_R \cup \{S'_{G_{H_r}}\}, mode = normal)}$$

La règle sémantique pour un service de type 2 est légèrement différente, car le comportement normal est remplacé par le scénario défini par le service déclenché.

$$\frac{\exists f = (g_f, P_f, \mathcal{R}_f, I_f) \in \mathcal{F} | g_f = true}{(S, S_R, mode = normal) \xrightarrow{e} (S_{\mathcal{G}_{P_f}}, S_R, mode = feature(f))}$$

Fin de réaction : Une réaction est un scénario fini (sans itération). Lorsqu'une réaction se termine, elle doit être retirée de la liste des réactions en cours.

$$\frac{\exists s_r \in S_R, s_r \xrightarrow{e} s_\emptyset}{(S, S_R, mode) \xrightarrow{e} (S, S_R - \{s_r\}, mode)}$$

où s_\emptyset est un état à partir duquel aucun événement n'est exécutable.

Fin de service : La fin d'un service est la règle la plus complexe. Comme les communication se font de manière asynchrone, un utilisateur peut avoir achevé le service avant le réseau (et inversement). La règle sémantique doit donc prendre en compte le fait que l'utilisateur ou le réseau peuvent avoir un ensemble d'événements à exécuter avant de revenir en mode normal.

$$\frac{S_{\mathcal{G}_H} \xrightarrow{e} S'_{\mathcal{G}_H}, \forall e_1 \dots e_n | S'_{\mathcal{G}_H} \xrightarrow{e_1} S_1 \dots \xrightarrow{e_n} S_n, \phi(e_i) \neq \phi(e), \text{ après avoir exécuté } e, \text{ la spécification revient au noeud } x}{(S_{\mathcal{G}_H}, S_R, mode = feature(f)) \xrightarrow{e} (S'_{\mathcal{G}_H} \circ S_{\mathcal{G}_x(P)}^0, S_R, mode = normal)}$$

où P est le HMSC décrivant le comportement en mode normal.

Jusqu'à présent, nous n'avons défini que des vues locales d'un système téléphonique, c'est à dire les interactions entre un utilisateur particulier et le réseau, les autres utilisateurs étant considérés comme l'extérieur du système. Nous allons maintenant définir comment ces différentes vues utilisateur se recoupent et se superposent pour définir un comportement global cohérent du réseau téléphonique.

Definition 8 : Une *valuation* d'un ensemble de variables est une fonction associant à toute variable une valeur dans son domaine.

Definition 9 : Un *état global* est la donnée d'un ensemble d'états locaux et d'une valuation de variables. Il sera noté par un couple $G = (\{ls_u\}_{u \in U_{sers}}, V)$ où chaque ls_u est l'état local associé à un utilisateur u , et V est une valuation.

L'état initial global d'une spécification est l'état $G_0 = (LS_0, V_0)$, composé de l'ensemble des états locaux initiaux $LS_0 = \{ls_u^0\}_{u \in U_{sers}}$, et d'une valuation initiale de variables V_0 . La valuation $V_{\{VA\}}$ obtenue après application d'un ensemble d'affectations $VA = \{v_i := exp_i\}_{i \in 1..n}$ à une valuation V est la valuation V dans laquelle toute variable v_i de VA a la valeur de exp_i (exp_i est calculée à partir des valeurs de V).

Definition 10 : Un ensemble d'affectations VA est dit *consistant* s'il ne contient pas de contradiction. Par exemple, $VA = \{A_Busy := true; Talking(A, B) := true; A_Busy := false\}$ contient une contradiction, car A_Busy ne peut pas être mis à

vrai et *faux*. VA est consistant pour une valuation V si l'application des affectations de VA à V dans n'importe quel ordre donne la même valuation $V_{\{VA\}}$.

Definition 11: Deux événements e et e' sont dits *unifiables* si et seulement si il existe une affectation VA_{IV} des variables d'instance telle que $e_{\{VA_{IV}\}} = e'_{\{VA_{IV}\}}$.

Une transition globale peut se produire lorsqu'un utilisateur est prêt à effectuer une transition locale a . Tirer cette transition peut également obliger à tirer d'autres transitions dans un autre état local où a peut être considéré comme stimulus extérieur, ou une évolution du réseau. Une transition globale consiste donc à exécuter un événement dans un état local, et modifier les autres états locaux en fonction de cette action. Lorsque plusieurs règles sémantiques peuvent s'appliquer, il existe une priorité entre ces règles. La création d'un service est prioritaire par rapport à une réaction et à l'exécution d'une action. Une exécution d'action est prioritaire par rapport à une réaction. Bien sûr, tout événement tiré dans un état local doit être unifiable avec les autres événements tirés. De plus, l'ensemble des affectations associées à chaque événement local tiré doit être consistant.

$$\frac{\exists u_1 \in Users | el_{u_1} \xrightarrow{a} el'_{u_1}, \exists u_2 \in Users - \{u_1\} | el_{u_2} \xrightarrow{e} el'_{u_2}, \text{unifiable}(a, e), v(a) \cup v(e) \text{ est consistant}}{(LS, V) \xrightarrow{a} (LS', V')}$$

où $LS' = LS - \{el_{u_1}; el_{u_2}\} \cup \{el'_{u_1}; el'_{u_2}\}$ et $V' = V_{\{v(a) \cup v(e)\}}$

$$\frac{\exists u_1 \in Users | el_{u_1} \xrightarrow{a} el'_{u_1}, \nexists u_2 \in Users - \{u_1\} | el_{u_2} \xrightarrow{e} el'_{u_2} \wedge \text{unifiable}(a, e)}{(LS, V) \xrightarrow{a} (LS', V')}$$

où $LS' = LS - \{el_{u_1}\} \cup \{el'_{u_1}\}$ et $V' = V_{\{v(a)\}}$

Considérons l'exécution ci dessous, impliquant deux utilisateurs A et B , n'ayant souscrit à aucun service. Le comportement de A et B sera décrit par deux HMSCs : la spécification du POTS, Figure 3 comme protocole normal, et la réaction R_1 , qui prévoit que le réseau retourne un message *Busy* lorsqu'un appel arrive sur une ligne occupée. Supposons que A essaie de joindre B et regardons quelles règles sont utilisées. Nous écrirons $A !m \text{ to } N$ lorsque l'instance A envoie un message m à l'instance N , et $A ?m \text{ from } N$ lorsque l'instance A reçoit un message m de l'instance N .

Action	Conséquences
A!Offhook to N	Après exécution de cet événement, A est occupé, la variable A_Busy est positionnée à <i>vrai</i> .
N ?Offhook from A	
N !Dialtone to A	
A ?Dialtone from N	
A !Dial(B) to N	Il y a un événement unifiable $v!Dial(B)toN$ dans la spécification de B , donc les états locaux de A et B évoluent. La variable d'instance u vaut à présent B . De plus, la variable B_Busy est positionnée à <i>vrai</i>
N ?Dial(B) from A	Les états locaux de A et B évoluent.

4. Détection d'interactions de services

Selon [KIM 95] : Une interaction de services se produit lorsque le comportement d'un service est affecté par le comportement d'un autre service ou d'une autre instance du même service . Les interactions de services peuvent être classées en différentes catégories, en fonction de leur cause, du nombre d'instances du service impliquées, etc. [CAM 94, KIM 95, SEF 99]. Par la suite, nous nous intéresserons à deux catégories d'interactions particulières : les conflits de déclenchement entre services, et la violation des exigences d'un service.

conflits de déclenchement : Dans notre environnement, les conflits de déclenchement entre deux services se caractérisent par une situation non-déterministe au cours de la simulation du système complet. La recherche des conflits de déclenchement consiste à détecter les états du système à partir desquels deux services peuvent être déclenchés par le même événement. Une approche similaire a été proposée dans un environnement synchrone basé sur LOTOS [FAC 94].

Soit \mathcal{F} un ensemble de services. Un couple de services $(f_1, f_2) \in \mathcal{F}^2$, gardés par g_1 et g_2 , et dont les événements déclencheurs sont e_1 et e_2 , est en conflit de déclenchement si et seulement si il existe un mot w tel que $G_0 \xrightarrow{w} G$, que $V(G) \models g_1 \wedge V(G) \models g_2$, et $G \xrightarrow{a} G' \wedge \text{unifiable}(a, e_1) \wedge \text{unifiable}(a, e_2)$. Notons que $(g_1 \wedge g_2 \neq \text{false}) \wedge \text{unifiable}(e_1, e_2)$ n'est pas une condition suffisante pour décider si (f_1, f_2) est une interaction de services, car le système peut très bien ne jamais atteindre d'état dans lequel $g_1 = \text{true} \wedge g_2 = \text{true}$.

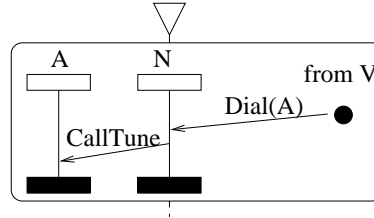


Figure 7 –. HMSC H_1 : Service Call Waiting.

Considérons les services Call Waiting (CW) et Call Forward on Busy (CFB). Le service CW permet à un utilisateur de recevoir un son d'avertissement de la part du réseau lorsque sa ligne est occupée et qu'un autre utilisateur essaie de le joindre. Un souscripteur au CW peut alors décider de commuter vers le nouvel appelant. Le conflit entre CW et CFB est un cas bien connu d'interaction [CAM 94]. L'environnement de simulation permet de détecter cette interaction. Supposons qu'un utilisateur A ait souscrit au CW, et ait redirigé ses appels vers un utilisateur C grâce au CFB lorsque sa ligne est occupée. La vision que A a du réseau est décrite par un nuplet $B = (N_A, \mathcal{F}_A)$, où N_A est la spécification du POTS, et où :

$$\mathcal{F}_A = \{ \begin{array}{l} CW_A = (\text{Talking}(A, x), H_1, \emptyset, \text{true}); \\ CFB_{A \rightarrow C} = (A_Busy \wedge \neg C_Busy, R_2, \emptyset, \text{true}) \end{array} \}.$$

La Figure 4-b donne une représentation graphique du HMSC R_2 du service CFB. La Figure 7 décrit la partie protocole H_1 du service CW. Ce service étant assez complexe, seul le début du service est fourni.

Action	Affectations	Action	Affectations
A !Offhook to N	A_Busy:=true	N ?Dial(B) from A	
N ?Offhook from A		N !Ring(B) to B	
N !Dialtone to A		N !Ringing to A	
A ?Dialtone from N		A ?Ring from N	
A !Dial(B)	B_Busy:=true, la variable d'instance u vaut B	B ?Ringing from N	
		B !Offhook to N	
		N ?Offhook from B	Talking(A,B) :=true

Considérons l'exécution de la table ci dessus. A présent, Si un utilisateur D compose le numéro de A , les services CFB et CW peuvent tous deux se déclencher. Il y a donc une interactions de services entre CFB et CW. L'étude de l'intersection des gardes des services peut aider à détecter du non-déterminisme. Cependant, cela n'est pas suffisant, car des gardes peuvent décrire des propriétés sur des ensembles de variables disjoints, et n'être équivalentes qu'en cours d'exécution. Considérons les gardes A_busy du CFB et $Talking(A, B)$ du CW. A partir des spécifications, il est évident que $Talking(A, B) \implies A_Busy$, mais une simple vérification syntaxique n'aurait pas permis de détecter l'interaction entre CFB et CW.

Violation d'exigence : Ce type d'interaction survient lorsque les exigences de plusieurs services ne sont pas compatibles. Une exigence de service est vue comme une propriété invariante qui doit être préservée par toute exécution du système lorsque le service a été activé. L'exigence I d'un service est *violée* s'il existe un état atteignable dans lequel la propriété I est fausse, c'est à dire s'il existe un mot w tel que $G_o \xrightarrow{w} G$, et $V(G) \not\models I$. Une recherche exhaustive n'est pas toujours possible, car le système étudié peut avoir un nombre infini d'états. A nouveau, la simulation interactive du système peut permettre de trouver un état dans lequel l'invariant du service est violé.

Étudions une autre interaction bien connue, le CFB et l'OCS. Supposons qu'un abonné A ait souscrit à l'OCS, ait mis l'utilisateur C dans sa liste de filtrage, et qu'un utilisateur B ait redirigé ses appels vers un utilisateur C grâce au CFB. L'exigence de l'OCS indique que l'on ne doit pas pouvoir atteindre d'état dans lequel A et C sont connectés. Considérons l'exécution ci dessous, et les affectations associées à chaque événement. Au cours de la simulation un état dans lequel l'exigence de l'OCS est violée a été atteint. Par conséquent, il y a une interaction entre OCS et CFB.

Action	Affectations
B !Dial(D)	Busy_D := true; Busy_B := true
A !Dial(B) to N	Busy_A := true; CFB déclenché
N ?Dial(B) from A	
N !Ring(C) to C	Busy_C := true
N !Ringing to A	
C ?Ring from N	
C !Offhook to N	
N ?Offhook from C	Talking(A,C):=true

5. Conclusion

Cet article a défini un environnement formel permettant de modéliser des réseaux téléphoniques et des services à l'aide de scénarios, puis de les simuler afin de détecter deux types d'interactions : les conflits de déclenchement, et la violation des exigences. Cette approche orientée utilisateur pourrait être étendue pour prendre en compte des services et exigences spécifiques au réseau, (cas dans lesquels un service influe sur le comportement du réseau), ou d'autres classes d'interactions, telles que les conflits liés aux ressources.

6. Bibliographie

- [FIW 92] 1st Int. Workshop on Feature Interactions in Telecom & Software Systems, Floride, Décembre 1992.
- [FIW 94] 2nd Int. Workshop on Feature Interactions in Networks, Pays-Bas, Mai 1994.
- [FIW 95] 3rd Int. Workshop on Feature Interactions in Telecom & Software Systems, Japon, 1995.
- [FIW 97] 4th Int. Workshop on Feature Interactions in Telecom & Software Systems, Canada, Juin 1997.
- [FIW 98] 5th Int. Workshop on Feature Interactions in Telecom & Software Systems, Suède, Octobre 1998.
- [BER 97] J.Bergstra, W.Bouma, Models for Feature Descriptions and Interactions, in Feature Interactions in Telecommunications and Distributed Systems IV, P.Dini et al. (Eds.) IOS Press 1997, pp 31- 45.
- [BLO 97] J. Blom, Formalisation of Requirements with Emphasize on Feature Interaction Detection, Feature Interactions in Telecommunications Networks IV, P.Dini et al. (Eds.) IOS Press 1997, p 61.
- [CAM 94] E.J. Cameron, N.D. Griffeth, Y.J. Lin, M.E. Nilson, W.K. Schnure, and H. Velthuijsen, A Feature Interaction Benchmark for Intelligent Network and Beyond , pp. 1-23, Amsterdam, IOS Press, Mai 1994.
- [FAC 94] M.Faci and L.Logrippo, Specifying Features and Analyzing their Interactions in a LOTOS Environment, Second International Workshop on Feature Interactions in Telecommunication Software Systems, IOS Press, 1994, pp. 136 - 151.
- [GRA 96] Graubmann.P, Rudolph.E, Grabowski.J. Tutorial on message sequence charts (msc'96). FORTE/PSTV'96, Octobre 1996.
- [HEL 98] L. Helouet, C. Jard, B. Caillaud, An Effective equivalence for sets of scenarios represented by HMSCs, Rapport de recherche INRIA n0 3499, Septembre 1998.
- [HEL 99] L.Helouet, A Simulation framework for Message Sequence Charts, Proceedings of the 9th SDL Forum, Juin 1999.
- [ITU 00] Z120 ITU-T, Norme Z.120, Message Sequence Charts (MSC 2000)
- [KEC 98] D.O. Keck and P. J. Kuehn, The Feature and Service Interaction problem in Telecommunication systems: A survey, IEEE Transactions on Software Engineering, Vol. 24, No. 10, Octobre 1998.
- [KIM 95] K.Kimble, H. Velthuijsen, Feature Interaction Benchmark Presented in the Panel Session Benchmarking, FIW '95, Kyoto, Japan, 11-13 Octobre, 1995.
- [MAUW 97] Mauw S. , Reniers M.A., High-level Message Sequence Charts. In A. Cavalli and A. Sarma, editors, SDL'97: Time for Testing - SDL, MSC and Trends, Proceedings of the Eighth SDL Forum, p 291-306, France, Septembre 1997.
- [SEF 99] A. Sefidcon and F. Khendek, A Pragmatic Approach for Feature Interaction Detection in Intelligent Networks, in Proceedings of the IEEE International Conference on Computer Communications and Networks (IEEE ICCCN'99), Boston, Octobre 1999.