Test formel de systèmes interactifs multimodaux : couplage ICARE – Lutess

Frédéric Jourde[1,2], Laurence Nigay[1] et Ioannis Parissis[2]
1- Laboratoire CLIPS-IMAG 2- Laboratoire LSR-IMAG
BP 53 38041 Grenoble Cedex 9 - FRANCE
{Prénom.Nom}@imag.fr

I Introduction

Aujourd'hui, l'efficacité recherchée n'est plus celle de la technologie prise isolément mais celle du couple "machine-humain". Il ne s'agit plus seulement d'exploiter au mieux les ressources techniques, mais de développer une technologie qui soit conforme aux attributs de la communication humaine. Et l'être humain est multimodal : voit, entend, parle, gesticule, manipule. Ce constat entraîne de nombreuses possibilités de modalités d'interaction et formes de multimodalité. La multiplicité des possibilités peut se voir comme un facteur de souplesse et de sa compagne immédiate, la complexité. Dans nos travaux, nous traitons le problème de la complexité de réalisation logicielle des systèmes interactifs multimodaux. La complexification de tels systèmes interactifs rend nécessaire leur développement et leur validation rigoureux. Pour cela, nous étudions l'application de techniques de test de logiciels réactifs synchrones basées sur l'outil Lutess [5] à la validation de systèmes interactifs multimodaux.
Dans cet article, nous limitons l'étude de techniques de test de logiciels réactifs synchrones basées sur Lutess à la validation de systèmes interactifs multimodaux dont l'architecture logicielle est selon le modèle à composants ICARE. ICARE est un environnement de développement à composants de systèmes interactifs multimodaux [2]. Aussi dans cet article, nous présentons d'abord l'approche ICARE puis les principes de tests formels avec l'outil Lutess. Nous expliquons ensuite notre couplage ICARE et Lutess basé sur l'architecture logicielle ICARE sous-jacente. Pour illustrer notre propos tout au long de l'article, nous considérons un système interactif multimodal YellowPage que nous présentons dans le paragraphe suivant.

II Système interactif multimodal YellowPage

Nous illustrons nos travaux en considérant le système interactif multimodal YellowPage que nous avons développé. YellowPage est une version très simplifiée d'un système comme PagesJaunes disponible à www.pagesjaunes.fr. En effet, notre système YellowPage permet de localiser un lieu sur une carte, à partir du nom et de l'adresse d'une personne. Pour cela, l'utilisateur réalise plusieurs tâches successives en rapport avec les lieux d'interaction de l'interface graphique présentée à la figure 1 : -1- Spécifier le nom d'une personne, -2- Spécifier l'adresse de cette personne, -3- Lancer une recherche, -4- Naviguer dans la carte (zoom, déplacement, recentrage, etc.).
YellowPage propose plusieurs modalités d'interaction en entrée, de l'utilisateur vers le système. Les trois dispositifs physiques disponibles sont : une souris, un clavier et un microphone. Ainsi, les tâches (1, 2, 3, et 4) peuvent être réalisées par l'utilisation de plusieurs modalités. La tâche de recherche peut par exemple être initiée soit par la souris, en sélectionnant le bouton "Search", soit par l'enfoncement de la touche "enter" du clavier, ou encore par la commande vocale "search".

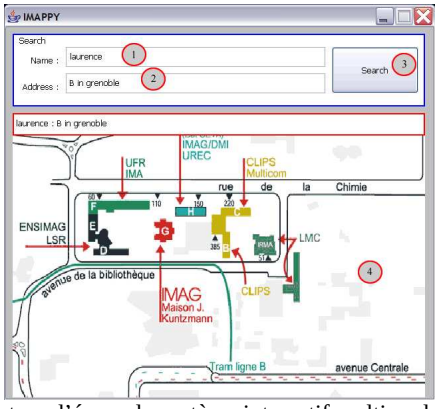

Figure 1 : Capture d'écran du système interactif multimodal YellowPage.

Au sein de ce prototype, pour les champs de texte "Name" et "Address", nous limitons le nombre de noms reconnus à trois (noms parmi "Cooper", "Laurence" et "Smith"), et le nombre d'adresses à deux, chacune décrite de trois manières (adresses parmi "Buidling < B ou C > in Grenoble", "< B ou C > in Grenoble" et "< B ou C> Grenoble").

III Architecture ICARE de YellowPage

L'interaction multimodale du système YellowPage est développée grâce à un assemblage de composants ICARE, qui constitue un moteur de fusion d'événements multimodaux en entrée. La plate-forme ICARE [2], pour Interaction-CARE (Complémentarité, Assignation, Redondance, Equivalence), permet aux concepteurs de manipuler graphiquement et d'assembler des composants logiciels ICARE afin de spécifier l'interaction multimodale pour une tâche donnée d'un système interactif. De cette spécification, le code de l'assemblage est automatiquement généré. Nous nous intéressons ici à l'architecture logicielle à composants ICARE résultante. Pour cela, nous présentons d'abord le modèle conceptuel à composants ICARE. Puis, nous exposons les combinaisons d'entrées proposées par YellowPage, accompagnées de certains schémas ICARE les implémentant.

Le modèle à composants ICARE comprend deux types de composants : les composants élémentaires qui permettent de définir des modalités d'interaction et les composants de composition (ou combinaison) qui permettent de spécifier l'usage combiné de modalités. Les composants de composition sont indépendants des modalités à composer.
Basés sur la définition d'une modalité d'interaction comme étant l'association d'un dispositif physique d avec un langage d'interaction L, <d, L> [6], *deux types de composants ICARE élémentaires* sont identifiés : les composants de type Dispositif et les composants de type Langage d'Interaction.

- Un composant Dispositif représente une couche supplémentaire du pilote d'un dispositif physique. Il s'agit du niveau physique d'une modalité. Par exemple, le composant Dispositif Souris abstrait les données (mouvements effectués ou pression sur les boutons) fournies par le pilote de la souris.
- Un composant Langage d'Interaction correspond au niveau logique d'une modalité d'interaction. Par exemple, un composant Langage d'Interaction peut abstraire les données d'un composant Dispositif Souris en une commande correspondant à la sélection dans un menu d'options.

Les composants de composition sont définis en se basant sur les propriétés CARE (Complémentarité, Assignation, Redondance, Équivalence) [4, 6]

- L'équivalence et l'assignation caractérisent la portée des choix en matière de modalités d'interaction. Deux modalités sont équivalentes pour une tâche élémentaire ou commande si l'utilisateur peut utiliser l'une ou l'autre des modalités pour réaliser la tâche. Au contraire lorsqu'une modalité est assignée à une tâche, l'utilisateur n'a pas le choix de la modalités.
- La redondance et la complémentarité qualifient la combinaison de modalités. Deux modalités sont utilisées de manière redondante si elles sont équivalentes et véhiculent la même information dans une fenêtre temporelle donnée. C'est le cas lorsque l'utilisateur énonce la commande vocale « search » tout en cliquant sur le bouton « Search » avec la souris.
  Deux modalités sont utilisées de façon complémentaire dans une fenêtre temporelle donnée si chacune véhicule une information utile pour obtenir une commande complète. Par exemple lorsque l'utilisateur énonce la commande vocale « Zoom in here » tout en sélectionnant une position sur la carte, deux modalités distinctes sont utilisées pour spécifier la commande de zoom sur un point.

Basés sur ces propriétés CARE, *quatre composants ICARE de composition* sont définis et permettent de combiner les données de 2 à n composants : un composant de composition pour la Complémentarité, un pour la Redondance, un pour l'Equivalence et un pour la Redondance/Equivalence. Ce dernier caractérise l'usage de modalités équivalentes qui peut être soit redondant (Redondance) soit exclusif (Equivalence). L'Assignation de CARE n'est pas explicite dans la spécification ICARE car elle est représentée par un simple lien entre deux composants. En effet, un composant A lié à un composant B implique que A est assigné à B.
La communication entre les composants ICARE est implémentée sous la forme d'événements ICARE (ICAREEvents). Cette communication est statique à l'exécution et est représentée dans un schéma ICARE par des flèches comme le montre la figure 2. Une flèche dénote un langage entre les composants émetteur et receveur. Les ICAREEvents circulant entre deux composants portent un des mots de ce langage. Les composants de type Dispositif émettent des ICAREEvents de bas niveau d'abstraction. Ceux-ci sont traduits au sein des composants de type Langage d'Interaction en événements de plus haut niveau d'abstraction, pour être ensuite fusionnés par les composants de composition afin d'obtenir des tâches ou commandes complètes traitées par le reste du système interactif.

Illustrons l'architecture à composants ICARE en considérant le système YellowPage. Nous décrivons d'abord les commandes simples sans paramètre décrits par le schéma ICARE de la figure 2. Celui-ci comprend trois composants de type Dispositif : "Mouse", "Keyboard" et "Microphone" associés à 3 composants de type Langage d'Interaction pour constituer 3 modalités. La souris est assignée à la commande rechercher, aussi dans le schéma ICARE le composant « Mouse » est assigné au composant Langage "Search". Les composants Langage "Simple Commands 1 et 2" traduisent les entrées provenant du clavier et du microphone en commandes parmi "zoom in", "zoom out", "up", "down", "right", "left", "center", et "search". Les trois modalités sont

fusionnées de manière redondante et équivalente grâce au composant de composition ICARE Redondance/Equivalence. Le composant Redondance/Equivalence définit une fenêtre temporelle T dans laquelle sera traitée la redondance d'information : par exemple si l'utilisateur énonce la commande vocale « search » et successivement clique sur le bouton «Search» avec la souris dans un intervalle de temps inférieur à T, alors les deux commandes sont fusionnées en une seule commande de recherche.

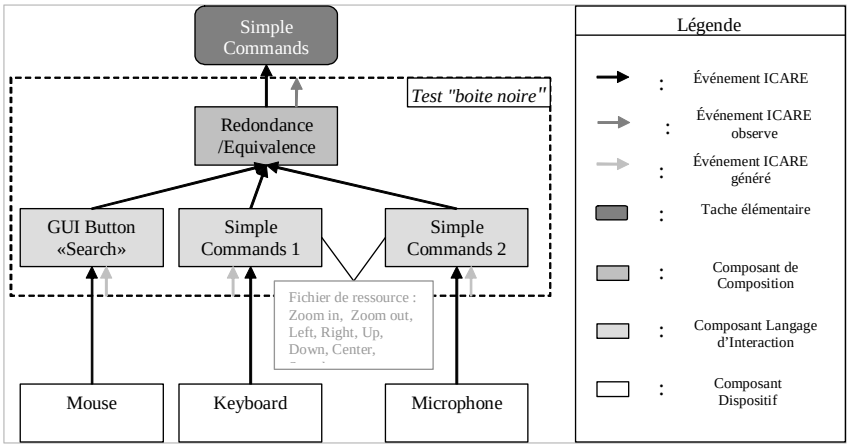

Figure 2 : Schéma ICARE pour des commandes simples sans paramètre de YellowPage.

Outre les commandes simples, l'utilisateur peut utiliser une ou deux modalités pour les tâches complémentaires de sélection d'un champ et de remplissage de ce champ. La sélection du champ peut s'effectuer par la souris en cliquant sur le champ ou oralement par le mot "name" ou "address". Le remplissage peut s'effectuer au clavier ou à l'oral (les valeurs autorisées étant définies au paragraphe II). Notons qu'un composant de composition Complémentarité fusionne les informations de sélection et de remplissage de manière ordonnée.

Enfin YellowPage propose des tâches de zoom avant/arrière et de centrage, sur un point donné de la carte, dont une partie du schéma ICARE est présentée à la figure 3. Pour effectuer un zoom, l'utilisateur peut spécifier "zoom in/out here" au clavier ou au microphone, et spécifier un point avec la souris. Par exemple l'utilisateur prononce "zoom in here" et pointe avec la souris aux coordonnées (x,y) de la carte, les deux actions étant effectuées à des dates comprises dans la fenêtre temporelle T. Le composant "Complementarity 1" fusionne alors ces deux entrées pour obtenir la commande complète "zoom in point(x,y)".

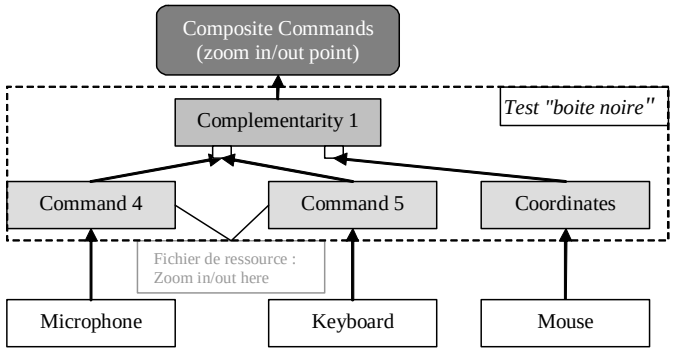

Figure 3 : Schéma ICARE pour les commandes zoom avant/arrière sur <point> de YellowPage.

IV Principe de validation avec Lutess

Notre approche de vérification repose sur l'hypothèse qu'un système interactif multimodal peut se comporter comme un système réactif synchrone, c'est-à-dire, que son comportement au cours du temps peut être représenté sous la forme d'une suite de couples <entrée, sortie>. Sous cette hypothèse, nous utilisons l'outil de vérification de logiciel réactif synchrone Lutess [3, 7]. Lutess construit automatiquement un générateur de données de test pour le programme sous test. Pour cela, il requiert en entrée une description de l'environnement écrite en langage Lustre [1] ainsi que des directives de guidage. Le test est effectué par cycles successifs action-réaction. Un cycle se déroule comme suit :

1. Lutess génère un vecteur d'entrées.
2. Lutess l'envoie au logiciel sous test puis attend.
3. Le logiciel sous test lit le vecteur d'entrées.
4. Le logiciel sous test réagit en envoyant un vecteur de sorties à Lutess.
5. Lutess lit les sorties du logiciel sous test.
6. Lutess vérifie la propriété décrite dans l'oracle en prenant en compte les entrées-sorties de ce cycle.

A la figure 4, nous situons ces étapes du cycle au sein de l'architecture globale qui regroupe Lutess et le logiciel sous test. Nous considérons le cas où l'architecture du logiciel sous test est selon le modèle à composants ICARE. Tester un logiciel comme YellowPage avec Lutess nécessite donc de spécifier l'environnement et l'oracle de test.

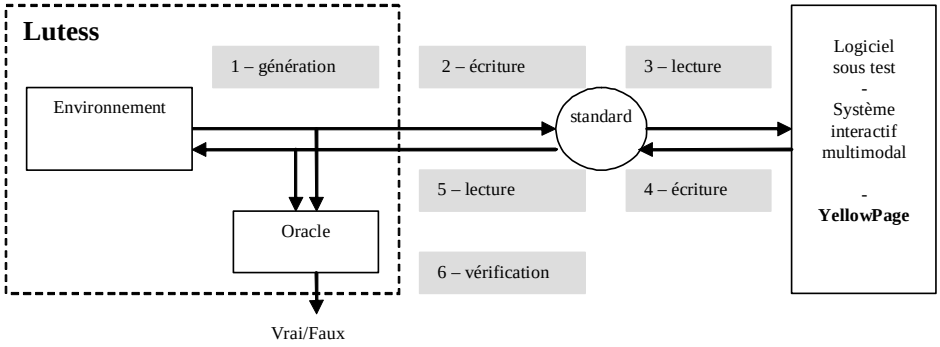

Figure 4 : Architecture globale et étapes des cycles de vérification d'un logiciel interactif multimodal avec Lutess.

La spécification de l'environnement définit des invariants liant les entrées et les sorties du logiciel sous test. Elle est enrichie par des directives de guidage de la génération :

- Spécification d'invariants : il s'agit de propriétés logiques et temporelles sur les entrées-sorties du logiciel sous test. Lutess garantit alors que l'ensemble des jeux d'essai qui sont soumis vont satisfaire ces invariants. Par exemple, si E1 et E2 sont deux entrées du logiciel sous test, il est possible d'exprimer la propriété : "E2 suit toujours E1" par l'expression Lustre "not E2 or pre E1".
- Spécification de probabilités conditionnelles : il s'agit d'affecter la probabilité P qu'une entrée E soit vraie, si une condition C est vraie, par l'expression "proba ((E, P, C))". Lutess vérifie à chaque cycle la validité de C, puis génère la valeur de E en fonction du résultat.
- Spécification de contraintes par rédaction de scénarios : il s'agit d'écrire des scénarios sous la forme d'une suite de <condition, action>. La génération de tests va être orientée vers la reproduction des séquences satisfaisant le scénario.

Lutess vérifie la validité d'une propriété logique et temporelle exprimée en Lustre dans un oracle. Lustre étant une logique temporelle du passé, on ne peut pas exprimer une référence au futur.

Dans nos travaux, nous utilisons l'outil Lutess pour tester des systèmes interactifs multimodaux dont l'architecture est selon le modèle ICARE. L'originalité de notre approche réside dans la vérification de propriétés ergonomiques de la multimodalité, comme les propriétés CARE [4, 6]. Pour cela, nous effectuons des tests de type boîte noire avec Lutess des assemblages de composants ICARE pour l'interaction multimodale en entrée. Le paragraphe suivant explique comment nous couplons Lutess aux composants ICARE puis au paragraphe VI nous présentons les résultats de deux expérimentations.

V Couplage ICARE-Lutess

Nous expliquons comment effectuer des tests formels par le couplage de Lutess à un système interactif multimodal développé avec la plateforme ICARE. Pour cela, nous détaillons d'abord la communication entre Lutess et le logiciel sous test puis les niveaux d'abstraction des tests effectués au regard de l'architecture logicielle ICARE du logiciel sous test. Enfin nous expliquons l'instrumentation complète automatique par l'extension apportée à la plateforme ICARE.

V.1 Communication ICARE-Lutess

Lutess communique avec l'extérieur par la lecture-écriture de booléens sur l'entrée-sortie standard. Ce type de communication implique le développement de deux mécanismes.

- Un mécanisme de communication réalise la lecture-écriture de vecteurs de booléens sur l'entrée-sortie standard du côté du logiciel sous test.
- Un mécanisme de traduction de la sémantique des événements permet à Lutess et au logiciel sous test de se comprendre. Ce mécanisme dépend des entrées-sorties choisies pour être simulées et observées sur le logiciel sous test. Ces entrées-sorties dépendent elles-mêmes des propriétés à vérifier.

La communication s'effectue au moyen d'événements ICAREEvents qui sont échangés au sein d'un schéma ICARE depuis les composants de type Dispositif (niveau inférieur d'un schéma ICARE) jusqu'aux tâches ou commandes (niveau supérieur d'un schéma ICARE). Chaque ICAREEvent porte une sémantique propre correspondant à un niveau d'abstraction dans la fonction qui abstrait les actions de l'utilisateur en tâche ou commande traitée par le système interactif. Lors des tests avec Lutess, nous simulons un événement par

l'instanciation d'un ICAREEvent et son envoi à un composant ICARE. Nous pouvons ainsi simuler le comportement d'un composant C, en émettant des ICAREEvents à destination des composants cibles de C.

La figure 5 détaille le code permettant la connexion entre ICARE et Lutess. Celui-ci comprend trois classes : "Codeur", "Décodeur" et "LutessInterface". La classe "Décodeur" comprend un ensemble de méthodes "Generate* (booléen [ ] B)", chacune correspondant à une flèche du schéma ICARE. Une méthode "Generate* (booléen [ ] B)" crée une instance d'ICAREEvent, lui attribue une sémantique en fonction de B, et l'envoie au composant "*". La Classe "Codeur" implémente des écouteurs d'ICAREEvents. Elle s'abonne aux différents composants ICARE, en sollicitant l'appel d'une méthode "From*" en cas d'événement. Il existe donc une méthode "From*" par composant ICARE, ce qui permet d'observer tous les fils d'interaction. Les méthodes "From*" traduisent l'ICAREEvent transmis en paramètre en un vecteur de booléens qui est ensuite stocké. La classe "LutessInterface" contient plusieurs méthodes pour la lecture et l'écriture sur l'entrée-sortie standard, ainsi qu'une méthode "ConnectToLutess". Celle-ci implémente l'attente mutuelle entre le logiciel sous test et Lutess. Elle comprend plusieurs étapes :

1. Lecture de vecteurs de booléens
2. Appels des méthodes de la classe Décodeur correspondant aux vecteurs lus
3. Attente de 1000 ms
4. Écriture des ICAREEvents observés

L'attente de 1000 ms permet de stabiliser la durée d'un cycle de simulation de manière à permettre l'expression de propriétés temporelles sur le temps réel, à partir du temps discret. Pour cela, il convient que le temps réel soit découpé en intervalles égaux correspondant à la durée d'un cycle. Nous avons mesuré au cours d'expérimentation qu'un cycle dure entre 0,2 et 0,8 ms. Nous introduisons un délai de 1000 ms qui rend négligeable les durées précédentes et rend ainsi stable la durée d'un cycle. Cela rend possible l'expression de propriétés temporelles sur le temps discret des cycles de test.

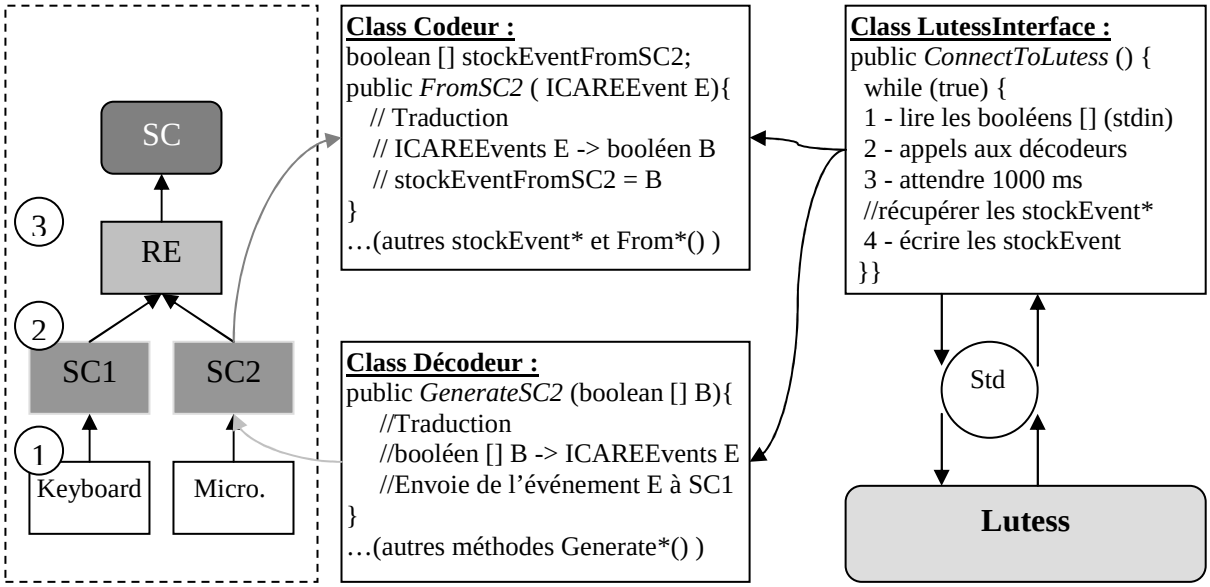

Figure 5 : Schéma détaillé de la connexion ICARE-Lutesse.
A gauche nous considérons une partie du schéma ICARE YellowPage de la figure 2.

V.2 Niveaux d'abstraction des événements simulés

L'instrumentation proposée pour la connexion ICARE-Lutess consiste à simuler des événements ICARE et donc à simuler le comportement de composants ICARE. Nous distinguons trois niveaux d'abstraction de simulation notés 1, 2 et 3 à la figure 5.

1. **Niveau dispositif :** il s'agit de simuler un composant ICARE de type Dispositif par la génération d'événements ICAREEvents à destination du composant Langage d'Interaction associé. Par exemple à la figure 5, nous simulons le composant Dispositif "Micro" en émettant des événements vers le composant "SC2, Simple Commands 2".
2. **Niveau langage d'interaction :** il s'agit de simuler un composant ICARE de type Langage d'Interaction. La simulation est réalisée en émettant des événements ICAREEvents à destination des composants cibles de type Composition ou Langage d'Interaction ou à destination du reste du système en simulant des tâches élémentaires abstraites ou commandes. Par exemple à la figure 5, pour simuler le

composant "SC2, Simple Commands 2", des événements sont générés à destination du composant de composition "RE, Redondance/Equivalence".

3. **Niveau composition :** il s'agit de simuler un composant ICARE de Composition par l'émission d'événements à destination de ses cibles : des composants de type Composition ou Langage d'Interaction ou encore le reste du système si il s'agit de tâches élémentaires abstraites. Pour exemple à la figure 5, pour simuler le composant "RE, Redondance/Equivalence", des événements sont générés à destination de la tâche abstraite "SC, Simple Commands".

Ainsi notre approche permet d'effectuer des tests en simulant des entrées de l'utilisateur du système interactif multimodal à différents niveaux d'abstraction. Il est donc possible de simuler des actions physiques de l'utilisateur (Dispositif) et des actions de plus haut niveau d'abstraction (Langage d'Interaction et Composition).

V.3 Extension de la plateforme ICARE

Nous avons intégré la génération de l'instrumentation au sein de la plateforme ICARE. L'utilisateur peut l'activer ou la désactiver par la boîte à cocher du menu de l'éditeur ICARE de la figure 6. Une grande partie du code de connexion présenté au paragraphe V.1 est générée automatiquement (étape 2 de la figure 6). Pour effectuer des tests, il convient néanmoins de décrire les fichiers environnement et oracle (étape 3 de la figure 6), puis de compléter le contenu de la boucle "while" de la classe "LutessInterface" (étape 4 de la figure 6). Ces activités manuelles de programmation dépendent des vérifications à effectuer et correspondent à l'écriture d'un total de moins de 100 lignes de code (JAVA/LUTRE). Après compilation (étape 5 de la figure 6), les tests peuvent alors être effectués (étape 6 de la figure 6). Tandis que la vérification de nouvelles propriétés sur les mêmes entrées-sorties entraîne la modification de l'oracle seulement (étape 3), effectuer des tests avec de nouvelles entrées-sorties implique de refaire les étapes 3, 4 et 5.

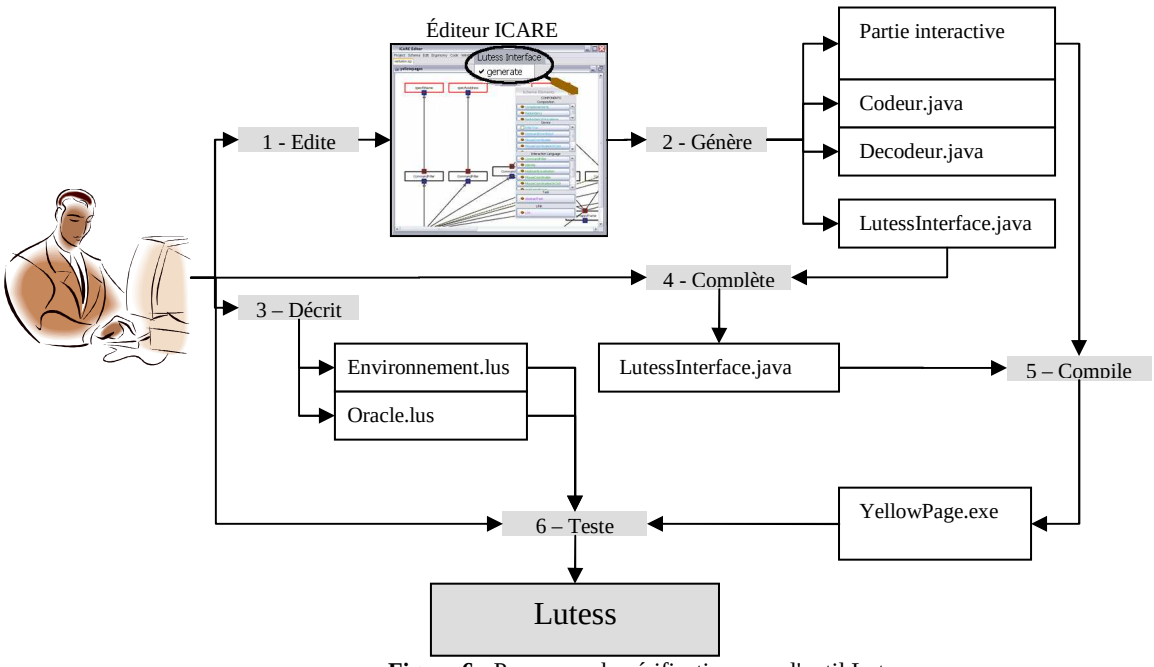

Figure 6 : Processus de vérification avec l'outil Lutess
d'un système interactif multimodal développé avec la plateforme ICARE.

VI Tests expérimentaux de YellowPage

Pour tester YellowPage, nous avons d'abord effectué l'instrumentation manuellement avant de l'automatiser dans la plateforme ICARE. Nous avons mis environ un jour pour programmer l'instrumentation et trois jours pour en assurer la correction. L'extension de la plateforme permet maintenant de produire l'instrumentation presque automatiquement. Il s'agit pour YellowPage d'environ 2400 lignes de JAVA, soit 1400 pour l'entrée, et 1000 pour la sortie.

Dans cette partie, nous présentons des résultats expérimentaux de tests de YellowPage en étudiant particulièrement des propriétés ergonomiques de la multimodalité : l'usage redondant/équivalent de modalités d'interaction. Nous testons la redondance et l'équivalence des entrées du système interactif multimodal YellowPage pour les commandes simples sans paramètre dont le schéma ICARE est présenté à la figure 2. La figure 2 souligne le niveau d'abstraction des tests effectués par un rectangle en pointillé intitulé test boîte noire sur le schéma ICARE. Le niveau d'abstraction des événements simulés est celui des dispositifs. En entrée, nous

simulons les composants ICARE de type Dispositif et en sortie nous observons les événements ICARE de plus haut niveau d'abstraction, le niveau des tâches élémentaires abstraites ou commandes.

VI.1 Fichier d'environnement

Pour ce test, Lutess doit générer des jeux d'essai correspondant aux trois composants de type Dispositif "Mouse", "Keyboard" et "Microphone". Nous démarrons le test de YellowPage en considérant que le nom et l'adresse sont spécifiés. Pour cela, nous décrivons un environnement qui produit dans un premier temps, un événement de recherche suivi par des événements de navigation "Zoom In". L'événement de recherche peut être spécifié par l'usage d'un des trois dispositifs tandis que les commandes de navigation sont effectuées au clavier et oralement. Afin d'obtenir un comportement réaliste des entrées, nous interdisons la génération de plusieurs entrées vocales simultanées en spécifiant un invariant. Nous observons l'affichage de la carte à chaque cycle. Le comportement "recherche puis navigation" est obtenu par la définition de probabilité affectée aux différentes entrées en fonction de l'affichage de la carte.

VI.2 Fichier d'oracle

Nous vérifions d'une part l'équivalence des entrées provenant des trois dispositifs simulés et d'autre part la redondance des entrées équivalentes. Notons que la fenêtre temporelle du composant Redondance/Equivalence est de 10000 ms (10 cycles). Il est possible d'exprimer le comportement attendu de la redondance pour une commande donnée. Par exemple, pour "Zoom in", si l'une des deux entrées équivalentes est vraie (clavier "Page Up" et commande vocale "zoom in"), alors la sortie "Zoom In" doit être vraie, sauf si elle l'a déjà été dans les 10000 ms (10 cycles) précédents. Nous écrivons cette propriété sous la forme du nœud LUSTRE de la figure 7.

```
node RE1(const DT:int ; entree, sortie:bool)returns(ok:bool)
let
      ok = not(entree) or ExactementUnDansFenetre(DT, sortie) ;
tel
```

Figure 7 : Nœud LUSTRE exprimant l'usage redondant/équivalent de modalités pour une commande.

Cette propriété est applicable aux entrées équivalentes. Par exemple pour "zoom in" : "RE1(10, Clavier_zoom_in or Voix_zoom_in, RE_zoom_in)". Nous exprimons la propriété finale dans l'oracle par un "et" logique des RE1 pour toutes les commandes simples telles que "zoom in", "up", "down", etc.

VI.3 Résultats des tests

Nous avons effectué plusieurs vérifications consécutives sans détecter de viol de l'oracle. La figure 8 montre une partie des résultats obtenus sous la forme d'un tableau. Nous garantissons la conformité de ce qui est présenté dans le tableau par rapport au fichier de résultats obtenus. Les cycles du test sont numérotés dans la colonne de gauche. Les entrées-sorties vraies pendant un cycle sont présentées respectivement dans les colonnes "Entrées" et "Sorties". La colonne "Oracle" indique la valeur de la propriété de l'oracle à chaque cycle. Les préfixes V, C, et R, correspondent respectivement à Voix, Clavier et Redondance/Equivalence afin de dénoter l'origine de l'événement. Le code IMG signifie que la carte est affichée.

Cycle	Entrées	Sorties	Oracle
10	-	-	vrai
11	V-search	IMG, RE-search	vrai
12	C-zoom_in	IMG, RE-zoom_in	vrai
13	V-zoom_in	IMG	vrai
14	V-zoom_in, C-zoom_in	IMG	vrai
15	-	IMG	vrai
16	V-search	IMG	vrai
17	C-zoom_in	IMG	vrai
18	C-zoom_in	IMG	vrai
19	C-zoom_in	IMG	vrai
21	-	IMG	vrai
22	C-zoom_in	IMG, RE-zoom_in	vrai
42	-	IMG	vrai
43	V-zoom_in	IMG, RE-zoom_in	vrai

Figure 8 : Tableau de synthèse des résultats partiels d'un test pour un usage redondant/équivalent de deux modalités.

Au cycle 11, l'entrée vocale "search" provoque la commande "search" en sortie. Le plan est affiché ce qui implique que la commande a bien été prise en compte par YellowPage. Au cycle 12, on observe une entrée clavier "zoom in", suivie de la commande "zoom in" en sortie. Durant les cycles 13 à 21, plusieurs entrées "zoom in" sont générées et ne provoquent aucune réponse de la part de YellowPage. Ce comportement est attendu puisque pendant les 10 cycles suivant une commande "zoom in" en sortie, aucune entrée "zoom in" ne doit être prise en compte (usage redondant des modalités). Après ces 10 cycles, nous observons au cycle 22 une entrée clavier "zoom in" qui déclenche la commande "zoom in" en sortie.
Enfin nous observons que les deux entrées clavier et voix sont équivalentes aux cycles 22 et 43 où la même commande "zoom in" en sortie est déclenchée suite à l'entrée clavier "zoom-in" au cycle 22 et l'entrée vocale "zoom in" au cycle 43.

VII Conclusion

L'approche de vérification combinant ICARE et Lutess est opérationnelle et efficace. L'originalité de notre approche réside dans la vérification formelle de propriétés ergonomiques. Il convient néanmoins de souligner que nos travaux ne concernent que la vérification de la partie du système interactif dédiée à l'interaction concrète entre l'utilisateur et le système. Le reste du système comme le contrôleur de dialogue qui gère l'enchaînement des tâches n'est pas traité dans nos travaux, ceci limitant les propriétés ergonomiques à vérifier.
Nous envisageons une intégration plus forte de la plateforme ICARE et de l'outil Lutess pour définir un environnement de développement et de test dédié à l'interaction multimodale qui soit complètement intégré, permettant ainsi de tester les assemblages ICARE au cours de leur production. Enfin pour le guidage de la génération des tests, nous souhaitons approfondir le lien existant entre les scénarios d'utilisabilité qui peuvent être issus de l'analyse de la tâche dans le cadre d'une conception itérative centrée sur l'utilisateur et les scénarios de guidage des tests utilisés par Lutess.

VIII Bibliographie

[1] d'Ausbourg, B., "Using Model Checking for the Automatic Validation of User Interface Systems", *International Eurographics Workshop on Design, Specification and Verification of Interactive Systems (*DSVIS), Abingdon, United Kingdom, June 3-5, 1998

[2] Bouchet, J., Nigay, L., Ganille, T., "ICARE Software for Rapidly Developing Multimodal Interface", *Proceedings of the 6th international conference on Multimodal interfaces* (ICMI), new York, pp. 251-258, 2004

[3] du Bousquet, L., Ouabdesselam, F., Richier, J.-L., Zuanon, N., "Lutess: a Specification Driven Testing Environment for Synchronous Software", *Proceedings of the international Conference of software engineering* (ICSE), ACM Press, pp. 267-276, 1999

[4] Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., Young, R., "Four Easy Pieces for Assessing the Usability of Multimodal Interaction : The CARE properties", *Proceedings of the INTERACT'95 conference*, S. A. Arnesen, D. Gilmore Eds., Chapman&Hall Publ., Lillehammer, Norway, pp. 115-120, 1995

[5] Madani, L., Oriat, C., Parissis I., Bouchet, B., Nigay, L., "Synchronous Testing of Multimodal Systems: an Operational Profile-Based Approach", *Proceedings of the International Symposium on Software Reliability Engineering* (ISSRE), Chicago, Illinois, USA, november 8-11, 2005

[6] Nigay, L., Coutaz, J., "Multifeature Systems : The CARE Properties and Their Impact on Software Design", in Intelligence and Multimodality in Multimedia Interfaces, AAAI Press, 1997

[7] Parissis, I., Ouabdesselam, F., "Specification-based Testing of Synchronous Software", *Proceeding of* ACM SIGSOFT *Fourth Symposium on the Foundations of Software Engineering*, ACM Press, pp. 127-134, 1996