THÈSE

Pour l'obtention du grade de

DOCTEUR DE DE L'ÉCOLE NATIONALE SUPÉRIEURE DE MÉCANIQUE ET D'AEROTECHNIQUE

Diplôme National - Arrêté du 7 août 2006

ECOLE DOCTORALE : SCIENCES ET INGENIERIE POUR L'INFORMATION

DOMAINE DE RECHERCHE : INFORMATIQUE ET APPLICATIONS

Présentée par

Linda MOHAND-OUSSAID

Conception et vérification formelles des interfaces homme-machine multimodales : application à la multimodalité en sortie

Directeur de thèse : **Yamine AIT-AMEUR**

Co-direction : **Idir AIT-SADOUNE**

Soutenue le 16 Décembre 2014
Devant la Commission d'Examen

JURY

M. Christophe KOLSKI	Professeur, Université de Valenciennes	Rapporteur
M. Hassan MOUNTASSIR	Professeur, Université de Franche-Comté	Rapporteur
M. Patrick GIRARD	Professeur, Université de Poitiers	Examinateur
M. Christian ATTIOGBE	Professeur, Université de Nantes	Examinateur
M. Yamine AIT-AMEUR	Professeur, INPT-ENSEEIHT, Toulouse	Examinateur
M. Idir AIT-SADOUNE	Professeur Assistant, SUPELEC, Gif-sur-Yvette	Examinateur

A la mémoire de mon père,

A ma mère,

A mon époux Menouar,

A mes soeurs Nora et Nawel.

Remerciements

Je tiens à remercier infiniment **Yamine AIT-AMEUR**, mon directeur de thèse, pour m'avoir donné ma chance en me proposant ce sujet de thèse et pour m'avoir accueilli au sein du laboratoire. Je le remercie particulièrement pour ses orientations scientifiques précieuses, sa disponibilité, sa patience et sa sympathie.

Mes sincères remerciements vont également à **Idir AIT-SADOUNE**, mon co-directeur de thèse, pour son aide sans faille, le temps conséquent qu'il m'a consacré, ses conseils avisés et ses encouragements.

Je tiens à remercier **Christophe KOLSKI** et **Hassan MOUNTASSIR** pour avoir accepté d'être les rapporteurs de mes travaux de thèse, ainsi que **Patrick GIRARD** et **Christian ATTIOGBE** pour avoir accepté d'être membres du jury en tant qu'examinateurs. Je souhaite leur exprimer ma profonde reconnaissance.

Je tiens à remercier **Emmanuel GROLLEAU**, Directeur du LIAS pour m'avoir accueillie au sein du laboratoire.

Je tiens également à remercier tous les membres du LIAS pour l'environnement de travail très agréable, qu'ils m'ont offert. Je pense en particulier à **Ladjel BELLATRECHE** et **Allel HADJALI** pour leurs encouragements, **Frédéric CARREAU** et **Claudine RAULT** pour leurs aides techniques et administratives respectives.

Sans oublier de remercier tous ceux qui me sont chers et m'ont soutenu, ma maman, mon époux, mes soeurs, mes beaux frères et mes neveux.

Enfin, à tous mes amis : **Nadia**, **Chedlia**, **Yassine**, **Younes**, **Amira**, **Selma**.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Tout système informatique (ordinateur, smartphone, borne interactive, jeux vidéo ...) requiert un échange d'informations avec l'utilisateur, aussi bien en entrée pour introduire les commandes ou requêtes de l'utilisateur, qu'en sortie pour présenter l'information à l'utilisateur. Les systèmes informatiques devenant de plus en plus interactifs et utilisés du grand public, l'interface homme-machine prend de plus en plus une part prépondérante dans la réussite ou l'échec du système informatique. Ceci a donné naissance à des thématiques de recherche en interaction Homme-Machine intéressées entre autres par : le développement de nouveaux modes d'interaction, la proposition de modèles de conception pour les interfaces, l'adaptation de l'interface au profil utilisateur et l'évaluation des interfaces.

Les progrès actuels en termes d'interaction Homme-Machine ont abouti à un nouveau type d'interfaces Homme-Machine : les interfaces multimodales. Ces interfaces utilisent des modes d'interaction de plus en plus proches de la communication humaine tels que la parole ou le geste afin de rendre l'interaction plus intuitive. Elles recourent à de nouveaux dispositifs d'interaction (manette, gant à retour d'effort, microphone ...), et sont basées sur un dialogue multi-fils privilégiant les échanges par question/réponse.

Les interfaces homme-machine multimodales offrent à l'utilisateur la possibilité de combiner, à un niveau sémantique, les modalités d'interaction afin d'augmenter la robustesse et l'utilisabilité de l'interface utilisateur d'un système. Plus particulièrement, les interfaces multimodales en sortie décomposent sémantiquement, dans un premier temps, l'information générée par le noyau fonctionnel, puis produisent les informations décomposées sur les différents médias et modalités afin de construire la présentation à restituer à l'utilisateur. Cette interaction, plus naturelle pour l'utilisateur certes, implique toutefois deux principes supplémentaires pour la mise en œuvre de ces interfaces par rapport aux interfaces WIMP.

D'une part, un mécanisme de composition temporelle faisant intervenir plusieurs schémas d'ordonnancement temporels pour les différentes actions interactives entreprises aussi bien par l'utilisateur que par le système, d'autre part, des mécanismes de fusion et de fission (décomposition) sémantiques des informations échangées.

Le développement des interfaces multimodales a donné lieu à plusieurs travaux dédiés à la conception de ces interfaces. Ces modèles avec une prévalence pour la multimodalité en entrée s'appuient la plupart du temps, sur des modèles initialement proposés pour les interfaces conventionnelles et adaptés au contexte multimodal, et ce par la modélisation des mécanismes de composition temporelle et sémantique.

Lorsque les interfaces multimodales sont implantées dans un domaine critique (transports, nucléaire, médecine, ...), il est primordial de concevoir ces interfaces selon les exigences fixées pour leur fonctionnement et d'assurer un certain nombre de propriétés fondamentales (sûreté, vivacité, utilisabilité) qui garantissent la qualité des interfaces conçues. Dans ce contexte, une approche de développement formel s'avère nécessaire pour le développement de l'interface multimodale au même titre que le noyau fonctionnel. Cette approche doit permettre de modéliser sans ambigüité l'interface multimodale et de procéder à la validation des propriétés pertinentes pour l'interface.

Dans nos travaux nous nous intéressons plus particulièrement à la conception et la vérification formelles des interfaces Homme-Machine multimodales en sortie. Ces interfaces supportent l'interaction multimodale de la machine vers l'utilisateur.

Contexte.

L'objectif principal de nos travaux est de proposer un cadre formel générique pour la conception et la vérification des interfaces multimodales en sortie. Ce modèle doté d'une syntaxe, d'une sémantique statique et dynamique, permet de décrire sans ambiguïté et indépendamment de toute technique formelle l'interface multimodale en sortie. Pour cela, nous nous sommes basés sur un modèle semi-formel existant et avons inclus toutes les descriptions pertinentes à notre sens pour l'interface multimodale en sortie. Le modèle proposé se décompose en deux modèles : le modèle de fission sémantique qui décrit la décomposition de l'information à restituer en informations élémentaires, et le modèle d'allocation qui spécifie l'allocation des modalités et média pour chaque information élémentaire.

Afin d'assister le concepteur de l'interface multimodale dans le processus de développement sûr de l'interface conformément aux exigences fixées pour cette dernière, nous propo-

sons de plonger le modèle formel générique, moyennant des transformations, dans une méthode formelle cible. Nous présentons dans ce mémoire un plongement dans une méthode de spécification et de validation formelles basée sur une approche descendante de modélisation et une validation orientée raffinement et démonstration de théorèmes (theorem-proving) : la méthode "B événementiel". Ce plongement dans B événementiel opérationnalise le modèle générique, il modélise la construction de l'interface par raffinements successifs et permet de vérifier des propriétés de sûreté, vivacité et utilisabilité sur les interfaces multimodales en sortie par preuve de théorème.

Plan du mémoire.

Ce mémoire est structuré en six chapitres. Un premier chapitre dédié aux systèmes interactifs en général et aux IHM multimodales en particulier pour lesquelles nous proposons un modèle de conception et de vérification formelles. Ce chapitre introduit les définitions préalables ainsi que les principaux concepts manipulés dans ce mémoire. Il présente ensuite, de la manière la plus exhaustive possible, les travaux réalisés autour de la problématique de la description et de la conception des IHM et des IHM multimodales et les différentes propriétés qui caractérisent ces interfaces. Nous mettons en évidence au terme de ce chapitre que ces modèles manquent de rigueur pour le développement des interfaces dans un cadre critique.

Le deuxième chapitre présente, dans un premier temps, une classification des approches de développement et de validation formelles dans le cadre général du génie logiciel. Dans un deuxième temps, il présente l'état actuel des recherches relatives aux développements formels des systèmes interactifs. Ce chapitre est conclu par une synthèse qui expose les bénéfices et les limites des approches formelles proposées pour le développement des IHM, et l'absence de modèle formel de conception pour les IHM multimodales en sortie. Cette synthèse est suivie par l'introduction de notre proposition qui comprend un modèle formel générique de conception et d'une formalisation B événementiel pour la conception des IHM multimodales en sortie.

Le troisième chapitre présente le premier volet de notre contribution qui consiste en un modèle formel générique pour la conception des IHM multimodales en sortie, il décrit les deux modèles qui le composent (fission sémantique et allocation) par la description de leur syntaxe, leurs sémantiques statique et dynamique, et une étude de cas, reprise tout au long de ce mémoire.

Le quatrième chapitre présente le second volet de notre proposition, il décrit la formalisation B événementiel du modèle de conception présenté dans le troisième chapitre. Après une brève introduction à la méthode B événementiel, la démarche de modélisation B Évènementiel des IHM multimodales en sortie est présentée, suivie des quatre modèles génériques (un modèle abstrait et trois raffinements) de développement B Évènementiel. Ces modèles génériques représentent des modèles cadres ou patrons à partir desquels les modèles B Évènementiel détaillés de la fission sémantique, décrits dans le cinquième chapitre et de l'allocation dans le sixième chapitre, sont dérivés.

Le cinquième chapitre présente de manière détaillée les développements B Évènementiel de la fission sémantique. Il décrit le modèle abstrait de l'interface ainsi que les raffinements relatifs aux différentes variantes possibles de la fission sémantique.

Le sixième chapitre présente les développements B Évènementiel de l'allocation. Il décrit les raffinements relatifs aux différentes variantes possibles de la composition et de la décomposition des présentations et de leur affectation. Il présente également la vérification des propriétés pertinentes de ces interfaces par preuve de théorème dans la plate-forme de support de B Évènementiel.

Enfin, ce mémoire se termine par une conclusion et une présentation des perspectives dégagées à l'issue de nos travaux.

Chapitre 1

Les Interfaces Homme-Machine multimodales

1 Introduction

Depuis quelques années, les interfaces Homme-Machine s'imposent comme un facteur déterminant dans le succès des applications interactives. Cet intérêt accordé aux interfaces Homme-Machine est à l'origine de différents travaux qui se sont intéressés à la définition, la caractérisation, la conception et l'implémentation des interfaces Homme-Machine. Ainsi, les premières interfaces apparues, dès la fin des années 60, sont les interfaces en ligne de commandes basées sur une interaction textuelle entre l'utilisateur et la machine, elles seront révolues lors de l'avènement en début des années 80 des interfaces WIMP *Windows, Icons, Menus and Pointing device* pour exprimer qu'elle utilise des metaphores de type : fenêtres, icônes, menus et dispositif de pointage. Les interfaces WIMP sont basées sur le principe WYSIWYG *What You See Is What You Get* pour ce que vous voyez est ce que vous obtenez. Ces interfaces offrent une interaction plus intuitive en réduisant les efforts d'apprentissage et ont grandement contribué à l'utilisation des machines par le grand public. Enfin, les interfaces post-WIMP dont le principe est de développer à l'extrême l'aspect intuitif de l'interaction afin de se rapprocher au plus de la communication entre humains, ont introduit des dispositifs d'interaction non conventionnels (manette de jeu, gant à retour d'effort ...) et utilisent de manière séquentielle et simultanée plusieurs modalités d'interaction conventionnelles (texte, manipulation directe) et non conventionnelles (parole, geste), elles sont également appelées *interfaces Homme-Machine multimodales*. La première interface multimodale fût proposée par Richard Bolt [6], elle intègre deux modalités d'interaction, la parole et le geste avec le paradigme "met ça là". Depuis, plusieurs travaux ont succédé couvrant plusieurs aspects tels que la définition des concepts, la conception, le développement et l'évaluation de ces interfaces.

Ce chapitre présente les travaux les plus représentatifs entrepris pour caractériser et modéliser les interfaces Homme-Machine de manière générale et les interfaces Homme-Machine multimodales en particulier.

2 Interaction Homme-Machine

L'interaction, la communication ou le dialogue Homme-Machine représente l'ensemble des mécanismes d'échange d'information entre un *humain* et une *machine* pour accomplir une *tâche* ou atteindre un *but* particulier pour l'humain. Elle est caractérisée (voir Figure 1.1) par le triplet : opérateur humain (ou *utilisateur*), *machine* et *environnement* de l'interaction.

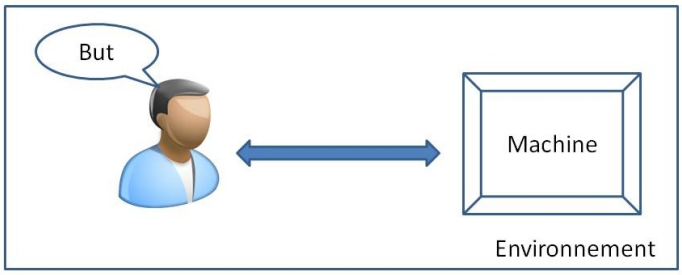

Figure 1.1 – L'interaction Homme-Machine

Selon Calvary[7], l'interaction Homme-Machine ou interaction Personne Système se réfère à l'ensemble des phénomènes cognitifs, matériels, logiciels et sociaux mis en jeu dans l'accomplissement de tâches sur support matériel (machine, système).

L'interaction Homme-Machine consiste en un échange d'informations dans deux sens donnant lieu à deux types d'interaction.

- **L'interaction en entrée :** consiste en un échange d'information de l'utilisateur vers la machine lorsque l'utilisateur entre une commande au système. Par exemple, une personne qui compose un numéro sur son téléphone pour appeler, une personne qui entre des mots-clés dans un moteur de recherche sur un ordinateur, une personne qui éteint un téléviseur, etc.
- **L'interaction en sortie :** consiste en un échange d'information du système vers l'utilisateur. L'interaction en sortie est de trois natures différentes.

    1. **L'interaction en sortie rétro-active :** elle survient suite à une interaction en entrée pour que l'utilisateur s'assure qu'il a bien entré sa commande. Par exemple, lorsqu'une personne compose un numéro sur son téléphone, l'interaction en sortie

consiste à afficher le numéro de téléphone composé et faire retentir la sonnerie indicative de la composition d'un numéro, etc.

2. **L'interaction en sortie de réponse :** elle survient suite à une interaction en entrée (commande/requête) de l'utilisateur pour répondre à cette requête. Par exemple, lorsqu'une personne effectue une recherche par mots-clés dans un moteur de recherche sur un ordinateur, l'interaction en sortie consiste à afficher la page de résultats de la recherche sur l'écran, etc.
3. **L'interaction en sortie sur initiative du système :** ce type d'interaction en sortie est déclenché par un stimulus produit par le système, le plus souvent ce type d'interaction en sortie restitue des informations de type alerte. Par exemple, lorsque le niveau de la batterie dans un téléphone atteint un seuil bas critique, une interaction en sortie est enclenchée par le système sous forme d'alerte.

3 Systèmes interactifs

Selon [8], un système interactif est un système dont le fonctionnement dépend d'informations fournies par un environnement externe qu'il ne contrôle pas.

Un système interactif (voir Figure 1.2) est un système informatique avec lequel communique l'utilisateur. Il est constitué d'un *noyau fonctionnel* qui regroupe les données et traitements du système informatique. Ces traitements sont encapsulés et intégrés dans une couche extérieure *Interface Homme-Machine* (IHM) qui permet à l'utilisateur d'agir sur le noyau fonctionnel sans en connaître tous les détails. Ainsi, chaque composant de l'interface Homme-Machine fait intervenir un ou plusieurs éléments du noyau fonctionnel.

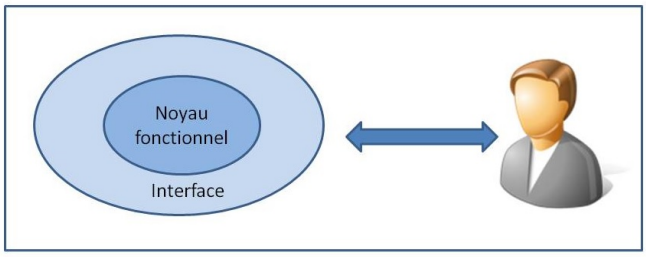

Figure 1.2 – Les systèmes interactifs

Par conséquent, une interface Homme-Machine est un ensemble de composants matériels et logiciels jouant le rôle d'intermédiaire entre l'utilisateur et le noyau fonctionnel d'une application.

4 Concepts de base

Le développement d'une interface Homme-Machine fait intervenir un certain nombre de composants et de notions. Il existe dans la littérature (voir [9], [10]) différentes définitions pour ces concepts suivant le point de vue (utilisateur / technologie) ou le niveau d'abstraction dans lequel on considère l'interface. Nous nous proposons de désambigüiser ces concepts en présentant ci-dessous les définitions considérées dans notre travail.

4.1 Information

Selon [11], l'information est l'objet de nature sémantique (en provenance du noyau fonctionnel et transitant par le contrôleur de dialogue) que le système doit présenter à l'utilisateur. Elle est produite par une fonction élémentaire du noyau fonctionnel suite à une interaction en entrée ou sur initiative du système, puis restituée à l'utilisateur.

4.2 Mode

Foley et al. [12] définissent le mode par un état ou une collection d'états dans lesquels seul un sous-ensemble de toutes les tâches interactives possibles est disponible.
Selon Nigay [13], un mode est un état ou une manière avec laquelle se fait l'interaction, et qui détermine son interprétation lorsqu'elle se produit. Dans l'interaction en entrée, le mode d'interaction est lié aux habiletés de l'utilisateur. Ainsi, les modes en entrée possibles sont : le mode *langage* dans lequel l'utilisateur utilise le langage pour interagir avec le système, par exemple : la saisie d'un texte ou l'utilisation de commande vocales et le mode *action* dans lequel l'utilisateur effectue des actions physiques de manipulation pour interagir avec le système interactif, par exemple : un clic de souris, un appui sur le bouton d'appel d'un téléphone. Dans l'interaction en sortie, le mode d'interaction est lié aux sens perceptuels de l'utilisateur qu'il sollicite lors de l'interaction (vue, ouïe, toucher ...). Ainsi, les principaux modes d'interaction en sortie utilisés sont : le mode *visuel* dans lequel le système utilise des objets devant être visualisés par l'utilisateur afin de percevoir l'information transmise tels qu'une image ou un texte, le mode *auditif* dans lequel le système utilise des objets sonores devant être écoutés par l'utilisateur afin de percevoir l'information tels qu'une sonnerie ou un discours produit par synthèse vocale, et le mode *tactile* dans lequel le système utilise des mécanismes haptiques produisant une sensation tactile ou kinesthésique pour percevoir l'information tels qu'une vibration, un retour d'effort.

4.3 Média

Un média est un dispositif physique dans un système informatique permettant le support de l'information transmise entre l'utilisateur et la machine. Il désigne le dispositif physique qui acquiert ou qui diffuse l'information [14].
Dans l'interaction en entrée, le média joue le rôle de capteur : *microphone*, *caméra*, *clavier*, *souris*, *manette*... Dans l'interaction en sortie, le média joue le rôle d'effecteur : *écran*, *haut-parleur*, *gant à retour d'effort*...

4.4 Modalité

Une modalité fait référence au type de canal de communication utilisé pour transmettre ou obtenir des informations [15]. Elle désigne la structure ou la forme que prend l'information lors de l'interaction et peut être définie à différents niveaux de granularité. Une modalité fait intervenir un seul mode d'interaction à la fois mais un mode d'interaction dispose de plusieurs modalités d'interaction. Une modalité peut être véhiculée par plusieurs médias, cependant elle est généralement produite par un seul média. En entrée, le mode langage peut utiliser les modalités : *commande textuelle* et *parole*, le mode action peut utiliser les modalités : *clic* et *double-clic*, *glisser*... En sortie, le mode visuel peut utiliser les modalités : *texte*, *image*, *diagramme*, *vidéo*..., le mode auditif peut utiliser les modalités : *bip*, *discours*, *sonnerie*..., le mode tactile utilise essentiellement la modalité *vibration*.

4.5 Énoncé

Un énoncé est une suite séquentielle d'événements élémentaires effectués par l'utilisateur, faisant intervenir des modalités véhiculées par des médias en entrée, intervenant dans la réalisation d'une même commande. Par exemple, saisir les mots-clé dans la barre de recherche puis appuyer sur le bouton rechercher constitue un énoncé pour la commande recherche par mots-clés. Un énoncé est dit "multimodal" s'il contient au moins un événement multimodal et au moins deux événements élémentaires provenant de deux médias distincts [16].

4.6 But

Un but est l'état du système que l'utilisateur souhaite obtenir lors de l'interaction. Par exemple, lors d'une recherche sur un moteur de recherche, le but de l'utilisateur est de voir la liste des pages Internet correspondant à une liste de mots-clés.

4.7 Tâche interactive

L'analyse et la modélisation des tâches humaines ont fait l'objet de nombreuses propositions dans la littérature en IHM ([17], [18]). Selon Normand, une tâche est un but que l'utilisateur vise à atteindre assorti d'une procédure (ou plan) qui décrit les moyens pour atteindre ce but. C'est un échange informationnel complexe ou élémentaire entre utilisateur et machine soit en entrée en vue de la réalisation d'un but, dans ce cas la tâche interactive correspond à un énoncé, soit en sortie en vue de présenter une information, dans ce cas la tâche interactive correspond à une présentation.

4.8 Présentation

Une présentation est un assemblage de composants de l'interface pour restituer une information générée par le noyau fonctionnel à l'utilisateur. Ces composants définissent la forme de la présentation tandis que leurs attributs fixent l'instance de la présentation [4]. Par exemple, l'affichage dans une fenêtre du nombre de résultats ainsi que le détail des pages trouvées constituent une présentation pour l'information relative au résultat de la recherche par mots-clés. Une présentation est dite multimodale adaptés à l'expression d'une information. Une présentation est dite multimodale si elle utilise deux modalités distinctes pour restituer une même information générée par le noyau fonctionnel.

5 Interfaces Homme-Machines multimodales

Une interface Homme-Machine multimodale est une interface qui permet de combiner en entrée et / ou en sortie plusieurs médias et modalités de manière dynamique et à un niveau sémantique [1]. Par exemple : la commande *mets ça là* est exécutée en montrant sur l'écran l'objet (*ça*) et l'emplacement (*là*) correspondant aux mots prononcés pendant ce temps devant un microphone. La machine répond alors *Dois-je déplacer aussi ce fichier ?* en anticipant sur l'intention de l'utilisateur.

Alors qu'un système multimédia utilise plusieurs modalités de manière préétablie pour véhiculer les informations échangées entre l'utilisateur et la machine, un système multimodal fait coopérer à un niveau sémantique, et de manière synchronisée, plusieurs modalités pour exprimer une même information échangée entre l'utilisateur et la machine. La mul-

1. La sémantique des informations échangées entre l'utilisateur et le système est partagée et véhiculée par plusieurs modalités.

timodalité introduit deux mécanismes essentiels : la fusion sémantique des informations issues des différentes modalités en entrée (multimodalité en entrée) et la fission sémantique de l'information générée par le noyau fonctionnel (multimodalité en sortie). Tout comme pour l'interaction dans un système multimédia, nous pouvons distinguer la multimodalité en entrée et la multimodalité en sortie.

multimodalité en entrée : type d'interaction qui consiste à fusionner les informations fournies par l'utilisateur via plusieurs modalités. Par exemple, l'interaction dans laquelle l'utilisateur exprime par la voix "*met ça là*" en cliquant sur un objet puis sur un emplacement de l'interface, constitue une interaction multimodale en entrée utilisant des informations introduites respectivement à travers les modalités *parole* via le média microphone, et *manipulation directe* via le média souris.

multimodalité en sortie : type d'interaction qui consiste à fissionner (décomposer) les informations générées par le noyau fonctionnel et à les restituer à l'utilisateur en exploitant les différentes modalités en sortie et en fonction du contexte d'interaction. Par exemple, l'interaction dans laquelle, l'utilisateur demande la liste des trains de la ville A vers la ville B, le système répond par synthèse vocale "*votre requête peut être satisfaite par les quatre trains suivants*" et affiche en même temps une liste détaillée des trains. Il s'agit d'une interaction multimodale en sortie répondant à la requête utilisateur par les informations "*votre requête peut être satisfaite par les quatre trains suivants*" et liste des trains, à travers les modalités *parole* via le média haut-parleur, et *texte* via le média écran.

6 Développement des interfaces Homme-Machine

Une interface Homme-Machine se doit d'offrir l'accès aux fonctionnalités offertes par le noyau fonctionnel et de couvrir les besoins utilisateur en termes d'interactions. Par conséquent, le développement des interfaces Homme/Machine constitue une activité pluridisciplinaire faisant intervenir informaticiens, ergonomes et psychologues, elle a donné lieu à plusieurs techniques et notations dédiées à la description et à la conception des interfaces Homme-Machine. Dans [19], une classification de ces notations en deux catégories est proposée.

6.1 Techniques et notations orientées description

Ces techniques le plus souvent utilisées par les psychologues et ergonomes, permettent de décrire les besoins des utilisateurs en terme d'utilisabilité [2]. Généralement, ces notations décrivent l'interface par un ensemble de tâches qui permettent d'atteindre un but d'interaction, elles sont par conséquent, dites centrées utilisateur. Les notations centrées utilisateur sont utilisées soit au début du processus de conception de l'IHM pour spécifier les besoins de l'utilisateur soit à la fin du processus de conception de l'IHM pour valider la conception de l'interface par rapport aux exigences utilisateur. La complexité des tâches lors de leur description est maitrisée par l'introduction de mécanismes de décomposition hiérarchiques associés à des opérateurs d'ordonnancement temporel. Plusieurs notations ont été proposées : GOMS (Goal, Operator, Method and Selection rules) [20], MAD [21] étendue par MAD* [GS97], UAN [22], GTA (Groupware Task Analysis) [23], CTT [24],

SADT/Petri [Abed et al., 2001] avec : Mourad Abed, Houcine Ezzedine, Christophe Kolski (2001). Modélisation des tâches dans la conception et l'évaluation des systèmes interactifs : la méthode SADT/Petri. In Kolski C. (Ed.), Analyse et Conception de l'IHM. Interaction Homme-machine pour les SI, Vol. 1, Hermès, Paris, pp. 145-174,

TOOD (Task Object-Oriented Description) [25], AMBOSS [26] et UsiXML (USer Interface eXtensible Markup Language) [27]. Dans [28] une étude comparative des principales notations est proposée. Nous décrivons brièvement ci-dessous quelques modèles de tâches représentatifs de cette classe de notations.

6.1.1 MAD

MAD (Méthode Analytique de Description) [21] est une notation qui décrit les tâches par décomposition structurelle suivant une dimension "logico-temporelle". Elle utilise les concepts de *tâche*, *d'action* et de *structure*.

La **tâche** représente un traitement plus ou moins complexe, sa syntaxe est définie par le n-uplet suivant :

<tâche> := <état-initial>, <état-final>, <but>, <préconditions>, <postconditions>, <corps>

Où :

- **état initial** définit les arguments d'entrée de la tâche ;
- **état final** définit les arguments de sortie de la tâche ;

2. Degré selon lequel un produit peut-être utilisé, par des utilisateurs identifiés, pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié (ISO 9241-11, 1998)

- **but** consiste en un sous-ensemble de l'état final qui définit explicitement le but recherché de l'exécution de la tâche ;
- **préconditions** sont un ensemble de propriétés logiques qui expriment les contraintes devant être satisfaites sur l'état initial afin de déclencher l'exécution de la tâche ;
- **postconditions** sont un ensemble de propriétés logiques qui expriment les contraintes devant être satisfaites sur l'état final après l'exécution de la tâche ;
- **corps** décrit le déroulement de la tâche. Une tâche peut être *simple* lorsqu'elle est décrite par une *action* (procédure insécable) ou *composée* lorsqu'elle fait appel à un enchainement structuré de sous-tâches décrit dans la partie **corps**. Ainsi, le corps d'une tâche décrit les sous-tâches composant la tâche ainsi que les relations temporelles ou logiques entre ces sous-tâches au moyen d'un constructeur (séquence, alternative, parallèle, boucle).

La notation MAD propose une représentation graphique de type arbre de décomposition logico-temporelle qui exprime les décompositions successives de la tâche en sous-tâches où les feuilles représentent les tâches simples.

6.1.2 UAN

UAN (User Action Notation) [22] est une notation orientée tâche qui permet d'exprimer les spécifications des interfaces à manipulation directe. Elle décrit le comportement de l'utilisateur et de l'interface lors de leur coopération pour l'exécution d'une tâche.
Le concept de tâche est le principal mécanisme d'abstraction de la notation. Une interface utilisateur est représentée comme une structure hiérarchique de tâches asynchrones, chaque tâche est décrite de manière textuelle dans un tableau au travers des actions qui la composent. Ces actions consistent en : actions de l'utilisateur (par exemple, un clic de souris), feedbacks de l'interface (par exemple, surbrillance d'un composant de l'interface) et état de l'interface (par exemple, la variable *selected* a pour valeur "file"). À tous les niveaux d'abstraction, les actions utilisateur ainsi que les tâches sont combinées avec des relations temporelles telles que : la séquence, l'entrelacement, ou la concurrence. La notation UAN a été étendue par la notation XUAN [29] pour modéliser de manière plus détaillée les relations temporelles entre tâches en subdivisant dans les diagrammes tabulaires de description des tâches les différentes sections : variables, pré-conditions et post-conditions, sous-tâches et relations temporelles.

6.1.3 CTT

CTT(ConcurTaskTrees)[24] permet de modéliser les tâches utilisateur de manière hiérarchique en utilisant des arbres qui décrivent la décomposition de la tâche en sous-tâches. Les tâches considérées par la notation sont de quatre types : abstraite, utilisateur, interaction et application. Les tâches de même niveau sont combinées à l'aide d'opérateurs temporels empruntés à l'algèbre de processus Lotos (séquence, parallèle, disjonction, répétition, interruption, concurrence, ...). La notation dispose d'un environnement graphique associé CTTE (ConcurTaskTree Environment) [30] permettant l'édition, la simulation et la génération de scenarii de tâches. Un exemple de tâche de connexion est décrit dans la Figure 1.3. La tâche *Login* est décomposée en trois tâches : deux tâches concurrentes (|||) *Fill username* et *Fill password* suivie (») de la tâche *Connect*.

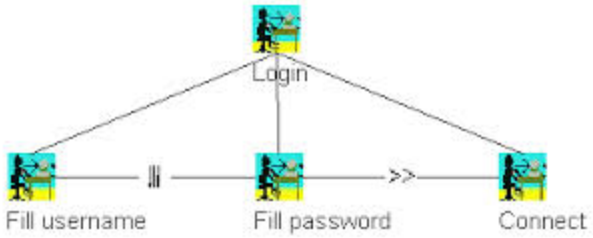

Figure 1.3 – Exemple d'une tâche de connexion en CTT

6.2 Techniques et notations orientées conception

Ces techniques et notations parfois nommées modèles d'architecture sont généralement utilisées par les informaticiens afin de décrire la structure statique du logiciel implémentant l'interface utilisateur décrite et/ou du système. Ces modèles séparent l'interface utilisateur du noyau fonctionnel ce qui permet de modifier l'interface sans affecter le noyau fonctionnel et inversement. Les modèles d'architecture permettent non seulement de structurer le système interactif mais également de disposer d'une organisation modulaire. Cette modularité facilite la réutilisation logicielle des composants, sa maintenance, son adaptation au contexte et son évolution. Un modèle d'architecture doit :

- préconiser une séparation fonctionnelle entre les services de l'application et ceux de l'interface ;
- définir une répartition des services de l'interface, qui se traduit par un ensemble de composants logiciels ;
- définir un protocole d'échange entre les constituants logiciels.

Plusieurs modèles d'architectures ont été proposés. Ils sont souvent classés en trois catégories.

6.2.1 Modèles globaux

Souvent nommés Macro-Modèle, modèles centralisés ou modèle généraux, ils décrivent l'organisation des différents modules constituant le système interactif sans détailler la structure interne de chaque module. Nous citons dans cette classe : les modèles Seeheim et ARCH. Le modèle Seeheim a été proposé au cours du séminaire sur les systèmes de gestion utilisateur en 1983 à Seeheim (R.F.A.) [31], il décompose un système interactif en trois composantes logiques : la présentation, le contrôle du dialogue et l'interface avec l'application. Le modèle ARCH (voir Figure 1.4), défini au cours de quatre séminaires regroupant des développeurs de gestionnaires d'interface utilisateur [32], il étend le modèle Seeheim par l'introduction de deux composantes supplémentaires : l'adaptateur de domaine et l'interaction.

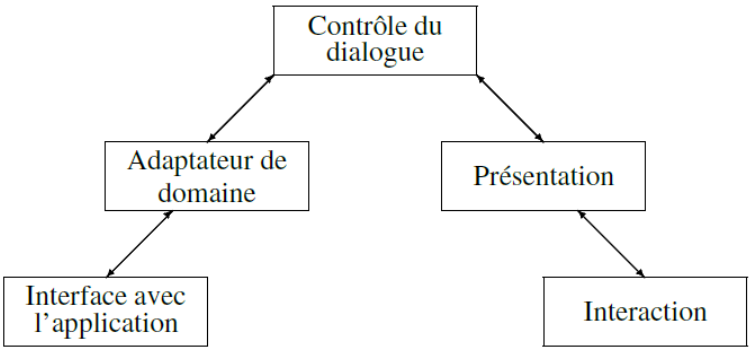

Figure 1.4 – Le modèle ARCH

6.2.2 Modèles multi-agent

A l'inverse des modèles généraux, les modèles multi-agents ou génériques définissent de manière précise les composants du système interactif ainsi que la communication entre eux, sans préciser leur nombre ou leur organisation. Le système interactif est décrit au moyen d'une hiérarchie d'agents, où chaque agent est constitué de plusieurs modules. Cette organisation permet de modifier facilement l'architecture de l'application, elle offre également les avantages de la conception itérative, et la compatibilité avec les langages à objets. Nous citons les modèles MVC [33], PAC [34], ALV [35] et H4 [36]. Une synthèse des principaux modèles multi-agent est présentée dans [37]. Nous décrivons dans ce qui suit les modèles les plus rencontrés dans la littérature : MVC et PAC.

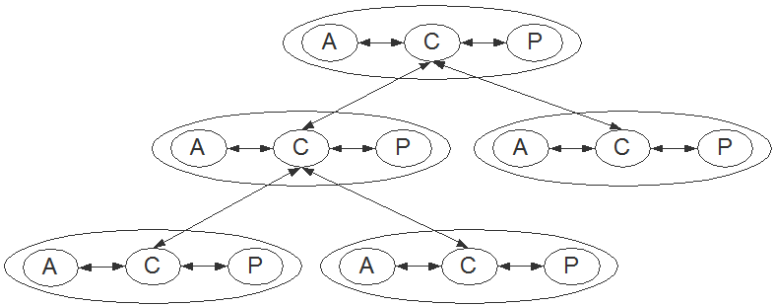

Figure 1.5 – Le modèle PAC [1]

Dans le modèle MVC (Modèle, Vue, Contrôleur), un agent MVC est constitué de trois modules : le modèle qui représente le noyau fonctionnel de l'agent, une ou plusieurs vues qui maintiennent une représentation du modèle perceptible par l'utilisateur, et un ou plusieurs contrôleurs qui interprètent les événements utilisateur, en les répercutant sur le modèle.
Le modèle PAC (présentation, Abstraction, Contrôle) (voir Figure 1.5) permet de décomposer un système interactif en plusieurs agents constitués de trois facettes : la présentation (*P*) assure la communication de l'agent avec l'utilisateur, l'Abstraction (*A*) regroupe les fonctionnalités de l'agent et le Contrôle (*C*) maintient la cohérence entre la présentation et l'abstraction.

6.2.3 Modèles à base d'interacteurs

Les modèles à base d'interacteurs sont des modèles multi-agents dans lesquels les comportements des agents sont décrits de manière plus précise en utilisant le principe de stimuli/réaction.
Dans les modèles à base d'interacteurs, le système interactif est décomposé en une collection d'unités autonomes appelées interacteurs (objets d'interaction), qui communiquent entre eux, avec le système ou avec l'utilisateur par le biais de stimuli (ou événements). Un interacteur est déclenché par une condition initiale, puis il fonctionne en gérant des dispositifs d'entrée et en produisant un effet de retour ; finalement il appelle une fonction qui déclenche une action à partir des paramètres de la manipulation.
La plupart des modèles à base d'interacteurs ont été formalisés suivant différentes techniques formelles afin de permettre la vérification de certaines propriétés (voir chapitre2). Les principaux modèles à base d'interacteurs sont : Cnuce [38], York [39], Pise [40] et ADC [41]. Nous décrivons dans ce qui suit les modèles les plus rencontrés dans la littérature : York et Cnuce.

Dans York, un interacteur est un composant dans la description d'un système interactif

qui encapsule un état, des événements d'entrée et de sortie qui manipulent cet état, et une présentation qui reflète cet état de façon perceptible par l'utilisateur.

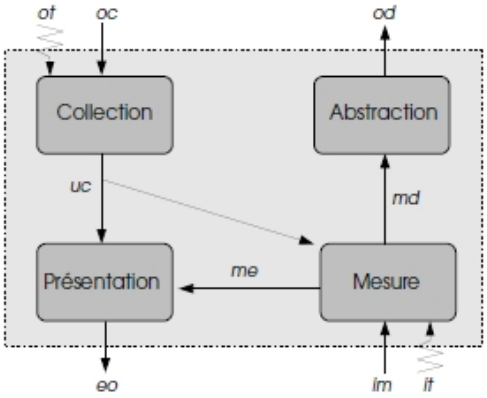

Figure 1.6 – Le modèle Cnuce [1]

Dans le modèle de Cnuce, un système interactif est décrit comme un graphe d'interacteurs communiquant entre eux. Au plus bas niveau, ils communiquent avec l'utilisateur et au plus haut niveau, ils communiquent avec l'application. Un interacteur Cnuce (voir Figure1.6) est une entité capable de réagir à des stimuli externes, il est décrit à un niveau abstrait en décrivant son comportement en réaction aux stimuli externes et au niveau concret en décrivant son fonctionnement interne. Il se compose de quatre composants qui communiquent entre eux : le composant Collection maintient une représentation abstraite de l'apparence externe de l'interacteur, le composant Présentation met à jour les éléments visibles par l'utilisateur, le composant Mesure reçoit et accumule les informations provenant de l'utilisateur et le composant Abstraction qui les convertit en données abstraites manipulables par l'application.

6.2.4 Modèles hybrides

Les modèles hybrides combinent un modèle global pour l'organisation du système interactif en modules et un modèle multi-agents pour la description interne du module. Parmi les modèles hybrides nous citons : PAC-Amodeus [42] et H^4 [36].

Le modèle PAC-Amodeus (voir Figure 1.7) utilise le modèle ARCH pour définir la structure modulaire du système interactif et détaille le composant contrôleur de dialogue en un ensemble d'objets coopératifs de type PAC.

Le modèle H^4 est basé également sur le modèle ARCH pour la description globale en composants, il décrit de manière précise tous les modules, et préconise la structuration de quatre d'entre eux en hiérarchies d'agents.

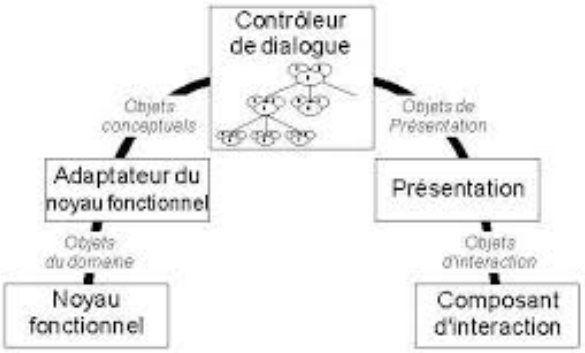

Figure 1.7 – Le modèle PAC-Amodeus

7 Les modèles de description du dialogue

Le dialogue sous-tend un fonctionnement de type conversation c'est-à-dire une intervention alternée entre l'homme et la machine [3]. Les modèles de description du dialogue ou modèles de dialogue permettent de représenter la dynamique de l'interaction homme-machine. Ce dialogue est lié à la sémantique du système interactif (ce qu'il doit faire) et à la présentation (visualisation du système). De nombreux formalismes ont été dédiés à la description du dialogue [43] : langages à évènements, langages orientés objet, théorie des graphes, systèmes de transition, réseaux de Petri, algèbres de processus et langages de flots de données. Nous présentons dans ce qui suit les formalismes les plus représentatifs pour la description du dialogue à savoir les formalismes à base d'états et les formalismes à base d'événements.

7.1 Modèles à base d'états

Les modèles à base d'état s'intéressent à représenter les différents états du système interactif lors du dialogue, nous citerons dans cette famille les formalismes les plus représentatifs : les automates à états et les réseaux de Petri.
Les automates à états, appelés également Systèmes de Transitions Étiquetés (STE), sont les premiers à avoir représenté le dialogue d'une application interactive. Un STE est un cadre mathématique formalisé permettant le raisonnement et qui dispose d'une représentation graphique. Dans ce cadre, un système interactif est représenté par un ensemble d'états, un ensemble d'actions, un état initial et une relation entre les états, appelée relation de transition. Le comportement du dialogue est établi par le déclenchement d'événements (étiquettes des

transitions) qui permettent le passage d'un état à un autre au moyen des transitions.
[44] fut le premier à spécifier le comportement d'une interface à l'aide de STE. Dans cette description, les états constituent les différents états possibles de l'interaction et les arcs sont étiquetés par les actions de l'utilisateur. L'exemple donné en Figure 1.8, repris de [2] décrit le système d'état-transitions de l'opération Copier/Coller sur une zone de texte.

La modélisation de systèmes complexes est obtenue par composition des systèmes de transitions représentant chaque sous-système en utilisant des opérations de produits cartésiens et de produits synchronisés. Cette composition conduit à une explosion combinatoire du nombre d'états du modèle ce qui rend l'examen de tous les états du modèle, lors de la vérification de certaines propriétés, très ardu.

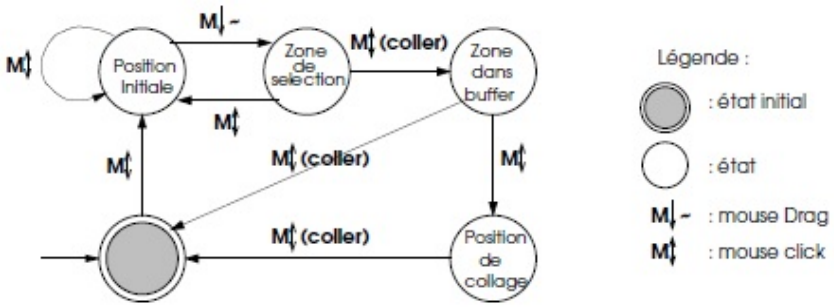

Figure 1.8 – Exemple de système de transition : Copier/Coller une zone de texte

Plusieurs extensions des automates afin de maitriser le problème de l'explosion du nombre des états [45], nous citerons les travaux de [46] sur les réseaux de transitions récursifs (RTN : Recursive Transition Networks) qui permettent une structuration hiérarchique d'un automate à états finis. [47] définit les réseaux de transitions augmentés (ATN : Augmented Transition Networks) qui introduisent la notion de registres (variables d'état de l'automate). Le formalisme des Statecharts [48] est également une extension des automates qui permet l'expression du parallélisme et de la synchronisation. Enfin, les STE sont associés à différentes techniques de preuve permettant la validation de nombreuses propriétés relatives au dialogue homme-machine (sûreté, vivacité, atteignabilité). C'est notamment le cas de la formalisation Lustre des interacteurs présentée en chapitre2.

Le second formalisme a états utilisé pour la description du dialogue des systèmes interactifs est les Réseaux de Petri (RdP), il s'agit d'un formalisme dérivé des automates, il possède une sémantique formelle particulièrement bien adaptée à la modélisation des sys-

tèmes concurrents.
Un réseau de Petri est composé de deux types d'objet : les places et les transitions. Les places correspondent aux états possibles du système et les transitions désignent les opérateurs de changement d'état. L'état du réseau est représenté par un ensemble de jetons répartis dans les places du réseau définissant le marquage du réseau. Places et transitions sont reliées par des arcs orientés. La figure 1.9 présente le réseau de Petri de l'exemple du Copier/Coller utilisé précédemment.

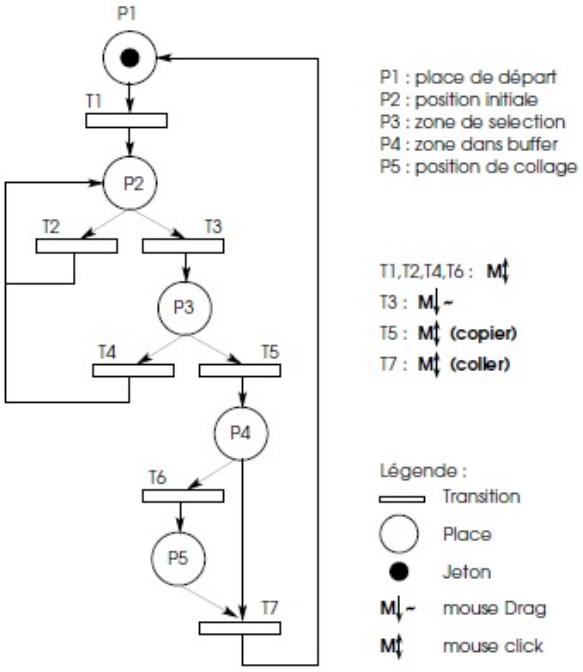

Figure 1.9 – Exemple de Réseau de Petri : Copier/Coller une zone de texte [2]

Les réseaux de Petri souffrent tout comme les automates du problème de l'explosion du nombre d'états. Des extensions ont été définies afin d'y remédier, parmi lesquels, les Réseaux de Petri à Objets (RPO) [49] utilisent le concept de structuration lié aux langages à objets, les Objets Coopératifs [50] introduisent la notion de modularité en définissant un système de communication entre les objets, les Objets Coopératifs Interactifs (ICO) [51], spécialement dédiés à la modélisation des IHM. Ils introduisent notamment la notion de présentation au sein des objets manipulés. Le formalisme réseaux de Petri offre la possibilité de vérifier des propriétés liées au graphe de marquage telles que l'absence de blocage (voir chapitre 2). D'autres travaux utilisant la méthode B ([52] et [53]) ont permis de modéliser le dialogue, ils sont présentés dans le chapitre 2.

7.2 Modèles à base d'évènements

Contrairement aux modèles précédents, les modèles à base d'événements [54] ne se concentrent pas sur la notion d'état, mais sur les événements permettant d'accéder à ces états. Ces modèles reposent sur les concepts d'événements, de gestionnaire d'événements et de traitements [51]. Lorsqu'un événement est émis, il est pris en charge par le gestionnaire d'événements qui le dirige vers une procédure de traitement qui exécute une modification d'état du système suivant le type et les paramètres de l'événement.
Les modèles à base d'événements possèdent un pouvoir d'expression important particulièrement adapté aux caractéristiques des systèmes interactifs telles que le parallélisme et la synchronisation. Il sont largement utilisés dans la plupart des langages de programmation (notamment le langage Java/Swing) et dans plusieurs langages de modélisation tels que Lustre [55] ou B Événementiel [56]. Une description plus détaillée des travaux utilisant les formalismes Lustre et B Événementiel pour la modélisation et la validation des interfaces est présentée en chapitre 2.

8 Propriétés des Interfaces Homme-Machine

Comme tout système, une Interface Homme-Machine est caractérisée par les propriétés : sûreté, équité, vivacité et atteignabilité auxquelles s'ajoutent des propriétés liées à l'utilisabilité du système. Ces dernières, issues des exigences d'ergonomes et de psychologues, traitent de la capacité de l'utilisateur à réaliser des tâches de manière efficace, confortable et sûre.

De nombreuses classifications des propriétés caractérisant les IHM, ont été proposées dans la littérature ([57], [58] et [59]). Nous décrivons ci-dessous la classification proposée dans [60] qui distingue deux classes de propriétés.

1. Les propriétés de *validité* caractérisent le fonctionnement attendu et souhaité par l'utilisateur.
2. Les propriétés de *robustesse* concernent la sûreté de fonctionnement du système.

8.1 Propriétés de validité

Les propriétés de validité regroupent les propriétés de *complétude* pour la réalisation d'un objectif donné et les propriétés de *flexibilité* pour : la représentation de l'information, le déroulement des tâches et pour l'adaptation du dialogue vis-à-vis de l'utilisateur.

8.1.1 Propriétés de complétude

[59] distingue la complétude des traitements (Task Completeness) qui est une propriété du système décrivant si le système est à même d'autoriser l'exécution correcte des traitements qui le composent, de la complétude des objectifs (Goal Completeness) qui est une propriété de l'utilisateur et du système exprimant si l'utilisateur peut atteindre un objectif donné au moyen du système.

8.1.2 Propriétés de flexibilité

La flexibilité d'une IHM caractérise la manière avec laquelle l'utilisateur et le système échangent des informations lors de l'exécution d'une tâche. [59] introduit trois propriétés composant la flexibilité : la *représentation de l'information*, l'*adaptation du dialogue* et le *déroulement des tâches*.

- *Représentation de l'information* : cette composante caractérise la *multiplicité des périphériques*, la *multiplicité de la représentation* et à la *réutilisabilité des données d'entrée et de sortie*.
    1. La multiplicité des périphériques est la capacité du système à offrir plusieurs périphériques d'entrée (souris, clavier, ...) et de sortie (écran, haut-parleur ...).
    2. La multiplicité de la représentation est la capacité du système à offrir plusieurs représentations d'un même concept. Dans le cas d'une horloge, il existe différentes formes différentes de représentation : numérique ou analogique.
    3. La réutilisabilité des données d'entrée et de sortie est la capacité du système à autoriser l'usage des entrées et des sorties précédentes comme entrées futures. Dans le cas des sorties du système, la technique du couper-coller est un concept utilisable comme données d'entrée tandis que les valeurs par défaut comme entrées de l'utilisateur sont réutilisables par le système en sortie.
- *Adaptation du dialogue* : cette catégorie englobe principalement les propriétés liées à l'adaptativité et l'adaptabilité.
    1. L'adaptativité est la capacité du système à s'adapter à l'utilisateur sans intervention explicite de sa part.
    2. L'adaptabilité (ou la reconfigurabilité) est la capacité du système à supporter la personnalisation de l'interface par l'utilisateur.
- *Déroulement des tâches* : cette propriété regroupe les propriétés liées à l'*atteignabilité*, la *non-préemption* et l'*interaction de plusieurs fils* (multithreading).

1. L'atteignabilité est la capacité du système à atteindre un état désiré du système à partir de l'état actuel.
2. La non-préemption est la capacité du système à fournir directement le prochain but à l'utilisateur.
3. L'interaction à plusieurs fils est la capacité du système à gérer plusieurs processus concurrents, ce qui permet à l'utilisateur de lancer plusieurs traitements en même temps.

8.2 Propriétés de robustesse

Les propriétés de robustesse englobent les propriétés liées à la *visualisation du système* comme l'observabilité, l'insistance et l'honnêteté, et les propriétés liées à la *gestion des erreurs* comme la prédictibilité et la tolérance aux écarts.

8.2.1 Visualisation du système

Les propriétés liées à la visualisation du système permettent d'assurer une représentation correcte de l'état du système interactif via l'interface. La visualisation se décline en trois propriétés principales.

- L'observabilité est la capacité pour l'utilisateur à évaluer l'état interne du système. Le système fait en sorte que toutes les informations disponibles soient visualisées à l'écran.
- L'insistance est la capacité du système à forcer la perception de l'état du système. Le système fait en sorte que les informations nécessaires à l'utilisateur soient affichées à l'écran.
- L'honnêteté est la capacité à rendre conforme l'état interne du système aux yeux de l'utilisateur.

8.2.2 Gestion des erreurs

Ce sont des propriétés qui caractérisent la capacité du système à recouvrer ou à prévenir les erreurs des utilisateurs.

- La prédictibilité est la capacité pour l'utilisateur de prévoir les états accessibles du système à partir d'un état courant observable.
- La tolérance aux écarts est la capacité du système à aider l'utilisateur lorsqu'une ou plusieurs erreurs se produisent.

En plus des propriétés qui caractérisent les Interfaces Homme-Machine de manière générale, le développement des IHM multimodales a donné naissance à de nouvelles propriétés d'utilisabilité appelées les propriétés CARE et qui sont spécifiques à cette classe d'interfaces.

8.3 Les propriétés CARE

L'utilisabilité des IHM multimodales peut être caractérisée par quatre propriétés : Complémentarité, Assignation, Redondance, Eéquivalence (CARE) [61]. Ces propriétés permettent d'apprécier les propriétés de flexibilité et de robustesse de l'IHM multimodale en décrivent les relations existantes entre les différentes modalités pour l'accomplissement d'une tâche.

- **Complémentarité** : désigne l'usage de plusieurs modalités pour la réalisation d'un but.
- **Assignation** : désigne l'usage exclusif d'une seule modalité pour la réalisation d'un but, et aucune autre modalité ne permet de le réaliser.
- **Redondance** : désigne l'usage de plusieurs modalités pour le même but, et d'une manière parallèle.
- **Équivalence** : désigne le choix offert à l'utilisateur pour réaliser son but avec une modalité de son choix parmi un ensemble de modalités permettant de le réaliser.

Les propriétés CARE peuvent être définies à différents niveaux de raffinement et peuvent être appliquées aussi bien du coté utilisateur pour les interactions en entrée, que du coté système pour les interactions en sortie. Nous donnons ci-après l'interprétation des propriétés CARE pour l'interaction multimodale en sortie.

- **La complémentarité :** la présentation fait intervenir plusieurs modalités et chaque modalité utilisée est nécessaire à la présentation. Exemple : la réponse à une requête de recherche de la carte d'une ville donnée est produite en combinant la phrase "voici la carte de la ville recherchée" par synthèse vocale sur haut-parleur, et la carte de la ville recherchée sur l'écran. Les modalités *discours* et *carte* sont donc complémentaires.
- **L'assignation (spécialisation) :** la même modalité (ou sous-ensemble de modalités) est choisie pour une présentation donnée. Exemple : la copie d'un répertoire vers un nouvel emplacement est présentée toujours en utilisant une image animée.
- **La redondance :** la présentation fait intervenir plusieurs modalités et les modalités expriment la même sémantique. Exemple : la présentation de l'information : "il n'existe aucun train qui satisfait votre requête" se fait par synthèse vocale et par texte affiché. Ainsi, ces deux modalités sont redondantes pour cette présentation.
- **L'équivalence :** la présentation peut être effectuée en utilisant différentes modalités permettant d'obtenir le même résultat avec un coût cognitif éventuellement différent.

Exemple : la présentation de l'information : "il n'existe aucun train qui satisfait votre requête" se fait au choix par synthèse vocale ou bien par texte affiché. Par conséquent, ces deux modalités sont équivalentes pour cette présentation.

Ainsi, l'équivalence et la redondance qui offrent à l'utilisateur des variantes d'interaction, favorisent la flexibilité et la robustesse de l'interface. Au contraire, l'assignation qui impose à l'utilisateur, l'emploi d'une modalité particulière pour une tâche interactive, contribue à la rigidité de l'interface et va à l'encontre des principes de flexibilité et robustesse. De même, la complémentarité fait intervenir plusieurs modalités qui sont assignées à des aspects différents et complémentaires de l'exécution d'une tâche interactive ; par conséquent, elle est tout comme l'assignation cause d'une certaine rigidité de l'interface.

Une définition formelle de ces propriétés reposant sur les concepts d'état, de but, de modalité et de relations temporelles a été proposée par [62]. Une autre définition a également été proposée dans [63], elle permet de préciser la définition précédente en prenant en compte les dispositifs physiques et les langages d'interactions associés.
Les propriétés CARE ont servi de base pour la définition de la plateforme de conception multimodale ICARE [64]. Des approches de vérification pour ces propriétés ont été proposées, elles utilisent le model checking basé sur SMV dans [65] et la preuve de théorème dans B Evènementiel dans [19].

9 Conception des IHM multimodales

La multimodalité peut être envisagée de différentes manières. Plusieurs travaux ont été consacrés à la caractérisation des IHM multimodales dans le but de les évaluer et de les comparer, ou de fournir des patterns de conception pour ces interfaces. Nous citons parmi ces travaux : l'espace de conception CASE [3] qui définit des schémas d'interaction multimodale du point de vue du système. Il définit l'utilisation des modalités pour véhiculer les informations échangées entre l'utilisateur et la machine ainsi que leur ordonnancement.

9.1 Les types de multimodalité : L'espace CASE

L'espace de conception CASE [3] caractérise les interfaces multimodales suivant les trois critères suivants : la temporalité, l'enchaînement des énoncés/présentations et leur contribution à la tâche interactive. Il propose de classer la multimodalité selon deux axes (voir Figure 1.10) :

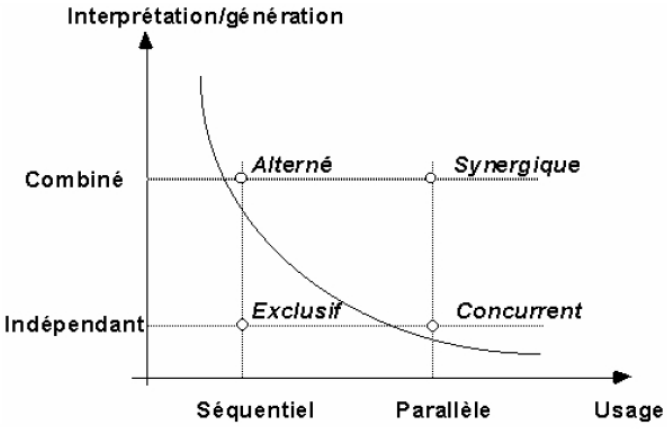

Figure 1.10 – L'espace de conception CASE [3]

- **L'usage des médias (séquentiel / parallèle) :** spécifie si les médias sont utilisés de manière séquentielle (l'un après l'autre) ou parallèle (en même temps) pour véhiculer l'information.
- **Les liens entre les informations (combiné ou indépendant) :** spécifient s'il existe un lien sémantique entre les informations échangées lors de la tâche interactive (introduites en entrée ou présentées en sortie) par les différentes modalités.

Le croisement des deux valeurs de chacun des axes ci-dessus donne naissance aux quatre types d'interaction multimodale suivants :

- **Concurrent (parallèle et indépendant) :** l'utilisation des médias est parallèle et les informations exprimées par les différentes modalités sont sémantiquement indépendantes. Exemple : l'apparition sur l'écran d'un message d'alerte "batterie faible" pendant que s'affichent dans une fenêtre les résultats d'une recherche par mots-clés.
- **Alterné (séquentiel et combiné) :** l'utilisation des médias est séquentielle et les informations exprimées par les différentes modalités sont sémantiquement dépendantes. Exemple : le résultat d'une requête de recherche des horaires de train, restitué à l'utilisateur par la phrase "les horaires de trains sont les suivants" par synthèse vocale sur haut-parleur suivie du tableau des horaires affiché sur écran.
- **Synergique (parallèle et combiné) :** l'utilisation des médias est parallèle et les informations exprimées par les différentes modalités sont sémantiquement dépendantes. Exemple : le résultat de la requête de recherche d'une carte d'une ville , restitué à l'utilisateur par la phrase "voici la carte de la ville recherchée" par synthèse vocale sur haut-parleur, tout en affichant en même temps la carte de la ville recherchée sur l'écran.
- **Exclusif (séquentiel et indépendant) :** l'utilisation des médias est séquentielle et les

informations exprimées par les différentes modalités sont sémantiquement indépendantes. Exemple : l'indication de la température sur écran suivie de l'annonce du taux d'humidité sur haut-parleur.

L'espace CASE a été raffiné, dans [66] et [9], dans le but de distinguer le parallélisme intervenant dans l'utilisation des médias du parallélisme relatif à la production des énoncés. Il propose de croiser trois axes : production des énoncés (séquentielle / parallèle), usage des médias (exclusif / parallèle) et nombre de médias par énoncé (un / plusieurs). Ceci aboutit à sept types de multimodalité (la configuration production séquentielle des énoncés et usage simultané des médias étant impossible) : exlusif, alterné, synergique, parallèle exlusif, parallèle simultané, parallèle alterné, parallèle synergique.

9.2 Les modèles de conception des interfaces Homme-Machine multimodales

Les modèles de conception dédiés à la conception des IHM multimodales sont peu nombreux, ils sont souvent consacrés soit à la multimodalité en entrée soit à la multimodalité en sortie avec une prédominance de la multimodalité en entrée et restent souvent dédiés à des champs d'application limités. Ainsi, certains modélisent un problème technique spécifique tel que la fusion des modalités ([67] propose un cadre orienté-objet pour modéliser la fusion dans la multimodalité en entrée et [68] étend le modèle PAC-Amodeus par un mécanisme de fusion pour modéliser la multimodalité en entrée), la désambiguïsation mutuelle [3] dans [69] et [67], l'adaptation des interfaces multimodales dans [70], soit ils sont dédiés à la modélisation d'une classe d'interfaces multimodales (les interfaces multimodales Web [71], les applications distribuées [72] ou les services Web [73]) ou à des modalités spécifiques telles que la reconnaissance des gestes [74], la reconnaissance de la parole [75] ou l'utilisation combinée de la parole et du geste [76].

D'autres consistent en des plate-formes de développement (voir étude comparative dans [77]) modélisant la composition multimodale des dispositifs d'interaction en entrée tel que ICON [78], ICARE [64] ou Open Interface [79] et d'entrée et sortie comme I*Chameleon [77],

De manière plus ciblée, la multimodalité en entrée a donné lieu à un modèle formel [16] tandis que la multimodalité en sortie a fait l'objet de deux modèles semi-formels : le modèle SRM et le modèle WWHT.

3. Utilisation conjointe de plusieurs modalités afin de lever l'ambiguïté sémantique induite par l'utilisation de chacune des modalités de manière séparée.

9.2.1 Le modèle SRM (Standard Reference Model)

Le modèle SRM [80] (voir Figure 1.11) est orienté but où le but représente l'information à présenter à l'utilisateur. Il construit la présentation multimodale au travers d'une architecture à cinq couches :

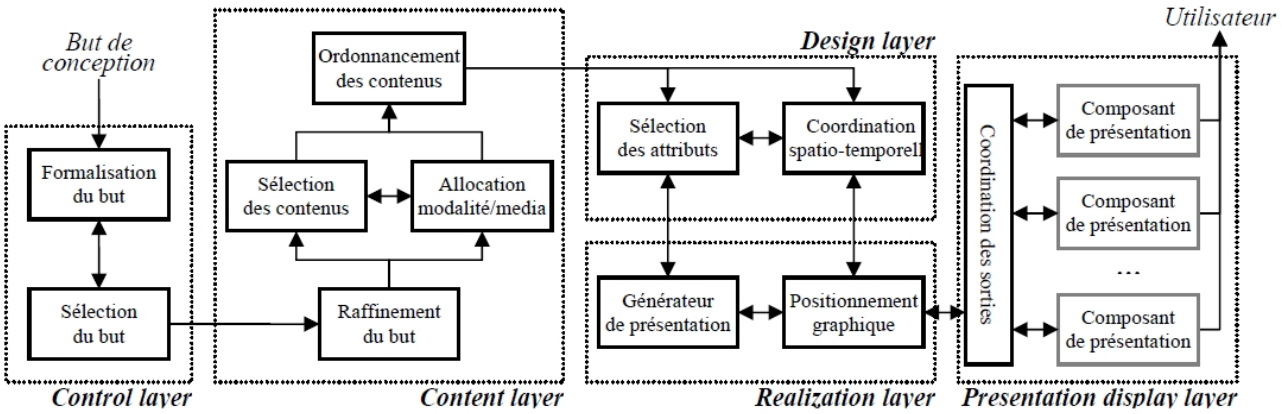

Figure 1.11 – Le modèle SRM [4]

- *Control layer ou couche contrôle* qui consiste à sélectionner le prochain but à atteindre ou la commande de présentation à exécuter.
- *Content layer ou couche contenu* qui raffine dans un premier temps le but en sous-but plus spécialisés, puis sélectionne pour chaque but élémentaire, le couple (modalité, média) qui va le restituer à l'utilisateur ainsi que le contenu de la présentation (par exemple, la suite de caractères contenus dans un texte affiché sur un écran). Les contenus ainsi sélectionnés sont ensuite ordonnés.
- *Design layer ou couche conception* qui fixe les attributs morphologiques (par exemple, la police et la taille des caractères pour un texte affiché sur un écran) et assure la coordination spatio-temporelle des contenus.
- *Realization layer ou couche réalisation* qui se charge de générer de manière effective la présentation.
- *Presentation display layer ou couche affichage de la présentation* dont le rôle est de distribuer les différents composants de la présentation aux médias adéquats et de coordonner les différentes composants afin d'aboutir à la présentation.

Ainsi, le modèle SRM décrit le processus de construction du système interactif multimodal en sortie, il abstrait le fonctionnement du noyau fonctionnel et modélise :

- La sélection de la commande de présentation à exécuter.
- Le raffinement du but de présentation en buts élémentaires.

- La sélection des couples modalité/média et contenus relatifs à chaque but élémentaire et leur ordonnancement.
- La détermination des attributs morphologiques et la coordination spatio-temporelle des contenus.
- La génération effective des présentations et leur coordination.

9.2.2 Le modèle WWHT (What, When, How, Then)

Le modèle conceptuel WWHT (What, Which, How, Then) [4] (voir Figure 1.12), repose sur le même principe de développement de l'interface multimodale en sortie du modèle SRM. Il en reprend les étapes à l'exception de celles de la couche *Control layer* et des étapes de génération effective de la présentation, pour les réorganiser en quatre étapes. D'autre part, il enrichit le modèle SRM en introduisant un mécanisme d'adaptation de la présentation multimodale au contexte d'interaction et la possibilité d'évolution d'une présentation multimodale en cas de changement du contexte. Il revient à se poser, lors du processus de conception d'une interface multimodale en sortie, les quatre questions suivantes :

- *What :* quelle information présenter ?
- *Which :* quelle présentation mutimodale choisir ?
- *How :* comment instancier cette présentation ?
- *Then :* comment faire évoluer cette présentation ?

Les réponses à ces questions conduisent à suivre le processus de conception de la présentation multimodale en sortie présenté en Figure 1.12, il se compose des quatre étapes suivantes :

- **Fission sémantique** : décomposition sémantique de l'information produite par le noyau fonctionnel en informations élémentaires à présenter à l'utilisateur. Cette décomposition peut éventuellement se réaliser en plusieurs étapes successives. Généralement, la fission sémantique est réalisée de façon manuelle par les concepteurs de l'interface multimodale en sortie.
- **Allocation de la présentation** : sélection pour chaque unité d'information élémentaire, de la présentation multimodale adaptée à l'état courant du contexte d'interaction, et regroupement en une même présentation multimodale. Chaque présentation consiste en un ensemble de couples (modalité, média) liés par des propriétés de redondance et/ou de complémentarité.
- **Instanciation** : détermination, selon l'état du contexte d'interaction, des contenus lexico-syntaxiques et des attributs morphologiques des modalités de la présentation.

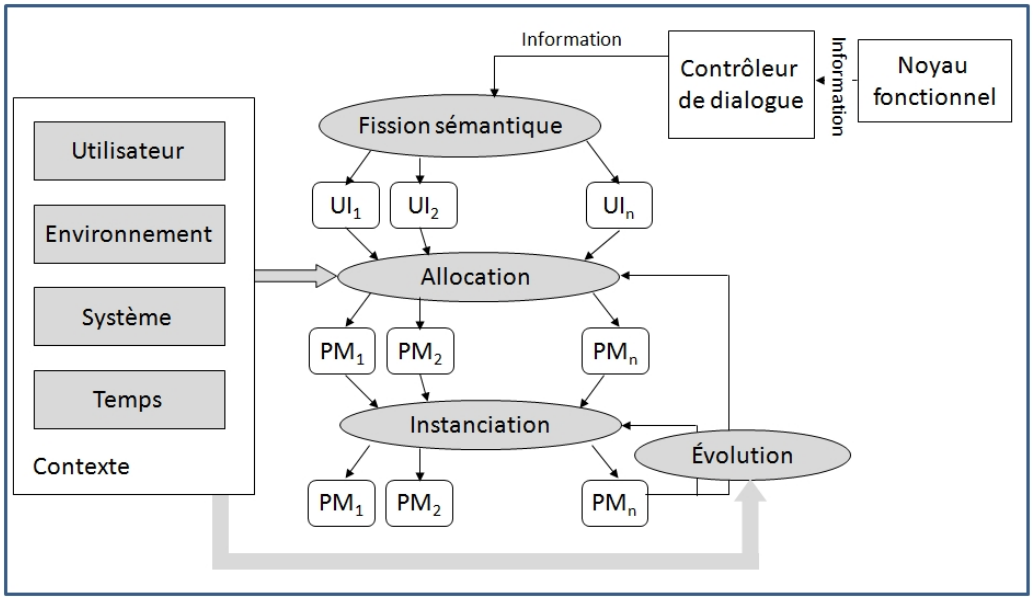

Figure 1.12 – Le modèle WWHT

– **Evolution** : évolution de la présentation multimodale suivant le changement du contexte. Cette évolution peut ramener la conception de la présentation, selon le degré de changement du contexte, soit à la phase allocation, soit à la phase d'instanciation.

Le modèle WWHT décrit le processus de construction du système interactif multimodal en sortie, il abstrait le fonctionnement du noyau fonctionnel et du contrôleur de dialogue et modélise le fonctionnement de la couche présentation qui construit les composants de l'interface.

9.2.3 Bilan sur les modèles de conception des IHM multimodales en sortie

Une comparaison entre les modèles SRM et WWHT est présentée dans Table 1.1. Elle confronte les deux modèles selon deux critères : les composants du système interactif dont ils modélisent le fonctionnement et les étapes modélisées lors de la construction de l'IHM multimodale en sortie. Elle montre que le modèle SRM modélise le fonctionnement du contrôleur de dialogue (couche contrôle) et de la présentation par les quatre autres couches (contenu, conception, réalisation et affichage). Le modèle WWHT modélise le fonctionnement de la présentation à travers ses quatre étapes (Fission sémantique, allocation, instanciation et évolution).

Le modèle SRM, modélise le processus de conception de la présentation dans sa globalité incluant au début la sélection de la commande de présentation à exécuter et à la fin le proces-

sus de rendu de la présentation. Cependant, il ne détaille pas les phases de raffinement du but ou d'allocation des modalités et média qu'il regroupe dans une même couche : *content layer*. Le modèle WWHT se focalise sur la conception de la présentation, il ne modélise pas la phase de sélection de l'information, et ne spécifie pas la génération effective de la présentation mais il détaille les phases de fission sémantique et d'allocation. Il modélise également les mécanismes d'adaptation au contexte et l'évolution de la présentation dans le temps.

	SRM	**WWHT**
Composant modélisé	- Contrôleur de dialogue - Présentation	- Présentation
Étape modélisée	- Sélection du but - Raffinement du but - Allocation modalité / média - Sélection du contenu - Sélection des attributs - Coordination des présentations	- Raffinement du but - Allocation modalité / média - Sélection du contenu - Sélection des attributs - Adaptation au contexte - Évolution de la présentation

Tableau 1.1 – Comparaison entre les modèles SRM et WWHT

10 Les systèmes multimodaux

Les systèmes multimodaux ont été mis en oeuvre dans divers domaines d'application. La multimodalité en entrée a suscité plus d'intérêt, elle est plus répandue que la multimodalité en sortie. Ainsi, il existe des systèmes multimodaux en entrée exclusivement, dont certains ont été développés pour des besoins réels tels que le système de [81] qui permet d'aider les chirurgiens à placer une vis dans le pédicule d'une vertèbre en utilisant deux modalités en entrée : un pointeur et un localisateur optique à trois dimensions, et le système VICO (Virtual Intelligent CO-Driver)[82] utilisé pour l'aide à la conduite d'automobile. Ce dernier utilise deux modalités en entrée : le geste à l'aide de l'écran tactile et la parole. D'autres systèmes ont été développés dans le but d'étudier l'interaction multimodale tels que [83] et [84] et d'autres sont développés dans un cadre expérimental dans des laboratoires de recherche tel que le système MATIS [85]. Parallèlement, des systèmes multimodaux en sortie exclusivement ont été développés. Parmi lesquels on citera : COMET [86] utilisé dans la génération automatique de diagnostics pour la réparation et la maintenance de radios portables militaires, par la combinaison des modalités visuelles (textes et graphiques), AlFresco [87] utilisé pour l'accès à l'information sur les fresques italiennes du 14^{ième} siècle par la coordination de modalités

visuelles (textes et images), PostGraphe [88] utilisé pour la génération de rapports statistiques par la coordination des modalités visuelles (textes et graphiques), MAGIC [89] pour la génération de briefings postopératoires cardiovasculaires, utilisant les modalités visuelle et auditive. Enfin, il existe des systèmes multimodaux en entrée et en sortie tel que SmartKom [90] permettant la gestion d'applications diverses : carnet d'adresses, accès aux services d'information, réservation d'hôtels, commande d'appareils domestiques. C'est un système multimodal symétrique utilisant les mêmes modalités (parole, geste, expression faciale) aussi bien en entrée qu'en sortie.

Notre choix s'est porté sur le système Smartkom comme étude de cas pour l'expression et la validation de nos modèles. Ce choix est motivé d'une part, par la forte multimodalité d'interaction qu'il offre par l'utilisation synergique de multiples modalités en sortie, et d'autre part, par le fait que l'interaction dans le système Smartkom est très proche de la communication entre humains grâce à des dialogues cohérents et coopératifs avec initiative mélangée. Nous décrivons ci-dessous de manière plus détaillée le système Smartkom.

10.1 Le système Smartkom

Le système de *SmartKom* [90], est un système multimodal symétrique flexible et adaptatif muni d'un système de dialogue à plusieurs initiatives et d'un agent conversationnel incorporé.

Smartkom utilise les mêmes modalités en entrée qu'en sortie (la parole, geste, expression faciale), la multimodalité en sortie est prise en charge par une interface utilisateur anthropomorphe [4]et affective à travers un agent conversationnel appelé *Smartakus* qui restitue l'information à l'utilisateur. Il s'agit d'un agent d'interface auto-animé avec un grand répertoire de gestes, de postures et d'expressions faciales. *Smartakus* emploie la langue du corps pour informer les utilisateurs qu'il attend leur entrée, qu'il les écoute, qu'il a des problèmes à comprendre leur entrée, ou qu'il travaille dur pour trouver une réponse à leurs questions. *SmartKom* est basé sur le paradigme de dialogue orienté-délégation situé (situated delegation-oriented dialogue paradigm) (SDDP) : l'utilisateur délègue une tâche à un assistant de communication virtuel. Lorsque la tâche est complexe, la délégation se fait au moyen d'un modèle simple de commande-et-contrôle. A contrario, lorsque la tâche est complexe, un dialogue de collaboration entre l'utilisateur et l'agent établit les spécifications de la tâche déléguée et les plans possibles de l'agent pour réaliser le but intentionnel de l'utilisateur. Ainsi, l'utilisateur aide *Smartakus* en cas de besoin, dans l'exécution de la tâche. *Smartakus*

4. Qui a la forme, l'apparence huamine

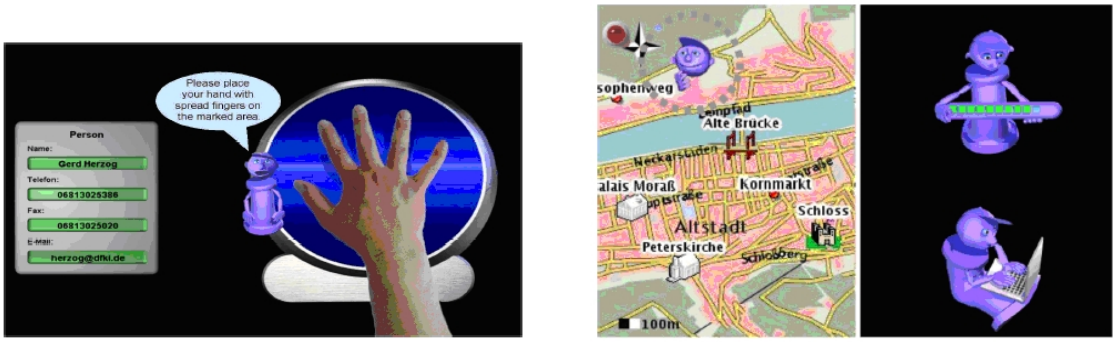

Figure 1.13 – Interaction dans *SmartKom*

accède à divers services numériques et appareils au nom de l'utilisateur, il assemble les résultats, et les présente à l'utilisateur.

SmartKom a été conçu comme un système transmutable qui peut s'engager dans différents types de tâches et dans différents contextes d'utilisation. Actuellement, l'utilisateur peut déléguer 43 types de tâches complexes à *Smartakus* moyennant des dialogues multimodaux dans les trois scénarios d'application suivants.

- un compagnon de communication qui aide avec le téléphone, le fax, l'email, et les tâches d'authentification ;
- un compagnon d'infotainment qui aide à choisir le contenu de médias et à actionner de divers appareils de TV (à l'aide d'un tablet PC en tant que client mobile) ;
- un compagnon de voyage mobile qui aide à la navigation et la récupération d'information à partir de services basés dans des endroits spécifiques (employant un PDA en tant que client mobile).

Exemple d'interaction dans *Smartkom*

Un scénario d'interaction illustrant le fonctionnement du système Smartkom au travers de son agent conversationnel Smartakus.

(1) Utilisateur : "Je voudrais aller au cinéma ce soir".

(2) Smartakus : [affiche une liste de titres de film] c'est une liste de films à Heidelberg.

(3) utilisateur : "Hmm, aucun de ces films ne semble être intéressant ... Montrez moi le programme TV."

(4) Smartakus : [affiche une liste TV] (voir Figure 1.14) ici [pointe la liste] est une liste des émissions TV Du soir.

(5) utilisateur : " Activez le troisième ! "

Figure 1.14 – Présentation des programme TV par *Smartakus*

11 Conclusion

A travers l'étude bibliographique que nous avons mené, nous avons pu constaté que les recherches dans le domaine de l'interaction Homme-Machine se sont intéressées à la description, la caractérisation et la conception des systèmes interactifs. Nous avons également constaté que la terminologie relative aux IHM n'était pas consensuelle, nous avons par conséquent introduit la terminologie considérée dans ce travail. Enfin, nous avons également présenté les principaux travaux de recherches menés dans le domaine de l'interaction Homme-Machine et plus particulièrement l'interaction multimodale. Cette étude nous a permis de faire les constats suivants :

1. La conception des IHM est une problématique multi-disciplinaire pouvant être envisagée sous différents angles : système, utilisateur, dialogue, architecture, composant, etc. Ceci a donné lieu à plusieurs notations et modèles qui sont employés selon l'objectif fixé pour ces interfaces. Les notations dédiées à la description des IHM (MAD, UAN, CTT) considèrent le système interactif dans sa globalité tandis que les notations orientées conception (Seeheim, ARCH, PAC MVC) sont plus focalisées sur la structure même de l'interface qu'elles séparent donc du noyau fonctionnel, le plus souvent abstrait dans cette spécification. Toutes ces notations sont le plus souvent graphiques et assez faciles d'accès, ce qui rend leur utilisation possible pour des non experts. Souvent, il est également nécessaire de vérifier un certain nombre de propriétés pour des IHM ou plus particulièrement des IHM multimodales nécessitant le respect d'un ensemble de

contraintes.

2. Les différents modèles et notations dédiés au développement des IHM constituent souvent une représentation partielle (description, architecture, dialogue) de l'interface. Les modèles proposés ne disposent pas d'une sémantique rigoureuse permettant d'effectuer des raisonnements pour affirmer ou infirmer le respect de propriétés sur ces interfaces, c'est ce qui a motivé plusieurs travaux visant à formaliser ces modèles, ils sont présentés dans le chapitre 2.
3. La mise en œuvre des interfaces multimodales repose souvent sur des modèles initialement proposés pour le développement des IHM conventionnelles. Cependant, elles s'en distinguent par la modélisation supplémentaire des mécanismes de fusion et fission sémantique des données, la combinaison concurrente et la synchronisation des modalités. Les modèles de conception des interfaces multimodales réutilisent souvent les notations proposées pour les interfaces conventionnelles, ils traitent souvent des parties ciblées du problème et sont le plus souvent dédiés à la multimodalité en entrée. La multimodalité en sortie a donné lieu à deux modèles de conception semi-formels qui s'intéressent à la modélisation de la présentation mais qui cependant ne permettent pas de garantir la satisfaction de spécifications dans un cadre critique.

Le but de notre travail est de proposer un modèle formel pour la conception des interfaces multimodales en sortie. Nous exploitons, pour cela, les résultats des recherches menées dans le domaine des IHM de manière générale, et des IHM multimodales en sortie de manière plus spécifique. Ainsi, nous formalisons dans le chapitre 3 le modèle WWHT dans un modèle générique qui permettra d'exprimer sans ambigüité le fonctionnement de l'interface multimodale en sortie et de valider les propriétés pertinentes pour ces interfaces.

Chapitre 2

Développements formels des systèmes interactifs

1 Introduction

Lorsqu'un système interactif est déployé dans un environnement critique, il est nécessaire de respecter scrupuleusement un ensemble de spécifications relatives aussi bien au fonctionnement interne de ce dernier (noyau fonctionnel) qu'au dialogue avec les utilisateurs (interface).

Les approches classiques de développement des logiciels (spécification, conception, codage et tests) sont dans l'incapacité de garantir qu'un logiciel satisfait parfaitement les exigences fixées pour ce dernier, l'utilisation de ces approches est très souvent couplée à de longues et couteuses phases de test et de relecture. Une autre alternative préconisée pour le développement des logiciels critiques est d'utiliser des démarches plus rigoureuses appelées démarches formelles, qui permettent l'expression de spécifications fiables : cohérentes et en accord avec les besoins et d'assurer par preuve formelle que certaines propriétés : sûreté, vivacité sont respectées.

Le plus souvent, le noyau fonctionnel et l'interface sont développés de manière séparée. C'est ainsi que le développement sûr des IHM a tiré profit des méthodes et langages développés pour la spécification fonctionnelle donnant lieu à plusieurs travaux de formalisation des interfaces que nous présentons dans ce chapitre.

2 Les modèles formels

Un modèle formel utilise une syntaxe associée à une sémantique rigoureuse et non ambiguë à base mathématique et logique, permettant d'effectuer des raisonnements. Moyennant

des règles de déduction ou d'inférence [1], il est possible de vérifier des propriétés particulières d'un système spécifié à l'aide de ces modèles. Enfin, les modèles formels utilisent une description abstraite du système, c'est pourquoi certaines approches de modélisation proposent des mécanismes de raffinement pour dériver le système concret.

3 La conception formelle

La conception formelle consiste à développer un système (logiciel ou matériel) en respectant strictement un ensemble de spécifications. Elle repose d'une part, sur des modèles formels qui spécifient le système à concevoir et les propriétés à respecter pour ce système, et d'autre part, des mécanismes de validation a priori tels que la preuve de théorème [91] et le model checking [92] ou a posteriori tel que les tests [93]. Un processus de conception formelle est considéré comme étant plus complexe qu'un processus de conception dit "traditionnel", cependant il s'avère nécessaire pour le développement des systèmes critiques.

Suivant [45], il existe principalement deux approches pour modéliser formellement un système.

3.1 Modélisation descendante ou par décomposition

Cette démarche consiste à modéliser le système global à un haut degré d'abstraction. Ce modèle abstrait est ensuite concrétisé en effectuant plusieurs étapes de raffinements. Le raffinement [94] permet d'enrichir la spécification avec de nouveaux éléments de description de la spécification. Cette démarche permet de préciser les modèles de la spécification au fur et à mesure des raffinements c'est-à-dire à chaque étape du développement. Les propriétés sont introduites au fur et à mesure et vérifiées à chaque étape du développement. Le principal atout de cette approche est que le raffinement permet de maitriser la complexité de la spécification en intégrant les détails de conception progressivement, il garantit également que les propriétés vérifiées dans les étapes précédentes sont toujours conservées au niveau de la nouvelle étape de raffinement.

3.2 Modélisation ascendante ou par composition

Cette approche consiste à modéliser le système par composition de ses sous-systèmes. C'est une démarche permettant la réutilisation de systèmes déjà définis dans la conception

1. Règles permettant de produire de nouveaux faits à partir de prémisses et d'axiomes.

de nouveaux systèmes. L'inconvénient majeur de cette approche est qu'elle ne garantit pas la conservation des propriétés des sous-systèmes. Le concepteur doit alors vérifier cette préservation de propriétés.

4 Les méthodes de spécification formelle

D'après Hinchey [95], une méthode formelle est un ensemble d'outils et de notations (avec une sémantique formelle), utilisés pour spécifier de manière non ambigüe les spécificités du système et supportant les preuves de propriétés sur ces spécifications et les preuves de correction d'une implémentation éventuelle par rapport à la spécification.

D'après Gaudel [96], une notation ou une technique est formelle si elle est sujette à des manipulations systématiques suivant un calcul mathématique ; une méthode est formelle si elle utilise une notation ou une technique formelle.

Plusieurs classifications des méthodes formelles existent dans la littérature, nous citons une première classification basée sur les théories sur lesquelles se basent les méthodes (ensembles, types algébriques, automates et logique typée d'ordre supérieur) présentée dans [97], une seconde classification suivant les types de modèles (syntaxique, sémantique et temporel) décrite dans [16]. Enfin, nous présentons une dernière classification proposée dans [98], elle classe les méthodes formelles en fonction de ce qu'elles décrivent du système (modèle, propriétés).

4.1 Les méthodes orientées modèles

Appelées également méthodes orientées états, elles spécifient de manière explicite, à l'aide de concepts issus de la théorie des ensembles et de la logique, les états du système généralement par la construction d'un système états-transitions, l'état est modifié par les opérations qui modélisent l'aspect comportemental du système. Les propriétés sont exprimées par des expressions logiques sur les variables du système, elles sont vérifiées soit par *preuve de théorème* soit par *vérification sur modèles (model checking)* (voir section 5).
Dans ces approches, les systèmes sont modélisés à l'aide des méthodes utilisant la théorie des ensembles telles que : *la notation Z* [99],*le langage VDM* [100] et *la méthode B* [101] ou les logiques typées d'ordre supérieur telles que : *Coq* [102], *HOL* [103], *PVS* [104] et *Isabelle* [105]. La notation Z, ne permettant pas un raffinage complet des spécifications, elle est restreinte à la spécification fonctionnelle. Les méthodes VDM et B qui proposent des mécanismes de raffinement impliquent des obligations de preuve. La méthode B étant la plus

élaborée, complète et outillée, elle connaît une expansion considérable en milieu industriel. En effet, elle repose sur un processus de raffinement progressif aboutissant à une description concrète automatisable. Le processus de preuve porte d'une part, sur la cohérence du modèle mathématique et d'autre part, sur la préservation des propriétés du modèle initial par les raffinements successifs. La méthode B dispose d'un outil appelé *Atelier B* comprenant un générateur d'obligations de preuve et un assistant à la preuve. Elle a été étendue pour donner lieu à la méthode B Évènementiel [56]. La méthode B Évènementiel sera présentée en détail dans le chapitre 3 ainsi que son utilisation pour formaliser le modèle de conception des IHM multimodales en sortie. Les logiques typées d'ordre supérieur sont dotées d'un haut niveau d'expressivité car elles permettent de considérer un prédicat comme un objet susceptible de varier, voire d'être manipulé comme argument d'une fonction ou d'un prédicat [97]. Ces logiques sont combinées à un système de types, elles proposent des mécanismes d'extraction de programme qui permettent de développer simultanément un programme fonctionnel et de le prouver. Le haut degré d'expressivité offert par la logique d'ordre supérieur lui permet de spécifier une variété de systèmes mais augmente la complexité du processus de preuve.
Les systèmes réactifs, concurrents et distribués sont quant à eux modélisés à l'aide de systèmes état/transition tels que les réseaux de Pétri [106], CSP [107] ou les automates à états étendus dans *SDL* [108] et *Estelle* [109]. Le système est modélisé par des processus synchronisés par envoi de message, par variables partagées ou par rendez-vous. Chaque processus est un système caractérisé par ses différents états qui sont reliés par des transitions. La vérification des propriétés sur cette classe de méthodes se fait par l'exploration du modèle ou par la preuve de théorème. Les modèles basés sur les automates à états étendus associent dans leur spécification, un modèle algébrique pour la description statique du système, ils en héritent des mécanismes de raffinement et de preuve. La partie dynamique de la spécification offre la possibilité d'utiliser une logique temporelle afin de vérifier des propriétés à un haut niveau d'abstraction par la technique du model checking.

4.2 Les méthodes orientées propriétés

Appelées également algébriques ou fonctionnelles, ces méthodes reposent sur une sémantique définie par la logique du premier ordre et une algèbre introduisant des variables et des opérations. Le système est décrit par un ensemble d'équations décrivant le comportement du système. L'axiomatisation de ces équations fournit les propriétés (comportementales) du système à développer. Le raffinement s'effectue en faisant correspondre à chaque type

abstrait, un type concret au moyen d'une fonction de concrétisation. Le mécanisme de raffinement employé exige d'effectuer certaines preuves qui assurent la cohérence entre types abstrait et concret. Le processus de preuve est souvent supporté par des outils qui utilisent des techniques de réécriture [110], de résolution équationnelle [111] ou de récurrence. Les modèles algébriques sont très peu outillés (ASL [112], *ACT ONE* [113], OBJ [114], *LPG* [115]), c'est pourquoi ils sont peu utilisés en pratique, à l'exception de *LOTOS* [116], qui est considéré comme un modèle hybride qui combine le modèle algébrique de ACT ONE pour la description des types abstraits de données au langage orienté modèle CCS [117] pour la spécification du comportement des processus et leurs interactions. Cette combinaison de modèles fait de LOTOS un langage adapté à la description des systèmes concurrents.

Selon [98], les approches orientées modèles conviennent à la spécification des systèmes car elles spécifient directement le système en décrivant son fonctionnement avec précision ce qui fait que certains changements mineurs sur une spécification peuvent remettre en cause toute la modélisation, tandis que les approches orientées propriétés décrivent le système par une conjonction d'axiomes spécifiant ce qui est attendu du système, cette spécification peut ainsi donner lieu à plusieurs implémentations possibles et la modification de la spécification n'est pas très couteuse, ce qui implique que les approches orientées propriétés se prêtent à la spécification des besoins.

5 Vérification formelle

La vérification formelle consiste à vérifier que les propriétés souhaitées sont respectées sur un système (modèle) donné. Cette vérification se fait par la confrontation d'une expression formelle des propriétés et d'un modèle mathématique, basé sur un langage formel, représentant le système. Le mécanisme de vérification doit examiner que les propriétés souhaitées sont satisfaites par les comportements du système. Il existe deux principales approches de vérification formelle : *la preuve de théorème* et *le model checking*.

5.1 Preuve de théorème

Introduite par Hoare [118], cette approche est basée sur une preuve mathématique, elle consiste à décrire le système et ses propriétés dans un modèle de sémantique axiomatisable exprimée par des axiomes et des règles d'inférence et à démontrer que les propriétés peuvent être obtenues à partir du système en utilisant les axiomes et règles d'inférence. Elle est utilisée dans des outils variés ; du plus automatisé au plus interactif et du plus général

au plus spécialisé. Certains sont guidés par une séquence de lemmes et de définitions, mais chaque théorème est prouvé automatiquement en utilisant des heuristiques pour l'induction, la réécriture et la simplification. Il existe des outils combinant plusieurs techniques, tels que Analytica [119] qui combine le theorem-proving avec l'algèbre symbolique de Mathematica. PVS et STep permettent d'utiliser le model-checking et la preuve. Enfin, les outils supportant les méthodes B et B Évènementiel combinent un développement par raffinement, un prouveur, un générateur de code et un outil de model checking (ProB).

L'approche preuve de théorème a l'avantage de pouvoir traiter des systèmes à nombre d'états indéfini. Elle ne repose pas sur une construction explicite et exhaustive d'un modèle de comportement de type états/transitions, très coûteux en mémoire, puisqu'elle est capable d'inférer des conclusions directement à partir d'une description d'événements ou d'opérations permettant de faire évoluer le système. Néanmoins, elle nécessite souvent l'intervention du concepteur dans un processus de preuve interactive.

5.2 Vérification sur modèle ou model checking

Le model checking est une approche algorithmique utilisée dans les méthodes formelles utilisant des formalismes à base d'automates, elle repose sur l'énumération de tous les états possibles du système, afin de s'assurer qu'aucun de ces états n'est en contradiction avec les comportements que l'on désire pour le système. Cette technique de vérification est le plus souvent complètement automatisée par des outils dédiés appelés model checkers tels que SMV [120] ou SPIN [121].
Il existe deux principales approches de model checking qui diffèrent sur la manière dont le comportement souhaité du système est décrit.

5.2.1 Approche basée sur la logique (hétérogène)

Dans cette approche issue des travaux de Quielle et Sifakis (1981) et Clarke et Emerson (1981), les comportements attendus du système sont décrits par un ensemble de propriétés exprimées dans une logique appropriée (temporelle [2] ou modale [3]), le système est généralement modélisé par un automate à états finis et l'algorithme de model checking consiste à vérifier si le modèle satisfait ces propriétés pour un ensemble donné d'états initiaux.

2. Logique qui permet de spécifier des propriétés qui change dans le temps.
3. Logique exprimant certains concepts modaux tels que l'obligartion, la possibilité ...

5.2.2 Approche basée sur le comportement (homogène)

Comme son nom l'indique, dans l'approche homogène, les comportements attendus aussi bien que ceux possibles sont exprimés dans la même notation à savoir les automates, la vérification repose sur la confrontation des comportements attendus et souhaités au moyen de relations d'équivalence ou de pré-ordre. Les relations d'équivalence expriment généralement la notion de « se comporter comme », tandis que les relations de pré-ordre représentent la notion de « se comporter au moins comme ». Plusieurs relations d'équivalence et de pré-ordre ont été définies. Les plus connues sont la relation d'équivalence de bisimulation [4] et la relation de pré-ordre d'inclusion [5].

Les deux approches hétérogène et homogène sont conceptuellement différentes pourtant des relations entre elles peuvent être établies. Ainsi, si à l'aide de l'approche basée sur la logique, il s'avère que deux modèles satisfont les mêmes propriétés alors on peut affirmer que ces deux modèles ont deux comportements équivalents. Inversement, et à fortiori, puisqu'en général l'approche basée le comportement est plus rapide que celle basée sur la logique, si deux modèles sont équivalents alors il est clair qu'ils satisfont les mêmes propriétés.

Du fait que le model checking procède par énumération exhaustive des états du système à vérifier, la représentation de tous les comportements possibles d'un système avec un grand nombre d'états, conduit rapidement à un dépassement des capacités de stockage en mémoire et d'exploration. Ce phénomène est connu sous le nom d'explosion combinatoire, il fait l'objet de plusieurs recherches qui ont donné lieu à des techniques d'abstraction permettant d'aborder la vérification des systèmes de grandes taille [122].

6 Utilisation des méthodes formelles pour le développement des IHM

Plusieurs travaux ont été dédiés à la spécification et à la vérification formelles des IHM. Les méthodes de spécification utilisées sont essentiellement orientées modèle, on peut citer : les réseaux de pétri, les flots de données, la méthode B, etc. La vérification formelle s'opère aussi bien par preuve de théorème que par model checking. Ces travaux ont traité des interfaces de type WIMP, les interfaces multimodales ont donné lieu à peu de travaux qui se sont le plus souvent contentés d'étendre des modèles proposés pour les interfaces

4. Fait qu'un automate puisse simuler tout comportement d'un autre automate et vice versa.
5. Un premier automate est inclus dans un second automate si tous les mots acceptés par le premier sont acceptés par le second.

WIMP afin de prendre en considération les mécanismes induits par la multimodalité avec une prévalence pour la multimodalité en entrée. Ci-dessous une revue non exhaustive des différentes approches employées.

6.1 Les algèbres de processus : LOTOS

L'algèbre de processus LOTOS [116] a été utilisée pour modéliser les interacteurs de Pise, Cnuce dans les travaux de Paterno et Faconti [123] et [124] et les interacteurs ADC dans les travaux de Markopoulos [125], [126]. Dans cette approche l'interface est modélisée par un ensemble d'interacteurs interconnectés entre eux. Chaque interacteur est spécifié en LOTOS. La composition des différents interacteurs est réalisée par le lien entre les ports de communication des différents processus LOTOS. Paterno, démarre la conception de l'interface à partir d'une spécification des tâches qu'il traduit en une hiérarchie d'interacteurs. Ensuite, chaque interacteur est traduit en Full-LOTOS [127] qui est un langage basé sur LOTOS de base pour sa partie spécification du contrôle et la spécification algébrique ACT-ONE [128] pour sa partie spécification des données. Enfin, la spécification LOTOS de l'interface est traduite en un système de transitions. Pour que cette traduction soit possible automatiquement, la spécification est traduite d'abord vers LOTOS de base. Cette traduction implique la perte des données et les paramètres ainsi que les gardes booléennes utilisées pour contraindre le comportement. Une fois le système de transitions correspondant à la spécification de l'interface obtenu, les propriétés sont spécifiées dans la logique ACTL et vérifiées à l'aide du model-checker de l'outil *Lite*[129]. Un ensemble de schémas de propriétés vérifiées à l'aide de la logique temporelle ACTL est défini dans [123].
Dans le travail de Markolpoulos [125], les deux approches de model-checking ont été utilisées. Une approche hétérogène qui consiste à traduire la spécification LOTOS en un système de transitions et la spécification des propriétés à l'aide de la logique temporelle ACTL. Une deuxième approche homogène consiste à spécifier les propriétés en LOTOS et la vérification de la relation de conformité entre le système de transitions de l'interface et celui de la propriété est réalisée à l'aide de l'outil CAESAR/ALDEBARAN [130]. L'avantage de LOTOS est qu'il est structuré et permet une description formelle de l'architecture. Cependant, au moment de la génération des systèmes de transitions, la spécification est traduite vers LOTOS de base ce qui implique la perte des données spécifiées. Le problème se pose essentiellement pour les évènements gardés. Avec la perte des gardes, le système obtenu n'est pas équivalent à celui d'origine. Pour éviter ce problème, l'utilisation des gardes dans la spécification est à éviter ce qui limite le pouvoir d'expression du langage. Un autre inconvénient est que si le

langage LOTOS est assez abordable par la communauté des non experts dans le domaine des techniques formelles, ce n'est pas le cas pour la spécification algébrique des données ACT-ONE. Ces travaux n'offrent pas une interface aidant le concepteur d'IHM à utiliser cette technique sans être expert.
LOTOS a été utilisé également pour les IHM multimodales. Dans [131] et [132], LOTOS est utilisé pour décrire les interacteurs modélisant l'interface de l'application multimodale Matis. La modélisation de l'interface débute par une modélisation des tâches, en utilisant la notation UAN. Cette modélisation est raffinée jusqu'au niveau des tâches interactives élémentaires. Les interacteurs ont été implémentés en LOTOS. Un ensemble de propriétés d'utilisabilité a été exprimé dans la logique temporelle ACTL et vérifié par le model checker de l'outil Lite.

6.2 Les réseaux de Petri

Palanque et al. utilisent dans [50] et [133] le formalisme des objets coopératifs et interactifs (ICO) (Interactive Coopérative Objects). C'est une technique de description formelle dédiée à la spécification, modélisation et implémentation des systèmes interactifs. Elle utilise les concepts de l'approche orientée objet (instanciation dynamique, classification, encapsulation et héritage) pour la description de la partie statique du système et les réseaux de Petri pour la description de sa partie dynamique ou comportement. Dans cette approche le système est modélisé en général selon le modèle d'architecture ARCH. Dans le formalisme ICO, un objet est une entité caractérisée par quatre composants : un objet coopératif avec des services utilisateur, une partie présentation, et deux fonctions (la fonction d'activation et la fonction de rendu) qui font le lien entre l'objet coopératif et la partie présentation [134].
Une fois que l'interface est décrite dans le formalisme ICO, la description est ensuite traduite vers les réseaux de Petri standards. La vérification est effectuée par analyse du graphe du marquage du réseau de Petri obtenu. Un ensemble de propriétés de dialogue ont été vérifiées telles que : l'absence de blocage, re-initialisabilité et les états atteignables. La vérification effectuée peut concerner toute ou une partie du système, mais dans cette approche, la spécification des propriétés n'est pas assistée par un langage de description des propriétés d'interaction. De même, aucune validation de tâche n'est effectuée et seules les propriétés d'atteignabilité sont vérifiées.
Le formalisme ICO est supporté par l'environnement PETSHOP [135], il fournit un éditeur-interpréteur ICO qui permet de spécifier le système par un ensemble d'objets ICO, d'exécuter et d'analyser les réseaux de Petri décrivant leurs comportements. L'outil PETSHOP permet

également de décrire le modèle de tâches en faisant appel à l'éditeur CTTE [30], permettant ainsi de mettre en relation la tâche utilisateur avec le service offert par le système pour la réaliser.

Dans [136], [134] et [137], le formalisme ICO est étendu pour modéliser les IHM multimodales, il permet représenter la fusion de deux modalités : le geste et la parole [138]. Pour cela, le formalisme ICO a intégré une modélisation du temps dans la description du comportement de l'ICO, mais également un ensemble de mécanismes tel que la communication par production et consommation d'événements (actions interactives plus ou moins abstraites opérées soit par l'utilisateur soit par le système). Ce mécanisme permet de synchroniser un ensemble de transitions par rapport à un événement, et d'émettre un événement lors du franchissement d'une transition. Un second mécanisme de structuration permettant de traiter des événements à différents niveaux d'abstraction (abstrait, physique) et enfin, un troisième mécanisme de rendu qui modélise le comportement des médias de sortie selon le service fourni par l'application.
Des interactions d'une application de réalité virtuelle ont été présentées dans [136], les interactions y sont modélisées à plusieurs niveaux d'abstraction dont le premier niveau concerne les actions physiques faisant intervenir les modalités [16].

6.3 Les approches synchrones à flot de données : Lustre

Lustre [55] est un langage synchrone à flot de données. Il a été utilisé dans [60] et [139] pour décrire des structures d'interacteurs adaptées de celle de York. Chaque interacteur est modélisé par un noeud Lustre et l'ensemble de l'interface est conçu en interconnectant un ensemble de noeuds. Une description de l'interface est faite d'abord à l'aide d'Uil/x [140], un langage décrivant la hiérarchie des objets de la présentation. Ensuite, elle est traduite en noeuds Lustre. Un interacteur est un noeud qui a en entrée/sortie les flots booléens correspondants aux signaux d'action, de réaction, d'activation, de représentation et aux états internes du modèle d'interacteur. Dans cette approche les propriétés sont exprimées dans le même formalisme que l'interface. Une propriété est spécifiée par un noeud Lustre et elle est vérifiée si le flot de sortie du noeud est vrai. La vérification est effectuée avec l'outil Lesar[141]. Contrairement aux autres model-checker, dans l'approche Lustre, l'automate représentant le système n'est pas généré avant le processus de vérification, mais pour chaque propriété un nouvel automate est généré. Les propriétés vérifiées par Lustre sont limitées par rapport à celles exprimées en logiques temporelles, la vérification de la réalisation d'une tâche utilisateur ne peut pas être automatisée et seules les propriétés de sûreté et une forme

restrictive de vivacité peuvent être vérifiées.

Le langage Lustre a été également utilisé dans [142] et [143] pour effectuer des tests logiciels afin de vérifier si le système développé vérifie les propriétés souhaitées. Ces travaux utilisent l'outil de vérification de logiciel réactif synchrone Lutess [144] pour vérifier un système interactif multimodal développé avec la plateforme ICARE [145]. Lutess construit automatiquement un générateur de données de test pour le programme sous test. Dans ces travaux, les propriétés CARE ont été spécifiées en Lustre, ensuite vérifiées à l'aide de l'outil Lutess. Ces travaux n'utilisent pas les méthodes formelles au cours de la conception du système mais y font appel à la fin du développement du système multimodal afin d'effectuer les tests logiciels.

6.4 Les machines à état

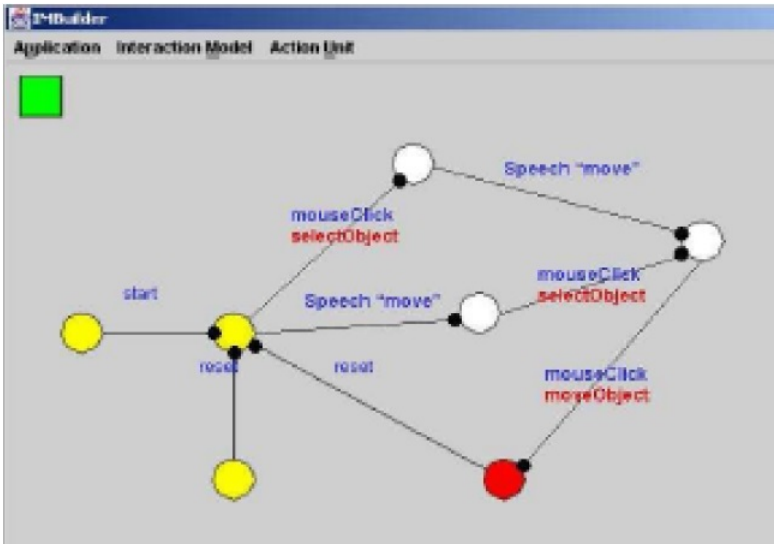

Figure 2.1 – Machine à état modélisant la commande "déplacer un objet", construite avec IMBuilder [5]

Les machines à état ou automates finis sont une technique de description qui consiste à décrire un système par une collection d'états, de transitions, et d'actions. Un état est une caractérisation statique du système par un ensemble de variables dites d'état. La transition correspond au passage d'un état de départ à un état cible, elle est exécutée lorsque l'automate est à l'état source et que l'évènement relatif à la transition se produit.
Les travaux de [146] montrent que les machines à états constituent un modèle adéquat pour modéliser l'interaction multimodale en entrée. Ils montrent que ce modèle formel permet aussi la modélisation des différentes contraintes temporelles et de synchronisation.

Un outil IMBuilder [5] (voir Figure. 2.1) a été développé pour spécifier les machines à états. Une liste d'entrées valides ainsi qu'une liste de commandes peuvent être créées ou importées d'un fichier XML.

Ces travaux restent de l'ordre du prototypage des interactions multimodales. Elles ont traité uniquement de la multimodalité en entrée, aucun modèle d'interaction n'est proposé et les propriétés d'utilisabilité ne sont pas modélisées ni vérifiées. Tous ces travaux ne traitent que partiellement la modélisation de l'interaction multimodale. Ils s'intéressent uniquement à prouver que telle technique ou que tel modèle formel peut aider à modéliser l'interaction multimodale. Aucun modèle de conception n'est pris en compte, ni une modélisation formelle des propriétés CARE n'est proposée.

6.5 La preuve de théorème : Z et B

La méthode Z a été utilisée, dans les travaux de Duke et Harisson, pour la description des IHM. Dans [58], les interacteurs ont été modélisés en utilisant le langage Z au moyen de plusieurs étapes de raffinement et les propriétés ont été vérifiées à l'aide d'invariants (propriétés devant toujours être vérifiées). Les travaux de [147] utilisent Z et CSP [107] pour modéliser les systèmes multimodaux. Les auteurs utilisent Z pour modéliser l'aspect statique de l'interface et l'algèbre CSP pour l'aspect dynamique. Ils modélisent le système Matis [85] en utilisant les interacteurs. Ces travaux font abstraction des modalités et évitent ainsi de prendre en charge les spécificités des systèmes multimodaux. Les problèmes de fusion et fission des modalités ainsi que la vérification des propriétés ne sont pas abordés.

Plusieurs travaux ont utilisé la méthode B et ses extensions pour la modélisation des IHM, ils se basent essentiellement sur le modèle ARCH comme architecture utilisée pour la description et la spécification du système interactif. Tout d'abord, [52] et [53] proposent une formalisation dans la méthode B d'un système interactif opérationnel de type WIMP (le Post-It Notes), cette spécification a permis la vérification de certaines propriétés ergonomiques telles que l'observabilité, l'insistance ou la robustesse. Dans les travaux de [148], une application est implémentée par raffinement jusqu'à la génération du code dans un langage de programmation et un ensemble de propriétés ont été vérifiées. Dans [149], une expérience comparative pour la modélisation et la vérification des IHM est présentée, elle confronte la preuve de théorème avec B et le model-checking avec Promela/SPIN.

D'autres travaux utilisant la méthode B ont été réalisés, en s'intéressant à une architecture orientée objet, telle que MVC ou PAC, le langage de la méthode B ne permettant pas d'utiliser directement les modèles à agents, une nouvelle architecture hybride appelée CAV [150] a

été développée. Elle constitue l'architecture de base pour un processus de développement B complet du système interactif allant jusqu'à la génération du code et la vérification de propriétés de sûreté et d'utilisabilité. Dans [2], B événementiel a été utilisé pour la modélisation du contrôleur de dialogue, cette démarche a permis de valider des tâches décrites en CTT.

La méthode B, dans sa version événementielle, a été utilisée dans [151] et [152] pour la modélisation et la validation de systèmes interactifs multimodaux en se limitant à la multimodalité en entrée. Le système a été modélisé dans B Évènementiel par un ensemble d'événements gardés (l'évènement est déclenché dès que sa garde est satisfaite). Les propriétés ont été exprimées sous forme d'invariants dans la logique du premier ordre. Les auteurs ont proposé une méthodologie de développement basée sur le raffinement [151] et une validation de tâches exprimées avec l'algèbre de processus CTT [152]. Une représentation des propriétés CARE en B a été également abordée dans [152].

6.6 Le model checking

SMV a été utilisé par Abowd et al. [153] pour la vérification d'une spécification du dialogue effectuée à l'aide du Simulateur d'Action (Action Simulator). L'IHM est spécifiée à l'aide du simulateur d'action, ensuite, la spécification est traduite dans le langage d'entrée de SMV qui permet de vérifier des propriétés exprimées dans la logique temporelle CTL. Dans ce travail, aucun modèle d'architecture n'est utilisé pour représenter l'IHM. Seul le dialogue est spécifié de manière tabulaire à l'aide du système de production propositionnel (PPS) [154]. L'outil offre aussi la possibilité de simuler le dialogue. Un ensemble de schémas génériques de propriétés de dialogue a été proposé. On y trouve le non blocage, la complétude des tâches, etc...

Dans [43], XTL(eXtended Temporal Logic) une logique temporelle dédiée à la description des systèmes interactifs est développée, elle permet la spécification de problèmes temporels par la modélisation de la concurrence, avec une sémantique du parallélisme et de l'entrelacement, les interruptions, les interruptions suivies par de l'entrelacement, etc.

SMV a également fait l'objet du travail de [155], où les interacteurs de York sont d'abord spécifiés dans un langage basé sur la logique modale MAL [156], ensuite la spécification est traduite vers le langage d'entrée de SMV où des propriétés exprimées en CTL sont vérifiées. Contrairement au travail précédent où seul le dialogue est modélisé, ici toute l'interface est spécifiée en utilisant l'architecture d'interacteur de York.

Enfin, [157] propose une utilisation de SMV et de SPIN pour vérifier les propriétés d'utilisabilité des IHM multimodales en entrée. Ces travaux utilisent une description opé-

rationnelle des interfaces par les systèmes états-transitions proposée par le même auteur. Ils reposent également sur la définition d'un cadre générique formalisant les propriétés CARE. Ces travaux proposent des règles de passage permettant de coder la description de l'interface multimodale dans le langage d'entrée des model-checkers, ainsi que les propriétés CARE dans les logiques : CTL (SMV) et LTL(SPIN). Ils ont permis de vérifier les propriétés de complémentarité, d'assignation et d'équivalence.

7 Synthèse et proposition

Cette section propose une synthèse relative aux travaux consacrées à la modélisation de l'interaction Homme-Machine. Nous nous focalisons plus particulièrement sur les approches dédiées à un développement sûr des IHM et plus particulièrement les IHM multimodales. Un bilan des travaux développés dans ce sens est dressé afin de déterminer de manière plus précise le champ d'action de notre proposition ainsi que l'approche la plus indiquée pour couvrir les besoins en terme de modélisation et de validation formelles des IHM multimodales en sortie.

7.1 Synthèse

Au terme de ce chapitre, nous avons pu constater que les approches formelles de développement des IHM permettent de spécifier et de valider le système interactif de manière plus ou moins complète selon les exigences fixées pour ce dernier (besoins utilisateur, fonctionnement du système interactif). Certaines approches se sont intéressées à spécifier le dialogue comme : les approches basées sur les automates, les réseaux de Petri (ICO) ou le model checking (SMV), elles permettent essentiellement de valider les besoins utilisateur. D'autres approches plus soucieuses de la conception du système interactif décrivent ses composants, nous pouvons citer dans cette catégorie les approches de modélisation par composition d'interacteurs en utilisant les langages : LOTOS, Z et Lustre. Enfin, une dernière catégorie d'approches permet de spécifier entièrement le système interactif, nous pouvons citer dans cette catégorie l'utilisation du model checking avec SMV pour la validation complète de l'interface, les travaux utilisant la méthode B pour les systèmes interactifs de type WIMP, et la méthode B Événementiel pour les systèmes interactifs multimodaux en entrée, ces travaux ont permis de spécifier le système interactif en couvrant tout le processus de développement (spécification, conception, validation, implémentation).

Ces différents travaux de formalisation des systèmes interactifs utilisent des modèles d'architecture différents. Les modèles à interacteurs sont adoptés dans les approches de : LOTOS, Lustre et SMV. Le modèle ARCH est utilisé dans les approches réseaux de Petri et quelques travaux utilisant la méthode B. Enfin, les modèles MVC, PAC et CAV sont employés pour d'autres approches utilisant la méthode B.

L'utilisation de ces approches formelles a permis la validation de manière ciblée de différentes propriétés. Les réseaux de Petri valident des propriétés de dialogue et de vivacité (absence de blocage). Les approches à flot de données avec Lustre valident des propriétés de sûreté et de vivacité. La méthode B vérifie des propriétés de sûreté, de vivacité, d'ergonomie et d'utilisabilité. Enfin, le model checking avec SMV permet de conduire la vérification de propriétés de dialogue, de sûreté, de vivacité et d'utilisabilité.

Les approches basées sur la vérification par preuve de théorème souvent accompagnées de mécanismes de raffinement introduisant les détails de conception progressivement, permettent de maîtriser la complexité de la spécification du système et de la preuve, cependant, elles requièrent souvent l'intervention du concepteur dans le processus de preuve. Les approches basées sur le model checking permettent de vérifier une large gamme de propriétés, elles sont entièrement automatisées mais se heurtent au problème de l'explosion des états.

Les IHM multimodales héritent des caractéristiques et des contraintes de développement des IHM conventionnelles, auxquelles viennent s'ajouter les exigences propres à l'interaction multimodale : modélisation des mécanismes de fusion et de fission, concurrence et synchronisation. Deux modèles semi-formels dédiés à la conception des IHM multimodales en sortie ont été proposés : le modèle SRM qui modélise le processus de construction de la présentation et le modèle WWHT qui intègre à ce processus de construction de la présentation des mécanismes d'adaptation au contexte et d'évolution. Ces modèles qui décrivent le principe de fission caractéristique de la multimodalité en sortie demeurent cependant très ambigus quant à la description des relations sémantiques et temporelles entre composants fissionnés et sur les différents de schémas de construction de la présentation finale (combinaison temporelle et sémantique). De plus, ils ne permettent d'effectuer aucune validation rigoureuse de ces interfaces.
Une formalisation générique du processus de construction de la présentation nous semble par conséquent nécessaire, elle permettra de décrire de manière détaillée, rigoureuse et indépendamment de tout langage de formalisation, la construction de la présentation.

La généricité de ce modèle offre l'avantage de pouvoir basculer vers une méthode de spécification choisie au moyen de règles de transformations et/ou d'interprétation et de procéder à la validation des propriétés attendues pour ces interfaces.

7.2 Notre proposition

La Figure 2.2 présente la vue globale de la proposition que nous formulons dans ce mémoire dont l'objectif est de définir une approche formelle et générique de conception et de vérification des IHM multimodales en sortie intervenant dans le cadre d'un processus de développement sûr d'un système interactif multimodal en sortie.

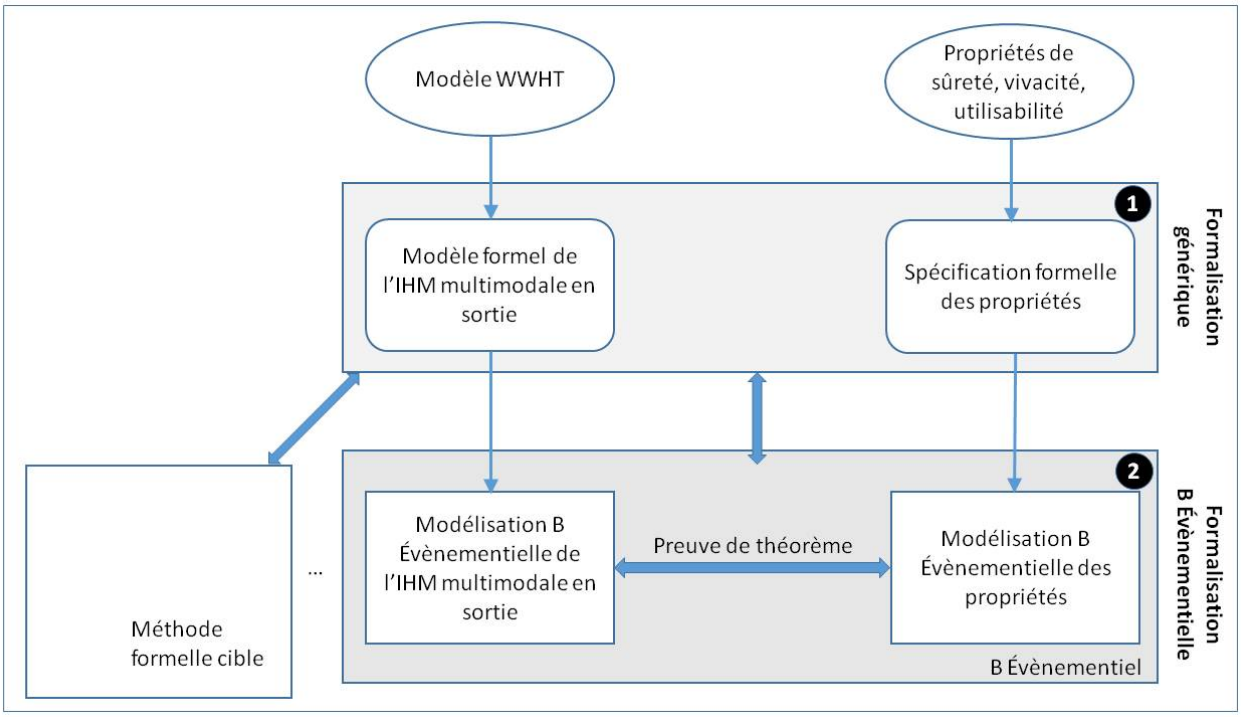

Figure 2.2 – Principes de l'approche proposée

L'approche proposée permet d'assister le concepteur de l'IHM multimodale en sortie dans son processus de développement (spécification formelle, conception et validation). Elle est basée sur le modèle semi-formel WWHT dont elle spécifie formellement les deux étapes : fission sémantique et allocation. L'instanciation étant fortement liée aux dispositifs physiques d'interaction déployés dans le système interactif à modéliser, elle est par conséquent spécifique à chaque système multimodal et ne peut être modélisée dans le modèle formel que nous proposons, qui se veut générique. Au même titre, l'évolution qui fait intervenir l'allocation et l'instanciation, n'est pas formalisée dans notre approche.

La modélisation formelle de la fission sémantique et de l'allocation spécifie formellement la construction de la présentation d'un système interactif par l'abstraction du noyau fonctionnel et du contrôleur de dialogue. Par conséquent notre approche peut utiliser les modèles d'architecture globaux : Seeheim et ARCH.

L'approche que nous proposons s'articule autour de deux étapes.

1. **Une formalisation générique :** cette première étape consiste en une formalisation des deux premières étapes du modèle WWHT (fission sémantique et allocation) ainsi que les propriétés pertinentes à la vérification de l'interface (sûreté, vivacité et utilisabilité) dans un modèle formel générique indépendant de toute notation formelle de manière à ce qu'il soit possible de le traduire dans une méthode formelle cible que le concepteur choisira pour valider ses spécifications.
   Le modèle formel est défini au moyen d'une syntaxe, d'une sémantique statique et dynamique. Le modèle WWHT a été enrichi par la définition de schémas de combinaison sémantique et d'ordonnancement temporel qui ont permis de désambigüiser les phases de fission sémantique et d'allocation.
2. **Une formalisation dans une méthode formelle cible :** cette seconde étape décrit la traduction du modèle formel générique ainsi que les propriétés à vérifier dans une méthode de spécification et de vérification formelles choisie par le concepteur. Nous proposons dans ce mémoire, une traduction du modèle générique dans la méthode B Événementiel. Ce choix est justifié d'une part, par le fait que la méthode B Événementiel repose sur une sémantique formelle permettant d'exprimer un large choix de propriétés pour un système interactif, et d'autre part, par le caractère événementiel de la méthode qui se prête à la modélisation des aspects de concurrence et de synchronisation. Enfin, le mécanisme de raffinement permettant la modélisation progressive du système et la preuve par théorème ne souffrant pas du problème de l'explosion combinatoire contribuent grandement à maitriser la complexité du processus de modélisation et à faciliter la phase de vérification.
   Ainsi, le modèle formel générique est entièrement formalisé en B Événementiel donnant lieu au modèle B Événementiel de fission sémantique et au modèle B Événementiel de l'allocation. Ces modèles sont instanciés sur des études de cas puis généralisés dans un processus de développement B Événementiel cadre dans lequel s'inscrivent les modèles spécifiques de fission sémantique et d'allocation.

Nous présentons notre proposition de manière plus détaillée dans les chapitres suivants. Le chapitre 3 présente l'approche de modélisation générique ainsi que le modèle formel que nous proposons, le chapitre 4 présente la démarche de modélisation B Événementiel adoptée ainsi que les modèles génériques de développement B Événementiel. Les chapitres 5 et 6 sont consacrés à la présentations détaillée des modèles B Événementiel de la fission sémantique et de l'allocation.

Chapitre 3

Un modèle formel générique pour les Interfaces Homme-Machine multimodales en sortie

1 Introduction

Le développement des interfaces homme-machine multimodales en sortie requiert l'utilisation de méthodes de développement adaptées. Ces méthodes doivent supporter le processus de génération de l'IHM multimodale en sortie à partir du moment où l'information est générée par le noyau fonctionnel jusqu'à sa présentation à l'utilisateur. Elles doivent modéliser la décomposition de l'information et la distribution de ses différentes composantes sur les média disponibles pour l'interface. Des modèles semi-formels tels que le modèle SRM orienté but de [80] et le modèle de conception WWHT de [158] ont été proposés pour spécifier les interfaces multimodales en sortie. Cependant, leur manque de formalisation implique qu'ils ne peuvent être intégrés dans un processus de développement formel d'un système interactif critique.

Nous proposons dans ce chapitre un modèle formel pour la conception des interfaces homme-machine multimodales en sortie. Ce modèle s'inspire du modèle WWHT, il reprend les deux premières étapes, fission sémantique et allocation qu'il enrichit avec des opérateurs de composition temporels et sémantiques. Ce modèle formel doté d'une syntaxe et de sémantiques statique et dynamique permet de spécifier sans ambigüité une interface multimodale en sortie. De plus, une fois plongé dans le modèle sémantique adopté, le modèle proposé permet la vérification de propriétés de robustesse et d'utilisabilité. Nous proposons dans les chapitres suivants une formalisation du modèle proposé dans un modèle sémantique à états basé sur la méthode B Évènementiel.

2 Démarche générale de modélisation

Nous proposons pour la modélisation des interfaces Homme-Machine multimodales, une approche formelle et générique basée sur le modèle WWHT. Il s'agit d'une démarche par transformations progressives qui repose sur les deux premières étapes du modèle WWHT : fission sémantique et allocation. Elle modélise le processus de conception des interfaces Homme-Machine multimodales dès que l'information est générée par le noyau fonctionnel jusqu'à la construction de la présentation multimodale à restituer à l'utilisateur. Une fois construite, cette présentation est instanciée par la détermination de ses caractéristiques opérationnelles. Ces caractéristiques sont : les contenus lexico-syntaxiques, par exemple, le contenu textuel à afficher dans le cas d'une modalité de type texte ou le fichier son employé pour une modalité de type alerte sonore, etc. et les attributs morphologiques, par exemple, la police et la taille des caractères employées pour afficher un texte ou le volume de la sonnerie, etc.

3 Modèle formel de conception des Interfaces Homme-Machine multimodales en sortie

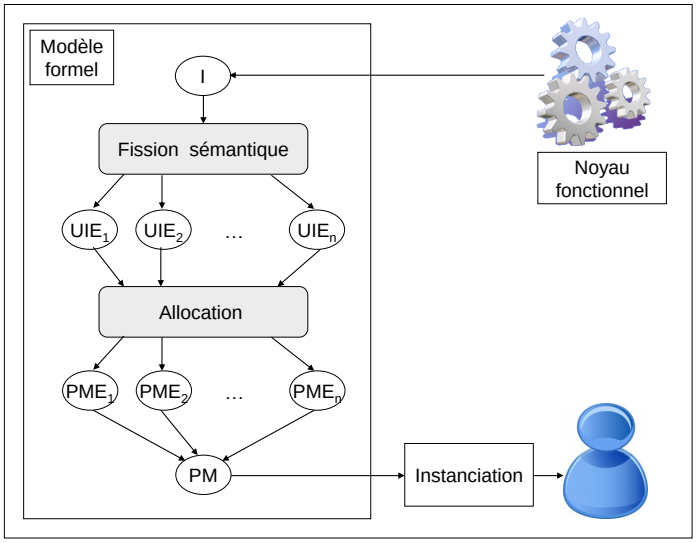

Figure 3.1 – Le modèle formel proposé pour la conception des IHM multimodales en sortie

L'approche que nous proposons s'articule autour d'un modèle formel qui capture les informations pertinentes lors du processus de construction de l'interface multimodale en

sortie. Le modèle formel proposé (voir Figure 3.1) modélise l'interface Homme-Machine multimodale à développer dans les différents stades de conception : information générée par le noyau fonctionnel, information fissionnée et présentation construite. Il est basé sur le modèle WWHT, dont il reprend les deux premières étapes donnant lieu à un modèle formel composé de deux autres modèles : le modèle de fission sémantique et le modèle d'allocation.

1. **Le modèle de fission sémantique :** exprime la décomposition de l'information à présenter à l'utilisateur en unités d'informations élémentaires. Le résultat de la phase de fission est une combinaison temporelle et sémantique des unités d'informations élémentaires.
2. **Le modèle d'allocation :** modélise la construction de la présentation à retourner à l'utilisateur par la construction des présentations multimodales élémentaires correspondantes aux informations fissionnées. Chaque information est allouée avec une présentation multimodale élémentaire qui est subdivisée par la suite en unités de présentations élémentaires. A chaque unité de présentation est alloué un couple (modalité, média) sélectionné pour produire l'information. Le résultat de la phase d'allocation est une combinaison temporelle et/ou sémantique des couples (modalité, média).

Suite à la phase d'allocation, la présentation multimodale obtenue modélisant l'interface multimodale en sortie est entièrement construite. Sa mise en œuvre en une interface concrète requiert la détermination des contenus lexico-syntaxiques et des attributs morphologiques des unités de présentation par la phase d'instanciation.

3.1 Étude de cas

Afin d'illustrer le modèle formel que nous proposons, nous utilisons un scénario d'interaction multimodale en sortie inspiré du système *SmartKom* [159]. Le scénario d'interaction multimodale sortie modélisé, que nous appelons *CityMap* (voir Figure 3.2), est issu d'un dialogue entre l'utilisateur et *Smartakus*, un agent conversationnel composant de l'interface multimodale en sortie. L'interaction en sortie survient en réaction à une demande de l'utilisateur d'afficher la carte de la ville de Heidelberg. L'agent conversationnel *Smartakus* répond à la demande de l'utilisateur par synthèse vocale : "Here you can see the map of the city", la carte de la ville de Heidelberg est affichée en même temps.

La réponse délivrée par le système *Smartkom* à l'utilisateur est calculée par le noyau fonctionnel de l'application. Elle est ensuite restituée à l'utilisateur par le biais de l'interface multimodale en sortie générée par le processus illustré en Figure 3.3. Il se compose des étapes

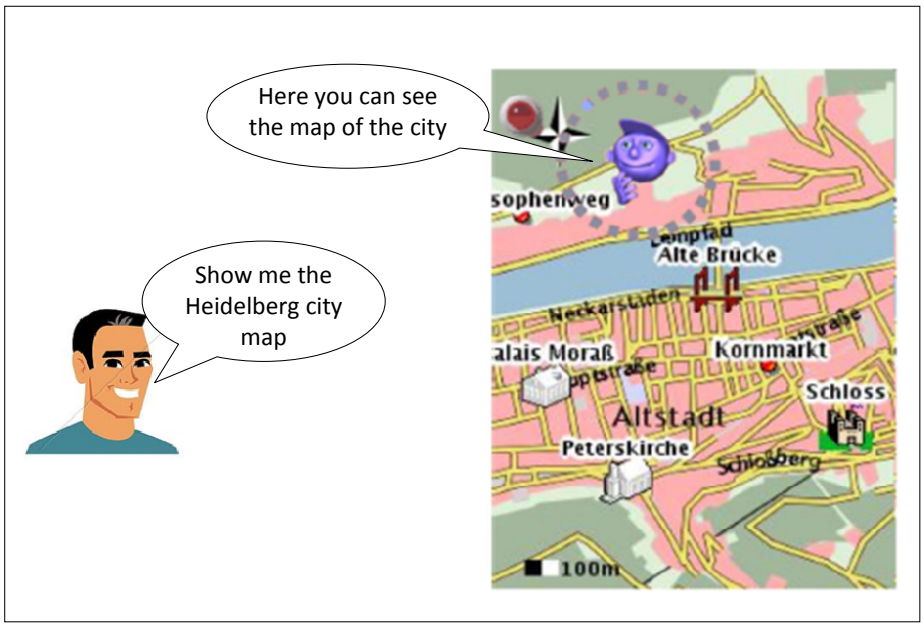

Figure 3.2 – Étude de cas : *CityMap*

suivantes.

1. **La phase de fission sémantique :** fissionne l'information *infoCityMap* qui exprime "voici la carte de la ville de Heidelberg" en deux informations parallèles et complémentaires : *infoSee* exprimant l'information "voici la carte de la ville " et *infoMap* décrivant la carte de la ville de Heidelberg. Ainsi, l'information *infoCityMap* est retournée à l'utilisateur par la production parallèle des informations complémentaires *infoSee* et *infoMap*.

2. **La phase d'allocation :** construit les présentations *presentSee* et *presentMap* correspondant respectivement à *infoSee* et *infoMap*. La présentation *presentSee* est décomposée en deux présentations complémentaires *presentSeeSpeech* produite par la parole sur le haut parleur et *presentSeeExpression* produite par des expressions faciales sur l'écran. La combinaison complémentaire des deux présentations : *presentSeeSpeech* et *presentSeeExpression* reproduit le discours de l'agent conversationnel *Smartakus*. La présentation *presentMap* est produite par l'affichage d'une image sur l'écran.

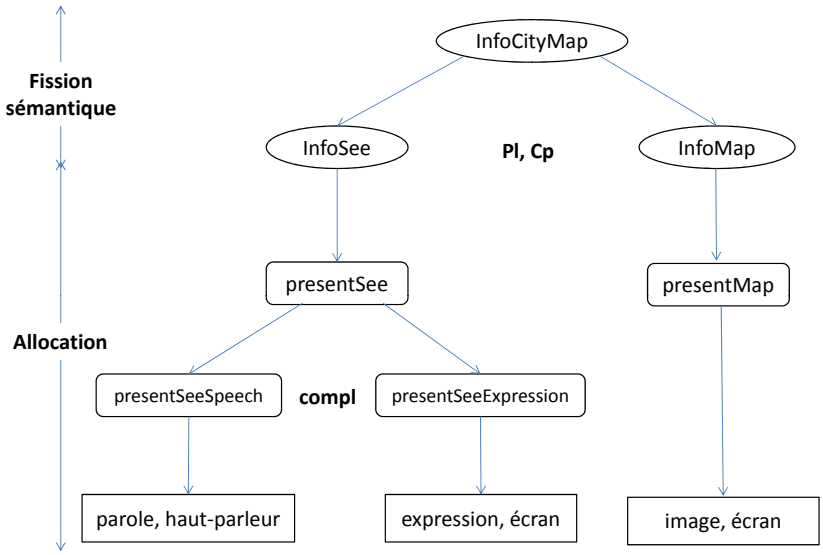

Figure 3.3 – Processus de modélisation de *CityMap*

3.2 Modèle de fission sémantique

Ce modèle exprime la fission ou la décomposition sémantique de l'information générée par le noyau fonctionnel en unités d'information élémentaires à présenter à l'utilisateur. Ces unités d'information élémentaires, seront par la suite, distribuées sur les différents média d'interaction en sortie, pour les combiner et les restituer à l'utilisateur en accord avec les choix faits par le concepteur de l'interface. Nous avons proposé au travers du modèle de fission sémantique, différentes possibilités en terme de fission d'informations, et d'ordonnancement temporel des informations fissionnées. La présentation du modèle comprend la description de la syntaxe et des sémantiques statique et dynamique.

3.2.1 Syntaxe

La syntaxe du modèle de fission met en relation l'information produite par le noyau fonctionnel avec les informations obtenues suite à la fission sémantique. Elle introduit, par conséquent, un ensemble d'opérateurs de composition pour les informations fissionnées. Ces opérateurs de composition caractérisent d'une part, les relations sémantiques existantes entre les informations fissionnées, et d'autre part, l'ordonnancement temporel des informations fissionnées lors de leur restitution.

Soit I l'ensemble des informations continues [1] à fissionner, et UIE l'ensemble des unités d'information élémentaires. La syntaxe de la fission sémantique est exprimée en grammaire BNF par la règle 3.1.

$$I ::= UIE \mid (op_{temp}, op_{sem})(I, I) \mid It(n, I) \; avec \; n \in \mathbb{N} \tag{3.1}$$

Où :

- op_{temp} est un opérateur binaire temporel appartenant à $TEMP$.
  $TEMP = \{An, Sq, Ct, Cd, Pl, Ch, In\}$
- op_{sem} est un opérateur binaire sémantique appartenant à SEM.
  $SEM = \{Cc, Cp, Cr, Pr, Tr\}$
- It est un opérateur temporel exprimant l'itération.

Les opérateurs de composition sont définis sur des traces d'évènements exprimant la production d'informations $i \in I$. Les opérateurs binaires temporels et sémantiques sont, en partie, inspirés des travaux de [160] sur les relations de composition temporelle, étendus par les travaux de [161] pour la composition sémantique.

$$op_{temp} : I \times I \rightarrow I$$
$$op_{sem} : I \times I \rightarrow I$$

Soit $op_{sem} \in SEM$ et soient i, j deux informations appartenant à I, alors les opérateurs op_{temp} sont définis comme suit.

- $(An,\ op_{sem})(i,\ j)$: i anachronique à j exprime que j survient après un intervalle de temps suite à la fin de i.
- $(Sq,\ op_{sem})(i,\ j)$: i séquentiel à j exprime que j survient juste à la fin de i.
- $(Cc,\ op_{sem})(i,\ j)$: i concomittant à j exprime que j survient après le début de i et qu'il se termine après i.
- $(Cd,\ op_{sem})(i,\ j)$: i coïncidant avec j exprime que j survient après le début de i et se termine avant la fin de i.
- $(Pl,\ op_{sem})(i,\ j)$: i parallèle à j exprime que i et j commencent et se terminent au même instant.
- $(Ch,\ op_{sem})(i,\ j)$: choix entre i et j exprime le choix déterministe entre i et j.
- $(In,\ op_{sem})(i,\ j)$: ordre indépendant entre i et j exprime que la relation temporelle entre i et j est indéterminée.

La démarche définie étant générique, nous ne connaissons pas l'effet des éléments de

1. Informations dont la restitution à l'utilisateur n'est pas instantanée (dure un temps non négligeable).

l'ensemble I, une fonction d'interprétation en précisera le sens. La connaissance de cette fonction permettra d'établir des propriétés relatives à cette interprétation.

Soit $op_{temp} \in TEMP$ et soient i, j deux informations appartenant à I, les opérateurs op_{sem} sont définis comme suit.

- $(op_{temp}, Cc)(i, j)$: i concurrent à j exprime que les interprétations de i et j sont indépendantes.
- $(op_{temp}, Cp)(i, j)$: i complémentaire à j exprime que les interprétations de i et j sont complémentaires sans être redondantes.
- $(op_{temp}, Cr)(i, j)$: i complémentaire et redondant à j exprime que les interprétations de i et j sont complémentaires et qu'une partie de chacune de leurs interprétations est redondante.
- $(op_{temp}, Pr)(i, j)$: i partiellement redondant à j exprime que l'interprétation de i est incluse entièrement dans l'interprétation de j.
- $(op_{temp}, Tr)(i, j)$: i totalement redondant à j exprime que les interprétations de i et j sont équivalentes.

La combinaison des opérateurs temporel et sémantique dans la règle syntaxique 3.1 suggère la combinaison possible de toute paire (op_{temp}, op_{sem}). Cependant, il apparaît que certaines combinaisons soient improbables voire interdites, d'autres contrairement sont conseillées. Nous donnons dans le Tableau 3.1 un ensemble de recommandations pour la combinaison de ces opérateurs.

op_{sem} \ op_{temp}	Cc	Cp	Cr	Pr	Tr
An	Possible	Possible	Possible	Possible	Possible
Sq	Possible	Souhaité	Souhaité	Possible	Possible
Ct	Possible	Souhaité	Souhaité	Possible	Possible
Cd	Possible	Souhaité	Souhaité	Possible	Possible
Pl	Possible	Souhaité	Souhaité	Possible	Possible
Ch	Interdit	Interdit	Interdit	Interdit	Souhaité
In	Possible	Possible	Possible	Possible	Possible

Tableau 3.1 – Recommandations pour les combinaisons des opérateurs temporels et sémantiques

Ainsi, l'utilisation des opérateurs sémantiques faisant intervenir la complémentarité tels que complémentaire et complémentaire et redondant est conseillée en combinaison avec les opérateurs de voisinage temporels tels que : séquentiel, concomitant, coïncidant et parallèle. De plus, l'opérateur temporel de choix qui implique l'exclusion d'une des informations fissionnées est combiné exclusivement à l'opérateur sémantique totalement redondant, cette

précaution évite la perte d'une partie de la sémantique de l'information fissionnée lors de la fission par choix.

L'itération est définie sur l'ensemble des entiers naturels et sur la trace finie d'évènements exprimant la production d'informations i.

$$It : \mathbb{N} \times I \rightarrow I$$

Soit i une information appartenant à I, et n un entier naturel supérieur ou égal à 1, $It(n, i)$ exprime la production itérative n fois de l'information i.

3.2.2 Sémantique statique

La sémantique statique du modèle de fission exprime les propriétés statiques de l'interface multimodale décrite par le modèle de la fission sémantique. Elle définit la durée de restitution d'une information i ainsi que son interprétation sémantique lors de l'interaction.
La durée de restitution d'une information i, est définie par l'introduction de ses bornes temporelles au moyen de la relation temporelle T.

Considérons $Time = \left\{t_j\right\}$ l'ensemble des instants discrets et les fonctions *start* et *end* définies sur I comme suit :

$$start : I \rightarrow Time \; \forall i \in I \; start(i) \text{ est l'instant de début de délivrance de } i.$$
$$end : I \rightarrow Time \; \forall i \in I \; end(i) \text{ est l'instant de fin de délivrance de } i.$$

Soit T la relation temporelle qui associe à chaque information l'instant de début et de fin de sa production.

$$T : I \rightarrow Time \times Time$$
$$\forall i \in I \; \exists \; (start(i), \; end(i)) \in Time \times Time \text{ tel que } start(i) < end(i)$$
$$\text{et } T(i) = (start(i), end(i))$$

L'interprétation sémantique d'une information est définie au moyen d'une fonction d'interprétation *int* qui permet d'associer à un élément d'information, l'interprétation sémantique le caractérisant lors de l'interaction. Elle est définie comme suit.

$$int : I \rightarrow D$$

Où D est le domaine d'interprétation associé aux informations. D est défini en fonction du système étudié, de l'utilisateur ou du concepteur de l'IHM. Ce domaine n'est pas précisé au niveau générique, il relève du noyau fonctionnel. Néanmoins, il est indispensable de définir la fonction d'interprétation pour établir les propriétés de complémentarité, redondance... des informations lors de la fission sémantique.

3.2.3 Sémantique dynamique

La sémantique dynamique du modèle de fission décrit les propriétés dynamiques de l'interface lors de la fission sémantique en utilisant les éléments décrits dans la sémantique statique. Elle décrit l'ordonnancement temporel des informations i ainsi que leurs relations sémantiques.
Les opérateurs temporels sont définis au moyen de la relation temporelle T comme suit.

$\forall op_{temp} \in \{An, Sq, Ct, Cd, Pl, Ch, In\}$

$\forall i, j \in I,\ avec\ T(i) = (start(i), end(i)), T(j) = (start(j), end(j))$

$\exists k \in I,\ avec\ k = (op_{temp}, op_{sem})(i, j)$ et :

$T(k) = (start(i), end(j))$ ssi $op_{temp} = An\ et\ end(i) < start(j)$

$T(k) = (start(i), end(j))$ ssi $op_{temp} = Sq\ et\ end(i) = start(j)$

$T(k) = (start(i), end(j))$ ssi $op_{temp} = Ct\ et\ start(i) < start(j) < end(i)$

$T(k) = (start(i), end(i))$ ssi $op_{temp} = Cd\ et\ start(i) < start(j) < end(j) < end(i)$

$T(k) = (start(i), end(i))$ ssi $op_{temp} = Pl\ et\ start(i) = start(j) \wedge end(i) = end(j)$

$k = i \vee k = j$ ssi $op_{temp} = Ch$

$T(k) = (start(k), end(k))$ ssi $op_{temp} = In$

L'opérateur temporel d'itération It est défini comme suit.

$It(0, i) = \delta$ où δ est l'information vide.

$It(1, i) = i$

$\forall i \in I,\ n > 1\ It(n, i) = Seq(It(n-1, i), i)$

Où Seq exprime la composition séquentielle définie au moyen de la relation temporelle T comme suit.

$\forall i,\ j \in I \times I$ avec $T(i) = (start(i), end(i)), T(j) = (start(j), end(j))$ et $end(i) = start(j)$

$\exists k \in I\ tel\ que\ k = Seq(i, j)$ et $T(k) = (start(i), end(j))$

La sémantique dynamique des opérateurs sémantiques op_{sem} ne peut être précisée à ce stade de la modélisation, elle requiert l'expression des relations de complémentarité, d'indépendance et de redondance sur le domaine D. Ces dernières sont formalisées à l'aide des éléments du modèle sémantique adopté lors de la modélisation de l'interface. Une modélisation de la sémantique dynamique des opérateurs sémantiques est proposée dans le chapitre5.

3.2.4 Application à l'étude de cas

Nous illustrons le modèle proposé pour la phase de fission sémantique en reprenant l'étude de cas relative à la carte de la ville de Heidelberg introduite en section 3.1. Pour cela, nous commençons par préciser le contexte de production de l'interface multimodale en sortie et qui consiste en la définition des différents ensembles : I, D, UIE.
I regroupe les informations intervenant dans la construction de l'interface CityMap. Il comprend les informations : $infoCityMap$ qui exprime *"voici la carte de la ville de Heidelberg"*, $infoSee$ qui exprime *"voici la carte de la ville"*, $infoMap$ qui exprime *"carte de Heidelberg"* et $emptyI$ qui exprime l'information vide.

$I = \{emptyI, infoCityMap, infoSee, infoMap\}$

D regroupe les interprétations sémantiques des informations de l'ensemble I. Il comprend les interprétations : $CityMap$ qui exprime la sémantique de $infoCityMap$, See qui exprime la sémantique de $infoSee$, Map qui exprime la sémantique de $infoMap$ et $emptyD$ qui exprime la sémantique de $emptyI$.

$D = \{emptyD, CityMap, See, Map\}$

UIE regroupe les unités d'information élémentaires intervenant dans la construction de l'interface CityMap et qui résultent du processus de fission sémantique. Il comprend les informations : $infoSee$ et $infoMap$.

$UIE = \{infoSee, infoMap\}$

Le processus de fission de l'information $infoCityMap$ (voir Figure 3.4) est exprimé par la formule suivante.

$$infoCityMap = (Pl, Cp)(infoSee, infoMap)$$

L'information $infoCityMap$ produite par le noyau fonctionnel, est fissionnée en deux informations complémentaires, restituées en parallèle $infoSee$ et $infoMap$, ce qui exprime que la restitution de l'information $infoCityMap$ est opérée par la production parallèle des informations complémentaires $infoSee$ et $infoMap$. Les informations $infoSee$ et $infoMap$ étant des unités d'information élémentaires, il n'est donc pas possible de les fissionner.

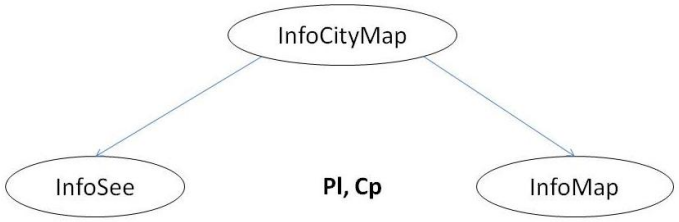

Figure 3.4 – Processus de fission de *CityMap*

3.3 Modèle d'allocation

Le modèle d'allocation proposé est basé sur la seconde étape du modèle WWHT. Il reprend les opérateurs proposés dans la phase d'allocation : opérateurs complémentaire et redondant, auxquels il ajoute les deux opérateurs : choix et itération. Ainsi, le modèle formel d'allocation formalise la construction de la présentation multimodale en sortie à restituer à l'utilisateur. Cette présentation est construite en trois étapes successives. Une première étape construit la présentation correspondant à l'information générée par le noyau fonctionnel, elle consiste en la combinaison des présentations multimodales correspondantes aux différentes informations élémentaires obtenues lors de la phase de fission sémantique. Une deuxième étape décompose les présentations élémentaires en unités de présentations élémentaires afin qu'en troisième étape, elles soient distribuées sur les différents média. La présentation multimodale ainsi construite correspond à une combinaison temporelle et sémantique des couples (modalité, média) affectés aux présentations élémentaires dans le but d'appliquer les choix d'utilisabilité faits pour l'IHM multimodale en sortie.

3.3.1 Syntaxe

La syntaxe du modèle d'allocation est décrite par trois règles syntaxiques. La première règle 3.2 définit la présentation multimodale globale (*pm*) correspondant à une information (*i*) en termes de présentations multimodales élémentaires (*pme*) correspondant aux informations fissionnées. Cette règle découle de la règle syntaxique de la fission sémantique, elle utilise deux règles qui expriment le collage[2] : 3.3 qui fait correspondre à une information (*i*) la présentation multimodale qui la restitue (*pm*) et 3.4 qui fait correspondre à une information

2. Application de type morphisme qui permet de faire correspondre à tout élément d'un premier ensemble, un élément d'un deuxième ensemble.

élémentaire (*uie*) la présentation multimodale élémentaire qui la restitue (*pme*).

$$PM ::= PME \mid (op'_{temp}, op'_{sem})(PM, PM) \mid It'(n, PM) \; avec \; n \in \mathbb{N} \tag{3.2}$$

Où :

- op'_{temp} est un opérateur binaire temporel appartenant à $TEMP'$.
  $TEMP' = \{An', Sq', Ct', Cd', Pl', Ch', In'\}$.
- op'_{sem} est un opérateur binaire sémantique appartenant à SEM'.
  $SEM' = \{Cc', Cp', Cr', Pr', Tr'\}$.
- It' est un opérateur temporel binaire exprimant l'itération.

$$PM ::= allocation(I) \tag{3.3}$$

$$PME ::= allocation(UIE) \tag{3.4}$$

Les opérateurs binaires temporels et sémantiques sont définis sur des traces d'évènements exprimant la production de présentations multimodales $pm_i \in PM$.

$$op_{temp'} : PM \times PM \rightarrow PM$$
$$op_{sem'} : PM \times PM \rightarrow PM$$

Soit $op'_{sem} \in SEM'$ et soient i, j deux présentations multimodales appartenant à PM, alors les opérateurs op'_{temp} sont définis comme suit.

- $(An', op'_{sem})(i, j)$: i anachronique à j exprime que j survient après un intervalle de temps suite à la fin de i.
- $(Sq', op'_{sem})(i, j)$: i séquentiel à j exprime que j survient juste à la fin de i.
- $(Ct', op'_{sem})(i, j)$: i concomittant à j exprime que j survient après le début de i et qu'il se termine après i.
- $(Cd', op'_{sem})(i, j)$: i coïncidant avec j exprime que j survient après le début de i et se termine avant la fin de i.
- $(Pl', op'_{sem})(i, j)$: i parallèle à j exprime que i et j commencent et se terminent au même instant.
- $(Ch', op'_{sem})(i, j)$: choix entre i et j exprime le choix déterministe entre i et j.
- $(In', op'_{sem})(i, j)$: ordre indépendant entre i et j exprime que la relation temporelle entre i et j est indéterminée.

La définition de relations sémantiques entre les présentations multimodales implique l'introduction d'une interprétation sémantique des présentations multimodales à l'image

des informations qu'elles restituent. Ainsi, une fonction d'interprétation est définie pour l'ensemble *PM*.

Soit $op'_{temp} \in TEMP'$ et soient i, j deux présentations multimodales appartenant à *PM*, alors les opérateurs op'_{sem} sont :

- $(op'_{temp}, Cc')(i, j)$: i concurrent à j exprime que les interprétations de i et j sont indépendantes.
- $(op'_{temp}, Cp')(i, j)$: i complémentaire à j exprime que les interprétations de i et j sont complémentaires sans être redondantes.
- $(op'_{temp}, Cr')(i, j)$: i complémentaire et redondant à j exprime que les interprétations de i et j sont complémentaires et qu'une partie de chacune de leurs interprétations est redondante.
- $(op'_{temp}, Pr')(i, j)$: i partiellement redondant à j exprime que l'interprétation de i est incluse entièrement dans l'interprétation de j ou bien que l'interprétation de j est entièrement incluse dans l'interprétation de i.
- $(op'_{temp}, Tr')(i, j)$: i totalement redondant à j exprime que les interprétations de i et j sont équivalentes.

L'itération est définie sur l'ensemble des entiers naturels et sur la trace finie d'évènements exprimant la production de présentations multimodales $i \in PM$.

$$It' : \mathbb{N} \times PM \rightarrow PM$$

Soit *pm* une information appartenant à *PM*, et n un entier naturel supérieur ou égal à 1, $It'(n, pm)$ exprime l'itération n fois de la présentation *pm*.

La seconde règle 3.5 décrit la décomposition des présentations multimodales élémentaires (*pme*), obtenues par la règle 3.4. Chaque présentation élémentaire correspond à une unité d'information élémentaire (*uie*), elle est décomposée en unités de présentation élémentaires (*upe*). Soit *PME* l'ensemble des présentations multimodales élémentaires, et *UPE* l'ensemble des unités de présentation élémentaires.

$$PME ::= UPE \mid compl(UPE, PME) \mid redun(UPE, PME) \mid choice(UPE, PME) \mid iter(n, PME)\, avec\, n \in N \tag{3.5}$$

Où

- *Compl* exprime la complémentarité représentationnelle entre deux présentations multimodales élémentaires pour la restitution d'une unité d'information élémentaire *uie* ;
- *redun* exprime la redondance représentationnelle entre deux présentations multimodales élémentaires pour la restitution d'une unité d'information élémentaire *uie* ;

– *choice* exprime le choix représentationnel entre deux présentations multimodales élémentaires pour la restitution d'une unité d'information élémentaire *uie* ;
– *Iter* exprime l'itération *n* fois d'une présentation multimodale élémentaire *pme*.

Les opérateurs de combinaison : *compl*, *redun*, *choice*, *Iter* sont définis sur des traces d'évènements exprimant la production des présentations multimodales élémentaires $pme \in PME$.

$$compl, redun, choice : PME \times PME \rightarrow PME$$
$$Iter : \mathbb{N} \times PME \rightarrow PME$$

Une fonction d'interprétation représentationnelle définie sur *PME* est introduite en sémantique statique. Elle définit la signification représentationnelle d'une présentation élémentaire pour produire une information élémentaire et permet d'exprimer précisément les opérateurs qui combinent les présentations multimodales élémentaires.

La troisième règle 3.6 décrit l'attribution des paires modalité/média (mod, med) aux unités de présentation élémentaires obtenues lors de la décomposition en règle 3.5.

Considérons *MOD* l'ensemble des modalités en sortie et *MED* l'ensemble des médias en sortie. Soit la relation *rest* qui détermine si une modalité peut être restituée par un média.

$$rest : MOD \times MED \rightarrow \{true,\ false\}$$

$\forall mod_i \in MOD, \forall med_j \in MED,\ rest(mod_i,\ med_j) = true$ exprime que mod_i peut être restituée par med_j.

Considérons *ITEM* l'ensemble des couples (mod, med), tels que la modalité *mod* est restituée par le média *med*.

$ITEM = \left\{(mod_i, med_j)\ tel\ que\ mod_i \in MOD \wedge med_j \in MED \wedge rest(mod_i,\ med_j)\right\}$

Et soit la fonction $affectation$ qui associe à une unité de présentation élémentaire *upe* un couple (modalité, média) dans *ITEM*.

$$affectation : UPE \rightarrow ITEM$$

L'allocation des modalités et médias aux unités de présentations élémentaires est exprimée par la règle BNF suivante.

$$affectation(UPE) ::= ITEM \tag{3.6}$$

3.3.2 Sémantique statique

La sémantique statique du modèle d'allocation exprime la durée de production des présentations multimodales *pm* et *pme* par la définition de leurs bornes temporelles, respectivement, au moyen des relations temporelles T' et T''.

Considérons les fonctions *start*′ et *end*′ définies sur *PM* comme suit.

start′ : $PM \longrightarrow Time$ $\forall i \in PM$, *start*′(i) est l'instant de début de i.

end′ : $PM \longrightarrow Time$ $\forall i \in PM$, *end*′(i) est l'instant de fin de i.

Soit T' la relation temporelle qui associe à chaque présentation multimodale son instant de début et de fin.

$$T' : PM \longrightarrow Time \times Time$$

$$\forall i \in PM \; \exists \; (start'(i), end'(i)) \; \in \; Time \times Time \text{ tel que}$$
$$start'(i) < end'(i) \text{ et } T'(i) = (start'(i), end'(i))$$

Considérons les fonctions *start*″ et *end*″ définies sur *PME* comme suit.

start″ : $PME \longrightarrow Time$ $\forall i \in PME$, *start*″(i) est l'instant de début de i.

end″ : $PME \longrightarrow Time$ $\forall i \in PME$, *end*″(i) est l'instant de end de i.

Soit T'' la relation temporelle qui associe à chaque présentation multimodale élémentaire son instant de début et de fin.

$$T'' : PME \longrightarrow Time \times Time$$

$\forall i \in PME \; \exists \; (start''(i), end''(i)) \; \in \; Time \times Time$

tel que $start''(i) < end''(i)$ et $T''(i) = (start''(i), end''(i))$

La sémantique statique définit également les fonctions d'interprétation : une fonction d'interprétation sémantique *int*′ pour *PM* à l'image de la fonction d'interprétation *int* pour I et une fonction d'interprétation représentationnelle *repres* pour exprimer l'interprétation représentationnelle d'une présentation multimodale élémentaire *pme* lorsqu'elle restitue une information *uie*.

$$int' : PM \rightarrow D'$$

Où D' est le domaine d'interprétation associé aux présentations multimodales. D' est défini en fonction du contexte de l'IHM ou de son concepteur.

Soit la fonction d'interprétation représentationnelle des présentations multimodales élémentaires *repres* définie comme suit.

$$repres : PME \rightarrow DR$$

Où *DR* est le domaine d'interprétation représentationnelle associé aux présentations multimodales élémentaires.

Il existe des contraintes d'utilisabilité de l'interface en sortie qui peuvent être liées à : l'environnement (par exemple, le bruit qui détourne de l'utilisation du mode vocal), le système (l'utilisation d'un PDA plaide en faveur de la modalité courbe plutôt que tableau) et l'utilisateur (un utilisateur mal-entendant favorisera le mode visuel au mode auditif). Dans le but de décrire ces contraintes, nous introduisons un ensemble de propriétés statiques définies sur les éléments syntaxiques du modèle permettant de décrire les propriétés d'utilisabilité de l'IHM.

- Le mode utilisé pour un média : $mode(med_i) \in \{visuel,\ auditif,\ TPK\}$. Par exemple, mode (haut-parleur) = vocal, mode (histogramme) = visuel.
- La partageabilité du média qui exprime la possibilité de partager ou pas le média entre plus d'une modalité de même ou de différentes natures évitant ainsi les collisions [3] sur un média : $partag(med_i) \in \{true, false\}$.
  Par exemple, $partag(écran) = true$, $partag(haut-parleur) = false$.
- Les priorités entre modalités : $priorite(mod_i) \in \mathbb{N}$.
  Par exemple, priorite(parole) > priorite(texte).
- Les priorités entre modes : $priorite(mode_k) \in \mathbb{N}$.
  Par exemple, priorite(visuel) > priorite(auditif) (cas d'un environnement bruyant).

3.3.3 Sémantique dynamique

La sémantique dynamique du modèle d'allocation décrit l'ordonnancement temporel des présentations multimodales *pm* et des *pme*. Elle définit les opérateurs *An'*, *Sq'*, *Ct'*, *Cd'*, *Pl'*, *Ch'*, *In'*, *It'* à l'aide de la relation *T'* pour les présentations *pm* et *T''* pour les présentations *pme*.

$\forall i \in PM, \forall j \in PM,\ avec\ T'(i) = (start'(i), end'(i)), T'(j) = (start'(j), end'(j))$

$\exists k \in PM$, avec $k = (op'_{temp}, op'_{sem})(i, j)$ et :

$T'(k) = (start'(i), end'(j))$ ssi $op'_{temp} = An'$ et $end'(i) < start'(j)$

$T'(k) = (start'(i), end'(j))$ ssi $op'_{temp} = Sq'$ et $end'(i) = start'(j)$

$T'(k) = (start'(i), end'(j))$ ssi $op'_{temp} = Ct'$ et $start'(i) < start'(j) < end'(i)$

$T'(k) = (start'(i), end'(i))$ ssi $op'_{temp} = Cd'$ et $start'(i) < start'(j) < end'(j) < end'(i)$

$T'(k) = (start'(i), end'(j))$ ssi $op'_{temp} = Pl'$ et $start'(i) = start'(j) \wedge end'(j) = end'(i)$

3. La représentation sur une même fenêtre temporelle de deux modalités sur un média non partageable comme de présenter un bip sonore avec une phrase produite par synthèse vocale.

$k = i \vee k = j$ ssi $op'_{temp} = Ch'$
$T'(k) = (start'(k), end'(k))$ ssi $op'_{temp} = In'$
L'itération sur les présentation multimodales it', est définie comme suit.

$It'(0, pm) = \lambda$ où λ est la présentation vide.
$It'(1, pm) = pm$
$\forall pm \in PM,\ n > 1\ It'(n, pm) = Seq'(It'(n-1, pm), pm)$

Où Seq' exprime la composition séquentielle définie au moyen de la relation temporelle T' comme suit.

$$\forall i,\ j \in PM \times PM \text{ avec } T'(i) = (start'(i), end'(i)), T'(j) = (start'(j), end'(j)) \text{ et } end'(i) = start'(j)$$
$$\exists k \in PM \text{ tel que } k = Seq'(i, j) \text{ et } T'(k) = (start(i), end(j))$$

De la même manière, l'opérateur *iter* est défini sur PME en utilisant l'opérateur séquentiel Seq''.
$\forall i \in PME,\ n \in \mathbb{N}$

$Iter(0, pme) = \lambda$ où λ est la présentation vide.
$Iter(1, pme) = pme$
$\forall pme \in PME,\ n > 1\ Iter(n, pme) = Seq''(Iter(n-1, pme), pme)$

Où l'opérateur séquentiel Seq'' est défini au moyen de la relation temporelle T'' comme suit :

$$\forall i,\ j \in PME \times PME \text{ avec } T''(i) = (start''(i), end''(i)), T''(j) = (start''(j), end''(j)) \text{ et}$$
$$end''(i) = start''(j)$$
$$\exists k \in PME \text{ tel que } k = Seq(i, j) \text{ et } T''(k) = (start''(i), end''(j))$$

L'opérateur de *choice* est défini comme suit.

$$\forall i,\ j \in PME \times PME\ \exists k \in PME \text{ tel que}$$
$$k = choice(i, j)\ \Rightarrow k = i \vee k = j$$

La sémantique dynamique des opérateurs *compl* et *redun* est liée aux fonctions d'interprétation *int'* et *repres*. Elle ne peut être explicitement exprimée que si les domaines d'interprétation D' et DR sont formellement définis.

3.3.4 Application à l'étude de cas

Nous illustrons le modèle d'allocation par l'exemple de la ville de Heidelberg précédemment introduit. Nous commençons d'abord par introduire l'environnement du processus d'allocation décrit par les ensembles *PM*, *PME*, *UPE* et *ITEM*.
PM regroupe les présentations intervenant dans la construction de l'interface CityMap. Il comprend les présentations : *presentCityMap* qui restitue l'information $infoCityMap$, *presentSee* qui restitue $infoSee$, *presentMap* qui restitue $infoMap$ et *emptyP* représentant la présentation vide.

$PM = \{emptyP, presentCityMap, presentSee, presentMap\}$

PME regroupe les présentations élémentaires intervenant dans la construction de l'interface CityMap. Il comprend les présentations qui restituent les informations produites lors de la fission sémantique : *presentSee* qui restitue $infoSee$ et *presentMap* qui restitue $infoMap$.

$PME = \{presentSee, presentMap\}$

UPE regroupe les unités de présentation élémentaires intervenant dans la construction de l'interface CityMap. Il comprend les présentations issues de la décomposition des présentations : *presentSeeSpeech*, *presentSeeExpression* et *presentMap*.

$UPE = \{presentSeeSpeech, presentSeeExpression, presentMap\}$

ITEM regroupe l'ensemble des couples (modalité, média) disponibles sur le système multimodal tel que la modalité est restituée par le média.

$ITEM = \{(parole, haut-parleur), (expression, écran), (image, écran)\}$

Le processus d'allocation (voir Figure 3.5) consiste à construire la présentation multimodale globale *presentCityMap*, correspondante à l'information *inoCityMap*. Il exploite pour cela, les informations fissionnées résultantes de la phase de fission sémantique décrite dans la section 3.2.4. Dans un premier temps, il construit les présentations multimodales élémentaires *presentSee* et *presentMap* correspondant à $infoSee$ et $infoMap$ moyennant la fonction *allocation*, ceci exprime que l'informtion $infoSee$ est restituée par la présentation *presentSee* et que l'informtion $infoMap$ est restituée par la présentation *presentMap*. Elles sont combinées de manière analogue aux informations qu'elles restituent (parallèle et complémentaire), ce qui signifie que la restitution de la présentation *presentCityMap* se fait par la production parallèle des présentations complémentaires *presentSee* et *presentMap*.

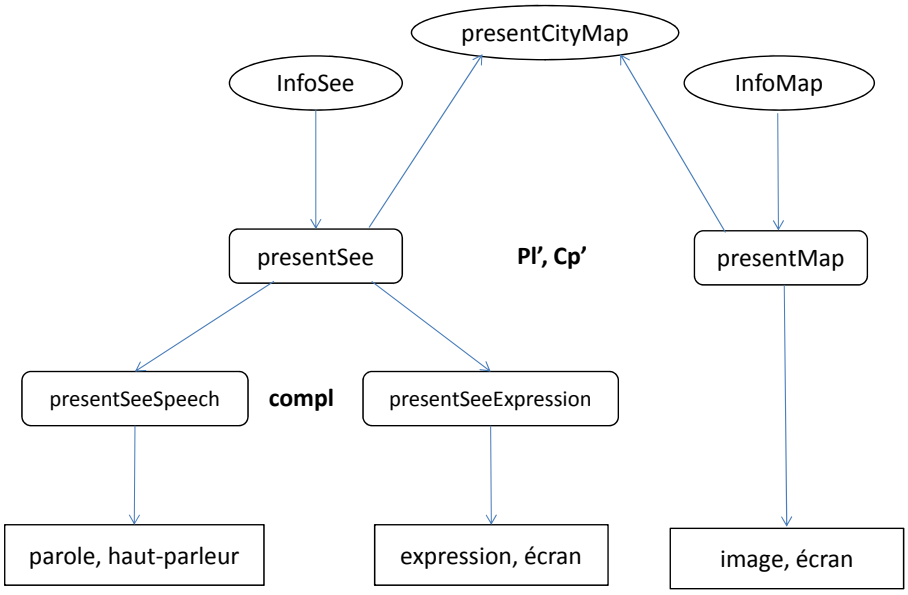

Figure 3.5 – Processus d'allocation de *CityMap*

$$presentCityMap = (Pl', Cp')(presentSee, presentMap)$$

Où :

$$presentSee = allocation(infoSee)$$
$$presentMap = allocation(infoMap)$$

Dans un deuxième temps, il décompose la présentation élémentaire *presentSee* en deux unités de présentation élémentaires complémentaires : *presentSeeSpeech* et *presentSeeExpression* ce qui signifie que les présentations *presentSeeSpeech* et *presentSeeExpression* sont complémentaires pour produire *presentSee*. La présentation *presentMap* étant unitaire, elle n'est pas décomposée.

$$presentSee = compl(presentSeeSpeech, presentSeeExpression)$$

Enfin, les unités de présentation élémentaires *presentSeeSpeech*, *presentSeeExpression* et *presentMap* sont allouées respectivement avec les couples (modalité, média) qui les restituent : *presentSeeSpeech* est produite par la parole sur le haut-parleur, *presentSeeExpression* est restituée par expression faciale sur l'écran et *presentMap* est restituée par une image sur l'écran.

$$affectation(presentMap) = (image, écran)$$
$$affectation(presentMapSpeech) = (parole, haut - parleur)$$
$$affectation(presentMapExpression) = (expression, écran)$$

3.4 Modélisation de l'espace CASE

Le modèle formel de l'espace CASE s'obtient à partir du modèle formel de conception générique précédemment décrit, en définissant les syntaxes relatives aux quatre types de multimodalités de l'espace. Nous commençons par définir les opérateurs temporels et sémantiques autorisés pour chacune des deux valeurs des deux axes de l'espace (usage des médias et lien entre les informations), puis nous donnons la grammaire de la syntaxe des quatres types résultant du croisement des deux axes (Concurrent, alterné, synergique et exclusif).

- **Usage des médias :** cet axe se rapporte à l'ordonnancement temporel et de ce fait, conditionne l'utilisation des opérateurs temporels qui combinent les présentations.
  - Séquentiel : réduit la grammaire aux opérateurs temporels ne faisant pas appel au parallélisme : anachronique, séquentiel, choix et itération.
  - Parallèle : limite la grammaire aux opérateurs temporels faisant appel aux différents types de parallèlisme : concomittant, coïncidant et parallèle.
- **Lien entre les informations :** cet axe se rapporte à la relation sémantique entre les informations véhiculées par les présentations et de ce fait, conditionne les opérateurs sémantiques.
  - Combiné : réduit les opérateurs sémantiques aux opérateurs : complémentaire, complémentaire et redondant, partiellement redondant et totalement redondant.
  - Indépendant : limite les opérateurs sémantiques à l'opérateur concurrent.

Ainsi, les quatre types de multimodalités peuvent être définis.

Type concurrent : en utilisant les opérateurs temporels concomittant, coïncidant et parallèle combinés avec l'opérateur sémantique concurrent.

$$PM ::= PME \mid (op'_{temp}, op'_{sem})(PM, PM) \text{ avec } op'_{temp} \in \{Ct', Cd', Pl'\} \text{ et } op'_{sem} \in \{Cc'\}$$

Type alterné : avec les opérateurs temporels anachronique, séquentiel, choix et itération combinés avec les opérateurs sémantiques complémentaire, complémentaire et redondant, partiellement redondant et totalement redondant.

$$PM ::= PME \mid (op'_{temp}, op'_{sem})(PM, PM) \mid It'(n, PM) \; avec \; n \in \mathbb{N} \text{ avec } op'_{temp} \in \{An', Sq', Ch'\}$$
$$\text{et } op'_{sem} \in \{Cp', Cr', Pr', Tr'\}$$

Type synergique : en utilisant les opérateurs temporels concomittant, coïncidant et parallèle combinés avec les opérateurs sémantiques : complémentaire, complémentaire et redondant, partiellement redondant et totalement redondant.

$$PM ::= PME \mid (op'_{temp}, op'_{sem})(PM, PM) \text{ avec } op'_{temp} \in \{Ct', Cd', Pl'\} \text{ et } op'_{sem} \in \{Cp', Cr', Pr', Tr'\}$$

Type exclusif : avec les opérateurs temporels anachronique, séquentiel, choix et itération combinés avec l'opérateur concurrent.

$$PM ::= PME \mid (op'_{temp}, op'_{sem})(PM, PM) \mid It'(n, PM) \textit{ avec } n \in \mathbb{N} \text{ avec } op'_{temp} \in \{An', Sq', Ch'\} \text{ et } op'_{sem} \in \{Cc'\}$$

3.5 Modélisation des propriétés CARE

Nous proposons d'exprimer les propriétés CARE par des modèles ou patrons de conception analogues à ceux proposés pour l'espace CASE. Ces modèles s'appuyent sur le modèle générique de conception. Chaque modèle est décrit au moyen d'une syntaxe. L'application d'un modèle garantit le respect de la propriété CARE correspondante à ce modèle.

- **La complémentarité :** une présentation p est complémentaire si elle fait intervenir deux présentations complémentaires p_i et p_j (combinées par un des opérateurs : Cp', Cr', $compl$), et que p_i et p_j ou les présentations qui les composent utilisent des modalités distinctes (deux modalités distinctes au moins).

$$p ::= op(p_i, p_j) \textit{ où } op \in \left\{(op'_{temp}, Cp' \mid Cr'), compl\right\}$$
$$p_i ::= op'(q_i, p_i) \; ; \; p_j ::= op'(q_j, p_j) \textit{ où } op' \in \left\{(op'_{sem}, op'_{temp}), compl, redun, choice\right\}$$
$$prj1(affect(q_i)) \neq prj1(affect(q_j))$$

- **L'assignation (spécialisation) :** une présentation p est assignée si elle, ou les présentations qui la composent, font intervenir une même modalité.

$$p ::= op(q, p) \textit{ où } op \in \left\{(op'_{sem}, op'_{temp}), compl, redun, choice\right\}$$
$$prj1(affect(q)) = prj1(affect(p))$$

– **La redondance :** une présentation p est redondante si elle fait intervenir deux présentations redondantes p_i et p_j (combinées par l'opérateur Tr'), et que p_i et p_j ou les présentations qui les composent utilisent des modalités distinctes (deux modalités distinctes au moins).

$$p ::= op(p_i, p_j) \text{ où } op \in \left\{(op'_{temp}, Tr'), redun\right\}$$
$$p_i ::= op'(q_i, p_i)\ ;\ \ p_j ::= op'(q_j, p_j) \text{ où } op' \in \left\{(op'_{sem}, op'_{temp}), compl, redun, choice\right\}$$
$$prj1(affect(q_i)) \neq prj1(affect(q_j))$$

– **L'équivalence :** une présentation p est équivalente si elle fait intervenir deux présentations au choix p_i et p_j (combinées par un des opérateur Ch', $choice$), et que p_i et p_j ou les présentations qui les composent utilisent des modalités distinctes (deux modalités distinctes au moins).

$$p ::= op(p_i, p_j) \text{ où } op \in \left\{(op'_{temp}, ch'), choice\right\}$$
$$p_i ::= op'(q_i, p_i)\ ;\ \ p_j ::= op'(q_j, p_j) \text{ où } op' \in \left\{(op'_{sem}, op'_{temp}), compl, redun, choice\right\}$$
$$prj1(affect(q_i)) \neq prj1(affect(q_j))$$

4 Conclusion

Nous avons présenté dans ce chapitre le modèle formel que nous proposons pour la conception des Interfaces Homme-Machine multimodales. Nous avons voulu à travers ce modèle lever les ambigüités posées par le modèle semi-formel WWHT, et ce par la définition de manière précise de différents schémas de composition et de combinaison temporelle et sémantique, lors des étapes de fission sémantique et d'allocation. Nous estimons que ces opérateurs sont suffisants pour exprimer tous les choix en matière de composition pour la conception d'une interface multimodale en sortie.

Notre modèle doté d'une syntaxe et de sémantiques statique et dynamique permet de spécifier l'interface multimodale tout au long du processus de sa construction. Nous nous sommes particulièrement appliqués à ce que le modèle que nous proposons soit générique. En effet, il peut être instancié sur des études de cas, il est paramétré par les fonctions d'interprétation sémantique des informations et des présentations permettant ainsi de modéliser l'interface qui restitue une information quelque soit son interprétation sémantique. Enfin, il

modélise plusieurs espaces de conception grâce aux contraintes que l'on peut imposer sur les règles syntaxiques qui le définissent. Nous présentons les définitions de l'espace CASE et des propriétés CARE selon le modèle que nous proposons.

Les chapitres suivants présentent la seconde partie de notre proposition qui correspond à la formalisation de ce modèle dans B Évènementiel. Le chapitre 4 propose un prolongement de cette formalisation dans la méthode B Évènementiel par la proposition d'un modèle B Évènementiel générique et les chapitres 5 et 6 présentent en détail les développements B Évènementiel relatifs respectivement aux étapes de fission sémantique et d'allocation.

Chapitre 4

Modélisation des Interfaces Homme-Machine multimodales en sortie dans B Évènementiel

1 Introduction

Dans le chapitre précédent, nous avons présenté un modèle formel générique de conception pour les IHM multimodales en sortie. Ce modèle doté d'une syntaxe et d'une sémantique statique et dynamique, permet de spécifier sans ambiguïté l'interface multimodale en sortie. Cependant, un tel modèle ne peut être exploité concrètement pour le développement d'une IHM multimodale en sortie. Ainsi, un plongement de ce modèle conceptuel dans un système sémantique cible est nécessaire. Ce plongement permet d'une part, de concrétiser les éléments syntaxiques et la sémantique statique du modèle, et d'autre part de, formaliser certains éléments de la sémantique dynamique qui requièrent le prolongement du modèle initial par un système sémantique adéquat.

Dans ce chapitre, nous présentons une formalisation du modèle conceptuel dans la méthode B Évènementiel. Nous décrivons la démarche générale de modélisation ainsi que les règles de formalisation qui permettent de développer les modèles B Événementiel génériques correspondant au modèle de conception proposé dans le chapitre 3.

2 La méthode B Évènementiel

B Événementiel [162] est une méthode formelle basée sur la logique du premier ordre et la théorie des ensembles, il s'agit d'une évolution de la méthode B [101] qui repose sur les travaux de Hoare [118] sur les pré-conditions et post-conditions, et de Dijkstra [163] sur

la plus faible pré-condition. Un modèle B Événementiel décrit un système états/transitions où les variables représentent l'état du système et les évènements représentent les transitions d'un état à un autre. Un modèle est caractérisé par l'ensemble des séquences (traces) d'événements autorisés sous couvert des propriétés énoncées. La description du modèle B Événementiel est accompagnée de la génération automatique d'obligations de preuves qui doivent être déchargées afin d'assurer la preuve de correction du modèle.
Le processus de modélisation de B Événementiel est incrémental. Il débute par le développement du modèle abstrait du système qui évolue progressivement vers un modèle concret par l'ajout de détails de conception à travers les étapes successives de raffinement [164]. Le raffinement préserve les propriétés prouvées dans le modèle abstrait. Par conséquent, il n'est pas nécessaire de les prouver une nouvelle fois au niveau du modèle obtenu par raffinement. Le processus de développement B Événementiel est supporté par la plate-forme Rodin [165]. Un modèle B Événementiel est constitué de deux parties, une partie dite statique, décrite dans le composant CONTEXT et une partie dite dynamique, décrite dans le composant MACHINE. Un modèle peut contenir uniquement des contextes, uniquement des machines, ou les deux à la fois. Dans le premier cas, le modèle représente une structure mathématique pure. Dans le deuxième cas, le modèle représente un modèle non paramétré. Enfin dans le troisième cas, le modèle est paramétré par les contextes. Les machines et les contextes ont plusieurs relations : une machine peut être raffinée par une autre machine, un contexte peut être étendu par un autre contexte, et une machine peut importer un ou plusieurs contextes. La Figure 4.1 présente dans le cas général, les deux composants avec leurs différentes clauses introduites par des mots-clés [166].

CONTEXT <identifiant_contexte> **EXTENDS** <liste_identifiant_contextes> **SETS** <liste_identifiant_ensembles> **CONSTANTS** <liste_identifiant_constantes> **AXIOMS** <label> : <prédicat> ... **THEOREMS** <label> : <prédicat> ... **END**	**MACHINE** <identifiant_machine> **REFINES** <identifiant_machine> **SEES** <liste_identifiant_contextes> **VARIABLES** <liste_identifiant_variables> **INVARIANTS** <label> : <prédicat> ... **THEOREMS** <label> : <prédicat> ... **VARIANT** <variant> **EVENTS** <liste_événements> **END**

Figure 4.1 – Structure d'un modèle B Événementiel

2.1 Le composant CONTEXT

Le composant CONTEXT formalise la partie statique du système modélisé. Généralement, le CONTEXT contient les définitions des types abstraits et des constantes ainsi que les propriétés liées aux constantes. Il est composé des clauses suivantes :

- CONTEXT représentant le nom du composant, unique dans un modèle.
- EXTENDS déclarant la liste des contextes qu'étend le contexte décrit.
- SETS définissant un ensemble de types abstraits ou d'ensembles énumérés.
- CONSTANTS décrivant les noms de constantes utilisées par le modèle.
- AXIOMS exprimant des propriétés liées aux constantes à l'aide d'expressions de la logique du premier ordre. Il s'agit des hypothèses considérées lors du processus de preuve.
- THEOREMS exprimant un ensemble de propriétés qui peuvent être déduites à partir des axiomes.

2.2 Le composant MACHINE

Le composant MACHINE formalise la partie dynamique du système modélisé (son comportement). Elle définit l'état du système, ses propriétés et les événements qui le font évoluer d'un état à un autre. La MACHINE est composée par les clauses suivantes :

- MACHINE représentant le nom du composant qui devrait être unique dans un modèle.
- REFINES déclarant le nom de la machine éventuellement raffinée par la machine décrite.
- SEES déclarant les contextes référencés explicitement par la machine. La machine peut utiliser les ensembles et variables définis dans les contextes référencés explicitement ou implicitement (contextes étendus par le contexte référencé).
- VARIABLES décrivant les noms des variables du modèle (état).
- INVARIANTS exprimant les propriétés invariantes du modèle à l'aide de prédicats de la logique du premier ordre, ces propriétés consistent en des propriétés de typage des variables ainsi que les propriétés de sûreté du modèle. Les invariants doivent être préservés par les événements de la clause EVENTS et vérifiées dans la machine et ses raffinements.
- THEOREMS exprimant un ensemble de propriétés qui peuvent être déduites à partir des invariants.
- VARIANT définissant l'expression du variant du modèle, une variables entière natu-

relle décroissante définissant l'ordre de prise de contrôle des évènements convergents.
– EVENTS déclarant les événements qui peuvent prendre le contrôle dans le modèle (transitions entre états). Chaque événement est décrit par une garde et par un corps constitué des substitutions traduisant le changement d'état du système. Un événement prend le contrôle lorsque sa garde est évaluée à vrai. Une machine possède un événement particulier qui correspond à l'initialisation du système modélisé.

2.3 Les événements

Un événement conduit à un changement d'état du système modélisé par le biais d'une transition. Il est soit paramétré et composé des clauses décrites dans la Table 4.1, soit non paramétré et composé des clauses décrites dans la Table 4.2.

<identifiant_événement> =
STATUS
ordinary, convergent, anticipated
REFINES
<liste_identifiant_événements>
ANY
<liste_identifiant_paramètres>
WHERE
<label> : <prédicat>
...
WITH
<label> : <témoin>
...
THEN
<label> : <action>
...
END

Tableau 4.1 – Structure d'un événement paramétré

<identifiant_événement> =
STATUS
ordinary, convergent, anticipated
REFINES
<liste_identifiant_événements>
WHEN
<label> : <prédicat>
...
THEN
<label> : <action>
...
END

Tableau 4.2 – Structure d'un événement non paramétré

– STATUS définissant le statut d'un événement. Il peut être ordinary, convergent (l'événement doit décrémenter la valeur du variant), ou anticipated (l'événement ne doit pas incrémenter la valeur du variant).
– REFINES déclarant l'événement abstrait raffiné éventuellement par l'événement décrit.
– ANY décrivant les noms des paramètres éventuels de l'événement.
– WHERE exprimant la garde de l'événement. Elle définit les conditions de la prise de contrôle de l'événement.
– WITH exprimant des témoins (prédicats déclarés dans l'évènement raffiné pour définir la valeur de chaque variable et/ou paramètre de l'événement abstrait qui disparait dans l'événement raffiné). Les témoins contribuent dans la preuve de simulation d'un événement abstrait par l'événement qui le raffine (voir section 2.4).

– THEN déclarant les actions de l'événement qui expriment les changements de l'état du système modélisé par le biais des changements de ses variables (substitutions).

B Événementiel offre trois types d'actions pouvant être déterministes ou non.

– Une action déterministe représentée par l'opérateur d'affectation " :=" agissant sur une variable en modifiant sa valeur, elle est illustrée comme suit :

$$< identifiant\_variable > \ := \ < expression >$$

Exemple :

$$x := x + z$$

– Une action non-déterministe représentée par l'opérateur *devient_tel_que* agissant sur un ensemble de variables et dont l'effet est représenté par un prédicat exprimant la relation entre la valeur des variables avant et après l'exécution de l'action. L'action est illustrée comme suit :

$$< liste\_identifiant\_variables > \ :| \ \ < prédicat >$$

Exemple :

$$x, y \ \ :| \ x' > y \wedge y' > x' + z$$

Où x', y' et z' sont les nouvelles valeurs des variables x, y et z suite à l'exécution de l'action.

– Une action non-déterministe représentée par l'opérateur *devient_appartenant_a* agissant sur une variable en modifiant son contenu avec une valeur indéterminée dans un ensemble de valeurs comme suit :

$$< identifiant\_variable > \ :\in \ < ensemble >$$

Exemple :

$$x \ \ :\in \ \ \{1, 2, 3\}$$

Les événements d'une machine B Événementiel sont atomiques. La sémantique associée aux événements est une sémantique d'entrelacement. Par conséquent, la sémantique d'un modèle B Événementiel est basée sur une trace d'événements entrelacés.

2.4 Les obligations de preuve

Les obligations de preuve (PO) représentent la partie devant être prouvée dans un modèle B Événementiel, elles sont de la forme : hypothèses $\vdash$ conclusion.

Les obligations de preuves sont générées automatiquement par un plugin de la plate-forme *Rodin* nommé the *proof obligation generator* en fonction du modèle et suivant un certain nombre de règles appelées règles des obligations de preuve. Une fois ces obligations de preuves générées, elles sont transmises au prouveur de la plateforme *Rodin* pour être prouvées soit

de manière automatique par le prouveur soit de manière interactive avec l'intervention du concepteur. La liste complète des règles d'obligation de preuve est décrite dans [56], ci-dessous, les règles les plus représentatives.

Afin de définir ces règles, nous introduisons un modèle B Événementiel type décrit dans la Figure 4.2. Dans le contexte *contexte_name*, s et c représentent respectivement les ensembles et constantes et $A(s,c)$ et $T(s,c)$ représentent respectivement les axiomes et théorèmes. Dans la machine *machine_name*, v représente les variables de la machine, $A(s,c,v)$ et $T(s,c,v)$ représentent respectivement les axiomes et théorèmes, $V(s,c,v)$ représente le variant de la machine, *evt_name* représente un évènement convergent, paramétré par le paramètre x et gardé par la garde $G(s,c,v,x)$. L'expression $BA(s,c,v,x,v')$ représente un prédicat sur la nouvelle valeur de v, notée v', après modification.

CONTEXT	**MACHINE**
contexte_name	machine_name
SETS	**SEES**
s	contexte_name
CONSTANTS	**VARIABLES**
c	v
AXIOMS	**INVARIANTS**
A(s,c)	I(s,c,v)
THEOREMS	**THEOREMS**
T(s,c)	T(s,c,v)
END	**VARIANT**
	V(s,c,v)
	EVENTS
	EVT evt_name
	STATUS convergent
	ANY *x* **WHERE**
	G(s, c, v, x)
	THEN
	v :| BA(s, c, v, x, v')
	END
	END

Figure 4.2 – Modèle B Événementiel type

- **Obligation de preuve de préservation d'invariant (INV) :** assure que, sous couvert de l'ensemble des axiomes, des invariants, des théorèmes et des gardes de l'événement, chaque invariant est préservé par tous les événements y compris l'initialisation, ce qui signifie que les substitutions opérées dans les actions des événements satisfont toujours l'invariant.

$$A(s,c), T(s,c), A(s,c,v), T(s,c,v), I(s,c,v), G(s,c,v,x), BA(s,c,v,x,v') \vdash inv(s,c,v')$$

- **Obligation de preuve de faisabilité (FIS) :** assure que, sous couvert de l'ensemble des axiomes, des invariants, des théorèmes et des gardes de l'événement, chaque action non-déterministe est réalisable, ceci est assuré s'il existe au moins une valeur possible pour la réalisation de l'action.

$$A(s,c), T(s,c), A(s,c,v), T(s,c,v), I(s,c,v), G(s,c,v,x) \vdash \exists v' \cdot BA(s,c,v,x,v')$$

– **Obligation de preuve de variant numérique (NAT) :** assure que, sous couvert de l'ensemble des axiomes, des invariants, des théorèmes et des gardes de l'événement, la garde de chaque événement de type covergent ou anticipated est dotée d'un variant de type entier naturel.

$$A(s,c), T(s,c), I(s,c,v), T(s,c,v), G(x,s,c,v) \vdash V(s,c,v) \in NAT$$

– **Obligation de preuve de variant (VAR) :** assure que, sous couvert de l'ensemble des axiomes, des invariants, des théorèmes et des gardes de l'événement, chaque événement de type covergent ou anticipated décrémente le variant (la nouvelle valeur du variant suite au déclenchement de l'évènement est strictement inférieure à sa valeur avant le déclenchement de l'évènement.).

$$A(s,c), T(s,c), I(s,c,v), T(s,c,v), G(x,s,c,v), AC(s,c,v,x,v') \vdash V(s,c,v') < V(s,c,v)$$

2.5 Le raffinement

L'opération de raffinement [164] offerte par la méthode B Événementiel consiste à reformuler, par étapes successives, une machine abstraite en une suite de machines plus précises (modèle concret) où plus de détails apparaissent, dans l'état du système en ajoutant des variables, et dans le comportement en détaillant les événements de l'abstraction ou en ajoutant de nouveaux événements. Un raffinement est une machine dont le comportement détaille celui de l'abstraction. Ainsi, il doit assurer que tous les comportements du raffinement sont des comportements de l'abstraction, et doit préserver l'invariant de l'abstraction. Un lien entre les variables d'abstraction et les variables du raffinement est défini par l'invariant dit de collage. Cet invariant de collage explicite la manière dont sont liées les variables du raffinement aux variables de la machine abstraite [166].

2.6 Les obligations de preuve du raffinement

Le développement de raffinements induit la génération de nouvelles obligations de preuves. Ces obligations de preuves portent d'une part sur la cohérence du raffinement en tant que modèle (établissement des obligations de preuves : INV, FIS, NAT, VAR pour les nouvelles structures introduites dans le raffinement : variables, événements, gardes, actions) et sur le raffinement de la machine abstraite exprimé par les règles d'obligation de preuve de raffinement. Ces dernières permettent d'établir la cohérence de la machine raffinée avec la machine abstraite ce qui permet d'assurer l'établissement dans la machine raffinée des obligations de preuves prouvées dans la machine abstraite sans avoir à les reprouver.

Nous présentons ci-dessous les deux règles d'obligation de preuve de raffinement les plus représentatives.

Afin de définir ces règles, nous introduisons un raffinement type décrit dans la Figure 4.3. Le raffinement *refinement_name* raffine *machine_name* présentée dans la Figure 4.2. Il importe le contexte *contexte_name* et définit les variables concrètes w, les invariants $J(s,c,v,w)$, les théorèmes $H(s,c,v,w)$ et l'événement *evt_ref_name* qui raffine l'événement *evt_name* de la machine *machine_name*. L'événement *evt_ref_name* contient les déclarations du paramètre y, de la garde $R(y,s,c,w)$ et de l'action $w :| BA'(s,c,w,y,w')$. L'expression $BA'(s,c,w,y,w')$ représente un prédicat sur la nouvelle valeur de w notée w' après modification. L'événement *evt_ref_name* contient également l'expression d'un témoin $v' : W(v',s,c,y,w')$ qui le lie à l'événement abstrait *evt_name*.

CONTEXT contexte_name **SETS** s **CONSTANTS** c **AXIOMS** A(s,c) **THEOREMS** T(s,c) **END**	**MACHINE** refinement_name **REFINES** machine_name **SEES** contexte_name **VARIABLES** w **INVARIANTS** J(s,c,v,w) **THEOREMS** H(s,c,v,w) **EVENTS** **EVT** evt_ref_name **ANY** *y* **WHERE** *R(s, c, w, y)* **WITH** v' : W(v',s,c,y,w') **THEN** *w :| BA'(s, c, w, y, w')* **END** **END**

Figure 4.3 – Raffinement type

- **Obligation de preuve de renforcement des gardes (GRD) :** assure que sous couvert de l'ensemble des axiomes, invariants et théorèmes des machines abstraites et raffinées et des gardes et témoins de l'événement raffiné, la garde concrète de l'événement concret est plus forte que la garde abstraite de l'événement abstrait (la garde concrète peut être déduite de la garde abstraite).

  $$A(s,c), T(s,c), I(s,c,v), T(s,c,v), J(s,c,v,w), R(y,s,c,w), W(v',s,c,y,w') \vdash G(x,s,c,v)$$

- **Obligation de preuve de simulation (SIM) :**
  assure que sous couvert de l'ensemble des axiomes, invariants et théorèmes des machines abstraites et raffinées et des gardes et témoins de l'événement raffiné, chaque action dans un événement abstrait est correctement simulée dans le raffinement correspondant (l'action abstraite peut être déduite de l'action concrète).

$$A(s,c), T(s,c), I(s,c,v), T(s,c,v), J(s,c,v,w), R(y,s,c,w), W(v',s,c,y,w'), BA'(s,c,w,y,w') \vdash BA(s,c,v,x,v')$$

2.7 Exemple de modèle B Évènementiel

Nous illustrons le processus de formalisation B Évènementiel par l'introduction d'un exemple emprunté à [166] relatif à la spécification d'une horloge présentée dans la Figure. 4.4.

CONTEXT ClockContext1
CONSTANTS
HR
AXIOMS
axm1 : $HR \in \mathbb{N}$
axm2 : $HR = 23$
END

MACHINE ClockMachine1
SEES ClockContext1
VARIABLES
hours
INVARIANTS
inv1 : $hours \in 0 \,..\, HR$
inv2 : $(hours \neq HR) \lor (hours = HR)$
EVENTS
Initialisation
begin
act1 : $hours :\in 0 \,..\, HR$
end

Event *incr* $\widehat{=}$
when
grd1 : $hours \neq HR$
then
act1 : $hours := hours + 1$
end
Event *zero* $\widehat{=}$
when
grd1 : $hours = HR$
then
act1 : $hours := 0$
end
END

Figure 4.4 – Spécification B Évènementiel abstraite d'une horloge

Dans un premier temps, la spécification modélise uniquement les heures. Le contexte *ClockContext*1 déclare une constante *HR* modélisant le nombre maximum d'heures dans une horloge. La machine *ClockMachine*1 importe le contexte *ClockContext*1 et utilise la constante *HR*. Elle décrit une variable *hours* pour les heures de l'horloge. Deux événements sont décrits. Le premier (événement *incr*) permet d'incrémenter la variable *hours* et prend le contrôle tant que $hours < HR$. Le second événement *zero* prend le contrôle lorsque $hours = HR$, il a pour effet de réinitialiser la variable *hours* à la valeur zéro. Le théorème de cette machine modélise la propriété de non blocage par la disjonction des gardes des événements de la machine *ClockMachine*1.

Dans la spécification du raffinement *ClockMachine*2 (voir Figure 4.5), la gestion des minutes est introduite dans la spécification de l'horloge. Le nouveau contexte *ClockContext*2

étend le contexte *ClockContext*1 en ajoutant la déclaration d'une nouvelle constante *MN* modélisant le nombre maximum de minutes dans une horloge. La machine *ClockMachine*2 raffine la machine *ClockMachine*1 et importe le contexte *ClockContext*2 pour utiliser la constante *MN*. Le raffinement *ClockMachine*2 déclare une nouvelle variable *minutes* modélisant les minutes de l'horloge, il raffine respectivement les événements abstraits *incr* et *zero* par les événements concrets *incr_ref* et *zero_ref* pour calculer en plus de la variable *hours* (heure), la variable *minutes* (minutes). Il introduit également un nouvel événement *ticTac* qui incrémente la variable *minutes* tant que $minutes < MN$. L'événement *ticTac* est déclaré convergent et doit décrémenter le variant $MN - minutes$ pour vérifier qu'il ne prend pas le contrôle indéfiniment.

CONTEXT ClockContext2
EXTENDS ClockContext1
CONSTANTS
MN
AXIOMS
axm1 : $MN \in \mathbb{N}$
axm2 : $MN = 59$
END

MACHINE ClockMachine2
REFINES ClockMachine1
SEES ClockContext2
VARIABLES
hours minutes
INVARIANTS
inv1 : $minutes \in 0 \mathinner{..} 59$
VARIANT
$MN - minutes$
EVENTS
Initialisation
begin
act1 : $hours :\in 0 \mathinner{..} 23$
act2 : $minutes :\in 0 \mathinner{..} 59$
end
Event *incr_ref* $\widehat{=}$
refines *incr*
when
grd1 : $hours \neq HR$
grd2 : $minutes = MN$
then
act1 : $hours := hours + 1$
act2 : $minutes := 0$
end
Event *zero_ref* $\widehat{=}$
refines *zero*
when
grd1 : $hours = HR$
grd2 : $minutes = MN$
then
act1 : $hours := 0$
act2 : $minutes := 0$
end
Event *ticTac* $\widehat{=}$
Status convergent
when
grd1 : $minutes \neq MN$
then
act1 : $minutes := minutes + 1$
end
END

Figure 4.5 – Spécification B Évènementiel raffinée d'une horloge

3 Démarche de modélisation avec B Évènementiel

La démarche de formalisation avec B Évènementiel des interfaces homme-machine multimodales en sortie, reflète le modèle générique proposé, elle exploite les mécanismes et les éléments du langage de la méthode B Évènementiel comme suit.

1. **La modélisation incrémentale par raffinements successifs :** permet de modéliser les différentes représentations de l'interface multimodale en sortie tout au long du processus de conception où chaque modèle B Évènementiel (abstrait/raffinement) correspond à un niveau de modélisation de l'interface dans le processus de conception (voir Figure 4.6).

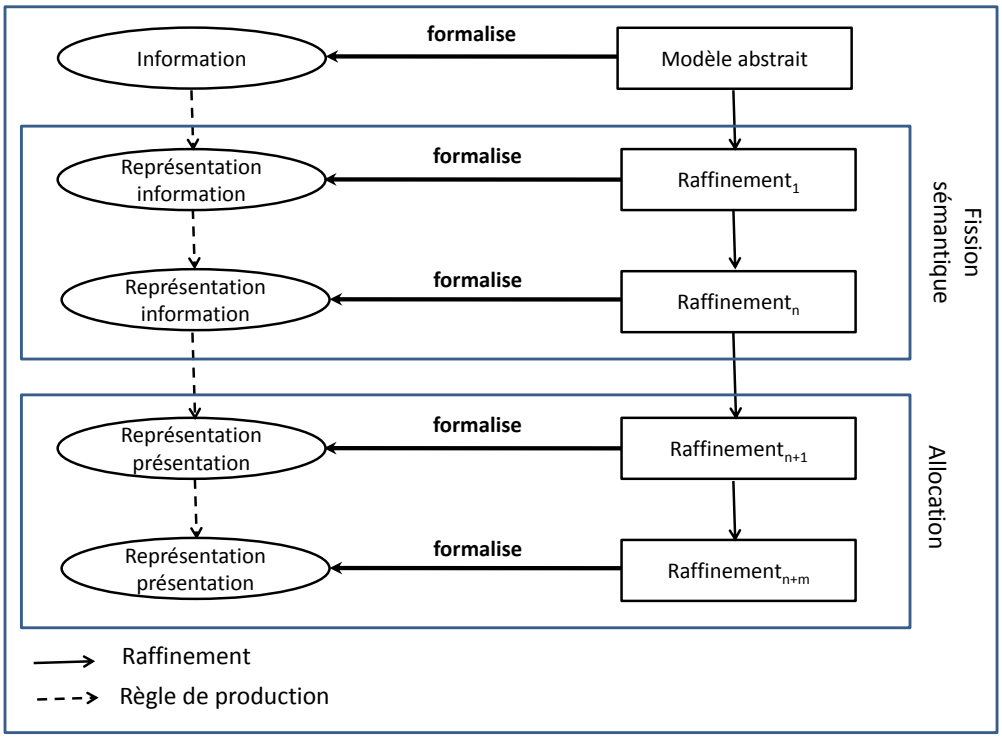

Figure 4.6 – Le processus de modélisation B Évènementiel de l'IHM multimodale en sortie

Pour cela, nous utilisons le principe introduit dans [167]. Les auteurs formalisent les opérateurs d'une algèbre de processus dans la méthode B événementiel par raffinements en exploitant les règles BNF définissant les opérateurs de cette algèbre de processus. Ainsi, la partie gauche de la règle BNF est formalisée par un modèle abstrait et la partie droite de la règle BNF est formalisée par le modèle raffiné. Nous utilisons ce même principe pour le développement B Évènementiel de notre interface en s'ap-

puyant sur les règles syntaxiques BNF définies dans les deux étapes du processus de conception.

- L'information produite par le noyau fonctionnel est formalisée par le modèle abstrait de l'interface.
- Le modèle abstrait est raffiné pour chaque application d'une règle syntaxique de la fission sémantique ($n - 1$ raffinements).
- Le dernier raffinement de la fission sémantique sera à son tour raffiné pour chaque application d'une règle syntaxique de l'allocation ($m - 1$ raffinements). Ainsi, le dernier raffinement de l'allocation constitue le dernier modèle du processus de développement B Évènementiel de l'interface, il formalise la présentation multimodale obtenue.

2. **Dichotomie statique / dynamique :** la structure du modèle B Évènementiel permet de formaliser les sémantiques statique et dynamique de notre modèle générique de manière séparée (voir Figure 4.7).

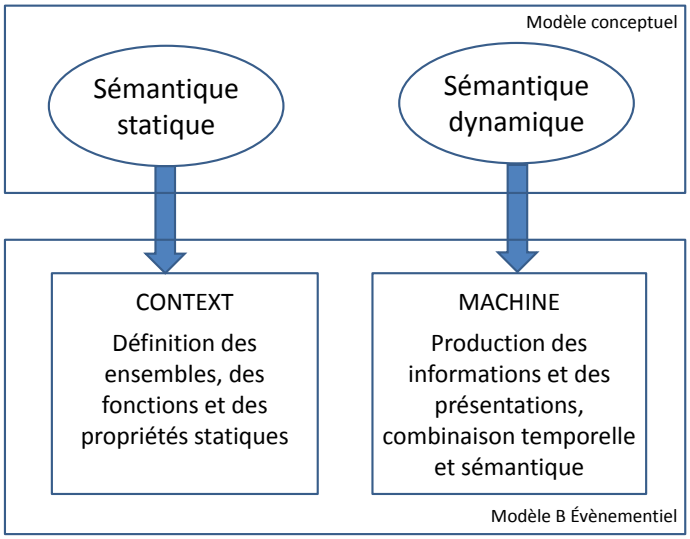

Figure 4.7 – Correspondance entre modèle conceptuel et modèle B Évènementiel

En effet, la sémantique statique du modèle conceptuel, qui exprime les propriétés statiques définies sur l'interface est formalisée par le composant CONTEXT du modèle B Évènementiel, par l'utilisation de la théorie des ensembles tandis que la sémantique dynamique exprimant les changements d'état de l'interface, est formalisée dans le composant MACHINE du modèle B Évènementiel par le biais des systèmes états/transitions.

4 Les modèles génériques de développement B Évènementiel

Le processus de développement B Évènementiel de l'interface multimodale en sortie (voir Figure 4.8) consiste à construire successivement plusieurs modèles B Évènementiel correspondant aux différentes étapes de transformation de l'interface Homme-Machine multimodale en sortie.

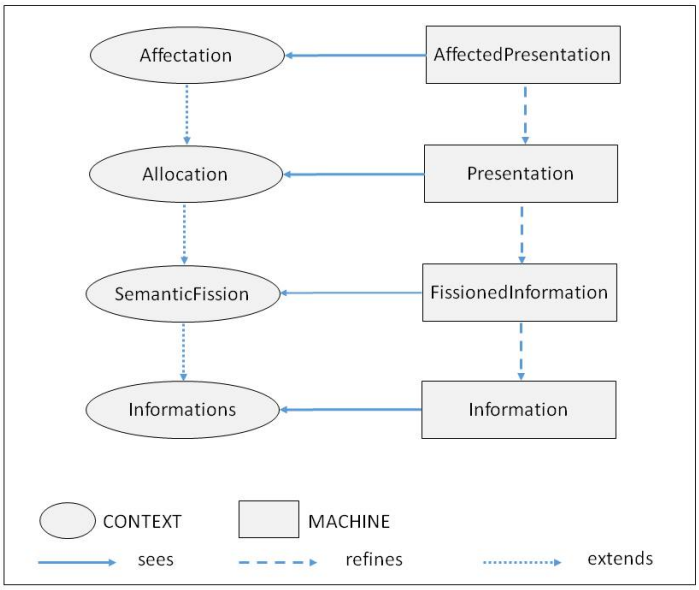

Figure 4.8 – Les modèles B Évènementiel génériques de développement des IHM multimodales en sortie

Nous présentons ci-après les modèles génériques pour chacune de ces étapes : modèle abstrait, raffinements de la fission sémantique, raffinements de l'allocation. Pour celà, nous nous basons sur les règles syntaxiques développées dans le chapitre 3. Ainsi, chaque règle syntaxique est formalisée par un modèle générique cadre à partir duquel sont dérivés les modèles spécifiques à chaque cas précis exprimé par la règle syntaxique. Les modèles spécifiques à la fission sémantique et à l'allocation sont détaillés respectivement dans les chapitres 5 et 6.

4.1 Le modèle générique abstrait

Le modèle générique abstrait (voir Figure 4.9) modélise la production de l'information i par le noyau fonctionnel et la construction de la présentation multimodale qui la restitue

à l'utilisateur. Il formalise les parties gauches des règles syntaxiques 4.1, et 4.2 ainsi que la règle de collage 4.3.

$$I ::= UIE \mid (op_{temp}, op_{sem})(I, I) \mid It(n, I) \; avec \; n \in \mathbb{N} \tag{4.1}$$

$$PM ::= PME \mid (op'_{temp}, op'_{sem})(PM, PM) \mid It'(n, PM) \; avec \; n \in \mathbb{N} \tag{4.2}$$

$$PM ::= allocation(I) \tag{4.3}$$

Il comprend le contexte et la machine décrits ci-dessous.

CONTEXT Informations
SETS
Information InterpretationDomain Presentation
CONSTANTS
interpretation allocation
AXIOMS
axm1 : *interpretation* ∈ *Information* → *InterpretationDomain*
axm2 : *allocation* ∈ *Information* → *Presentation*
END

MACHINE Information
SEES Informations
VARIABLES
i d p varSeq
INVARIANTS
inv1 : $i \in Information$
inv2 : $d \in InterpretationDomain$
inv3 : $p \in Presentation$
inv4 : $varSeq \in \{0, 1\}$
VARIANT
varSeq
EVENTS
Initialisation
begin
act1 : $i :\in Information$
act2 : $d :\in InterpretationDomain$
act3 : $p :\in Presentation$
act4 : $varSeq := 1$
end

Event *information* $\widehat{=}$ /*Délivrance de *i* */
Status convergent
any
x
where
grd1 : $x \in Information$
grd2 : $varSeq = 1$
then
act1 : $i := x$
act2 : $varSeq := varSeq - 1$
end
Event *interpretation* $\widehat{=}$ /*calcul de d et construction de *p*/
when
grd1 : $varSeq = 0$
then
act1 : $d := interpretation(i)$
act2 : $p := allocation(i)$
end
END

Figure 4.9 – Le modèle générique abstrait

4.1.1 Le CONTEXT *Informations*

Il décrit l'environnement statique de l'interface constitué des ensembles : *Information* qui regroupe les informations intervenant lors du processus de génération de l'interface et

Presentation qui regroupe les présentations intervenant lors du processus de génération de l'interface, il définit également la fonction *allocation* (*axm*2) qui alloue à chaque information la présentation qui la restitue à l'utilisateur. Enfin, il définit la sémantique des informations manipulées en introduisant l'ensemble des interprétations sémantiques des informations manipulées *InterpretationDomain* et la fonction *interpretation* (*axm*1) qui affecte à chaque information son interprétation sémantique.

4.1.2 La MACHINE *Information*

Elle modélise la production de l'information à fissionner. Elle fait intervenir pour cela deux évènements en séquence : l'évènement *information* qui modélise la production de l'information *i* à fissionner suivi de *interpretation*, l'évènement qui calcule *d* l'interprétation de l'information *i* (*act*1) ainsi que *p* la présentation qui la restitue (*act*2). L'ordonnancement séquentiel des deux évènements *information* et *interpretation* est garanti par l'introduction du variant *varSeq* initialisé à 1.

4.2 Le modèle générique de la fission sémantique

Le modèle générique de la fission sémantique (voir Figure 4.10) modélise la production de l'information *i* par la combinaison des informations fissionnées *i*1 et *i*2. Il formalise la partie droite de la règle syntaxique 4.1 et comprend le contexte et la machine décrits ci-dessous.

4.2.1 Le CONTEXT *SemanticFission*

Il étend le CONTEXT *Informations* par l'introduction de l'opérateur sémantique générique *semanticoperator*. Ce dernier correspond à l'opérateur op_{sem} qui relie les informations fissionnées (*axm*1).

4.2.2 La MACHINE *FissionedInformation*

Elle raffine la MACHINE *Information* afin de produire l'information *i* en terme des informations fissionnées *i*1 et *i*2. Elle fait intervenir pour cela quatre événements.

- l'évènement *information* est raffiné par la production de deux informations supplémentaires *i*1 et *i*2 (*act*1), dont la combinaison sémantique des interprétations est égale à l'interprétation de *i* (*grd*2).
- deux nouveaux évènements *interpretation*1 et *interpretation*2 sont introduits, ils calculent respectivement *d*1 (*act*1) et *d*2(*act*1) les interprétations sémantiques de *i*1 et *i*2.

CONTEXT SemanticFission
EXTENDS Informations
CONSTANTS
semanticOperator
AXIOMS
axm1 : $semanticOperator \in InterpretationDomain \times InterpretationDomain \rightarrow InterpretationDomain$
END

MACHINE FissionedInformation
REFINES Information
SEES SemanticFission
VARIABLES
i d p varSeq var i1 i2 d1 d2
INVARIANTS
inv1 : $i1 \in Information$
inv2 : $d1 \in InterpretationDomain$
inv3 : $var \in \mathbb{N}$
inv4 : $varSeq < 1 \Rightarrow interpretation(i) = semanticOperator(interpretation(i1) \mapsto interpretation(i2))$
inv5 : $var = 0 \Rightarrow d1 = interpretation(i1) \wedge d2 = interpretation(i2)$ ...
VARIANT
var
EVENTS
Initialisation
begin
act1 : $i1 :\in Information$
act3 : $d1 :\in InterpretationDomain$
act5 : $var :\in \mathbb{N}_1$ ...
end
Event *information* $\widehat{=}$ /*Délivrance de *i* */
Status convergent
refines *information*
any
$x, x1, x2$
where
grd1 : $x \in Information$
grd2 : $interpretation(x) = semanticOperator(interpretation(x1) \mapsto interpretation(x2))$
grd3 : $varSeq = 1$
grd4 : $var \in \mathbb{N}_1$ ...
then
act1 : $i, i1, i2 := x, x1, x2$
act2 : $varSeq := varSeq - 1$
end

Event *interpretation* $\widehat{=}$ /*calcul de *d* et construction de *p*/
refines *interpretation*
when
grd1 : $varSeq = 0$
grd2 : $var = 0$
then
act1 : $d := semanticOperator(d1 \mapsto d2)$
act2 : $p := allocation(i)$
end
Event *interpretation1* $\widehat{=}$ /*calcul de d1*/
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $var \in \mathbb{N}_1$
then
act1 : $d1 := interpretation(i1)$
act2 : $var :| (var' \in \mathbb{N} \wedge var' < var)$
end
Event *interpretation2* $\widehat{=}$ /*calcul de d2*/
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $var \in \mathbb{N}_1$
then
act1 : $d2 := interpretation(i2)$
act2 : $var :| (var' \in \mathbb{N} \wedge var' < var)$
end
END

Figure 4.10 – Le modèle générique de la fission sémantique

– l'évènement *interpretation* est raffiné, il calcule *d* au moyen de la combinaison sémantique de $d1$ et $d2$ ($act1$).

L'ordonnancement temporel des deux évènements *interpretation*1 et *interpretation*2 étant défini pas un opérateur temporel, il est formalisé selon l'approche définie dans [167]. Les au-

teurs ont formalisé les opérateurs de composition temporelle : séquentiel, parallèle, choix et itération dans B Évènementiel par la combinaison de variants et de gardes. Ils utilisent un variant spécifique à chaque cas d'ordonnancement temporel. Ce variant permet de déclencher les événements selon l'ordre établi, par l'utilisation de gardes conditionnées sur la valeur du variant. Le variant est décrémenté dans chaque évènement, permettant de déclencher l'évènement suivant.

Nous introduisons donc un variant générique *var* (*inv*3) qui décroît dans les deux évènements *interpretation*1 (*grd*2) et *interpretation*2 (*grd*2) et qui est nul au déclenchement de l'évènement *interpretation* (*grd*2). L'invariant de collage est exprimé par les invariants : *inv*4 qui définit la relation entre les interprétations de i, $i1$ et $i2$ lors de leur production et *inv*5 qui détermine les valeurs de $d1$ et de $d2$ en fonction de $i1$ et $i2$ après leur calcul dans les événements *interpretation*1 et *interpretation*2.

4.3 Le modèle générique de l'allocation

Le modèle générique de l'allocation (voir Figure 4.11) raffine le modèle générique de la fission sémantique, il modélise la combinaison temporelle et/ou sémantique de deux présentations multimodales pour constituer une présentation résultante.

Il formalise, d'une part, la partie droite de la règle syntaxique 4.4 pour décrire la combinaison temporelle et sémantique des présentations multimodales élémentaires *pme* qui composent la présentation multimodale globale *pm* en utilisant la règle de collage 4.5. D'autre part, il formalise la partie droite de la règle syntaxique 4.6 pour décrire la décomposition des *pme* en unités de présentation élémentaires *upe*. Le modèle générique de l'allocation comprend le contexte et la machine décrits ci-dessous.

$$PM ::= PME \mid (op'_{temp}, op'_{sem})(PM, PM) \mid It'(n, PM) \; avec \; n \in \mathbb{N} \quad (4.4)$$

$$PME ::= allocation(UIE) \quad (4.5)$$

$$PME ::= UPE \mid compl(UPE, PME) \mid redun(UPE, PME) \mid choice(UPE, PME) \mid iter(n, PME) \quad (4.6)$$

4.3.1 Le CONTEXT *Allocation*

Il étend le CONTEXT *SemanticFission* par l'introduction de l'opérateur de composition *combinationOperator* qui combine les présentations mutli-modales (*axm*1). Cet opérateur générique couvre aussi bien les opérateurs sémantiques op'_{sem} mais également les opérateurs *compl* et *redon*.

CONTEXT Allocation
EXTENDS SemanticFission
CONSTANTS
combinationOperator
AXIOMS
axm1 : $combinationOperator \in Presentation \times Presentation \rightarrow Presentation$
END

MACHINE Presentation
REFINES FissionedInformation
SEES Allocation
VARIABLES
i d p i1 i2 d1 d2 p1 p2 varSeq var
INVARIANTS
inv1 : $p1 \in Presentation$
inv2 : $p2 \in Presentation$
inv3 : $varSeq < 1 \Rightarrow allocation(i) = combinationOperator(allocation(i1) \mapsto allocation(i2))$
inv4 : $var = 0 \Rightarrow p1 = allocation(i1) \wedge p2 = allocation(i2)$
EVENTS
Initialisation
begin
act1 : $p1 :\in Presentation$
act2 : $p2 :\in Presentation$
act3 : $var :\in \mathbb{N}_1$ ...
end
Event *information* $\widehat{=}$ /*Délivrance de *i* */
Status convergent
any
$x, x1, x2$
where
grd1 : $interpretation(x) = semanticOperator(interpretation(x1) \mapsto interpretation(x2))$
grd2 : $allocation(x) = combinationOperator(allocation(x1) \mapsto allocation(x2))$
grd3 : $varSeq = 1$
grd4 : $var \in \mathbb{N}_1$ ...
then
act1 : $i, i1, i2 := x, x1, x2$
act2 : $varSeq := varSeq - 1$
end
Event *presentation* $\widehat{=}$ /*construction de *p* */
refines *interpretation*
when
grd1 : $varSeq = 0$
grd2 : $var = 0$
then
act1 : $d := semanticOperator(d1 \mapsto d2)$
act2 : $p := combinationOperator(p1 \mapsto p2)$
end
Event *presentation1* $\widehat{=}$ /*construction de *p*1 */
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $var \in \mathbb{N}_1$
then
act1 : $d1 := interpretation(i1)$
act2 : $var :| (var' \in \mathbb{N} \wedge var' < var)$
act3 : $p1 := allocation(i1)$
end
Event *presentation2* $\widehat{=}$ /*construction de *p*2 */
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $var \in \mathbb{N}_1$
then
act1 : $d2 := interpretation(i2)$
act2 : $var :| (var' \in \mathbb{N} \wedge var' < var)$
act3 : $p2 := allocation(i2)$
end
END

Figure 4.11 – Le modèle générique de l'allocation

4.3.2 La MACHINE *Presentation*

Elle raffine la MACHINE *FissionedInformation* pour produire les présentations correspondantes aux informations fissionnées. Elle fait intervenir pour cela quatre événements.

– l'évènement *information* est raffiné par l'introduction d'une garde supplémentaire (*grd*2), qui exprime que la combinaison sémantique des présentations relatives aux

informations fissionnées est égale à la présentation qui restitue *i*.

- les évènements *interpretation*1 et *interpretation*2 sont raffinés par les évènements *presentation*1 et *presentation*2 qui construisent en plus des interprétations, les présentations *p*1 (*act*3) et *p*2 (*act*3) correspondantes respectivement à *i*1 et *i*2. L'ordonnancement temporel de *presentation*1 et *presentation*2 est garanti par le variant générique *var* introduit dans le raffinement précédemment. Il assure que les présentations sont produites dans le même ordre temporel que les informations qu'elles restituent.
- l'évènement *interpretation* est raffiné par l'évènement *presentation* qui construit *p* au moyen de la combinaison de *p*1 et *p*2 (*act*2).

L'invariant de collage est exprimé par les invariants : *inv*3 qui définit la relation entre les présentations relatives à *i*, *i*1 et *i*2 lors de leur production et *inv*4 qui détermine les valeurs de *p*1 et de *p*2 en fonction de *i*1 et *i*2 après leur construction dans les évènements *presentation*1 et *presentation*2. Il traduit les règles de collage 4.3 et 4.5.

4.4 Le modèle générique de l'affectation

Le modèle générique de l'affectation (voir Figure 4.12) raffine le modèle générique de l'allocation, il modélise l'affectation des modalités et des médias aux unités de présentations élémentaires *upe* ($upe \in UPE$). Il formalise la partie droite de la règle syntaxique 4.7. Le modèle générique de l'affectation comprend le contexte et la machine décrits ci-dessous.

$$affectation(UPE) ::= ITEM \tag{4.7}$$

4.4.1 Le CONTEXT *Affectation*

Il étend le CONTEXT *Allocation* par l'introduction des ensembles : *Modality* pour les modalités, *Media* pour les médias et *PresentUnit* pour l'ensemble des unités de présentations multimodales. Il s'agit d'un sous-ensemble de *Presentation* (*axm*1) qui regroupe les présentations obtenues suite à la décomposition opérée lors de la phase d'allocation et qui sont allouées avec un couple (*modality*, *media*). L'allocation des couples (*modality*, *media*) se fait de manière séparée : d'une part, la modalité est affectée à la présentation, ainsi, le CONTEXT *Affectation* définit la fonction *affectation* qui affecte une modalité à une présentation unitaire (*axm*2). D'autre part, un média est sélectionné dans l'ensemble des médias qui peuvent restituer la modalité pour la véhiculer. Par conséquent, la fonction *linkModalityMedia* est introduite (*axm*2). Elle affecte à une modalité l'ensemble des médias qui peuvent la restituer.

CONTEXT Affectation
EXTENDS Allocation
SETS
presentationUnit Modality Media
CONSTANTS
affectation linkModalityMedia
AXIOMS
axm1 : $presentationUnit \subseteq Presentation$
axm2 : $affectation \in presentationUnit \rightarrow Modality$
axm3 : $linkModalityMedia \in Modality \rightarrow \mathbb{P}(Media)$
END

MACHINE AffectedPresentation
REFINES Presentation
SEES Affectation
VARIABLES
item i i1 i2 d d1 d2 p p1 p2 varSeq var
INVARIANTS
inv1 : $item \in presentationUnit \rightarrow Modality \times Media$
EVENTS
Initialisation
begin
act1 : $item :\in presentationUnit \rightarrow Modality \times Media$
act2 : $varSeq := 1$
act3 : $var :\in \mathbb{N}_1$
act4 : ...
end
Event *information* $\widehat{=}$ /*Délivrance de *i* */
...
Event *presentation* $\widehat{=}$ /*construction de *p* */
...
Event *presentation1* $\widehat{=}$ /*construction de *p*1 */
...
Event *presentation2* $\widehat{=}$ /*construction de *p*2 */
...

Event *affectation1* $\widehat{=}$ /*affectation de *p*1 */
any
m
where
grd1 : $p1 \in presentationUnit$
grd2 : $m \in linkModalityMedia(affectation(p1))$
grd3 : $var = 0$
then
act1 : $item(p1) := (affectation(p1) \mapsto m)$
end
Event *affectation2* $\widehat{=}$ /*affectation de *p*2 */
any
m
where
grd1 : $p2 \in presentationUnit$
grd2 : $m \in linkModalityMedia(affectation(p2))$
grd3 : $var = 0$
then
act1 : $item(p2) := (affectation(p2) \mapsto m)$
end
END

Figure 4.12 – Le modèle générique de l'affectation

4.4.2 La MACHINE *AffectedPresentation*

Elle raffine la MACHINE *Presentation* par l'introduction d'une variable de type fonction *item*. Elle affecte à une unité de présentation un couple (*modality*, *media*) (*inv*1). La MACHINE *AffectedPresentation* introduit deux nouveaux évènements *affectation*1 et *affectation*2 qui permettent d'affecter respectivement à *p*1 et *p*2 (*act*1), sous la condition qu'elles appartiennent à *presentationUnit* (*grd*1), les couples (*modality*, *media*) qui les restituent. La modalité est allouée à la présentation au moyen de la fonction *affectation*, le média par contre, est

sélectionné au moment du déclenchement des évènements $affectation1$ et $affectation2$ ($grd2$) en utilisant la fonction *linkModalityMedia*.

L'ordonnancement temporel des évènements $affectation1$ et $affectation2$ par rapport aux évènements *Presentation*1 et *Presentation*2 est assuré par la garde ($grd3$) dans $affectation1$ et $affectation2$. Elle assure que les évènements $affectation1$ et $affectation2$ ne sont déclenchés qu'une fois que les présentations $p1$ et $p2$ sont calculées respectivement dans *Presentation*1 et *Presentation*2.

4.5 Bilan des obligations de preuve

Les modèles décrits ci-dessus ont généré 56 obligations de preuves (OP) (voir Table 4.3) dont 52 ont été prouvées de manière automatique par le prouveur de la plate-forme *Rodin*. 4 ont nécessité l'intervention du concepteur dans une preuve interactive, dont 2 obligations de preuve de préservation d'invariant sur $inv5$ dans $FissionedInformation$ et 2 obligations de preuve de préservation d'invariant sur $inv4$ dans *Presentation*.

Composant	OP automatiques	OP manuelles	OP totales	OP non prouvées
Information	9	0	9	0
FissionedInformation	22	2	24	0
Presentation	13	2	15	0
AffectedPresentation	8	0	8	0
Total	52	4	56	0

Tableau 4.3 – Les obligations de preuves

4.6 Instanciation des modèles B Évènementiel génériques

L'utilisation des modèles génériques pour la formalisation d'une interface multimodale en sortie concrète nécessite la définition concrète des ensembles décrivant le contexte de l'interface et la précision des opérateurs de combinaison temporelle et sémantique combinant les informations et les présentations. Dans un premier temps, l'interface est décrite dans le modèle formel générique par une série de formules la définissant dans les différentes étapes : délivrance de l'information, fission sémantique, allocation et affectation. Ces formules sont obtenues par l'application des règles définissant la syntaxe de chaque étape. Dans un deuxième temps, les codes B Évènementiel décrivant l'interface sont construits par instanciation des modèles B Évènementiel génériques comme suit :

1. l'extension du contexte générique par un contexte spécifique à l'interface concrète. Il s'agit de créer un nouveau contexte concret qui étend, en utilisant la clause *EXTENDS*,

le contexte générique correspondant à la règle syntaxique appliquée. Le contexte concret définit en extension les ensembles et fonctions introduits dans le modèle générique, il définit de manière plus précise les opérateurs sémantiques dont seule la signature est définie au niveau générique. La définition des différents opérateurs sémantiques dans des contextes spécifiques est détaillée dans les chapitre 5 et 6.

2. l'enrichissement de la machine générique en accord avec l'interface concrète. Contrairement au contexte, il n'y a pas lieu de créer une nouvelle machine. La machine générique est enrichie par les détails de l'interface concrète en explicitant les opérateurs sémantiques employés pour combiner les informations et les présentations et en définissant le variant de manière à assurer l'ordonnancement temporel adéquat pour la production des informations et des présentations. Les différentes variantes d'ordonnancement temporel ainsi que les variants qui les définissent sont détaillés dans les chapitre 5 et 6.

4.7 Application à l'étude de cas

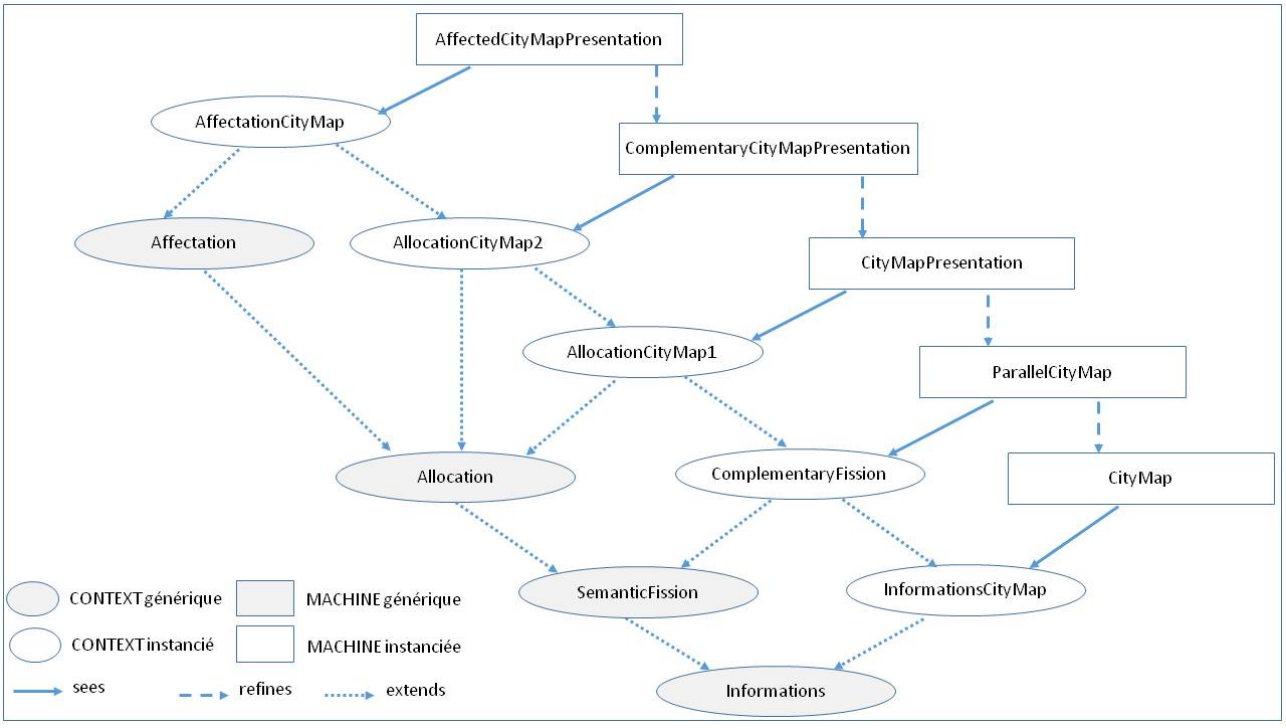

Figure 4.13 – Le processus de développement B Évènementiel de *CityMap*

Conformément à la démarche décrite ci-dessus, le processus de développement B Évènementiel de l'interface multimodale *CityMap* exploite les définitions de l'interface *CityMap*

dans le modèle formel générique que nous avons présenté dans le chapitre 3 et que nous récapitulons ci-dessous. Il consiste à développer pour chaque étape de transformation de l'information $infoCityMap$, le modèle B Évènementiel correspondant par instanciation du modèles générique relatif à la règle syntaxique invoquée par la définition. Il est présenté dans Figure.4.13.

$$infoCityMap = (Pl, Cp)(infoSee, infoMap) \tag{4.8}$$

$$presentCityMap = (Pl', Cp')(presentSee, presentMap) \tag{4.9}$$

$$presentSee = compl(presentSeeSpeech, presentSeeExpression) \tag{4.10}$$

$$image, écran) = affectation(presentMap) \tag{4.11}$$

$$(parole, haut - parleur) = affectation(presentMapSpeech) \tag{4.12}$$

$$(expression, écran) = affectation(presentMapExpression) \tag{4.13}$$

4.7.1 Modèle abstrait

Le modèle abstrait (voir Figure 4.14) formalise la délivrance de $infoCityMap$ et la construction de la présentation $presentCityMap$ qui la restitue. Il formalise les parties gauches des définitions 4.8 et 4.9. Il se compose des éléments suivants.

- le contexte instancié $InformationCityMap$ qui étend le contexte générique $Information$ par la définition en extension des ensembles qui constituent l'environnement de l'interface : $Information$, $interpretationDomain$ et $Presentation$ et des fonctions $allocation$ et $interpretation$ ($axm1$, $axm2$, $axm3$, $axm4$, $axm5$, $axm6$, $axm7$, $axm8$).
- la machine $CityMap$ qui enrichit la machine $Information$ par l'attribution de la valeur $infoCityMap$ à l'information délivrée dans l'événement $information$ ($act1$).

CONTEXT InformationsCityMap
EXTENDS Informations
SETS

$Information = \{infoCityMap, infoSee, infoMap, emptyI\}$
$InterpretationDomain = \{CityMap, See, Map, emptyD\}$
$Presentation = \{presentCityMap, presentSee, presentMap, presentSeeSpeech, presentSeeExpression, emptyP\}$

AXIOMS

axm1 : $interpretation(infoCityMap) = CityMap$
axm2 : $interpretation(infoSee) = See$
axm3 : $interpretation(infoMap) = Map$
axm4 : $interpretation(emptyI) = emptyD$
axm5 : $allocation(infoCityMap) = presentCityMap$
axm6 : $allocation(infoSee) = presentSee$
axm7 : $allocation(infoMap) = presentMap$
axm8 : $allocation(emptyI) = emptyP$

END

MACHINE CityMap
SEES InformationsCityMap
VARIABLES

$i\ d\ p\ varSeq$

INVARIANTS

inv1 : $i \in Information$
inv2 : $d \in InterpretationDomain$
inv3 : $p \in Presentation$
inv4 : $varSeq \in \{0, 1\}$

VARIANT

$varSeq$

EVENTS
Initialisation
begin

act1 : $i :\in Information$
act2 : $d :\in InterpretationDomain$
act3 : $p :\in Presentation$
act4 : $varSeq := 1$

end

Event $information \widehat{=}$ /* délivrance de $infoCityMap$ */
Status convergent
when

grd1 : $varSeq = 1$

then

act1 : $i := infoCityMap$
act2 : $varSeq := varSeq - 1$

end
Event $infoCityMap \widehat{=}$ /* interprétation de $infoCityMap$ */
when

grd1 : $varSeq = 0$

then

act1 : $d := interpretation(i)$
act3 : $p := allocation(i)$

end
END

Figure 4.14 – *CityMap* : le modèle abstrait

4.7.2 Modèle de fission sémantique

Le modèle raffiné de fission sémantique (voir Figure 4.15) formalise la fission sémantique de $infoCityMap$. Il correspond à la partie droite de la définition 4.8 et se compose des éléments suivants.

– le contexte instancié *ComplementaryFission* qui étend aussi bien le contexte abstrait $InformationCityMap$, que le contexte générique *SemanticFission* en instanciant l'opérateur générique *semanticOperator* par l'opérateur spécifique *complementaryOperator* ($axm1$, $axm2$, $axm3$, $axm4$, $axm5$).

CONTEXT ComplementaryFission
EXTENDS SemanticFission InformationsCityMap
CONSTANTS

complementaryOperator

AXIOMS

axm1 : $complementaryOperator \subseteq semanticOperator$
axm2 : $dom(complementaryOperator) = \{See \mapsto Map, emptyD \mapsto Map, See \mapsto emptyD\}$
axm3 : $complementaryOperator(See \mapsto Map) = CityMap$
axm4 : $complementaryOperator(emptyD \mapsto Map) = emptyD$
axm5 : $complementaryOperator(See \mapsto emptyD) = emptyD$

END

MACHINE parallelCityMap
REFINES CityMap
SEES ComplementaryFission
VARIABLES

i d p varSeq varPar1 varPar2 i1 i2 d1 d2

INVARIANTS

inv1 : $varPar1 \in \{0,1\}$
inv2 : $varPar2 \in \{0,1\}$
inv3 : $varSeq < 1 \Rightarrow i = infoCityMap \wedge i1 = infoSee \wedge i2 = infoMap$
inv4 : $varSeq < 1 \Rightarrow interpretation(i) = semanticOperator(interpretation(i1) \mapsto interpretation(i2))$
inv5 : $varPar1 = 0 \wedge varPar2 = 0 \Rightarrow d1 = interpretation(i1) \wedge d2 = interpretation(i2)$ ...

VARIANT

$varPar1 + varPar2$

EVENTS
Initialisation
begin
act1 : $varSeq := 1$
act2 : $varPar1 := 1$ ...
act3 : $varPar2 := 1$ ...
end

Event $information \widehat{=}$ /* délivrance de $infoCityMap$ */
Status convergent
refines *information*
when
grd1 : $varSeq = 1$
grd2 : $varPar1 = 1$
grd3 : $varPar2 = 1$
then
act1 : $i, i1, i2 := infoCityMap, infoSee, infoMap$
act2 : $varSeq := varSeq - 1$
end

Event $infoCityMap \widehat{=}$ /* interprétation de $infoCityMap$ */
refines *infoCityMap*
when
grd1 : $varSeq = 0$
grd2 : $varPar1 = 0$
grd3 : $varPar2 = 0$
then
act1 : $d := complementaryOperator(d1 \mapsto d2)$
act2 : $p := allocation(i)$
end

Event $infoSee \widehat{=}$ /* interprétation de $infoSee$ */
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varPar1 = 1$
then
act1 : $d1 := interpretation(i1)$
act2 : $varPar1 := varPar1 - 1$
end

Event $infoMap \widehat{=}$ /* interprétation de $infoMap$ */
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varPar2 = 1$
then
act1 : $d2 := interpretation(i2)$
act2 : $varPar2 := varPar2 - 1$
end

END

Figure 4.15 – *CityMap* : le 1er raffinement

– la machine *ParallelCityMap* qui enrichit la machine *FissionedInformation* par le modèle de l'opérateur temporel parallèle en remplaçant le variant générique *var* par le variant

spécifique *varPar*1 + *varPar*1. Il permet de déclencher de manière parallèle les évènements *infoSee* (*grd*2) et *infoMap* (*grd*2) qui produisent respectivement *See* et *Map*, les interprétations relatives aux informations *infoSee* et *infoMap*. Ils sont suivis de l'évènement *infoCityMap* (*grd*2, *grd*3) qui produit *CityMap*, la présentation relative à l'information fissionnée *infoCityMap* en combinant les interprétations complémentaires *See* et *Map* (*act*1).

4.7.3 Modèle d'allocation : combinaison

Le premier modèle raffiné d'allocation (voir Figure 4.16) formalise la combinaison des présentations qui restituent *infoSee*, *infoMap*. Il correspond à la partie droite de la définition 4.9 et se compose des éléments suivants.

- le contexte instancié *AllocationCityMapCombination* qui étend le contexte précédent *ComplementaryFission* mais également le contexte générique *Allocation* en instanciant l'opérateur générique *combinationOperator* par l'opérateur spécifique complémentaire *PcomplementaryOperator* (*axm*1, *axm*2, *axm*3, *axm*4, *axm*5).
- la machine *CityMapPresentation* qui enrichit la machine *Presentation* en précisant le modèle de l'opérateur temporel en remplaçant le variant générique *var* par le variant spécifique *varPar*1 + *varPar*1. Ceci permet de déclencher en parallèle *presentationSee* (*grd*2), *presentationMap* (*grd*2) suivis de *presentationCityMap* (*grd*2, *grd*3) qui produisent respectivement les présentations *presentSee*, *presentMap* et *presentCityMap* dans le même ordre temporel que les évènements qu'ils raffinent *infoSee*, *infoMap* et *infoCityMap*. La présentation relative à l'information *infoCityMap* est obtenue par la combinaison complémentaire des présentations construites *presentSee*, *presentMap* dans l'événement *presentCityMap* (*act*2).

CONTEXT AllocationCityMapCombination
EXTENDS Allocation ComplementaryFission
CONSTANTS
PcomplementaryOperator
AXIOMS
axm1 : $PcomplementaryOperator \subseteq combinationOperator$
axm2 : $PcomplementaryOperator(presentSee \mapsto presentMap) = presentCityMap$
axm3 : $dom(PcomplementaryOperator) = \{presentSee \mapsto presentMap, emptyP \mapsto presentMap, presentSee \mapsto emptyP, presentSeeSpeech \mapsto presentSeeExpression, presentSeeSpeech \mapsto emptyP, emptyP \mapsto presentSeeExpression\}$
axm4 : $PcomplementaryOperator(presentSee \mapsto presentMap) = presentCityMap$
axm5 : $PcomplementaryOperator(presentSee \mapsto emptyP) = emptyP$
axm6 : $PcomplementaryOperator(emptyP \mapsto presentMap) = emptyP$
END

MACHINE CityMapPresentation
REFINES parallelCityMap
SEES AllocationCityMapCombination
VARIABLES
i d p i1 i2 d1 d2 p1 p2 varSeq varPar1 varPar2
INVARIANTS
inv1 : $varSeq < 1 \Rightarrow allocation(i) = combinationOperator(allocation(i1) \mapsto allocation(i2))$
inv2 : $varPar1 = 0 \wedge varPar2 = 0 \Rightarrow p1 = allocation(i1) \wedge p2 = allocation(i2)$ ...
VARIANT
$varPar1 + varPar2$
EVENTS
Initialisation
begin
act1 : $varSeq := 1$
act2 : $varPar1 := 1$
act3 : $varPar2 := 1$ ...
end
Event *information* $\widehat{=}$ /* délivrance de *infoCityMap* */
...
Event *presentationCityMap* $\widehat{=}$ /* construction de *presentCityMap* */
refines *infoCityMap*
when
grd1 : $varSeq = 0$
grd2 : $varPar1 = 0$
grd3 : $varPar2 = 0$
then
act1 : $d := complementaryOperator(d1 \mapsto d2)$
act2 : $p := PcomplementaryOperator(p1 \mapsto p2)$
end

Event *presentationSee* $\widehat{=}$ /* construction de *presentSee* */
Status convergent
extends *infoSee*
when
grd1 : $varSeq = 0$
grd2 : $varPar1 = 1$
then
act1 : $d1 := interpretation(i1)$
act2 : $varPar1 := varPar1 - 1$
act3 : $p1 := allocation(i1)$
end
Event *presentationMap* $\widehat{=}$ /* construction de *presentMap* */
Status convergent
extends *infoMap*
when
grd1 : $varSeq = 0$
grd2 : $varPar2 = 1$
then
act1 : $d2 := interpretation(i2)$
act2 : $varPar2 := varPar2 - 1$
act3 : $p2 := allocation(i2)$
end
END

Figure 4.16 – *CityMap* : le 2ème raffinement

4.7.4 Modèle d'allocation : décomposition

Le second modèle raffiné d'allocation (voir Figure 4.17) formalise la décomposition de la présentation *presentSee* en deux présentations complémentaires *presentSeeSpeech* et

presentSeeExpression. Il correspond à la partie droite de la définition 4.10 et se compose des éléments suivants.

CONTEXT AllocationCityMapDecomposition
EXTENDS AllocationCityMapCombination
AXIOMS
axm1 : $PcomplementaryOperator(presentSeeSpeech \mapsto presentSeeExpression) = presentSee$
axm2 : $PcomplementaryOperator(presentSeeSpeech \mapsto emptyP) = emptyP$
axm3 : $PcomplementaryOperator(emptyP \mapsto presentSeeExpression) = emptyP$
END

MACHINE ComplementaryCityMapPresentation
REFINES CityMapPresentation
SEES AllocationCityMapDecomposition
VARIABLES
p11 p12 varPar1 varPar2 varPar3 varPar4 i d p i1 i2 d1 d2 p1 p2

INVARIANTS
inv1 : $p11 \in Presentation$
inv2 : $varPar3 \in \{0,1\}$
inv3 : $varPar4 \in \{0,1\}$
inv4 : $varPar3 = 0 \wedge varPar4 = 0 \Rightarrow p11 = presentSeeSpeech \wedge p12 = presentSeeExpression$ ...
VARIANT
$varPar3 + varPar4$
EVENTS
Initialisation
begin
act1 : $p11 :\in Presentation$
act2 : $varPar3 := 1$
act3 : $varPar4 := 1$ ...
end
Event *information* $\widehat{=}$ /* délivrance de $infoCityMap$ */
...
Event *presentationCityMap* $\widehat{=}$ /* construction de *presentCityMap* */
...
Event *presentationSee* $\widehat{=}$ /* construction de *presentSee* */
Status convergent
refines *presentationSee*
when
grd1 : $varSeq = 0$
grd2 : $varPar1 = 1$
grd3 : $varPar3 = 0$
grd4 : $varPar4 = 0$
then
act1 : $d1 := interpretation(i1)$
act2 : $varPar1 := varPar1 - 1$
act3 : $p1 := PcomplementaryOperator(p11 \mapsto p12)$
end

Event *presentationMap* $\widehat{=}$ /* construction de *presentMap* */
...
Event *presentationSeeSpeech* $\widehat{=}$ /* construction de *presentSeeSpeech* */
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varPar1 = 1$
grd3 : $varPar3 = 1$
then
act1 : $p11 := presentSeeSpeech$
act2 : $varPar3 := 0$
end
Event *presentationSeeExpression* $\widehat{=}$ /* construction de *presentSeeExpression* */
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varPar1 = 1$
grd3 : $varPar4 = 1$
then
act1 : $p12 := presentSeeExpression$
act2 : $varPar4 := 0$
end
END

Figure 4.17 – *CityMap* : le 3ème raffinement

– le contexte instancié *AllocationCityMapDecomposition* qui étend le contexte précédent *AllocationCityMapCombination*, et le contexte générique *Allocation* en instanciant l'opérateur *combinationOperator* par l'opérateur *PcomplementaryOperator* (*axm*1, *axm*2, *axm*3).

– la machine *ComplementaryCityMapPresentation* qui enrichit *Presentation*, la machine générique en précisant le modèle de l'opérateur temporel en remplaçant le variant générique *var* par le variant spécifique *varPar*3 + *varPar*4. Ceci permet de déclencher en parallèle les événements *presentationSeeSpeech* (*grd*3) et *presentationSeeExpression* (*grd*3) qui produisent respectivement les présentations composites *presentSeeSpeech* et *presentSeeExpression*. Ils sont suivis de l'évènement *presentationSee* (*grd*3, *grd*4) qui produit *presentSee* en combinant les présentations complémentaires *presentSeeSpeech* et *presentSeeExpression*.

4.7.5 Modèle d'affectation

Le modèle raffiné d'affectation (voir Figure 4.18) formalise l'affectation de chacune des présentations *presentSeeSpeech*, *presentSeeExpression* et *presentMap* avec les couples modalité/média qui les restituent à l'utilisateur. Il correspond à la partie droite des définitions 4.11, 4.12 et 4.13 et comprend les composants suivants.

– le contexte instancié $Affectation CityMap$ qui étend d'une part, le contexte de décomposition *AllocationCityMapDecomposition*, et le contexte générique $Affectation$ par la définition des ensembles *presentationUnit*, $modality$ et *media* et de la fonction $affectation$ (*axm*1, *axm*2, *axm*3).

– la machine $AffectedCityMapPresentation$ qui enrichit la machine $AffectedPresentation$ en précisant le modèle de l'opérateur temporel qui permet de déclencher de manière parallèle les évènements $affectationSeeSpeech$ (*grd*2), $affectationSeeExpression$ (*grd*2) et $affectationMap$ (*grd*2) affectant respectivement *presentSeeSpeech*, *presentSeeExpression* et *presentMap*. Ces évènements sont déclenchés après la construction des présentations *presentSeeSpeech*, *presentSeeExpression* et *presentMap* respectivement dans les évènements *presentationSeeSpeech*, *presentationSeeExpression* et *presentationMap*. Ainsi, la valeur de *m* est instanciée dans les trois évènements $affectationSeeSpeech$ (*grd*1), $affectationSeeExpression$ (*grd*1) et $affectationMap$ (*grd*1) et les présentations sont affectées avec les couples (modalité, média) correspondants.

CONTEXT AffectationCityMap
EXTENDS Affectation AllocationCityMapDecomposition
SETS

$presentationUnit = \{presentMap, presentSeeSpeech, presentSeeExpression\}$
$Modality = \{picture, speech, expression\}$
$Media = \{speaker, screen\}$

AXIOMS

axm1 : $affectation(presentSeeSpeech) = speech$
axm2 : $affectation(presentSeeExpression) = expression$
axm3 : $affectation(presentMap) = picture$

END

MACHINE AffectedCityMapPresentation
REFINES ComplementaryCityMapPresentation
SEES AffectationCityMap
VARIABLES

item varSeq varPar1 varPar2 varPar3 ...

INVARIANTS

inv1 : $item \in presentationUnit \rightarrow Modality \times Media$

EVENTS
Initialisation
begin

act1 : $varSeq := 1$
act2 : $varSeq1 := 2$
act3 : $varPar1 := 1$
act4 : $varPar2 := 1$
act5 : $item :\in presentationUnit \rightarrow Modality \times Media$

end

Event *information* $\widehat{=}$ /* délivrance de *infoCityMap* */
...
Event *presentationCityMap* $\widehat{=}$ /* construction de *presentCityMap* */
...
Event *presentationSee* $\widehat{=}$ /* construction de *presentSee* */
...
Event *presentationMap* $\widehat{=}$ /* construction de *presentMap* */
...
Event *presentationSeeSpeech* $\widehat{=}$ /* construction de *presentSeeSpeech* */
...
Event *presentationSeeExpression* $\widehat{=}$ /* construction de *presentSeeExpression* */
...

Event *affectationMap* $\widehat{=}$ /* affectation de *presentMap* */
any
m
where
grd1 : $m \in linkModalityMedia(affectation(presentMap))$
grd2 : $varPar2 = 0$
grd3 : $varSeq = 0$
then
act1 : $item(p1) := (affectation(p2) \mapsto m)$
end

Event *affectationSeeSpeech* $\widehat{=}$ /* affectation de *presentSeeSpeech* */
any
m
where
grd1 : $m \in linkModalityMedia(affectation(presentSeeSpeech))$
grd2 : $varPar3 = 0$
grd3 : $varSeq = 0$
then
act1 : $item(p11) := (affectation(p11) \mapsto m)$
end

Event *affectationSeeExpression* $\widehat{=}$ /* affectation de *presentSeeExpression* */
any
m
where
grd1 : $m \in linkModalityMedia(affectation(presentSeeExpression))$
grd2 : $varPar4 = 0$
grd3 : $varSeq = 0$
then
act1 : $item(p12) := (affectation(p12) \mapsto m)$
end

END

Figure 4.18 – *CityMap* : le 4ème raffinement

4.7.6 Bilan des obligations de preuve

L'instanciation des modèles génériques pour formaliser l'étude de cas a engendré des enrichissements et des raffinements qui ont généré 98 obligations de preuves (voir Figure 4.4) dont 88 ont été prouvées de manière automatique et 10 ont nécessité l'intervention du concepteur dans une preuve interactive : 1 obligation de preuve portant sur la bonne définition de l'axiome (*axm*3) dans *ComplementaryFission*, 2 obligations de preuve de préservation d'invariant sur *inv*4 et *inv*5 dans *ParallelCityMap*, 1 obligation de preuve portant sur la bonne définition de l'axiome (*axm*2) dans *AllocationCityMapCombination*, 2 obligations de preuve de préservation d'invariant sur *inv*1 et *inv*2 dans *CityMapPresentation*, 1 obligation de preuve portant sur la simulation de l'action *presentSee*/*act*3 dans *ComplementaryCityMapPresentation* et 3 obligations de preuve de préservation d'invariant sur *inv*1 dans *AffectedCityMapPresentation*.

Composant	OP automatiques	OP manuelles	OP totales	OP non prouvées
InformationCityMap	8	0	8	0
CityMap	9	0	9	0
ComplementaryFission	2	1	3	0
ParallelCityMap	24	2	26	0
AllocationCityMapCombination	4	1	5	0
CityMapPresentation	12	2	14	0
AllocationCityMapDecomposition	3	0	3	0
ComplementaryCityMapPresentation	15	1	16	0
AffectationCityMap	3	0	3	0
AffectedCityMapPresentation	8	3	11	0
Total	**88**	**10**	**98**	**0**

Tableau 4.4 – *CityMap* : bilan des obligations de preuves

5 Conclusion

Nous avons présenté dans ce chapitre une formalisation B Évènementiel du modèle conceptuel que nous proposons. Cette formalisation s'articule autour d'une démarche de modélisation qui précise l'enchaînement des développements B Évènementiel à effectuer et d'un ensemble de modèles génériques qui formalisent les différentes étapes de construction de l'interface à partir du modèle de conception formel décrit dans le chapitre 3.

La combinaison du processus de développement B Évènementiel et des modèles génériques constitue une approche générique de développement B Évènementiel pour les IHM

multimodales en sortie. Nous avons également présenté une approche d'instanciation des modèles génériques B Événementiel permettant, moyennant l'application de mécanismes de raffinements, de développer les modèles B Évènementiel qui spécifient une IHM multimodale en sortie concrète.

Les modèles génériques B Évènementiel représentent des modèles cadres à partir desquels les modèles B Évènementiel détaillés de la fission sémantique et de l'allocation sont dérivés respectivement dans les chapitre 5 et chapitre 6. Ils présentent les modèles spécifiques aux différents schémas de composition sémantique et temporelle lors des phases de fission sémantique et allocation.

Chapitre 5

Modélisation de la fission sémantique avec B Évènementiel : Généralisation

1 Introduction

Dans le chapitre précédent, nous avons présenté les développements B Évènementiel génériques pour le développement d'une interface multimodale en sortie. Ces modèles génériques développés suivant un processus incrémental, formalisent les deux étapes de construction de l'interface multimodale en sortie : la fission sémantique et l'allocation. Nous avons également présenté des mécanismes de raffinement et d'enrichissement qui permettent la mise en œuvre de ces modèles sur une étude de cas. Dans ce chapitre, nous présentons les modèles B Évènementiel spécifiques à la fission sémantique. Ils constituent des instanciations du modèle générique de fission sémantique présenté dans le chapitre 4. Ils formalisent la totalité des cas de fission sémantique : binaire et itérative et définissent de manière détaillée les contextes et machines qui correspondent aux différents schémas de combinaison sémantique et temporelle des informations fissionnées. Ces modèles spécifiques, peuvent donc être directement instanciés et combinés (contexte/ machine) afin de formaliser une IHM multimodale en sortie décrite selon le modèle formel générique présenté en chapitre 3.

2 Démarche de modélisation avec B Évènementiel

La formalisation B Évènementiel de la fission sémantique utilise la démarche de modélisation des modèles génériques présentée dans le chapitre 4. Il s'agit d'une démarche incrémentale par raffinements successifs, basée sur le principe de séparation entre sémantique statique et dynamique. La démarche de modélisation B Évènementiel de la fission sémantique débute par l'invocation du modèle générique abstrait, il est commun à tous les

cas de fission sémantique et constitue la première spécification de l'interface multimodale en sortie. Il est raffiné par le modèle de fission sémantique. Ce dernier enrichit le modèle générique de fission sémantique en formalisant les schémas de combinaison temporelle et sémantique des informations fissionnées. Par conséquent, les différents modèles B Évènementiel de fission sémantique constituent les différentes réalisations possibles du modèle générique de fission sémantique. Chaque contexte spécifique correspond à un cas spécifique de combinaison sémantique du contexte générique et chaque machine spécifique correspond à un cas spécifique d'ordonnancement temporel des informations fissionnées dans la machine.

3 Le modèle de fission sémantique

Le modèle de fission sémantique modélise la production des informations fissionnées, il correspond à la partie droite de la règle syntaxique de la fission sémantique que nous rappelons dans 5.1. La fission sémantique est dite binaire lorsqu'elle produit les informations $i1$ et $i2$ et itérative lorsqu'elle produit n fois l'information $i1$. Nous présentons ci-après les raffinements correspondant aux cas de fission binaire et itérative.

$$I ::= UIE \mid (op_{temp}, op_{sem})(I, I) \mid It(n, I) \; avec \; n \in \mathbb{N} \tag{5.1}$$

3.1 Le modèle de fission sémantique binaire

Le modèle de la fission sémantique binaire modélise la restitution de l'information i par la production des informations fissionnées $i1$ et $i2$. Il est dérivé du modèle générique de fission sémantique présenté dans la Figure 4.10. Il définit les différentes réalisations du contexte générique *SemanticFission* selon les opérateurs sémantiques op_{sem}, et il enrichit la machine générique $FissionedInformation$ selon les opérateurs temporels op_{temp}. Nous détaillons ci-après ces différentes réalisations pour les contextes et machines.

3.1.1 Le composant CONTEXT

Les contextes spécifiques découlant du contexte générique $Semanticfission$, sont obtenus par la définition de l'opérateur sémantique employé pour combiner les interprétations sémantiques des informations fissionnées (concurrent, complémentaire, complémentaire et redondant, partiellement redondant et totalement redondant). Par conséquent, il existe au-

tant de variantes de contextes que d'opérateurs sémantiques. Ces variantes sont présentées ci-dessous.

1. **Le cas concurrent.** Le CONTEXT *ConcurrentFission* (voir Figure 5.1) définit l'opérateur sémantique concurrent (*concurrentOperator*). L'axiome (*axm*2) définit la concurrence sémantique. Il exprime que la production séparée (combinée à l'interprétation vide) de chacune des deux interprétations concurrentes n'altère pas leur sémantique.

CONTEXT ConcurrentFission
CONSTANTS
concurrentOperator
AXIOMS
axm1 : $concurrentOperator \in InterpretationDomain \times InterpretationDomain \rightarrow InterpretationDomain$
axm2 : $\forall x, y \cdot (x \in InterpretationDomain \setminus \{emptyD\} \land y \in InterpretationDomain \setminus \{emptyD\} \land x \mapsto y \in dom(concurrentOperator) \land emptyD \mapsto y \in dom(concurrentOperator) \land x \mapsto emptyD \in dom(concurrentOperator) \Rightarrow (concurrentOperator(emptyD \mapsto y) = y \land concurrentOperator(x \mapsto emptyD) = x))$
END

Figure 5.1 – Le CONTEXT concurrent

2. **Le cas complémentaire.** Le CONTEXT *ComplementaryFission* (voir Figure 5.2) définit l'opérateur sémantique complémentaire (*complementaryOperator*). L'axiome (*axm*2) définit la complémentarité sémantique. Il exprime que deux interprétations complémentaires ne peuvent être produites séparément (combinée à l'interprétation vide) sans perdre leur sémantique.

CONTEXT ComplementaryFission
CONSTANTS
complementaryOperator
AXIOMS
axm1 : $complementaryOperator \in InterpretationDomain \times InterpretationDomain \rightarrow InterpretationDomain$
axm2 : $\forall x, y \cdot (x \in InterpretationDomain \setminus \{emptyD\} \land y \in InterpretationDomain \setminus \{emptyD\} \land x \mapsto y \in dom(complementaryOperator) \land emptyD \mapsto y \in dom(complementaryOperator) \land x \mapsto emptyD \in dom(complementaryOperator) \Rightarrow (complementaryOperator(emptyD \mapsto y) = emptyD \land complementaryOperator(x \mapsto emptyD) = emptyD))$
END

Figure 5.2 – Le CONTEXT complémentaire

3. **Le cas partiellement redondant.** Le CONTEXT *PartialRedundantFission* (voir Figure 5.3) définit l'opérateur sémantique partiellement redondant (*partialredundantOperator*). L'axiome (*axm*2) définit la redondance sémantique partielle. Il exprime que dans une combinaison partiellement redondante, la première interprétation sémantique est ex-

primée par la deuxième. Par conséquent, la première interprétation sémantique peut être remplacée par l'interprétation vide.

CONTEXT PartialRedundantFission
CONSTANTS
partialredundantOperator
AXIOMS
axm1 : $partialredundantOperator \in InterpretationDomain \times InterpretationDomain \rightarrow InterpretationDomain$
axm2 : $\forall x, y \cdot (x \in InterpretationDomain \setminus \{emptyD\} \land y \in InterpretationDomain \setminus \{emptyD\} \land x \mapsto y \in dom(partialredundantOperator) \land emptyD \mapsto y \in dom(partialredundantOperator) \Rightarrow (partialredundantOperator(x \mapsto y) = partialredundantOperator(emptyD \mapsto y)$
END

Figure 5.3 – Le CONTEXT partiellement redondant

4. **Le cas totalement redondant.** Le CONTEXT *TotalRedundantFission* (voir Figure 5.4) définit l'opérateur sémantique totalement redondant (*totalredundantOperator*). L'axiome (*axm*2) définit la redondance sémantique totale. Il exprime que dans une combinaison totalement redondante, la première interprétation sémantique est exprimée par la deuxième et vice versa. Par conséquent, la première ou la deuxième interprétation peut être remplacée par l'interprétation vide.

CONTEXT TotalRedundantFission
CONSTANTS
totalredundantOperator
AXIOMS
axm1 : $totalredundantOperator \in InterpretationDomain \times InterpretationDomain \rightarrow InterpretationDomain$
axm2 : $\forall x, y \cdot (x \in InterpretationDomain \setminus \{emptyD\} \land y \in InterpretationDomain \setminus \{emptyD\} \land x \mapsto y \in dom(totalredundantOperator) \land emptyD \mapsto y \in dom(totalredundantOperator) \land x \mapsto emptyD \in dom(totalredundantOperator) \Rightarrow totalredundantOperator(x \mapsto y) = totalredundantOperator(emptyD \mapsto y) \land totalredundantOperator(x \mapsto y) = totalredundantOperator(x \mapsto emptyD)$
END

Figure 5.4 – Le CONTEXT totalement redondant

5. **Le cas complémentaire et redondant.** Le cas complémentaire et redondant requiert l'utilisation d'une sémantique dotée d'une relation d'ordre et d'inclusion sémantiques. La sémantique que nous utilisons n'est pas dotée de cette relation d'inclusion, par conséquent, le cas complémentaire et redondant ne peut être exprimé selon le modèle sémantique choisi.

3.1.2 Le composant MACHINE

Les machines spécifiques de fission sémantique formalisent l'ordonnancement temporel de production des informations fissionnées. Par conséquent, elles spécialisent la MACHINE générique *FissionedInformation* en fonction des opérateurs temporels : anachronique, séquentiel, concomittant, coïncidant, parallèle, choix et indépendant. Pour définir ces différents variantes d'ordonnancement temporel, nous nous basons sur les travaux de [167]. Les auteurs ont formalisé les opérateurs de composition temporelle : séquentiel, parallèle, choix et itération dans B Évènementiel. Ils ont également montré que ces quatre opérateurs sont des opérateurs de base permettant d'exprimer tout autre type de composition temporelle dans une algèbre de processus. Nous proposons d'utiliser ces modèles pour enrichir la machine générique *FissionedInformation*.
Nous définissons donc trois machines spécifiques dédiées à la fission sémantique suivant l'ordonnancement : séquentiel, parallèle et par choix. Les machines relatives aux opérateurs temporels : anachronique, concomittant, coïncident et indépendant sont dérivées des définitions de ces opérateurs en fonction des opérateurs de base comme suit.

1. **L'ordonnancement anachronique.** La composition anachronique de deux évènements x et y est définie par la composition séquentielle des trois évènements : x, *emptyEvent* et y, où *emptyEvent* est un évènement dont l'action consiste en la substitution vide *skip*. La séquence étant un opérateur associatif, la composition séquentielle des trois évènements peut être exprimée par deux applications successives de la séquence binaire . Ainsi, l'opérateur anachronique est défini comme suit :

$$An(x, y) = Sq(Sq(x, emptyEvent), y)$$

2. **L'ordonnancement concomittant.** La composition concomitante de deux évènements x et y est définie seulement si x et y peuvent être exprimés par la composition séquentielle de deux évènements ($x1$ et $x2$ pour x, $y1$ et $y2$ pour y). Ainsi, l'opérateur concomittant est défini par la composition séquentielle des trois évènements : $x1$, la composition parallèle de $x2$, $y1$ et $y2$.

$$x = Sq(x1, x2) \wedge y = Sq(y1, y2) \Rightarrow Cc(x, y) = Sq(Sq(x1, Pl(x2, y1)), y2)$$

3. **L'ordonnancement coïncident.** La composition coïncidente de deux évènements x et y est définie seulement si le premier évènement x peut être exprimé par la composition séquentielle de trois évènements $x1$, $x2$ et $x3$. L'opérateur coïncident est, par conséquent,

défini par la composition séquentielle des trois évènements : $x1$, la composition parallèle de $x2$, y et $x3$.

$$x = Sq(Sq(x1, x2), x3) \Rightarrow Cd(x, y) = Sq(Sq(x1, Pl(x2, y)), x3).$$

4. **L'ordonnancement indépendant.**
   La composition indépendante de deux évènements x et y dénote une composition temporelle indéterminée. Nous la définissons par la disjonction des compositions : séquentielle et parallèle.

$$In(x, y) = Sq(x, y) \vee Pl(x, y)$$

Les machines spécifiques aux opérateurs : séquentiel, parallèle et choix se distinguent de la machine générique *FissionedInformation* par le fait qu'ellent substituent le variant générique *var* par un variant spécifique. Ce dernier permet d'ordonnancer les événements produisant les informations fissionnées selon l'ordre établi par l'opérateur temporel qui les combine. Nous présentons ci-après ces machines, combinées au contexte correspondant à l'opérateur sémantique qui relie les informations fissionnées (concurrent, complémentaire, complémentaire et redondant, partiellement redondant, totalement redondant). Nous représentons cet opérateur sémantique par *semanticOperator* et par le contexte générique *Semanticfission* dans la clause *SEES*.

1. **L'ordonnancement séquentiel.** La machine *SequentialFissionedInformation* (voir Figure 5.5) enrichit la machine générique *FissionedInformation* en remplaçant le variant générique *var* par le variant spécifique *varSeq*1 initialisé à 2 (*act*2). Il conditionne l'évènement *information* par la garde (*grd*6), qui assure que les interprétations d, $d1$ et $d2$ sont calculées après la délivrance des informations i, $i1$ et $i2$. Il est ensuite décrémenté successivement dans les évènements *interpretation*1 (*act*2) et *interpretation*2 (*act*2) déclenchant ainsi de manière séquentielle les évènements *interpretation*1, *interpretation*2 et *interpretation* par les gardes (*grd*2). L'invariant de collage (*inv*3) permet d'établir les valeurs de $d1$ et $d2$ en fonction de $i1$ et $i2$ calculées respectivement dans les évènements *interpretation*1 et *interpretation*2.

MACHINE SequentialFissionedInformation
REFINES Information
SEES SemanticFission
VARIABLES

$i\ d\ p\ varSeq\ varSeq1\ i1\ i2\ d1\ d2$

INVARIANTS

inv1 : $varSeq1 \in \{0, 1, 2\}$
inv2 : $varSeq < 1 \Rightarrow interpretation(i) = semanticOperator(interpretation(i1) \mapsto interpretation(i2))$
inv3 : $varSeq1 = 0 \Rightarrow d1 = interpretation(i1) \wedge d2 = interpretation(i2)$ ...

VARIANT

$varSeq1$

EVENTS
Initialisation
begin
act1 : $varSeq := 1$
act2 : $varSeq1 := 2$ ...
end

Event $information \widehat{=}$ /*Délivrance de i */
Status convergent
refines $information$
any
$x\ x1\ x2$
where
grd1 : $x \in Information$
grd2 : $x1 \in Information$
grd3 : $x2 \in Information$
grd4 : $interpretation(x) = semanticOperator(interpretation(x1) \mapsto interpretation(x2))$
grd5 : $varSeq = 1$
grd6 : $varSeq1 = 2$
then
act1 : $i, i1, i2 := x, x1, x2$
act2 : $varSeq := varSeq - 1$
end

Event $interpretation \widehat{=}$ /*calcul de d et construction de p*/
refines $interpretation$
when
grd1 : $varSeq = 0$
grd2 : $varSeq1 = 0$
then
act1 : $d := semanticOperator(d1 \mapsto d2)$
act2 : $p := allocation(i)$
end

Event $interpretation1 \widehat{=}$ /*calcul de $d1$*/
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varSeq1 = 2$
then
act1 : $d1 := interpretation(i1)$
act2 : $varSeq1 := varSeq1 - 1$
end

Event $interpretation2 \widehat{=}$ /*calcul de $d2$*/
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varSeq1 = 1$
then
act1 : $d2 := interpretation(i2)$
act2 : $varSeq1 := varSeq1 - 1$
end

END

Figure 5.5 – L'ordonnancement séquentiel

2. **L'ordonnancement parallèle.** La machine *ParallelFissionedInformation* (voir Figure 5.6) enrichit la machine générique *FissionedInformation* en remplaçant le variant générique *var* par la combinaison de deux variants spécifiques *varPar*1 et *varPar*2 initialisés à 1(*act*2 et *act*3). Ils conditionnent l'évènement *information* par les gardes (*grd*6) et (*grd*7) qui assurent que les interprétations d, $d1$ et $d2$ sont calculées après la délivrance de i, $i1$ et $i2$. *varPar*1 déclenche l'événement *interpretation*1 (*grd*2) qui le décrémente à 1 (*act*2)

et *varPar*2 déclenche l'événement *interpretation*2 (*grd*2) qui le décrémente à 0 (*act*2). L'évènement *interpretation* est déclenché lorsque *varPar*1 et *varPar*2 sont à 0 (*grd*2 et *grd*3). Les invariants de collage (*inv*4 et *inv*5) permettent d'établir les valeurs de *d*1 et *d*2 en fonction de *i*1 et *i*2, respectivement calculées dans les évènements *interpretation*1 et *interpretation*2.

MACHINE ParallelFissionedInformation
REFINES Information
SEES SemanticFission
VARIABLES

i d p varSeq varPar1 varPar2 i1 i2 d1 d2

INVARIANTS

inv1 : $varPar1 \in \{0,1\}$
inv2 : $varPar2 \in \{0,1\}$
inv3 : $varSeq < 1 \Rightarrow interpretation(i) = semanticOperator(interpretation(i1) \mapsto interpretation(i2))$
inv4 : $varPar1 = 0 \Rightarrow d1 = interpretation(i1)$
inv5 : $varPar2 = 0 \Rightarrow d2 = interpretation(i2)$ ...

VARIANT

$varPar1 + varPar2$

EVENTS
Initialisation
begin
act1 : $varSeq := 1$
act2 : $varPar1 := 1$
act3 : $varPar2 := 1$ ...
end

Event *information* $\widehat{=}$ /*Délivrance de *i* */
Status convergent
refines *information*
any
x x1 x2
where
grd1 : $x \in Information$
grd2 : $x1 \in Information$
grd3 : $x2 \in Information$
grd4 : $interpretation(x) = semanticOperator(interpretation(x1) \mapsto interpretation(x2))$
grd5 : $varSeq = 1$
grd6 : $varPar1 = 1$
grd7 : $varPar2 = 1$
then
act1 : $i, i1, i2 := x, x1, x2$
act2 : $varSeq := varSeq - 1$
end
END

Event *interpretation* $\widehat{=}$ /*calcul de *d* et construction de *p**/
refines *interpretation*
when
grd1 : $varSeq = 0$
grd2 : $varPar1 = 0$
grd3 : $varPar2 = 0$
then
act1 : $d := semanticOperator(d1 \mapsto d2)$
act2 : $p := allocation(i)$
end

Event *interpretation1* $\widehat{=}$ /*calcul de *d*1*/
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varPar1 = 1$
then
act1 : $d1 := interpretation(i1)$
act2 : $varPar1 := varPar1 - 1$
end

Event *interpretation2* $\widehat{=}$ /*calcul de *d*1*/
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varPar2 = 1$
then
act1 : $d2 := interpretation(i2)$
act2 : $varPar2 := varPar2 - 1$
end

Figure 5.6 – L'ordonnancement parallèle

3. **L'ordonnancement par choix.** La machine *ChoiceFissionedInformation* (voir Figure 5.7) enrichit la machine générique *FissionedInformation* en introduisant une variable supplémentaire *dChoice* qui représente l'interprétation choisie, et en remplaçant le variant générique *var* par le variant spécifique *varChoice*.

MACHINE ChoiceFissionedInformation
REFINES Information
SEES SemanticFission
VARIABLES
i d p varSeq varChoice i1 i2 d1 d2 dChoice
INVARIANTS
inv1 : $varChoice \in \{0,1,2\}$
inv2 : $dChoice \in InterpretationDomain$
inv3 : $varSeq < 1 \Rightarrow interpretation(i) = semanticOperator(interpretation(i1) \mapsto interpretation(i2))$
inv4 : $varChoice = 0 \Rightarrow (d1 = interpretation(i1) \wedge dChoice = d1) \vee (d2 = interpretation(i2) \wedge dChoice = d2)$ ...
VARIANT
varChoice

EVENTS
Initialisation
begin
act1 : $dChoice :\in InterpretationDomain$
act2 : $varSeq := 1$
act3 : $varChoice :\in \{1,2\}$ ...
end
Event *information* $\widehat{=}$ /*Délivrance de *i* */
Status convergent
refines *information*
any
x x1 x2
where
grd1 : $x \in Information$
grd2 : $x1 \in Information$
grd3 : $x2 \in Information$
grd4 : $interpretation(x) = semanticOperator(interpretation(x1) \mapsto interpretation(x2))$
grd5 : $varSeq = 1$
grd6 : $varChoice \in \{1,2\}$
then
act1 : $i,i1,i2 := x,x1,x2$
act2 : $varSeq := varSeq - 1$
end
END

Event *interpretation* $\widehat{=}$ /*calcul de *d* et construction de *p**/
refines *interpretation*
when
grd1 : $varSeq = 0$
grd2 : $varChoice = 0$
then
act1 : $d := dChoice$
act2 : $p := allocation(i)$
end
Event *interpretation1* $\widehat{=}$ /*calcul de *d*1*/
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varChoice = 1$
then
act1 : $d1 := interpretation(i1)$
act2 : $dChoice := d1$
act3 : $varChoice := 0$
end
Event *interpretation2* $\widehat{=}$ /*calcul de *d*2*/
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varChoice = 2$
then
act1 : $d2 := interpretation(i2)$
act2 : $dChoice := d2$
act3 : $varChoice := 0$
end

Figure 5.7 – L'ordonnancement par choix

Ce dernier est initialisé de manière aléatoire soit à 1 soit à 2 (*act*3). Il conditionne l'évènement *information* par la garde (*grd*6), qui assure que les interprétations *d*, *d*1 et *d*2 sont calculées après la délivrance des informations *i*, *i*1 et *i*2. S'il est initialisé à 1, il déclenche *interpretation*1 (*grd*2) qui affecte à *dChoice* la valeur de *d*1 (*act*2). S'il est initialisé à 2, il déclenche *interpretation*2 (*grd*2) qui affecte à *dChoice* la valeur de *d*2 (*act*2). *varChoice* est décrémenté à 0 dans les deux évènements *interpretation*1 et *interpretation*2 (*act*3), déclenchant ainsi l'événement *interpretation* (*grd*2). L'invariant de collage (*inv*4) permet d'établir la valeur de *dChoice* en fonction de *i*1 et *i*2, calculées respectivement dans les évènements *interpretation*1 ou *interpretation*2.

3.2 Le modèle de fission sémantique itérative

Le modèle de la fission sémantique itérative (voir Figure 5.8) modélise la production de l'information *i* par la production *n* fois de l'information fissionnée *i*1. Nous présentons ci-après le contexte et la machine qui le composent.

3.2.1 La partie CONTEXT

Le composant CONTEXT *iterativeFission* définit l'opérateur de combinaison sémantique *Iteration*. Il permet de combiner *n* fois et de manière itérative, une même interprétation sémantique pour obtenir une nouvelle interprétation sémantique (*axm*1). L'axiome (*axm*3) exprime que si on combine cette interprétation sémantique, un nombre de fois inférieur à *n*, l'interprétation sémantique résultante est identique à l'interprétation sémantique initiale.

3.2.2 La partie MACHINE

La MACHINE *iterativeInformation* modélise la production itérative *n* fois de l'information fissionnée *i*1. L'évènement abstrait *information* est raffiné, il délivre en plus de l'information *i*, l'information fissionnée *i*1 (*act*3). L'interprétation sémantique de *i*1, *d*1 est combinée *n* fois afin de produire l'interprétation sémantique de *i* (*grd*3). Un nouvel évènement *interpretation*1 est introduit, il calcule *d*1 (*act*1) l'interprétation de l'information *i*1 produite à la $i^{ème}$ itération (*grd*1). L'évènement *interpretation* est raffiné, il calcule *d* l'interprétation de l'information *i* en fonction de *d*1 (*act*1). La production itérative *n* fois de *d*1 est assurée par l'introduction du variant *varIter* initialisé à *n* (*act*1). Il est décrémenté *n* fois dans l'évènement *interpretation*1 (*act*2). Au bout de *n* itérations, il atteint 0 et déclenche l'évènement *interpretation* (*grd*1).

CONTEXT iterativeFission
CONSTANTS
Iteration n
AXIOMS
axm1 : $Iteration \in \mathbb{N} \times InterpretationDomain \rightarrow InterpretationDomain$
axm2 : $n \in \mathbb{N}_1$
axm3 : $\forall i, x \cdot (i \in 0 \mathinner{..} n-1 \wedge x \in InterpretationDomain) \Rightarrow Iteration(i \mapsto x) = x$
END

MACHINE iterativeInformation
REFINES Information
SEES iterativeFission
VARIABLES
d d1 i1 varIter i p varSeq
INVARIANTS
inv1 : $i1 \in Information$
inv2 : $d1 \in InterpretationDomain$
inv3 : $varIter \in 0 \mathinner{..} n$
inv4 : $varSeq = 0 \Rightarrow interpretation(i) = Iteration(n \mapsto interpretation(i1))$
inv5 : $varIter < n \Rightarrow d1 = interpretation(i1)$
VARIANT
varIter
EVENTS
Initialisation
begin
act1 : $varIter := n$
act2 : $varSeq := 1$ ...
end
Event *information* $\widehat{=}$ /*Délivrance de *i* */
Status convergent
any
x x1
where
grd1 : $x \in I$
grd2 : $x1 \in I$
grd3 : $interpretation(x) = Iteration(n \mapsto interpretation(x1))$
grd4 : $varSeq = 1$
grd5 : $varIter = n$
then
act1 : $i := x$
act2 : $varSeq := 0$
act3 : $i1 := x1$
end
Event *interpretation1* $\widehat{=}$ /*calcul de *d*1*/
Status convergent
when
grd1 : $varIter \in 1 \mathinner{..} n$
grd2 : $varSeq = 0$
then
act1 : $d1 := Iteration(n - varIter \mapsto interpretation(i1))$
act2 : $varIter := varIter - 1$
end
Event *interpretation* $\widehat{=}$ /*calcul de *d* et construction de *p**/
refines *interpretation*
when
grd1 : $varIter = 0$
grd2 : $varSeq = 0$
then
act1 : $d := Iteration(n - varIter \mapsto d1)$
act2 : $p := allocation(i)$
end
END

Figure 5.8 – Le modèle de la fission sémantique itérative

L'invariant de collage (*inv*6) permet d'établir la valeur de $d1$ en fonction de $i1$, calculée aux n itérations de l'évènement *interpretation*1.

4 Bilan des obligations de preuve

Les différents modèles relatifs à la fission sémantique ont généré 91 obligations de preuves (OP) (voir Table 5.1) dont 87 ont été prouvées de manière automatique par le prouveur de la plate-forme *Rodin*, 4 ont nécessité l'intervention du concepteur dans une preuve interactive. Dans *SequentialFissionedInformation*, la preuve interactive a porté sur la préservation de l'invariant (*inv*2) par l'initialisation, la préservation de l'invariant (*inv*3) par l'évènement *interpretation*2 et une obligation de preuve de simulation sur (*act*1) dans *interpretation*. Dans *ParallelFissionedInformation*, la preuve interactive a porté sur deux obligations de preuves de variant dans les événements *interpretation*1 et *interpretation*2, deux obligations de preuves de variant numérique dans les événements *interpretation*1 et *interpretation*2 et une obligation de preuve de simulation sur (*act*1) dans *interpretation*. Dans *ChoiceFissionedInformation*, la preuve interactive a porté sur deux obligations de preuve de préservation de l'invariant (*inv*4) dans les événements *interpretation*1 et *interpretation*2 et sur une obligation de preuve de simulation sur (*act*1) dans *interpretation*.

Composant	OP automatiques	OP manuelles	OP totales	OP non prouvées
Informations	0	0	0	0
ConcurrentFission	1	0	1	0
ComplementaryFission	1	0	1	0
PartialRedundantFission	1	0	1	0
TotalRedundantFission	1	0	1	0
Information	7	0	7	0
SequentialFissionedInformation	29	3	32	0
ParallelFissionedInformation	28	3	31	0
ChoiceFissionedInformation	14	3	17	0
Total	82	9	91	0

Tableau 5.1 – La fission sémantique : les obligations de preuves

5 Instanciation du modèle B Évènementiel de fission sémantique

La formalisation B Évènementiel de la fission sémantique d'une interface multimodale en sortie concrète, commence par sa description dans le modèle conceptuel de fission séman-

tique que nous présentons dans le chapitre 3. A partir de cette définition, la formalisation B Évènementiel est opérée en deux étapes :

- l'invocation du modèle abstrait générique présenté dans la Figure 4.9 et son instanciation par la définition en extension des ensembles et fonctions.
- la construction du modèle de fission sémantique qui raffine le modèle abstrait. Il combine le contexte correspondant à l'opérateur sémantique et la machine correspondante à l'opérateur temporel définissant l'interface dans le cas de la fission binaire. Il invoque le modèle de la fission itérative dans le cas de la fission itérative. Une fois le modèle de fission sémantique construit, il est instancié par la définition en extension des ensembles et fonctions. Cette étape est renouvelée pour chaque fission sémantique appliquée aux informations résultantes jusqu'à aboutir à des unités d'information élémentaires.

6 Application à l'étude de cas

Conformément à la démarche décrite ci-dessus, l'application des modèles de fission sémantique à l'étude de cas *CityMap* (voir Figure 5.9) implique la construction de deux modèles :

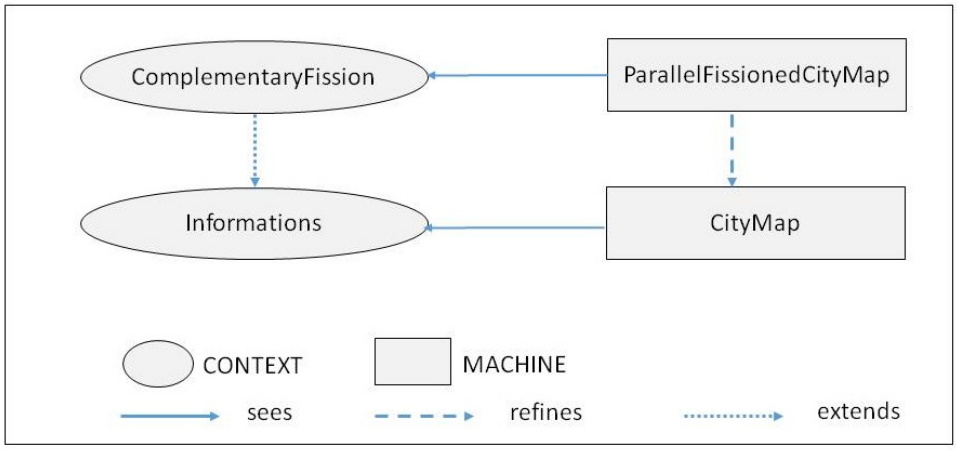

Figure 5.9 – *CityMap* : processus de formalisation B Évènementiel de la fission sémantique

1. Un modèle abstrait qui décrit la production de l'information $infoCityMap$ présenté en Figure 5.10. Il combine le contexte $Informations$ obtenu par la définition en extension des ensembles : $Information$, $InterpretationDomain$ et $Presentation$ et les fonctions : $interpretation$ et $allocation$, et la machine $CityMap$ qui instancie la variable i par la valeur $infoCityMap$ ($information$: $act1$).

CONTEXT Informations
SETS
$Information = \{emptyI, infoCityMap, infoSee, infoMap\}$
$InterpretationDomain = \{emptyD, cityMap, see, map\}$
$Presentation = \{emptyP, presentCityMap, presentSee, presentMap\}$
CONSTANTS
$interpretation\ allocation$
AXIOMS
axm1 : $interpretation \in Information \rightarrow InterpretationDomain$
axm2 : $interpretation(emptyI) = emptyD$
axm3 : $interpretation(infoCityMap) = cityMap$
axm4 : $interpretation(infoSee) = see$
axm5 : $interpretation(infoMap) = map$
axm6 : $allocation \in Information \rightarrow Presentation$
axm7 : $allocation(emptyI) = emptyP$
axm8 : $allocation(infoCityMap) = presentCityMap$
axm9 : $allocation(infoSee) = presentSee$
axm10 : $allocation(infoMap) = presentMap$
END

MACHINE CityMap
SEES Informations
VARIABLES
$i\ d\ p\ varSeq$
INVARIANTS
inv1 : $i \in Information$
inv2 : $d \in InterpretationDomain$
inv3 : $p \in Presentation$
inv4 : $varSeq \in \{0, 1\}$
EVENTS
Initialisation
begin
act1 : $i :\in Information$
act2 : $d :\in InterpretationDomain$
act3 : $p :\in Presentation$
act4 : $varSeq := 1$
end

Event $information \widehat{=}$ /* délivrance de $infoCityMap$ */
Status convergent
where
grd1 : $varSeq = 1$
then
act1 : $i := infoCityMap$
act2 : $varSeq := varSeq - 1$
end
Event $infoCityMap \widehat{=}$ /* calcul de $cityMap$ */
where
grd1 : $varSeq = 0$
then
act1 : $d := interpretation(i)$
act2 : $p := allocation(i)$
end
END

Figure 5.10 – *CityMap* : le modèle abstrait

2. Un raffinement modélisant la fission sémantique de l'information $infoCityMap$ en une combinaison complémentaire et parallèle des informations $infoSee$ et $infoMap$ présenté en Figure 5.11. Il combine le contexte complémentaire *ComplemlentaryFission* qui définit en extension l'opérateur *complementaryOperator* ($axm2, axm3, axm4, axm5$), et la machine parallèle *ParallelFissionedCityMap* qui instancie respectivement les variables i, $i1$ et $i2$ par les valeurs $infoCityMap$, $infoSee$ et $infoMap$ ($information : act1$).

   en combinant le contexte complémentaire et la machine parallèle dans laquelle les variables i, $i1$ et $i2$ sont instanciées par $infoCityMap$, $infoSee$ et $infoMap$.

CONTEXT ComplementaryFission
EXTENDS Informations
CONSTANTS

complementaryOperator

AXIOMS

axm1 : $complementaryOperator \in InterpretationDomain \times InterpretationDomain \rightarrow InterpretationDomain$
axm2 : $dom(complementaryOperator) = \{see \mapsto map, emptyD \mapsto map, see \mapsto emptyD\}$
axm3 : $complementaryOperator(see \mapsto map) = cityMap$
axm4 : $complementaryOperator(emptyD \mapsto map) = emptyD$
axm5 : $complementaryOperator(see \mapsto emptyD) = emptyD$

END

MACHINE ParallelFissionedCityMap
REFINES CityMap
SEES ComplementaryFission
VARIABLES

i d p varSeq varPar1 varPar2 i1 i2 d1 d2

INVARIANTS

inv1 : $i1 \in Information$
inv2 : $d1 \in InterpretationDomain$
inv3 : $varPar1 \in \{0, 1\}$
inv4 : $varPar2 \in \{0, 1\}$
inv5 : $varSeq < 1 \Rightarrow i = infoCityMap \wedge i1 = infoSee \wedge i2 = infoMap$
inv6 : $varSeq < 1 \Rightarrow interpretation(i) = semanticOperator(interpretation(i1) \mapsto interpretation(i2))$
inv7 : $varPar1 = 0 \Rightarrow d1 = interpretation(i1) \wedge varPar2 = 0 \Rightarrow d2 = interpretation(i2)$ ...

VARIANT

$varPar1 + varPar2$

EVENTS
Initialisation

begin

act1 : $i1 :\in Information$
act2 : $d1 :\in InterpretationDomain$
act3 : $varSeq := 1$
act4 : $varPar1 := 1$
act5 : $varPar2 := 1$ ...

end

Event $information \widehat{=}$ /* délivrance de $infoCityMap$, $infoSee$ et $infoMap$ */
Status convergent

where

grd1 : $varSeq = 1$
grd2 : $varPar1 = 1$
grd3 : $varPar2 = 1$

then

act1 : $i, i1, i2 := infoCityMap, infoSee, infoMap$
act2 : $varSeq := varSeq - 1$

end

Event $infoCityMap \widehat{=}$ /* calcul de $cityMap$ */
refines *interpretation*

when

grd1 : $varSeq = 0$
grd2 : $varPar1 = 0$
grd3 : $varPar2 = 0$

then

act1 : $d := complementaryOperator(d1 \mapsto d2)$
act2 : $p := allocation(i)$

end

Event $infoSee \widehat{=}$ /* calcul de *see* */
Status convergent

when

grd1 : $varSeq = 0$
grd2 : $varPar1 = 1$

then

act1 : $d1 := interpretation(i1)$
act2 : $varPar1 := varPar1 - 1$

end

Event $infoMap \widehat{=}$ /* calcul de *map* */
Status convergent

when

grd1 : $varSeq = 0$
grd2 : $varPar2 = 1$

then

act1 : $d2 := interpretation(i2)$
act2 : $varPar2 := varPar2 - 1$

end

END

Figure 5.11 – *CityMap* : la fission sémantique complémentaire et parallèle

Les développements B Évènementiel de la fission sémantique de l'exemple *CityMap* décrits ci-dessus ont généré 56 obligations de preuves (OP) (voir Table 5.2) dont 52 ont été prouvées de manière automatique par le prouveur de la plate-forme *Rodin*, 4 ont nécessité l'intervention du concepteur dans une preuve interactive.

Composant	OP automatiques	OP manuelles	OP totales	OP non prouvées
Informations	8	0	8	0
ComplementaryFission	4	0	4	0
CityMap	9	0	9	0
ParallelFissionedCityMap	24	2	26	0
Total	45	2	47	0

Tableau 5.2 – Fission sémantique de *CityMap* : les obligations de preuves

7 Conclusion

Nous avons présenté dans ce chapitre les développements B Évènementiel de la fission sémantique. Ils permettent de modéliser formellement la première étape de construction de l'interface multimodale en sortie : la fission sémantique. Ces modèles sont dérivés du modèle cadre présenté dans le chapitre 4 démontrant ainsi sa généricité.
Les modèles de fission sémantique sont génériques, ils permettent moyennant la définition en extension des ensembles et fonctions définissant le contexte et l'instanciation des variables de la machine de formaliser la fission sémantique de toute interface multimodale en sortie définie selon le modèle conceptuel présenté dans le chapitre 3. Néanmoins, ils sont plus précis que le modèle générique de fission sémantique présenté dans le chapitre 4, car ils sont dédiés aux différentes variantes (opérateur sémantique / opérateur temporel) définies dans le modèle de conception formel. L'utilisation de ces modèles pour le développement d'une interface concrète consiste à combiner et instancier les modèles dédiés aux opérateurs sémantiques et temporels invoqués par la définition de l'interface sans introduire de raffinements comme c'est le cas pour le modèle générique de fission sémantique.

Le dernier raffinement des développements B Évènementiel pour la fission sémantique constitue le modèle à raffiner lors de l'allocation. Les modèles B Évènementiel pour l'allocation sont détaillés dans le chapitre 6.

Chapitre 6

Modélisation de l'allocation avec B Évènementiel : Généralisation

1 Introduction

Dans le chapitre précédent, nous avons présenté les modèles B Évènementiel relatifs à la première étape de développement d'une interface multimodale en sortie : la fission sémantique. Dans ce chapitre, nous présentons les modèles relatifs à la phase qui succède à la fission sémantique : l'allocation. Ainsi, ces modèles prennent pour modèle abstrait le dernier raffinement de la fission sémantique. Ils sont dérivés des modèles génériques d'allocation et d'affectation présentés dans le chapitre 4 et définissent la totalité des cas de la phase d'allocation. Ils permettent de modéliser la dernière étape dans le processus de modélisation de l'interface multimodale en sortie moyennant trois raffinements.

- un premier raffinement construit la présentation globale correspondante à l'information générée par le noyau fonctionnel en combinant les présentations relatives aux informations fissionnées.
- un deuxième raffinement modélise la décomposition des présentations élémentaires en unités de présentation.
- un troisième raffinement modélise l'affectation des unités de présentations avec les modalités et les médias.

Nous présentons également dans ce chapitre, la démarche proposée pour la vérification de propriétés pertinentes sur les interfaces multimodales en sortie. Nous l'illustrons en présentant les modèles développés pour la vérification de propriétés de sûreté et de vivacité.

2 Démarche de modélisation avec B Évènementiel

La modélisation B Évènementiel de l'allocation utilise la démarche de formalisation définie pour la modélisation générique des IHM multimodales en sortie présentée dans le chapitre 4. Il s'agit d'une démarche incrémentale par raffinements successifs basée sur le principe de séparation entre sémantique statique et dynamique.
La démarche de modélisation B Évènementiel de l'allocation débute par la formalisation de la combinaison des présentations correspondantes aux informations fissionnées. Le modèle de combinaison des présentations raffine le dernier modèle de fission sémantique. Les différentes variantes de ce modèle constituent les différentes réalisations possibles du modèle générique d'allocation pour la combinaison de présentations. Chaque contexte correspond à un cas spécifique de combinaison sémantique du contexte générique et chaque machine correspond à un cas spécifique d'ordonnancement temporel de production des présentations combinées dans la machine. Le modèle de combinaison des présentations est ensuite raffiné par le modèle de décomposition des présentations. Les différentes variantes de ce modèle constituent les différentes réalisations possibles du modèle générique d'allocation pour la décomposition des présentations. Chaque contexte correspond à un cas spécifique de décomposition sémantique du contexte générique et chaque machine correspond à un cas spécifique d'ordonnancement temporel de production des présentations décomposées dans la machine. Enfin, le modèle de décomposition des présentations élémentaires et raffiné par le modèle d'affectation des modalités et média. Il invoque le modèle générique d'affectation.
Nous présentons ci-après les trois modèles intervenant dans la formalisation de l'allocation : combinaison des présentations, décomposition des présentations élementaires et affectation.

3 Le modèle de combinaison des présentations

Le modèle relatif à la combinaison des présentations multimodales est le premier raffinement de la phase d'allocation. Il succède au dernier raffinement de la phase de fission sémantique qu'il raffine. Il modélise la première règle syntaxique d'allocation que nous rappelons dans 6.1.

$$PM ::= PME \mid (op'_{temp}, op'_{sem})(PM, PM) \mid It'(n, PM) \; avec \; n \in \mathbb{N} \tag{6.1}$$

Cette règle décrit la construction des présentations multimodales *pm* relatives aux informations *i* obtenues lors de la phase de fission sémantique. Par conséquent, l'établissement de

ce raffinement impose l'achèvement de la phase de fission sémantique assuré par la condition que les informations fissionnées soient élémentaires. Les présentations multimodales ainsi construites sont combinées par un opérateur sémantique op'_{sem} analogue à l'opérateur temporel qui combine les informations fissionnées op_{sem}. Elles sont produites dans le même ordre temporel que les informations qu'elles restituent afin de construire la présentation multimodale globale p. Ainsi, à l'image de la fission sémantique, la combinaison peut être binaire lorsqu'elle combine les présentations $p1$ et $p2$ relatives aux informations $i1$ et $i2$ et itérative lorsqu'elle combine les n présentations $p1$ relatives à l'information $i1$. Nous présentons ci-après les modèles correspondant aux cas de la combinaison binaire et de la combinaison itérative.

3.1 Le modèle de combinaison binaire

Le modèle de combinaison binaire des présentations réalise le modèle générique d'allocation (voir Figure 4.11). Il décrit la combinaison sémantique et temporelle de deux présentations $p1$ et $p2$, qui restituent respectivement les informations $i1$ et $i2$, pour produire une présentation globale p qui restitue l'information i. Par conséquent, les relations sémantiques et temporelles qui lient $p1$ et $p2$ sont analogues aux relations sémantiques et temporelles qui lient $i1$ et $i2$. Le modèle de combinaison binaire se compose des éléments suivants.

3.1.1 Le composant CONTEXT

Les contextes de combinaison définissent les opérateurs sémantiques op'_{sem} découlant de l'opérateur générique *combinationOperator*, défini dans le contexte générique *allocation*. L'opérateur op'_{sem} combine les présentations de manière analogue au contexte de la fission sémantique. Il y a donc autant de variantes de contextes de combinaison que d'opérateurs sémantiques op'_{sem} : concurrent, complémentaire, complémentaire et redondant, partiellement redondant et totalement redondant. Nous présentons ci-après les contextes qui formalisent ces opérateurs.

1. **Le cas concurrent.** Le CONTEXT *ConcurrentCombination* (voir Figure 6.1) définit l'opérateur sémantique concurrent (*PconcurrentOperator*) sur les présentations . L'axiome (*axm*2) définit la concurrence sémantique. Il exprime que la production séparée de chacune des deux présentations concurrentes n'altère pas leur sémantique.

CONTEXT ConcurrentCombination
EXTENDS ConcurrentFission
CONSTANTS
PconcurrentOperator
AXIOMS
axm1 : $PconcurrentOperator \in Presentation \times Presentation \rightarrow Presentation$
axm2 : $\forall x, y \cdot (x \in Presentation \setminus \{emptyP\} \land y \in Presentation \setminus \{emptyP\} \land x \mapsto y \in dom(PconcurrentOperator) \land emptyP \mapsto y \in dom(PconcurrentOperator) \land x \mapsto emptyP \in dom(PconcurrentOperator) \Rightarrow (PconcurrentOperator(emptyP \mapsto y) = y \land PconcurrentOperator(x \mapsto emptyP) = x))$
END

Figure 6.1 – Combinaison des présentations : le CONTEXT concurrent

2. **Le cas complémentaire.**

Le CONTEXT *ComplementaryCombination* (voir Figure 6.2) définit l'opérateur sémantique complémentaire (*PcomplementaryOperator*) sur les présentations. L'axiome (*axm*2) définit la complémentarité sémantique. Il exprime que deux présentations complémentaires ne peuvent être produites séparément sans perdre leur sémantique.

CONTEXT ComplementaryCombination
EXTENDS ComplementaryFission
CONSTANTS
PcomplementaryOperator
AXIOMS
axm1 : $PcomplementaryOperator \in Presentation \times Presentation \rightarrow Presentation$
axm2 : $\forall x, y \cdot (x \in Presentation \setminus \{emptyP\} \land y \in Presentation \setminus \{emptyP\} \land x \mapsto y \in dom(PcomplementaryOperator) \land emptyP \mapsto y \in dom(PcomplementaryOperator) \land x \mapsto emptyP \in dom(PcomplementaryOperator) \Rightarrow (PcomplementaryOperator(emptyP \mapsto y) = emptyP \land PcomplementaryOperator(x \mapsto emptyP) = emptyP))$
END

Figure 6.2 – Combinaison des présentations : le CONTEXT complémentaire

3. **Le cas partiellement redondant.**

CONTEXT PartialRedundantCombination
EXTENDS PartialRedundantFission
CONSTANTS
PpartialredundantOperator
AXIOMS
axm1 : $PpartialredundantOperator \in Presentation \times Presentation \rightarrow Presentation$
axm2 : $\forall x, y \cdot (x \in Presentation \setminus \{emptyP\} \land y \in Presentation \setminus \{emptyP\} \land x \mapsto y \in dom(PpartialredundantOperator) \land emptyP \mapsto y \in dom(PpartialredundantOperator) \Rightarrow (PpartialredundantOperator(x \mapsto y) = PpartialredundantOperator(emptyP \mapsto y)$
END

Figure 6.3 – Combinaison des présentations : le CONTEXT partiellement redondant

Le CONTEXT *PartialRedundantCombination* (voir Figure 6.3) définit l'opérateur sémantique (*PpartialredundantOperator*) partiellement redondant sur les présentations.

L'axiome (*axm*2) définit la redondance sémantique partielle. Il exprime que dans une combinaison partiellement redondante, la sémantique de la première présentation est exprimée par la sémantique de la deuxième, et par conséquent, la première présentation peut être remplacée par la présentation vide *emptyP*.

4. **Le cas totalement redondant.** Le CONTEXT *TotalRedundantCombination* (voir Figure 6.4) définit l'opérateur sémantique totalement redondant (*PtotalredundantOperator*) sur les présentations . L'axiome (*axm*2) définit la redondance sémantique totale. Il exprime que dans une combinaison totalement redondante, la sémantique de la première présentation est exprimée par la sémantique de la deuxième et vice versa. Par conséquent, la première ou la deuxième présentation peuvent être remplacées par la présentation vide *emptyP*.

CONTEXT TotalRedundantCombination
EXTENDS TotalRedundantFission
CONSTANTS
PtotalredundantOperator
AXIOMS
axm1 : $PtotalredundantOperator \in Presentation \times Presentation \rightarrow Presentation$
axm2 : $\forall x, y \cdot (x \in Presentation \setminus \{emptyP\} \land y \in Presentation \setminus \{emptyP\} \land x \mapsto y \in dom(PtotalredundantOperator) \land emptyP \mapsto y \in dom(PtotalredundantOperator) \land x \mapsto emptyP \in dom(PtotalredundantOperator) \Rightarrow PtotalredundantOperator(x \mapsto y) = PtotalredundantOperator(emptyP \mapsto y) \land PtotalredundantOperator(x \mapsto y) = PtotalredundantOperator(x \mapsto emptyP)$
END

Figure 6.4 – Combinaison des présentations : le CONTEXT totalement redondant

5. **Le cas complémentaire et redondant.** Le cas complémentaire et redondant requiert l'utilisation d'une sémantique dotée d'une relation d'ordre et d'inclusion sémantiques sur *Presentation*. La sémantique que nous utilisons n'est pas dotée de cette relation d'inclusion, par conséquent, le cas complémentaire et redondant ne peut être exprimé selon le modèle sémantique choisi.

3.1.2 Le composant MACHINE

La machine de combinaison des présentations construit les présentations $p1$ et $p2$ relatives aux informations fissionnées $i1$ et $i2$. Elle combine les présentations $p1$ et $p2$ pour construire la présentation globale p. La MACHINE générique *Presentation* présentée dans la Figure 4.11 est spécialisée en fonction de l'opérateur temporel op'_{temp} qui combine les présentations multimodales correspondant aux informations fissionnées. Il y a donc autant de variantes de machine que d'opérateurs temporels.
Cet ordre temporel est analogue à l'ordre de production des informations fissionnées exprimé par op_{temp}. Il est modélisé en exploitant le variant introduit dans le modèle de fission qu'il raffine, pour ordonnancer les informations fissionnées. La machine spécifique aux différents ordonnancements temporels se distingue de la machine générique *Presentation* par le fait qu'elle définit le variant générique *var* par un variant spécifique qui permet d'ordonnancer les événements produisant les présentations selon l'ordre établi par l'opérateur temporel. Nous présentons ci-après les machines spécifiques aux opérateurs temporels de base : séquentiel, parallèle et choix. Ces machines sont combinées au contexte correspondant à l'opérateur sémantique qui relie les présentations (concurrent, complémentaire, complémentaire et redondant, partiellement redondant, totalement redondant). Nous représentons cet opérateur sémantique par *PsemanticOperator* et par le contexte générique *SemanticCombination* dans la clause *SEES*. Les opérateurs temporels : anachronique, concomitant, coïncidant et indépendant peuvent être exprimés en utilisant les opérateurs de base comme présentés dans la section 3.1.2.

1. **L'ordonnancement séquentiel.** La machine *SequentialPresentation* (voir Figure 6.5) raffine la machine *SequentialFissionedInformation* (voir Figure 5.5). Elle enrichit la machine générique *Presentation* en remplaçant le variant générique *var* par le variant spécifique *varSeq*1 initialisé à 2 (*act*3). Il conditionne l'évènement *information* par la garde (*grd*7), qui assure que les présentations p, $p1$ et $p2$ sont calculées après la délivrance des informations i, $i1$ et $i2$. Il est ensuite décrémenté dans les évènements *presentation*1 (*act*3), *presentation*2 (*act*3) déclenchant ainsi de manière séquentielle les évènements *presentation*1, *presentation*2 et *presentation* par les gardes *grd*2. L'invariant de collage (*inv*4) permet d'établir les valeurs de $p1$ et $p2$ en fonction de $i1$ et $i2$, construites respectivement dans les évènements *presentation*1 et *presentation*2.

MACHINE SequentialPresentation
REFINES SequentialFissionnedInformation
SEES SemanticCombination
VARIABLES

$i\ d\ p\ varSeq\ varSeq1\ i1\ i2\ d1\ d2\ p1\ p2$

INVARIANTS

inv1 : $p1 \in Presentation$
inv2 : $varSeq1 \in \{0, 1, 2\}$
inv3 : $varSeq < 1 \Rightarrow allocation(i) = combinationOperator(allocation(i1) \mapsto allocation(i2))$
inv4 : $varSeq1 = 0 \Rightarrow p1 = allocation(i1) \wedge p2 = allocation(i2)$ ...

VARIANT

$varSeq1$

EVENTS
Initialisation
begin
act1 : $p1 :\in Presentation$
act2 : $varSeq := 1$
act3 : $varSeq1 := 2$ ...
end

Event $information \widehat{=}$ /*Délivrance de i */
Status convergent
any
$x\ x1\ x2$
where
grd1 : $x \in Information$
grd2 : $x1 \in Information$
grd3 : $x2 \in Information$
grd4 : $interpretation(x) = semanticOperator(interpretation(x1) \mapsto interpretation(x2))$
grd5 : $allocation(x) = combinationOperator(allocation(x1) \mapsto allocation(x2))$
grd6 : $varSeq = 1$
grd7 : $varSeq1 = 2$
then
act1 : $i, i1, i2 := x, x1, x2$
act2 : $varSeq := varSeq - 1$
end
END

Event $presentation \widehat{=}$ /*construction de p */
refines $interpretation$
when
grd1 : $varSeq = 0$
grd2 : $varSeq1 = 0$
then
act1 : $d := semanticOperator(d1 \mapsto d2)$
act2 : $p := combinationOperator(p1 \mapsto p2)$
end

Event $presentation1 \widehat{=}$ /*construction de $p1$ */
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varSeq1 = 2$
then
act1 : $d1 := interpretation(i1)$
act2 : $p1 := allocation(i1)$
act3 : $varSeq1 := varSeq1 - 1$
end

Event $presentation2 \widehat{=}$ /*construction de $p2$ */
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varSeq1 = 1$
then
act1 : $d2 := interpretation(i2)$
act2 : $p2 := allocation(i2)$
act3 : $varSeq1 := varSeq1 - 1$
end

Figure 6.5 – Combinaison des présentations : l'ordonnancement séquentiel

2. **L'ordonnancement parallèle.** La machine *ParallelFissionedInformation* (voir Figure 6.6) raffine la machine *ParallelFissionedInformation* (voir Figure 5.6). Elle enrichit la machine générique *Presentation* en remplaçant le variant générique *var* par la combinaison de deux variants spécifiques *varPar*1 et *varPar*2 initialisés à 1.

MACHINE ParallelPresentation
REFINES ParallelFissionnedInformation
SEES SemanticCombination
VARIABLES

$i\ d\ p\ varSeq\ varPar1\ varPar2\ i1\ i2\ d1\ d2\ p1\ p2$

INVARIANTS

inv1 : $p1 \in Presentation$
inv2 : $varPar1 \in \{0,1\}$
inv3 : $varPar2 \in \{0,1\}$
inv4 : $varSeq < 1 \Rightarrow allocation(i) = combinationOperator(allocation(i1) \mapsto allocation(i2))$
inv5 : $varPar1 = 0 \Rightarrow p1 = allocation(i1)$
inv6 : $varPar2 = 0 \Rightarrow p2 = allocation(i2)$ ...

VARIANT

$varPar1 + varPar2$

EVENTS
Initialisation
begin
act1 : $p1 :\in Presentation$
act2 : $varSeq := 1$
act3 : $varPar1 := 1$
act4 : $varPar2 := 1$ ...
end

Event *information* $\widehat{=}$ /*Délivrance de i */
Status convergent
any
$x\ x1\ x2$
where
grd1 : $x \in Information$
grd2 : $x1 \in Information$
grd3 : $x2 \in Information$
grd4 : $interpretation(x) = semanticOperator(interpretation(x1) \mapsto interpretation(x2))$
grd5 : $allocation(x) = combinationOperator(allocation(x1) \mapsto allocation(x2))$
grd6 : $varSeq = 1$
grd7 : $varPar1 = 1$
grd8 : $varPar2 = 1$
then
act1 : $i, i1, i2 := x, x1, x2$
act2 : $varSeq := varSeq - 1$
end
END

Event *presentation* $\widehat{=}$ /*construction de p */
refines *interpretation*
when
grd1 : $varSeq = 0$
grd2 : $varPar1 = 0$
grd3 : $varPar2 = 0$
then
act1 : $d := semanticOperator(d1 \mapsto d2)$
act2 : $p := combinationOperator(p1 \mapsto p2)$
end

Event *presentation1* $\widehat{=}$ /*construction de $p1$ */
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varPar1 = 1$
then
act1 : $d1 := interpretation(i1)$
act2 : $p1 := allocation(i1)$
act3 : $varPar1 := varPar1 - 1$
end

Event *presentation2* $\widehat{=}$ /*construction de $p2$ */
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varPar2 = 1$
then
act1 : $d2 := interpretation(i2)$
act2 : $p2 := allocation(i2)$
act3 : $varPar2 := varPar2 - 1$
end

Figure 6.6 – Combinaison des présentations : l'ordonnancement parallèle

Ils conditionnent l'évènement *information* par les gardes (*grd*7 et *grd*8), qui assurent que les présentations p, $p1$ et $p2$ sont calculées après la délivrance des informations i, $i1$ et $i2$.

*varPar*1 déclenche *presentation*1 (*grd*2) qui le décrémente à 0 (*act*3) et *varPar*2 déclenche *presentation*2 (*grd*2) qui le décrémente à 0 (*act*3). L'évènement *presentation* est déclenché lorsque *varPar*1 et *varPar*2 sont à 0 (*grd*2 et *grd*3). Les invariants de collage (*inv*5 et *inv*6) permettent d'établir les valeurs de $p1$ et $p2$ en fonction de $i1$ et $i2$, construites respectivement dans les évènements *presentation*1 et *presentation*2.

3. **L'ordonnancement par choix.** La machine *ChoiceFissionedInformation* (voir Figure 6.7) raffine la machine *ChoiceFissionedInformation* (voir Figure 5.7).
   Elle enrichit la machine générique *Presentation* en introduisant une variable supplémentaire *pChoice* et en remplaçant le variant générique *var* par le variant spécifique *varChoice*. *varChoice* est initialisé de manière aléatoire soit à 1 soit à 2 (*act*3). S'il est initialisé à 1, il déclenche *presentation*1 (*grd*2) qui affecte à *pChoice* la valeur de $p1$ (*act*4), s'il est initialisé à 2, il déclenche *presentation*2 (*grd*2) qui affecte à *pChoice* la valeur de $p2$ (*act*4). *varChoice* est décrémenté à 0 dans les deux évènements (*act*5) déclenchant ainsi *presentation*. L'invariant de collage (*inv*4) permet d'établir la valeur de *pChoice* en fonction de $i1$ et $i2$ suite à sa construction dans l'évènement *presentation*1 ou *presentation*2.

3.2 Le modèle de combinaison itérative

Le modèle de combinaison itérative des présentations (voir Figure 6.8) raffine le modèle de fission itérative des informations (voir Figure 5.8). Il décrit la combinaison itérative n fois de la présentation $p1$ qui restitue l'information $i1$ pour produire la présentation globale p qui restitue l'information i. Le modèle de combinaison itérative se compose des éléments suivants.

3.2.1 La partie CONTEXT

Le composant CONTEXT *IterativeCombination* étend le contexte *IterativeFission* (voir Figure 5.8). Il définit l'opérateur *PIteration* (*axm*1) qui combine n fois et de manière itérative, une même présentation pour obtenir une nouvelle présentation. L'axiome (*axm*3) décrit que si on combine une présentation, un nombre de fois inférieur à n, la présentation résultante est identique à la présentation initiale.

3.2.2 La partie MACHINE

La MACHINE *IterativePresentation* raffine la machine décrivant la fission sémantique *iterativeInformation* (voir Figure 5.8). Elle modélise la production n fois de la présentation

MACHINE ChoicePresentation
REFINES ChoiceFissionnedInformation
SEES SemanticCombination
VARIABLES

$i\ d\ p\ varSeq\ varChoice\ i1\ i2\ d1\ d2\ p1\ p2\ dChoice\ pChoice$

INVARIANTS

inv1 : $pChoice \in Presentation$
inv2 : $varChoice \in \{0, 1, 2\}$
inv3 : $varSeq < 1 \Rightarrow allocation(i) = combinationOperator(allocation(i1) \mapsto allocation(i2))$
inv4 : $varChoice = 0 \Rightarrow (p1 = allocation(i1) \wedge pChoice = p1) \vee (p2 = allocation(i2) \wedge pChoice = p2$ ...

VARIANT

$varChoice$

EVENTS
Initialisation
begin
act1 : $p1 :\in Presentation$
act2 : $varSeq := 1$
act3 : $varChoice :\in \{1, 2\}$
act4 : $pChoice :\in Presentation$ ...
end

Event $information \widehat{=}$ /*Délivrance de i */
Status convergent
any
$x\ x1\ x2$
where
grd1 : $x \in Information$
grd2 : $x1 \in Information$
grd3 : $x2 \in Information$
grd4 : $interpretation(x) = semanticOperator(interpretation(x1) \mapsto interpretation(x2))$
grd5 : $allocation(x) = combinationOperator(allocation(x1) \mapsto allocation(x2))$
grd6 : $varSeq = 1$
grd7 : $varChoice \in \{1, 2\}$
then
act1 : $i, i1, i2 := x, x1, x2$
act2 : $varSeq := varSeq - 1$
end

END

Event $presentation \widehat{=}$ /*construction de p */
refines $interpretation$
when
grd1 : $varSeq = 0$
grd2 : $varChoice = 0$
then
act1 : $d := dChoice$
act2 : $p := pChoice$
end

Event $presentation1 \widehat{=}$ /*construction de $p1$ */
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varChoice = 1$
then
act1 : $d1 := interpretation(i1)$
act2 : $p1 := allocation(i1)$
act3 : $dChoice := d1$
act4 : $pChoice := p1$
act5 : $varChoice := 0$
end

Event $presentation2 \widehat{=}$ /*construction de $p2$ */
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varChoice = 2$
then
act1 : $d2 := interpretation(i2)$
act2 : $p2 := allocation(i2)$
act3 : $dChoice := d2$
act4 : $pChoice := p2$
act5 : $varChoice := 0$
end

Figure 6.7 – Combinaison des présentations : l'ordonnancement par choix

multimodale élémentaire $p1$ qui restitue l'information élémentaire $i1$ afin de produire la présentation globale p qui restitue i. En plus de calculer $d1$ et d respectivement dans les

CONTEXT IterativeCombination
EXTENDS IterativeFission
CONSTANTS
PIteration n
AXIOMS
axm1 : $PIteration \in \mathbb{N} \times Presentation \rightarrow Presentation$
axm2 : $n \in \mathbb{N}_1$
axm3 : $\forall i, x \cdot (i \in 0 \mathinner{..} n-1 \wedge x \in Presentation) \Rightarrow PIteration(i \mapsto x) = x$
END

MACHINE IterativePresentation
REFINES iterativeInformation
SEES IterativeCombination
VARIABLES
d d1 i1 varIter i p varSeq p1
INVARIANTS
inv1 : $p1 \in Presentation$
inv2 : $varIter \in 0 \mathinner{..} n$
inv3 : $varSeq = 0 \Rightarrow allocation(i) = PIteration(n \mapsto allocation(i1))$
inv4 : $varIter < n \Rightarrow p1 = allocation(i1)$
VARIANT
varIter
EVENTS
Initialisation
begin
act4 : $varIter := n$
act5 : $varSeq := 1$ ...
end
Event *information* $\widehat{=}$ /*Délivrance de *i* */
Status convergent
extends *information*
any
x x1
where
grd1 : $x \in Information$
grd2 : $x1 \in Information$
grd3 : $interpretation(x) = Iteration(n \mapsto interpretation(x1))$
grd4 : $allocation(x) = PIteration(n \mapsto allocation(x1))$
grd5 : $varSeq = 1$
grd6 : $varIter = n$
then
act1 : $i := x$
act2 : $i1 := x1$
act3 : $varSeq := 0$
end

Event *presentation1* $\widehat{=}$ /*construction de *p*1 */
refines *interpretation1*
Status convergent
when
grd1 : $varIter \in 1 \mathinner{..} n$
grd2 : $varSeq = 0$
then
act1 : $d1 := Iteration(n - varIter \mapsto interpretation(i1))$
act2 : $p1 := PIteration(n - varIter \mapsto allocation(i1))$
act3 : $varIter := varIter - 1$
end
Event *presentation* $\widehat{=}$ /*construction de *p* */
refines *interpretation*
when
grd1 : $varIter = 0$
grd2 : $varSeq = 0$
then
act1 : $d := Iteration(n - varIter \mapsto d1)$
act2 : $p := PIteration(n - varIter \mapsto p1)$
end
END

Figure 6.8 – Combinaison des présentations : l'itération

évènements *presentation*1 et *presentation*, elle construit les présentations *p*1 (*act*2) et *p* (*act*2). L'ordre iteratif de production des présentations correspond à celui du calcul des interpré-

tations dans la machine *iterativeInformation* qu'elle raffine. Par conséquent, il est modélisé en exploitant le variant *varIter* introduit dans la machine *iterativeInformation*. L'invariant de collage (*inv*4) permet d'établir la valeur de $p1$ en fonction de $i1$ à chaque itération.

3.3 Les obligations de preuve

Les différents modèles relatifs à la combinaison des présentations ont généré 51 obligations de preuves (OP) (voir Table 6.1) dont 49 ont été prouvées de manière automatique par le prouveur de la plate-forme *Rodin*, deux preuves de simulation portant sur la construction de la présentation globale dans les machines *ChoicePresentation* et *IterativePresentation* ont nécessité l'intervention du concepteur dans une preuve interactive.

Composant	OP automatiques	OP manuelles	OP totales	OP non prouvées
ConcurrentCombination	1	0	1	0
ComplementaryCombination	1	0	1	0
PartialRedundantCombination	1	0	1	0
TotalRedundantCombination	1	0	1	0
IterativeCombination	2	0	2	0
SequentialPresentation	15	0	15	0
ParallelPresentation	16	0	16	0
ChoicePresentation	6	1	7	0
IterativePresentation	6	1	7	0
Total	49	2	51	0

Tableau 6.1 – La combinaison des présentations : les obligations de preuves

4 Le modèle de décomposition des présentations

Le modèle relatif à la décomposition des présentations multimodales est le second raffinement de la phase d'allocation. Il succède au raffinement relatif à la combinaison des présentations multimodales. Il modélise la seconde règle syntaxique d'allocation que nous rappelons dans 6.2. Cette règle décrit la décomposition des présentations multimodales *pm* construites lors du dernier raffinement (combinaison des présentations). Les présentations multimodales *pm* sont décomposées afin d'être distribuées aux différentes modalités et média, jusqu'à aboutir à des unités de présentation élémentaires *upe*. La décomposition est dite binaire lorsqu'elle produit les présentations $p1$ et $p2$ et itérative lorsqu'elle produit n fois la présentation $p1$. Les présentations ainsi décomposées sont combinées à l'aide des opérateurs

compl, *redun*, *choice* et *iter*. Nous présentons ci-après les modèles correspondant aux cas de la décomposition binaire et de la décomposition itérative.

$$PME ::= UPE \mid compl(UPE, PME) \mid redun(UPE, PME) \mid choice(UPE, PME) \mid iter(n, PME)\, avec\, n \in N \quad (6.2)$$

4.1 Le modèle de décomposition binaire

Le modèle de décomposition binaire des présentations élémentaires raffine le modèle de composition des présentations (voir section 3). Il est dérivé du modèle générique d'allocation (voir Figure 4.11). Il décrit la décomposition d'une présentation élémentaire p en deux présentations élémentaires $p1$ et $p2$. Les présentations $p1$ et $p2$ peuvent être sémantiquement complémentaires ou redondantes. L'ordonnancement temporel de la production de $p1$ et $p2$ est soit explicite pour la décomposition par choix, ou alors implicite pour les décompositions complémentaires et redondantes, nous choisissons d'appliquer l'ordonnancement parallèle pour les présentations complémentaires ou redondantes. Le modèle de décomposition binaire définit les différentes réalisations du contexte générique *allocation* selon les deux opérateurs de décomposition sémantique (*compl* et *redon*) et il enrichit la machine *Presentation* selon les opérateurs de combinaison temporelle (parallèle et choix). Nous présentons ci-après les contextes et machines qui composent le modèle de décomposition binaire.

4.1.1 Le composant CONTEXT

Le composant CONTEXT définit l'opérateur qui combine les présentations obtenues suite à la décomposition. Il est dérivé du contexte générique *allocation* et raffine le contexte de combinaison des présentations correspondant au raffinement précédent. Nous représentons ce contexte de combinaison par *Combination*. Il existe deux variantes de contextes pour les deux opérateurs qui combinent sémantiquement les présentations décomposées : *compl* et *redun*. Nous présentons ci-après le CONTEXT *ComplementaryDecomposition* relatif à l'opérateur *compl* et le CONTEXT *RedundantDecomposition* relatif à l'opérateur *redun*.

– **le cas complémentaire.** Le CONTEXT *ComplementaryDecomposition* (voir Figure 6.9) définit l'opérateur sémantique complémentaire (*ComplementaryDecompositionOperator*) sur les présentations. L'axiome (*axm*2) définit la complémentarité sémantique. Il exprime que deux présentations complémentaires ne peuvent être produites séparément

sans perdre leur sémantique.

CONTEXT ComplementaryDecomposition
EXTENDS Combination
CONSTANTS
ComplementaryDecompositionOperator
AXIOMS
axm1 : $ComplementaryDecompositionOperator \in Presentation \times Presentation \rightarrow Presentation$
axm2 : $\forall x, y \cdot (x \in Presentation \setminus \{emptyP\} \land y \in Presentation \setminus \{emptyP\} \land x \mapsto y \in dom(ComplementaryDecompositionOperator) \land emptyP \mapsto y \in dom(ComplementaryDecompositionOperator) \land x \mapsto emptyP \in dom(ComplementaryDecompositionOperator) \Rightarrow (ComplementaryDecompositionOperator(emptyP \mapsto y) = emptyP \land ComplementaryDecompositionOperator(x \mapsto emptyP) = emptyP))$
END

Figure 6.9 – Décomposition des présentations élémentaires : le CONTEXT complémentaire

– **le cas redondant.** Le CONTEXT *RedundantDecomposition* (voir Figure 6.10) définit l'opérateur sémantique redondant sur les présentations (*RedundantDecompositionOperator*). L'axiome *axm*2 définit la redondance sémantique. Il exprime que dans une combinaison redondante, la sémantique de la première présentation est exprimée par la sémantique de la deuxième, et par conséquent, la première présentation peut être remplacée par la présentation vide.

CONTEXT RedundantDecomposition
EXTENDS Combination
CONSTANTS
RedundantDecompositionOperator
AXIOMS
axm1 : $RedundantDecompositionOperator \in Presentation \times Presentation \rightarrow Presentation$
axm2 : $\forall x, y \cdot (x \in Presentation \setminus \{emptyP\} \land y \in Presentation \setminus \{emptyP\} \land x \mapsto y \in dom(RedundantDecompositionOperator) \land emptyP \mapsto y \in dom(RedundantDecompositionOperator) \Rightarrow (RedundantDecompositionOperator(x \mapsto y) = (RedundantDecompositionOperator(emptyP \mapsto y))))$
END

Figure 6.10 – Décomposition des présentations élémentaires : le CONTEXT redondant

4.1.2 Le composant MACHINE

La machine du modèle de décomposition des présentations construit les présentations $p1$ et $p2$ résultantes de la décomposition de p. Elle raffine la machine qui combine les présentations correspondantes aux informations fissionnées que nous notons *CombinedPresentation*.

Elle spécialise la machine générique *Presentation* présentée dans la Figure 4.11 d'une part, en modélisant la décomposition d'une présentation v en deux présentations $v1$ et $v2$, formalisée dans l'évènement *decomposition* (*act*1), et d'autre part, en spécifiant l'ordre de production des présentations décomposées en fonction de l'opérateur temporel qui les combine. Cet ordre temporel est formalisé en remplaçant le variant générique *var* par un variant spécifique à chaque cas d'ordonnancement temporel employé pour combiner les présentations décomposées. Seul l'ordonnancement temporel par choix est utilisé pour combiner les présentations décomposées.

Pour la décomposition complémentaire et redondante, l'ordonnancement temporel n'est pas précisé, ainsi, nous considérons que les présentations sont produites de manière parallèle. Par conséquent, nous développons dans ce qui suit deux machines relatives aux cas d'ordonnancement : parallèle, et choix. Ces machines sont combinées au contexte correspondant à l'opérateur sémantique qui relie les présentations (complémentaire et redondant). Nous représentons cet opérateur sémantique par *DecompositionOperator* et par le contexte générique *Decomposition* dans la clause *SEES*.

1. **L'ordonnancement parallèle**

   La machine *ParallelDecomposedPresentation* (voir Figure 6.11) modélise la production des présentations $p1$ et $p2$ en parallèle. Elle spécialise la machine générique *Presentation* en remplaçant le variant générique *var* par deux variants spécifiques *varPar*1 et *varPar*2 initialisés à 1 (*act*3 et *act*4). *varPar*1 déclenche *presentation*1 (*grd*2) qui le décrémente à 0 (*act*2) et *varPar*2 déclenche *presentation*2 (*grd*2) qui le décrémente à 0 (*act*2). L'évènement *presentation* est déclenché lorsque *varPar*1 et *varPar*2 sont à 0 (*grd*2 et *grd*3). L'invariant de collage (*inv*7) établit les valeurs de $p1$ et $p2$ calculées respectivement en fonction de $v1$ et $v2$ dans les événements *Presentation*1 et *Presentation*2.

MACHINE ParallelDecomposedPresentation
REFINES CombinedPresentation
SEES Decomposition
VARIABLES

$p\ p1\ p2\ v\ v1\ v2\ varSeq\ varPar1\ varPar2$

INVARIANTS

inv1 : $p1 \in Presentation$
inv2 : $v \in Presentation$
inv3 : $varSeq \in \{0,1\}$
inv4 : $varPar1 \in \{0,1\}$
inv5 : $varPar2 \in \{0,1\}$
inv6 : $varSeq < 1 \Rightarrow v = DecompositionOperator(v1 \mapsto v2)$
inv7 : $varPar1 = 0 \Rightarrow p1 = v1) \wedge varPar2 = 0 \Rightarrow p2 = v2$ ...

VARIANT

$varSeq + varPar1 + varPar2$

EVENTS
Initialisation
begin
act1 : $v :\in Presentation$
act2 : $varSeq := 1$
act3 : $varPar1 := 1$
act4 : $varPar2 := 1$ ...
end

Event *decomposition* $\widehat{=}$ /* décomposition de v*/
Status convergent
any
$x\ x1\ x2$
where
grd1 : $x \in Presentation$
grd2 : $x1 \in Presentation$
grd3 : $x2 \in Presentation$
grd4 : $x = DecompositionOperator(x1 \mapsto x2)$
grd5 : $varSeq = 1$
grd6 : $var \in \mathbb{N}_1$
then
act1 : $v, v1, v2 := x, x1, x2$
act2 : $varSeq := varSeq - 1$
end
END

Event *presentation* $\widehat{=}$ /* construction de p*/
refines *presentation*
when
grd1 : $varSeq = 0$
grd2 : $varPar1 = 0$
grd3 : $varPar2 = 0$
then
act1 : $p := DecompositionOperator(p1 \mapsto p2)$
end

Event *presentation1* $\widehat{=}$ /* construction de $p1$*/
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varPar1 = 1$
then
act1 : $p1 := v1$
act2 : $varPar1 := varPar1 - 1$
end

Event *presentation2* $\widehat{=}$ /* construction de $p2$*/
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varPar2 = 1$
then
act1 : $p2 := v2$
act2 : $varPar2 := varPar2 - 1$
end

Figure 6.11 – Décomposition des présentations élémentaires : l'ordonnancement parallèle

2. **L'ordonnancement par choix** La machine *ChoiceDecomposedPresentation* (voir Figure 6.12) modélise la production par choix d'une des présentations $p1$ ou $p2$. Elle spécialise la machine *Presentation* en introduisant une variable supplémentaire *pChoice* et en remplaçant le variant générique *var* par le variant spécifique *varChoice*. Ce dernier est initialisé

de manière aléatoire soit à 1 soit à 2 (*act*4). S'il est initialisé à 1, il déclenche *presentation*1 (*grd*2) qui affecte à *pChoice* la valeur de *p*1 (*act*2). S'il est initialisé à 2, il déclenche l'événement *presentation*2 (*grd*2) qui affecte à *pChoice* la valeur de *p*2 (*act*2). *varChoice* est décrémenté à 0 dans les deux évènements (*act*3) déclenchant ainsi *presentation*.

MACHINE ChoiceDecomposedPresentation
REFINES CombinedPresentation
SEES Decomposition
VARIABLES
p p1 p2 v v1 v2 varSeq varChoice pChoice
INVARIANTS
inv1 : $p1 \in Presentation$
inv2 : $v \in Presentation$
inv3 : $varChoice \in \{0, 1, 2\}$
inv4 : $pChoice \in Presentation$
inv5 : $varSeq < 1 \Rightarrow v = ChoiceDecompositionOperator(v1 \mapsto v2)$
inv6 : $varChoice = 0 \Rightarrow p1 = v1 \vee p2 = v2$
VARIANT
$varSeq + varChoice$

EVENTS
Initialisation
begin
act1 : $v :\in Presentation$
act2 : $pChoice :\in Presentation$
act3 : $varSeq := 1$
act4 : $varChoice :\in \{1, 2\}$
end
Event *decomposition* $\widehat{=}$ /* décomposition de *v**/
Status convergent
any
x x1 x2
where
grd1 : $x \in Presentation$
grd2 : $x1 \in Presentation$
grd3 : $x2 \in Presentation$
grd4 : $x = DecompositionOperator(x1 \mapsto x2)$
grd5 : $varSeq = 1$
grd6 : $var \in \mathbb{N}_1$
then
act1 : $v, v1, v2 := x, x1, x2$
act2 : $varSeq := varSeq - 1$
end
END

Event *presentation* $\widehat{=}$ /* construction de *p**/
refines *presentation*
when
grd1 : $varSeq = 0$
grd2 : $varChoice = 0$
then
act1 : $p := pChoice$
end
Event *presentation1* $\widehat{=}$ /* construction de *p*1*/
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varChoice = 1$
then
act1 : $p1 := v1$
act2 : $pChoice := p1$
act3 : $varChoice := 0$
end
Event *presentation2* $\widehat{=}$ /* construction de *p*2*/
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varChoice = 2$
then
act1 : $p2 := v2$
act2 : $pChoice := v2$
act3 : $varChoice := 0$
end

Figure 6.12 – Décomposition des présentations élémentaires : l'ordonnancement par choix

4.2 Le modèle de décomposition itérative

Le modèle de décomposition itérative des présentations (voir Figure 6.13) raffine le modèle de composition des présentations (voir section 3). Il décrit la décomposition itérative de la présentation p en n présentations $p1$. Il se compose des éléments suivants.

CONTEXT iterativeDecomposition
EXTENDS Combination
CONSTANTS
$PIter$ n
AXIOMS
axm1 : $PIter \in \mathbb{N} \times Presentation \rightarrow Presentation$
axm2 : $n \in \mathbb{N}_1$
axm3 : $\forall i, x \cdot (i \in 0 \mathinner{..} n-1 \wedge x \in Presentation) \Rightarrow PIter(i \mapsto x) = x$
END

MACHINE iterativeDecomposedPresentation
REFINES Presentation
SEES iterativeDecomposition
VARIABLES
p $p1$ v $v1$ $varIter$ i $varSeq$
INVARIANTS
inv1 : $varIter \in 0 \mathinner{..} n$
inv2 : $varSeq = 0 \Rightarrow v = PIter(n \mapsto v1)$
inv3 : $varIter < n \Rightarrow p1 = v1$ ...
VARIANT
$varIter$
EVENTS
Initialisation
begin
act1 : $varIter := n$
act2 : $varSeq := 1$ ...
end
Event *decomposition* $\widehat{=}$ /* décomposition de v*/
Status convergent
any
x $x1$
where
grd1 : $x \in Presentation$
grd2 : $x1 \in Presentation$
grd3 : $x = Iter(n \mapsto x1)$
grd4 : $varSeq = 1$
grd5 : $varIter = n$
then
act1 : $v, v1 := x, x1$
act2 : $varSeq := 0$
end

Event *presentation1* $\widehat{=}$ /* construction de $p1$*/
Status convergent
when
grd1 : $varIter \in 1 \mathinner{..} n$
grd2 : $varSeq = 0$
then
act1 : $p1 := PIter(n - varIter \mapsto v1)$
act2 : $varIter := varIter - 1$
end
Event *presentation* $\widehat{=}$ /* construction de p*/
refines *presentation*
when
grd1 : $varIter = 0$
grd2 : $varSeq = 0$
then
act1 : $p := PIter(n - varIter \mapsto v1)$
end
END

Figure 6.13 – Décomposition des présentations élémentaires : l'itération

4.2.1 La partie CONTEXT

Le composant CONTEXT *IterativeDecomposition* raffine le contexte de combinaison des présentations correspondant au raffinement précédent. Nous représentons ce contexte de combinaison par *Combination*. Il définit l'opérateur *PIter* (*axm*1) qui combine n fois et de manière itérative, une même présentation pour obtenir une nouvelle présentation. L'axiome (*axm*3) décrit que si on combine une présentation, un nombre de fois inférieur à n, la présentation résultante est identique à la présentation initiale.

4.2.2 La partie MACHINE

La MACHINE *IterativeDecomposedPresentation* raffine la machine décrivant la combinaison des présentations que nous notons *CombinedPresentation*. Elle modélise la production n fois de la présentation multimodale composite $p1$ afin de produire la présentation globale p. Tout d'abord, elle modélise la décomposition d'une présentation v en n présentations $v1$, formalisée dans l'évènement *decomposition* (*act*1). Ensuite, elle produit la présentation $p1$ n fois dans l'événement *presentation*1 (*act*1). Au bout de la $n$ième itération, elle produit la présentation p dans l'événement *presentation* (*act*1). L'ordre iteratif de production des présentations est assuré par le variant *varIter* initialisé à n (*act*1), il conditionne la production de $p1$ (*presentation*1 : *grd*1). Lorsqu'il atteint 0 (*presentation* : *grd*1), il produit p. L'invariant de collage (*inv*3) permet d'établir la valeur de $p1$ en fonction de $v1$ à chaque itération.

4.3 Bilan des obligations de preuve

Les différents modèles relatifs à la décomposition des présentations ont généré 49 obligations de preuves (OP) (voir Table 6.2) dont 44 ont été prouvées de manière automatique par le prouveur de la plate-forme *Rodin*, 5 ont nécessité l'intervention du concepteur dans une preuve interactive. Elles consistent en 3 obligations de preuves dans *ParallelPresentation*, dont deux portant sur deux obligations de preuves de variant dans les événements *presentation*1 et *presentation*2 et une obligation de preuve de simulation sur l'action (*act*1) dans l'événement *presentation*. Deux obligations de preuves dans les machines *ChoiceDecomposedPresentation* et *IterativeDecomposedPresentation* portant sur la simulation de la construction de la présentation globale p .

Composant	OP automatiques	OP manuelles	OP totales	OP non prouvées
ComplementaryDecomposition	1	0	1	0
RedundantDecomposition	1	0	1	0
iterativeDecomposition	1	0	1	0
ParallelPresentation	23	3	26	0
ChoiceDecomposedPresentation	10	1	11	0
iterativeDecomposedPresentation	8	1	9	0
Total	44	5	49	0

Tableau 6.2 – La décomposition des présentations : les obligations de preuves

5 Le modèle d'affectation

Le modèle relatif à l'affectation des modalités et médias est le dernier raffinement de la phase d'allocation. Il modélise l'affectation des unités de présentation élémentaires sur les différents médias. Il succède au raffinement relatif à la décomposition des présentations multimodales et modélise la troisème règle syntaxique de l'allocation que nous rappelons dans 6.3.

$$ITEM ::= affectation(UPE) \tag{6.3}$$

Ainsi, les unités de présentation élémentaires *upe* obtenues lors de la phase de décomposition sont allouées avec des couples (*modalité*, *média*) suivant le raffinement décrit en Figure 6.14. Il est dérivé du modèle générique d'affectation présenté dans le chapitre 4. Il se compose des éléments suivants.

5.1 La partie CONTEXT

Le composant CONTEXT *affectation* étend le contexte de décomposition des présentations correspondant au raffinement précédent. Nous représentons ce contexte de décomposition par *Decomposition*. Il définit les ensembles *Media* pour les médias, *Modality* pour les modalités et *PresentUnit* (*axm*1) pour les unités de présentation multimodales obtenues lors du raffinement précédent relatif à la décomposition des présentations élémentaires. Il définit également la fonction *affectation* (*axm*2) qui affecte une modalité à une unité de présentation élémentaire. D'autre part, le contexte définit l'ensemble des médias qui peuvent restituer la modalité pour la véhiculer par le biais de la fonction *linkModalityMedia* (*axm*3) qui affecte à une modalité l'ensemble des médias qui peuvent la restituer.

CONTEXT Affectation
EXTENDS Decomposition
SETS
presentationUnit Modality Media
CONSTANTS
affectation linkModalityMedia
AXIOMS
axm1 : $presentationUnit \subseteq Presentation$
axm2 : $affectation \in presentationUnit \rightarrow Modality$
axm3 : $linkModalityMedia \in Modality \rightarrow \mathbb{P}(Media)$
END

MACHINE AffectedPresentation
REFINES DecomposedPresentation
SEES Affectation
VARIABLES
item p
INVARIANTS
inv1 : $item \in presentationUnit \rightarrow Modality \times Media$
inv2 : $p \in Presentation$
EVENTS
Initialisation
begin
act1 : $item :\in presentationUnit \rightarrow Modality \times Media$
act2 : $p :\in Presentation$
end

Event *affectation* $\widehat{=}$ /* affectation de la modalité et du média */
any
m
where
grd1 : $p \in presentationUnit$
grd2 : $m \in linkModalityMedia(affectation(p1))$
then
act1 : $item(p) := (affectation(p) \mapsto m)$
end
END

Figure 6.14 – Affectation des modalités et des médias

5.2 La partie MACHINE

La MACHINE *AffectedPresentation* raffine la machine décrivant la décomposition des présentations que nous notons *DecomposedPresentation*. Elle introduit pour chaque unité de présentation élémentaire un évènement *affectation* qui affecte à l'unité de présentation élémentaire p ($grd1$) au moyen de la fonction *item* ($act1$), le couple ($modality$, $media$) qui la restitue. La modalité est allouée à la présentation au moyen de la fonction *affectation*, le média par contre, est sélectionné au moment du déclenchement de l'évènement *affectation* dans l'ensemble des médias qui peuvent restituer la modalité, en utilisant la fonction *linkModalityMedia* ($grd2$).

6 Bilan des obligations de preuve

Les différents modèles relatifs à l'affectation ont généré 8 obligations de preuves (OP) (voir Table 6.3) qui ont été totalement prouvées de manière automatique par le prouveur de

la plate-forme *Rodin*.

Composant	OP automatiques	OP manuelles	OP totales	OP non prouvées
Affectation	0	0	0	0
AffectedPresentation	8	0	8	0
Total	8	0	8	0

Tableau 6.3 – L'affectation des modalités et des médias : les obligations de preuves

7 Instanciation du modèle B Évènementiel d'allocation

La formalisation B Évènementiel de l'allocation d'une interface multimodale en sortie concrète, commence par sa description dans le modèle d'allocation générique que nous présentons dans le chapitre 3. A partir de cette définition, la formalisation B Évènementiel est opérée en trois étapes :

- la construction du modèle de combinaison des présentations qui raffine le modèle de fission sémantique. Il combine le contexte correspondant à l'opérateur sémantique et la machine correspondante à l'opérateur temporel définissant la combinaison des présentations. Il invoque le modèle de la combinaison itérative dans le cas de la combinaison itérative. Une fois le modèle de combinaison construit, il est instancié par la définition en extension des ensembles et fonctions. Cette étape est renouvelée pour toutes les combinaisons de présentations opérées lors de la phase d'allocation.
- la construction du modèle de décomposition des présentations qui raffine le modèle de combinaison des présentations. Il combine le contexte correspondant à l'opérateur sémantique et la machine correspondante à l'opérateur temporel définissant la décomposition des présentations. Il invoque le modèle de la décomposition itérative dans le cas de la décomposition itérative. Une fois le modèle de décomposition construit, il est instancié par la définition en extension des ensembles et fonctions. Cette étape est renouvelée pour toutes les décompositions de présentations opérées lors de la phase d'allocation, jusqu'à aboutir aux unités de présentations élémentaires.
- la construction du modèle d'affectation des présentations qui raffine le modèle de décomposition des présentations. Il est instancié par la définition en extension des ensembles et fonctions.

8 Application à l'étude de cas

Conformément à la démarche décrite ci-dessus, l'application des modèles de l'allocation à l'étude de cas *CityMap* aboutit aux trois raffinements identifiés dans la Figure 6.15).

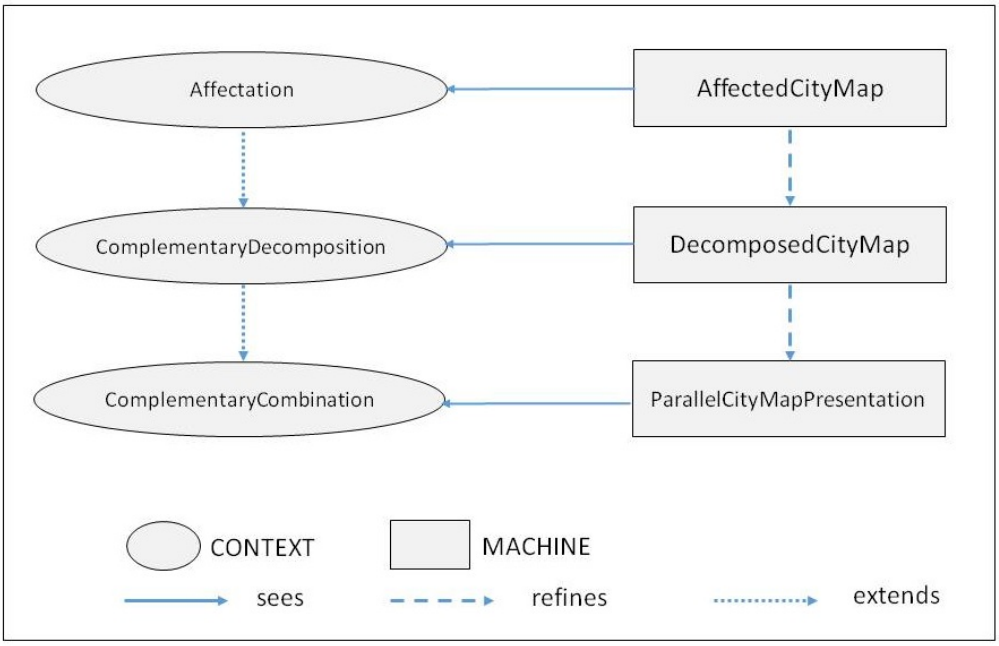

Figure 6.15 – *CityMap* : processus de formalisation B Évènementiel de l'allocation

8.1 Le modèle de combinaison

Le premier raffinement (voir Figure. 6.16) est relatif à la combinaison complémentaire et parallèle des présentations *presentSee* et *presentMap* correspondantes respectivement aux informations fissionnées $infoSee$ et $infoMap$. Il succède au dernier raffinement de la fission sémantique (voir Figure 5.11). Il en étend le contexte (*ComplementaryFission*) et en raffine la machine *ParallelFissionedCityMap*. Ce raffinement instancie le contexte de combinaison complémentaire *ComplementaryCombination* par les axiomes (*axm*2, *axm*3 et *axm*4) et la machine de combinaison parallèle *ParallelPresentation* en instanciant les variables i, $i1$ et $i2$ par $infoCityMap$, $infoSee$ et $infoMap$.

CONTEXT ComplementaryCombination
EXTENDS ComplementaryFission
CONSTANTS

complementaryOperator

AXIOMS

axm1 : $complementaryOperator \in Presentation \times Presentation \rightarrow Presentation$
axm2 : $complementaryOperator(presentSee \mapsto presentMap) = presentCityMap$
axm3 : $complementaryOperator(emptyP \mapsto presentMap) = emptyP$
axm4 : $complementaryOperator(presentSee \mapsto emptyP) = emptyP$

END

MACHINE ParallelCityMapPresentation
REFINES ParallelFissionedCityMap
SEES ComplementaryCombination
VARIABLES

i d p i1 i2 d1 d2 p1 p2 varSeq varPar1 varPar2

INVARIANTS

inv1 : $p1 \in Presentation$
inv2 : $p2 \in Presentation$
inv3 : $varSeq < 1 \Rightarrow allocation(i) = PcomplementaryOperator(allocation(i1) \mapsto allocation(i2))$
inv4 : $varPar1 = 0 \Rightarrow p1 = allocation(i1)$
inv5 : $varPar2 = 0 \Rightarrow p2 = allocation(i2)$

EVENTS
Initialisation

begin

act1 : $varSeq := 1$
act2 : $varPar1 := 1$
act3 : $varPar2 := 1$
act4 : $p1 :\in Presentation$
act5 : $p2 :\in Presentation$

end

Event *information* $\widehat{=}$ /*délivrance de infoCityMap*/
Status convergent
extends *information*

when

grd1 : $varSeq = 1$
grd2 : $varPar1 = 1$
grd3 : $varPar2 = 1$

then

act1 : $i, i1, i2 := infoCityMap, infoSee, infoMap$
act2 : $varSeq := varSeq - 1$

end

Event *presentationCityMap* $\widehat{=}$ /*construction de presentCityMap*/
refines *infoCityMap*

when

grd1 : $varSeq = 0$
grd2 : $varPar1 = 0$
grd3 : $varPar2 = 0$

then

act1 : $d := complementaryOperator(d1 \mapsto d2)$
act2 : $p := PcomplementaryOperator(p1 \mapsto p2)$

end

Event *presentationSee* $\widehat{=}$ /*construction de presentSee*/
Status convergent
extends *infoSee*

when

grd1 : $varSeq = 0$
grd2 : $varPar1 = 1$

then

act1 : $d1 := interpretation(i1)$
act2 : $varPar1 := varPar1 - 1$
act3 : $p1 := allocation(i1)$

end

Event *presentationMap* $\widehat{=}$ /*construction de presentMap*/
Status convergent
extends *infoMap*

when

grd1 : $varSeq = 0$
grd2 : $varPar2 = 1$

then

act1 : $d2 := interpretation(i2)$
act2 : $varPar2 := varPar2 - 1$
act3 : $p2 := allocation(i2)$

end

END

Figure 6.16 – *CityMap* : la combinaison complémentaire et parallèle des présentations

8.2 Le modèle de décomposition

CONTEXT ComplementaryDecomposition
EXTENDS ComplementaryCombination
CONSTANTS

complementaryOperator

AXIOMS

axm1 : $complementaryOperator \in Presentation \times Presentation \rightarrow Presentation$
axm2 : $complementaryOperator(presentSeeSpeech \mapsto presentSeeExpression) = presentSee$
axm3 : $complementaryOperator(emptyP \mapsto presentSeeExpression) = emptyP$
axm4 : $complementaryOperator(presentSeeSpeech \mapsto emptyP) = emptyP$

END

MACHINE DecomposedCityMap
REFINES ParallelCityMapPresentation
SEES ComplementaryDecomposition
VARIABLES

p11 p12 varPar1 varPar2 varSeq i d p i1 i2 d1 d2 p1 p2 varPar3 varPar4

INVARIANTS

inv1 : $p11 \in Presentation$
inv2 : $varPar3 \in \{0,1\}$
inv3 : $varPar4 \in \{0,1\}$
inv4 : $varPar3 = 0 \Rightarrow p11 = presentSeeSpeech$
inv5 : $varPar4 = 0 \Rightarrow p12 = presentSeeExpression$
...

VARIANT

$varPar3 + varPar4$

EVENTS
Initialisation
begin

act1 : $p11 :\in Presentation$
act2 : $varSeq := 1$
act3 : $varPar1 := 1$
act4 : $varPar2 := 1$
act5 : $varPar3 := 1$
act6 : $varPar4 := 1$ ...

end

Event *information* $\widehat{=}$ /*délivrance de infoCityMap*/
...

Event *presentationCityMap* $\widehat{=}$ /*construction de presentCityMap*/
...

Event *presentationMap* $\widehat{=}$ /*construction de presentMap*/
...

Event *presentationSee* $\widehat{=}$ /*construction de presentSee*/
Status convergent
refines *presentationSee*
when

grd1 : $varSeq = 0$
grd2 : $varPar1 = 1$
grd3 : $varPar3 = 0$
grd4 : $varPar4 = 0$

then

act1 : $d1 := interpretation(i1)$
act2 : $varPar1 := varPar1 - 1$
act3 : $p1 := PcomplementaryOperator(p11 \mapsto p12)$

end

Event *presentationSeeSpeech* $\widehat{=}$ /*construction de presentSeeSpeech*/
Status convergent
when

grd1 : $varSeq = 0$
grd2 : $varPar1 = 1$
grd3 : $varPar3 = 1$

then

act1 : $p11 := presentSeeSpeech$
act2 : $varPar3 := varPar3 - 1$

end

Event *presentationSeeExpression* $\widehat{=}$ /*construction de presentSeeExpression*/
Status convergent
when

grd1 : $varSeq = 0$
grd2 : $varPar1 = 1$
grd3 : $varPar4 = 1$

then

act1 : $p12 := presentSeeExpression$
act2 : $varPar4 := varPar4 - 1$

end

END

Figure 6.17 – *CityMap* : la décomposition complémentaire des présentations

Le second raffinement (6.17) est relatif à la décomposition de la présentation *presentSee* en deux présentations complémentaires *presentSeeSpeech* et *presentSeeExpression*. Il succède au raffinement de combinaison des présentations (voir Figure 6.16). Il en étend le contexte *ComplementaryCombination* et en raffine la machine *ParallelCityMapPresentation*. Ce raffinement instancie le contexte *ComplementaryDecomposition* de décomposition complémentaire par les axiomes (*axm*2, *axm*3 et *axm*4) et la machine *ParallelDecomposedPresentation* qui produit les présentations décomposées en parallèle en instanciant les variables v, $v1$ et $v2$ par *presentSee*, *presentSeeSpeech* et *presentSeeExpression*.

8.3 Le modèle d'affectation

Le troisième raffinement (voir Figure. 6.18) est relatif à l'affectation des modalités et des médias aux présentations *presentSeeSpeech*, *presentSeeExpression* et *presentMap*. Il succède au raffinement de décomposition des présentations (voir Figure 6.17). Il en étend le contexte (*ComplementaryDecomposition*) et en raffine la machine *DecomposedCityMap*. Ce raffinement instancie d'une part, le contexte $affectation$ par la définition des ensembles *presentationUnit*, *Modality* et *Media* et des fonctions $affectation$ (*axm*3, *axm*4 et *axm*5) et *linkModalityMedia* (*axm*6, *axm*7 et *axm*8). D'autre part, il instancie la machine $AffectedPresentation$ en instanciant la variable p dans les évènements $affectationSeeSpeech$, $affectationSeeExpression$ et $affectationMap$, par les valeurs des unités de présentation élémentaires *presentSeeSpeech*, *presentSeeExpression* et *presentMap*.

8.4 Bilan des obligations de preuve

Les différents modèles relatifs à l'allocation de l'étude de cas *CityMap* ont généré 55 obligations de preuves (OP) (voir Table 6.4) dont 47 ont été prouvées de manière automatique par le prouveur de la plate-forme *Rodin*, les 8 restantes ont été prouvées par preuve interactive. Parmi ces obligations de preuves effectuées de manière interactive, 2 portent sur le variant dans *ParallelCityMapPresentation*, 2 portent sur le variant et une sur la simulation dans la construction de la variable $p1$ dans *DecomposedCityMap*.

CONTEXT AffectationCityMap
EXTENDS ComplementaryDecomposition
SETS

presentationUnit = {*presentMap*, *presentSeeSpeech*, *presentSeeExpression*}
Modality = {*picture*, *speech*, *expression*}
Media = {*speaker*, *screen*}

CONSTANTS

affectation linkModalityMedia

AXIOMS

axm1 : $affectation \in presentationUnit \rightarrow Modality$
axm2 : $linkModalityMedia \in Modality \rightarrow \mathbb{P}(Media)$
axm3 : $affectation(presentSeeSpeech) = speech$
axm4 : $affectation(presentSeeExpression) = expression$
axm5 : $affectation(presentMap) = picture$
axm6 : $linkModalityMedia(speech) = speaker$
axm7 : $linkModalityMedia(expression) = screen$
axm8 : $linkModalityMedia(picture) = screen$

END

MACHINE AffectedCityMap
REFINES DecomposedCityMap
SEES AffectationCityMap
VARIABLES

item i i1 i2 d d1 d2 p p1 p2 varSeq
varPar1 varPar2 varPar3 varPar4 p11 p12

INVARIANTS

inv1 : $item \in presentationUnit \rightarrow Modality \times Media$

EVENTS
Initialisation
begin

act1 : $varSeq := 1$
act2 : $varPar1 := 1$
act3 : $varPar2 := 1$
act4 : $varPar3 := 1$
act5 : $varPar4 := 1$

end
Event *information* $\widehat{=}$ /*délivrance de infoCityMap*/
...
Event *presentationCityMap* $\widehat{=}$ /*construction de presentCityMap*/
...
Event *presentationSee* $\widehat{=}$ /*construction de presentSee*/
...
Event *presentationMap* $\widehat{=}$ /*construction de presentMap*/
...
Event *presentationSeeSpeech* $\widehat{=}$ /*construction de presentSeeSpeech*/
...
Event *presentationSeeExpression* $\widehat{=}$ /*construction de presentSeeExpression*/
...

Event *affectationMap* $\widehat{=}$ /*affectation de presentMap*/
any
m
where
grd1 : $m \in linkModalityMedia(affectation(presentMap))$
grd2 : $varPar2 = 0$
grd3 : $varSeq = 0$
then
act1 : $item(p1) := (affectation(p2) \mapsto m)$
end
Event *affectationSeeSpeech* $\widehat{=}$ /*affectation de presentSeeSpeech*/
any
m
where
grd1 : $m \in linkModalityMedia(affectation(presentSeeSpeech))$
grd2 : $varPar1 = 0$
then
act1 : $item(p11) := (affectation(p11) \mapsto m)$
end
Event *affectationSeeExpression* $\widehat{=}$ /*affectation de presentSeeExpression*/
any
m
where
grd1 : $m \in linkModalityMedia(affectation(presentSeeExpression))$
grd2 : $varPar2 = 0$
then
act1 : $item(p12) := (affectation(p12) \mapsto m)$
end
END

Figure 6.18 – *CityMap* : l'affectation des modalités et média

Composant	OP automatiques	OP manuelles	OP totales	OP non prouvées
ComplementaryCombination	5	0	5	0
ComplementaryDecomposition	3	0	3	0
Affectation	3	0	3	0
ParallelCityMapPresentation	15	2	17	0
DecomposedCityMap	13	3	16	0
AffectedCityMap	11	0	11	0
Total	50	5	55	0

Tableau 6.4 – L'allocation de *CityMap* : les obligations de preuve

9 Vérification des propriétés

Une fois que l'interface multimodale en sortie est formellement modélisée, nous devons pouvoir garantir la satisfaction de propriétés de sûreté, de vivacité et d'utilisabilité sur cette interface. Les modèles B Évènementiel proposés sont génériques permettant ainsi de vérifier plusieurs types de propriétés. L'utilisation de B Évènementiel offre la possibilité de garantir de telles propriétés par preuve de théorème en utilisant le prouveur de la plate-forme *Rodin*. Nous avons identifié deux façons de procéder à la vérification de ces propriétés, par la définition d'un invariant ou par la définition d'un raffinement. Nous illustrons ci-après ces deux méthodes par la vérification d'une propriété de vivacité : l'absence de blocage et la vérification d'une propriété de sûreté : la propriété d'absence de collisions.

9.1 La vérification par définition d'invariant

Elle consiste à introduire dans la machine un invariant exprimant la propriété à vérifier sans renforcer les gardes des évènements intervenant dans la satisfaction de la propriété. La propriété est considérée comme vraie lorsque la préservation de l'invariant exprimant la propriété à vérifier est établie dans toute la machine. Ainsi, la propriété de non-blocage est vérifiée dans les différents modèles B Évènementiel développés. Elle est modélisée dans chaque machine par un invariant de disjonction de gardes qui assure qu'à tout moment, la garde d'au moins un événement est activée. Un exemple de la vérification de la propriété de non-blocage est présenté dans la Figure 6.19, il montre la machine relative à la fission sémantique séquentielle présentée dans la Figure 5.5, pour laquelle est vérifiée la propriété d'absence de blocage. Cette dernière est vérifiée par l'introduction de l'invariant (*inv*4) qui

consiste en la disjonction des gardes des évènements : *in formation*, *interprtation*, *interprtation*1 et *interprtation*2.

MACHINE SequentialFissionedInformation
SEES SemanticFission
VARIABLES
...
INVARIANTS
inv4 : $(\forall x, x1, x2 \cdot (x \in Information \land x1 \in Information \land x2 \in Information \land interpretation(x) = complementaryOperator(interpretation(x1) \mapsto interpretation(x2))) \land varSeq = 1 \land varSeq1 = 2) \lor (varSeq = 0 \land varSeq1 = 0) \lor (varSeq = 0 \land varSeq1 = 2) \lor (varSeq = 0 \land varSeq1 = 1)$ ...
VARIANT
...

EVENTS
Initialisation
begin
...
end

Event *information* $\widehat{=}$ /* délivrance de i*/
Status convergent
refines *information*
any
x x1 x2
where
grd1 : $x \in Information$
grd2 : $x1 \in Information$
grd3 : $x2 \in Information$
grd4 : $interpretation(x) = semanticOperator(interpretation(x1) \mapsto interpretation(x2))$
grd5 : $varSeq = 1$
grd6 : $varSeq1 = 2$
then
...
end

Event *interpretation* $\widehat{=}$ /* calcul de d*/
refines *interpretation*
when
grd1 : $varSeq = 0$
grd2 : $varSeq1 = 0$
then
...
end

Event *interpretation1* $\widehat{=}$ /* calcul de d1*/
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varSeq1 = 2$
then
...
end

Event *interpretation2* $\widehat{=}$ /* calcul de d2*/
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varSeq1 = 1$
then
...
end

END

Figure 6.19 – Vérification de la propriété d'absence de blocage

9.2 La vérification opérationnelle par raffinement

Elle consiste à assurer la satisfaisabilité de la propriété dans la machine par l'introduction d'un raffinement dédié à exhiber les caractéristiques intervenant dans satisfaisabilité de la propriété. La propriété est alors exprimée par un invariant, elle est considérée comme vraie lorsque la préservation de cet invariant est établie dans toute la machine. A titre d'exemple,

une propriété de sûreté, appelée absence de collisions peut être vérifiée, elle assure qu'aucune collision ne peut se produire sur un média non partageable. Une collision consiste à produire deux modalités dans la même fenêtre temporelle, sur un média non partageable tel que la présentation en même temps, d'un bip sonore et d'un discours produit par synthèse vocale. Afin de garantir la propriété d'absence de collisions, il est nécessaire de contrôler l'allocation des médias. Par conséquent, la vérification de la propriété d'absence de collisions introduit un nouveau raffinement, successif au raffinement relatif à l'affectation. Dans ce raffinement, nous avons besoin de savoir, à tout moment, le nombre de modalités allouées aux médias. La propriété d'absence de collisions est vérifiée par l'introduction d'un invariant exprimant que si le média n'est pas partageable, alors la cardinalité de l'utilisation de ce média doit être inférieure à 1 (les médias ne peuvent pas être attribués à plus d'une modalité). Ainsi, le raffinement relatif à la vérification de la propriété d'absence de collisions (voir Figure. 6.20) comprend le contexte et de la machine suivants.

9.2.1 Le composant CONTEXT

La partie CONTEXT *CollisionFreeness* définit la fonction de partageabilité *share* sur les médias ($axm1$). Il étend le contexte d'affectation $Affectation$.

9.2.2 Le composant MACHINE

La MACHINE générique *CollisionFree* raffine la machine qui affecte les présentations avec les modalités et les média $AffectedPresentation$ (voir Figure 6.14). Elle contrôle l'allocation des modalités aux média, elle introduit pour chaque média, l'ensemble variable *mediaUse*, sous ensemble de l'ensemble des modalités ($inv1$) qui recense les modalités qui emploient le média, il est initialisé à l'ensemble vide ($act1$). Elle introduit également pour chaque évènement $affectation$ deux évènements : *alloc* et $free$. L'événement *alloc* précède l'évènement $affectation$. Il alloue la modalité au média en ajoutant la modalité à l'ensemble *mediaUse* ($act1$). L'événement $free$ survient après l'évènement $affectation$, il libère le média de la modalité en supprimant la modalité de l'ensemble *mediaUse* ($act1$). L'ordonnancement séquentiel des évènements *alloc*, $affectation$ et $free$ est assuré par le variant *varSeq*.

La propriété d'absence de collision est vérifiée par l'introduction de l'invariant ($inv4$) qui exprime que si le média n'est pas partageable, la cardinalité de son utilisation doit être inférieure à 2.

CONTEXT CollisionFreeness
EXTENDS Affectation
CONSTANTS
share
AXIOMS
axm1 : $share \in Media \rightarrow BOOL$
END

MACHINE CollisionFree
REFINES AffectedPresentation
SEES CollisionFreeness
VARIABLES
mediaUse varSeq p item
INVARIANTS
inv1 : $mediaUse \subseteq Modality$
inv2 : $varSeq \in \{0, 1, 2\}$
inv3 : $p \in UPresentation$
inv4 : $\forall x \cdot (x \in Media \land share(x) = FALSE) \Rightarrow card(mediaUse) < 2)$
VARIANT
varSeq
EVENTS
Initialisation
begin
act1 : $mediaUse := \emptyset$
act2 : $varSeq := 2$
act3 : $p :\in UPresentation$
act4 : $item :\in presentationUnit \rightarrow Modality \times Media$
end
Event *alloc* $\widehat{=}$ /*allocation du média*/
Status convergent
when
grd1 : $varSeq = 2$
then
act1 : $mediaUse := mediaUse \cup \{affectation(p)\}$
act2 : $varSeq := 1$
end

Event *affectation* $\widehat{=}$ /*affectation de la modalité et du média*/
Status convergent
refines *affectation*
any
m
where
grd1 : $p \in presentationUnit$
grd2 : $m \in linkModalityMedia(affectation(p1))$
grd3 : $varSeq = 1$
then
act1 : $item(p) := (affectation(p) \mapsto m)$
act3 : $varSeq := 0$
end
Event *free* $\widehat{=}$ /*libération du média*/
when
grd1 : $varSeq = 0$
then
act1 : $mediaUse := mediaUse \setminus \{affectation(p)\}$
end
END

Figure 6.20 – Vérification de la propriété d'absence de collisions

La vérification de la propriété d'absence de collision sur l'étude de cas *CityMap* est effectuée au moyen du raffinement présenté dans Figure. 6.21. Il se compose du contexte et de la machine décrits ci-après.

1. **Le CONTEXT** *CollisionFree***.** Il définit la fonction *share*, l'axiome *axm*2 définit le média écran comme un média partageable et l'axiome *axm*3 définit le média haut-parleur comme un média non partageable.

CONTEXT CollisionFreeness
EXTENDS Affectation
CONSTANTS
share
AXIOMS
axm1 : $share \in Media \rightarrow BOOL$
axm2 : $share(screen) = TRUE$
axm3 : $share(speaker) = FALSE$
END

MACHINE CollisionFreeCityMap
REFINES AffectedCityMap
SEES collisionFreeness
VARIABLES
speakerUse screenUse varSeq1 varSeq2 item ...
INVARIANTS
inv1 : $speakerUse \subseteq Modality$
inv2 : $screenUse \subseteq Modality$
inv3 : $varSeq1 \in \{0,1,2\}$
inv4 : $varSeq2 \in \{0,1,2\}$
inv5 : $varSeq3 \in \{0,1,2\}$
inv6 : $card(speakerUse) < 2$
VARIANT
$varSeq1 + varSeq2 + varSeq3$
EVENTS
Initialisation
begin
act1 : $item :\in presentationUnit \rightarrow Modality \times Media$
act2 : $speakerUse := \emptyset$
act3 : $screenUse := \emptyset$
act4 : $varSeq1 := 2$
act5 : $varSeq2 := 2$
act6 : $varSeq3 := 2$
end
Event *information* $\widehat{=}$ /*délivrance de i*/
...
Event *presentationSee* $\widehat{=}$ /* construction de presentSee*/
...
Event *presentationSeeExpression* $\widehat{=}$ /* construction de presentSeeExpression*/
...
Event *allocMap* $\widehat{=}$ /* allocation de l'écran*/
...
Event *affectationMap* $\widehat{=}$ /* affectation de presentMap */
...
Event *freeMap* $\widehat{=}$ /* libération de l'écran*/
...

Event *allocSeeExpression* $\widehat{=}$ /* allocation de l'écran*/
Status convergent
when
grd1 : $varSeq = 0$
grd2 : $varPar1 = 1$
grd3 : $varPar4 = 0$
grd4 : $varSeq3 = 2$
then
act1 : $screenUse := screenUse \cup \{affectation(p12)\}$
act2 : $varSeq3 := varSeq3 - 1$
end
Event *affectationSeeExpression* $\widehat{=}$ /* affectation de presentSeeExpression */
Status convergent
any
m
where
grd1 : $m \in linkModalityMedia(affectation(presentSeeExpression))$
grd2 : $varSeq = 0$
grd3 : $varPar1 = 1$
grd4 : $varPar4 = 0$
grd5 : $varSeq3 = 1$
then
act1 : $item(p12) := (affectation(p12) \mapsto m)$
act2 : $varSeq3 := varSeq3 - 1$
end
Event *freeSeeExpression* $\widehat{=}$ /* libération de l'écran*/
when
grd1 : $varSeq = 0$
grd2 : $varPar1 = 1$
grd3 : $varPar4 = 0$
grd4 : $varSeq3 = 0$
then
act1 : $screenUse := screenUse \setminus \{affectation(p12)\}$
end
END

Figure 6.21 – *CityMap* : vérification de la propriété d'absence de collisions

2. **La MACHINE** *CollisionFreeCityMap***.** Elle instancie la machine *CollisionFree* en introduisant deux variables de type ensemble *speakerUse* et *screenUse* pour la variable *mediaUse*. Ces ensembles regroupent les modalités utilisant respectivement les média haut-parleur et écran. Elle introduit également pour chaque événement : *presentationSeeSpeech*, *presentationSeeExpression* et *presentationMap* les événements d'allocation *alloc* et de libération $free$ des modalités pour contrôler l'attribution de l'écran et du haut-parleur aux modalités : *presentSeeSpeech*, *presentSeeExpression* et *presentMap*.
   La propriété d'absence de collisions est exprimée au moyen de l'invariant (*inv*6), il exprime que pour le média non partageable haut-parleur, le nombre de modalités allouées (contenues dans *speakerUse*) doit être inférieur à 2.

Le développement de ce raffinement a induit 21 obligations de preuves (OP) (voir Table 6.5) dont 13 ont été prouvées de manière automatique par le prouveur de la plateforme *Rodin*, les 8 restantes ont été prouvées par preuve interactive. Elles portent sur la bonne définition des ensembles *speakerUse* et *screenUse* dans les différents événements d'allocation et libération.

Composant	OP automatiques	OP manuelles	OP totales	OP non prouvées
CollisionFreeness	2	0	2	0
CollisionFreeCityMap	11	8	19	0
Total	13	8	21	0

Tableau 6.5 – Vérification de l'absence de collisions dans *CityMap* : les obligations de preuves

10 Conclusion

Nous avons présenté dans ce chapitre les développements B Évènementiel de l'allocation. Ils permettent de modéliser formellement la deuxième étape de construction de l'interface multimodale en sortie : l'allocation. Les modèles présentés représentent une succession de raffinements qui prennent pour modèle abstrait le modèle de fission sémantique présenté dans le chapitre 5. Ils modélisent les étapes de combinaison des présentations correspondant aux informations fissionnées, de décomposition des présentations et d'affectation des modalités et média. Ces modèles ont été dérivés du modèle générique cadre présenté dans le chapitre 4, ils en démontrent la généricité.
Ces modèles sont génériques, ils permettent moyennant la définition en extension des ensembles et fonctions définissant l'allocation de l'interface de spécifier toute interface multi-

modale en sortie allouée selon le modèle conceptuel que nous proposons dans le chapitre 3. Cependant, ils sont plus précis que les modèles génériques d'allocation et d'affectation présentés dans le chapitre 4, car chaque modèle est dédié à une variante (opérateur sémantique / opérateur temporel) décrivant la combinaison et la décomposition, telle que définie dans le modèle formel de conception. L'utilisation de ces modèles pour la modélisation d'une interface concrète consiste à combiner et instancier les modèles dédiés aux opérateurs sémantiques et temporels invoqués par la définition de l'interface sans introduire de raffinements supplémentaires.
Le dernier raffinement des développements B Évènementiel pour l'allocation constitue le dernier modèle dans le processus de formalisation de l'interface Homme-Machine multimodale en sortie. Il formalise la présentation multimodale à instancier afin de construire l'interface multimodale en sortie.

Nous avons également présenté la possibilité d'effectuer la vérification de propriétés sur ces modèles de deux manières. Par la définition d'un invariant telle que pour la propriété d'absence de blocage ou par l'introduction d'un nouveau raffinement telle que pour la propriété d'absence de collisions.

Conclusion générale et perspectives

Le travail présenté dans cette thèse s'intéresse au développement sûr des IHM multimodales en sortie. Son objectif est de proposer une solution méthodologique et opérationnelle pour assister le concepteur d'une interface multimodale en sortie dans un contexte critique. Cette solution doit permettre au concepteur de spécifier toutes les caractéristiques jugées pertinentes pour le développement de l'interface, mais elle doit également offrir la possibilité d'affirmer le respect d'exigences établies très tôt dans le processus de conception de ces interfaces.

L'étude bibliographique menée sur les systèmes interactifs et sur les développements formels entrepris pour les concevoir nous a permis, d'une part, d'identifier le modèle WWHT. Il s'agit du modèle semi-formel décrivant le plus exhaustivement la construction de l'interface multimodale en sortie bien qu'il ne décrit pas les relations sémantiques et temporelles entre composants fissionnés et les différents schémas de construction de la présentation finale (combinaison temporelle et sémantique).

D'autre part, nous avons constaté que la méthode B Événementiel constituait une méthode éprouvée pour la formalisation des systèmes interactifs : les principes de raffinement et de validation par preuve de théorème sur lesquels elle repose permettent de maitriser la complexité du processus de modélisation et de validation des propriétés et d'échapper au problème de l'explosion combinatoire. A fortiori, de par son caractère événementiel, elle se prête bien à la modélisation des aspects de concurrence et de synchronisation souvent induits par la multimodalité.

Méthodologie

La proposition présentée dans ce mémoire pour la conception et la vérification formelles des interfaces multimodales en sortie se décline en deux volets.

1. Un modèle de conception formel et générique pour la construction des interfaces multimodales en sortie, indépendamment de toute technique de spécification formelle.
2. Une formalisation avec la méthode B Évènementiel du modèle de conception formel et générique par l'application de règles de transformation.

Modèle de conception formel et générique

Le modèle de conception générique que nous proposons est formel, il permet spécifier l'interface multimodale en sortie conformément aux deux étapes de fission sémantique et d'allocation suivant une démarche de modélisation progressive. Il est doté d'une syntaxe, d'une sémantique statique et dynamique permettant de décrire avec rigueur le fonctionnement de l'interface multimodale en sortie. Il est également générique, dans la mesure où il peut être implémenté dans diverses techniques formelles.
Le modèle de conception que nous proposons est basé sur le modèle semi-formel WWHT, qui a été développé pour répondre aux besoins des concepteurs en matière de modélisation des interfaces multimodales en sortie. Notre modèle formel détaille et précise le modèle WWHT par la proposition de schémas de combinaison et de décomposition temporelles et sémantiques. Il permet au travers des modèles de fission sémantique et d'allocation, de capturer les caractéristiques pertinentes des interfaces multimodales en sortie lors du processus de conception et d'exhiber les propriétés d'utilisabilité pour ces interfaces (espace CASE, propriétés CARE).

Formalisation B Évènementiel

La formalisation B Évènementiel que nous proposons constitue un plongement et une opérationnalisation du modèle de conception que nous proposons. Une démarche de modélisation incrémentale par raffinements successifs a été proposée. Elle permet de décrire les différentes représentations de l'interface modélisée lors de son processus de conception.
Les développements B Évènementiel modélisent les étapes de fission sémantique et d'allocation, par la formalisation des principes de : délivrance des informations, production des présentations, affectation des présentations, combinaison et décomposition temporelles et sémantiques. Ils sont développés à l'aide de règles d'interprétation appliquées au modèle conceptuel générique. Ils permettent d'une part, de construire par raffinements successifs l'interface multimodale en sortie. Et d'autre part, de procéder à la vérification de propriétés pertinentes telles que l'absence de collision ou le non blocage par l'exploitation du méca-

nisme de preuve de théorème offert par la méthode B Évènementiel.

Application

Des mécanismes d'instanciation ont également été proposés. Ils permettant de formaliser une interface concrète à l'aide des modèles B Évènementiel de fission sémantique et d'allocation développés. Ces mécanismes ont permis de formaliser des études de cas inspirées d'interfaces réelles. L'objectif de ces instanciations est double. D'une part, elles garantissent l'expressivité du modèle et justifient les schémas de combinaison et de décomposition temporelles et sémantiques que nous avons proposés. D'autre part, elles permettent de déduire un processus de formalisation B Évènementiel générique pour les interfaces multimodales en sortie. Ce processus généralise les modèles B Évènementiel de fission sémantique et d'allocation proposés.

Généralisation B Évènementiel

La généralisation du processus de développement B Évènementiel propose un cadre de formalisation dans lequel s'inscrivent les modèles B Évènementiel de fission sémantique et d'allocation proposés. Il formalise moyennant quatre modèles génériques, la conception de l'interface multimodale en sortie. Ces modèles constituent des modèles cadres à partir desquels les modèles B Évènementiel détaillés de la fission sémantique et de l'allocation sont dérivés par l'utilisation de mécanismes de raffinement et de spécialisation.

Nous avons également proposé une approche d'instanciation pour les modèles B Évènementiel génériques. Elle a permis, combinée aux mécanismes de raffinement et de spécialisation proposés, de construire des développements spécifiques à une interface concrète.

Par conséquent, la formalisation B Évènementiel d'une interface multimodale en sortie concrète peut s'opérer de deux manières différentes. Une première méthode spécifique consiste à combiner les composants B Évènementiel dédiés à chaque cas précis des étapes de fission sémantique et d'allocation. Une deuxième méthode générique consiste à raffiner et spécialiser les modèles B Évènementiel génériques. La confrontation de ces deux approches pour le développement de notre étude de cas, a montré que la méthode de développement spécifique demandait au concepteur de l'interface un effort supérieur de modélisation et un effort moindre d'instanciation. La méthode de développement générique demande quant à elle, un moindre effort de modélisation et un effort supérieur d'instanciation, elle induit un

plus grand nombre d'obligation de preuves en raison des raffinements introduits.

Perspectives

La réalisation des travaux présentés dans cette thèse permet d'envisager quatre perspectives principales.

Garantie de l'expressivité du modèle conceptuel

Comme première perspective, il nous paraît important de définir d'autres instanciations du modèle proposé, par la formalisation de nouvelles études de cas plus complexes afin de démontrer l'expressivité et la pertinence des schémas de combinaison temporelle et sémantique que nous avons définis, et de mesurer la complexité des développements B Évènementiel obtenus.

Garantie de la généricité du modèle conceptuel

Une deuxième perspective, serait de plonger le modèle conceptuel proposé dans une autre technique de spécification et de validation formelles afin d'en assurer la généricité. Une technique utilisant une validation par model checking serait privilégiée, elle permettrait d'envisager une utilisation coopérative des techniques de validation par preuve de théorème et par model checking pour vérifier toutes les propriétés pertinentes pour les interfaces multimodales en sortie.

Passage à l'échelle et production automatisée des interfaces multimodales

Une troisième perspective est relative à la production automatisée des interfaces multimodales. En effet, les développements B Évènementiel que nous proposons dans ces travaux de thèse sont accomplis manuellement. Ce qui implique que l'approche que nous proposons est difficilement envisageable dans le cadre du développement d'une interface multimodale réelle, bien plus complexe que les études de cas que nous avons modélisé. Cette complexité est atténuée par la généralisation du processus de développement B Évènementiel, cependant le développement d'un outil automatisant la conception des modèles B Évènementiel à partir de patrons de conception du modèle générique s'avère fort utile. Elle permettrait

la production automatique de spécifications B Évènementiel à partir de descriptions syntaxiques de l'interface.

Étude de la composition des modèles B Évènementiel

Enfin, une dernière perspective consisterait à étudier la composition des modèles B Évènementiel proposés. En effet, les modèles de développement B Évènementiel que nous proposons formalisent des interfaces obtenues à partir d'une seule information produite par le noyau fonctionnel. En réalité, une interface multimodale produit plusieurs présentations multimodales pour toutes les informations générées par le noyau fonctionnel de l'application qu'elle interface. Par conséquent, la spécification d'une système multimodal en sortie impose la composition des modèles qui formalisent les différentes présentations qu'il produit. L'étude de cette composition de modèles devra également statuer sur les propriétés vérifiées sur les modèles de présentation composés, afin de vérifier si elles sont préservées sur le modèle global du système multimodal.

Bibliographie

[1] Gaelle Calvary, Joelle Coutaz, and Laurence Nigay. From single-user architectural design to pac* : a generic software architecture model for cscw. In *In CHI '97*, pages 242–249 (1997). ix, 16, 17

[2] Mickaël Baron. *Vers une approche sûre du développement des Interfaces Homme-Machine*. Thèse de Doctorat, Thesis, LIP6 (2004). ix, 19, 20, 49

[3] J Caelen and J Coutaz. Interaction multimodale homme-machine : quelques problèmes généraux. , **91**, pages 41–57 (1991). ix, 18, 25, 26

[4] C. Rousseau. *Prï¿½sentation multimodale et contextuelle de l'information*. Thèse de Doctorat, universitï¿½ Paris sud XI-Orsay (2006). ix, 10, 28, 29

[5] Marie-Luce Bourguet. Outil de prototypage pour la conception et l'evaluation d'interfaces utilisateur multimodales. In *Proceedings of the 14th French-speaking conference on Human-computer interaction (Conférence Francophone sur l'Interaction Homme-Machine)*, pages 239–242. ACM (2002). ix, 47, 48

[6] Richard A. Bolt. *"put-that-there" : Voice and gesture at the graphics interface*. SIGGRAPH Comput. Graph. **14**(3), 262–270 July (1980). 5

[7] Gaëlle Calvary. *Ingénierie de l'interaction homme machine : rétrospective et perspectives*. Traité des Sciences et Techniques de l'Information-Interaction homme-machine et recherche d'information, C. Paganelli Ed., Hermès pages 19–63 (2002). 6

[8] Peter Wegner. *Why interaction is more powerful than algorithms*. Communications of the ACM **40**(5), 80–91 (1997). 7

[9] Y. Bellik. *Interfaces Multimodales : concepts, modï¿½les et architecture*. Thèse de Doctorat, LIMSI, Universitï¿½ d'Orsay (1995). 8, 27

[10] Laurence Nigay and Joï¿½lle Coutaz. *Espaces conceptuels pour l'interaction multimï¿½dia et multimodale*. Technique et Science Informatiques, spï¿½cial Multimï¿½dia et Collecticiel, Vol 15(9) pages 1195–1225 (1996). 8

[11] Len Bass, Ross Faneuf, Reed Little, Niels Mayer, Bob Pellegrino, Scott Reed, Robert Seacord, Sylvia Sheppard, and Martha R Szczur. *A metamodel for the runtime architecture of an interactive system*. SIGCHI Bulletin **24**(1), 32–37 (1992). 8

[12] James D Foley, Victor L Wallace, and Peggy Chan. The human factors of computer graphics interaction techniques. In *Human-computer interaction*, pages 67–121. Prentice Hall Press (1990). 8

[13] Laurence Nigay and Joëlle Coutaz. A design space for multimodal systems : concurrent processing and data fusion. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pages 172–178. ACM (1993). 8

[14] Yacine Bellik and Daniel Teil. Définitions terminologiques pour la communication multimodale. , **92**, pages 229–232 (1992). 9

[15] J Coutaz. Multimedia and multimodal user interfaces : a taxonomy for software engineering research issues. In *St Petersburg HCI Workshop* (1992). 9

[16] N. Kamel. *Un cadre formel gï¿½nï¿½ric pour la specification et la vï¿½rification des interfaces multimodales. Cas de la multimodalitï¿½ en entrï¿½e*. Thèse de Doctorat, Universitï¿½ de Poitiers, France (2006). 9, 27, 39, 46

[17] Dan Diaper and Neville Stanton. *The handbook of task analysis for human-computer interaction*. CRC Press (2003).

[18] Christophe Kolski. *Analyse et conception de l'ihm, interaction homme-machine pour les systèmes d'information, vol. 1*. Hermès, Paris, ISBN pages 2–7462 (2001).

[19] Y. Ait-Ameur, I. Ait-Sadoune, M. Baron, and J-M. Mota. *Vï¿½rification et validation formelles de systï¿½mes interactifs fondï¿½es sur la preuve : application aux systï¿½mes multi-modaux*. Journal d'Interaction Personne-Systï¿½me **1**(1), 1–30 September (2010). Article 3. 11, 25

[20] Stuart K. Card, Allen Newell, and Thomas P. Moran. *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA (1983). 12

[21] DL Scapin and C Pierret-Golbreich. Mad : Une méthode analytique de description des tâches. In *Colloque sur l'engeniérie des interfaces homme-machine, Sophia-Antipolis, France, INRIA* (1989). 12

[22] H. Rex Hartson, Antonio C. Siochi, and D. Hix. *The uan : A user-oriented representation for direct manipulation interface designs*. ACM Trans. Inf. Syst. **8**(3), 181–203 July (1990). 12, 13

[23] Gerrit C. Van Der Veer, Bert F. Lenting, and Bas A. J. Bergevoet. *Gta : Groupware task analysis - modeling complexity*. Acta Psychologica **91**, 297–322 (1996). 12

[24] Fabio Paternò. *Concurtasktrees : an engineered approach to model-based design of interactive systems*. The handbook of analysis for human-computer interaction pages 483–500 (2002). 12, 14

[25] Adel Mahfoudhi, Mourad Abed, and Dimitri Tabary. From the formal specifications of users tasks to the automatic generation of the hci specifications. In Ann Blandford, Jean Vanderdonckt, and Phil Gray, editors, *People and Computers XVâInteraction without Frontiers*, pages 331–347. Springer London (2001). 12

[26] Matthias Giese, Tomasz Mistrzyk, Andreas Pfau, Gerd Szwillus, and Michael von Detten. Amboss : A task modeling approach for safety-critical systems. In *Engineering Interactive Systems*, pages 98–109. Springer (2008). 12

[27] Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, and Víctor López-Jaquero. Usixml : a language supporting multi-path development of user interfaces. In *Engineering human computer interaction and interactive systems*, pages 200–220. Springer (2005). 12

[28] Quentin Limbourg, Costin Pribeanu, and Jean Vanderdonckt. Towards uniformed task models in a model-based approach. In *IN DSV-ISï¿½01, year = 2001, pages = 13–15, publisher = .* 12

[29] Phil Gray, David England, and Steve McGowan. Xuan : Enhancing uan to capture temporal relationships among actions. In *BCS HCI*, pages 301–312 (1994). 13

[30] Fabio Paternò, Giulio Mori, and Riccardo Galiberti. Ctte : An environment for analysis and development of task models of cooperative applications. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '01, pages 21–22, New York, NY, USA (2001). ACM. 14, 46

[31] G. E. Pfaff, editor. *User Interface Management Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA (1985). 15

[32] Len Bass, Reed Little, Robert Pellegrino, Scott Reed, Robert Seacord, Sylvia Sheppard, and Martha R Szezur. The arch model : Seeheim revisited. In *User Interface Developpers' Workshop* (1991). 15

[33] Glenn E. Krasner and Stephen T. Pope. *A cookbook for using the model-view controller user interface paradigm in smalltalk-80*. J. Object Oriented Program. **1**(3), 26–49 August (1988). 15

[34] Joëlle Coutaz. *Pac : An object oriented model for implementing user interfaces*. ACM SIGCHI Bulletin **19**(2), 37–41 (1987). 15

[35] Ralph D. Hill. The abstraction-link-view paradigm : Using constraints to connect user interfaces to applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, pages 335–342, New York, NY, USA (1992). ACM. 15

[36] Depaulis Fabrice, Jambon Francis, Girard Patrick, and Guittet Laurent. *Le modï¿½le d'architecture logicielle h4 : Principes, usages, outils et retours d'expï¿½rience dans les applications de conception technique*. Revue d'Interaction Homme-Machine **200**(1), 93–129 (2006). 15, 17

[37] Christophe Kolski, Peter Forbrig, Bertrand David, Patrick Girard, Chi Dung Tran, and Houcine Ezzedine. Agent-based architecture for interactive system design : Current approaches, perspectives and evaluation. In *Human-Computer Interaction. New Trends*, pages 624–633. Springer (2009). 15

[38] G Faconti and Fabio Paternò. An approach to the formal specification of the components of an interaction. , **90**, pages 481–494 (1990). 16

[39] David J. Duke and Michael D. Harrison. *Abstract interaction objects*. Computer Graphics Forum **12**(3), 25–36 (1993). 16

[40] Fabio Paternò. *A formal specification of appearance and behaviour of visual environments*. Software Engineering Journal **8**, 154–164(10) May (1993). 16

[41] Panos Markopoulos. *On the expression of interaction properties within an interactor model*. Springer (1995). 16

[42] Joëlle Coutaz. *Interfaces homme-ordinateur : conception et réalisation*. Dunod (1990). 17

[43] P Brun. *Xtl : a temporal logic for the formal development of interactive systems*. Formal Methods for Human-Computer Interaction pages 121–139 (1997). 18, 49

[44] Robert JK Jacob. *Using formal specifications in the design of a human-computer interface*. Communications of the ACM **26**(4), 259–264 (1983). 19

[45] Alexandre Cortier. *Contribution ï¿½ la validation formelle de systï¿½mes interactifs Java*. Thèse de Doctorat, Universitï¿½ Paul Sabatier - Institut Supï¿½rieur de l'Aï¿½ronautique et de l'Espace (ISAE-Supaero) (2008). Disponible sur http ://alexandre.cortier.free.fr. 19, 38

[46] William A Woods. Transition network grammars for natural language analysis. In *Readings in natural language processing*, pages 71–87. Morgan Kaufmann Publishers Inc. (1986). 19

[47] Anthony I Wasserman. User software engineering and the design of interactive systems. In *Proceedings of the 5th international conference on Software engineering*, pages 387–393. IEEE Press (1981). 19

[48] David Harel. *Statecharts : A visual formalism for complex systems*. Science of computer programming **8**(3), 231–274 (1987). 19

[49] Christophe Sibertin-Blanc. High level petri nets with data structure. In *6th european workshop on Application and Theory of Petri Nets*, pages 141–168 (1985). 20

[50] Rémi Bastide and Philippe A Palanque. Petri net objects for the design, validation and prototyping of user-driven interfaces. , **90**, pages 625–631 (1990). 20, 45

[51] Philippe Palanque. *Modélisation par Objets Coopératifs Interactifs d'interfaces homme-machine dirigées par l'utilisateur*. Thèse de Doctorat, ANRT Université Pierre Mendès France Grenoble 2 (1992). 20, 21

[52] Yamine Aït-Ameur, Patrick Girard, and Francis Jambon. A uniform approach for the specification and design of interactive systems : the b method. In *Eurographics Workshop on Design, Specification, and Verification of Interactive Systems (DSVIS'98), Vol. Proceedings (Eds, Markopoulos, P. and Johnson, P.), Abingdon, UK*, pages 333–352 (1998). 20, 48

[53] Yamine Aït-Ameur, Patrick Girard, and Francis Jambon. Using the b formal approach for incremental specification design of interactive systems. In *Engineering for Human-Computer Interaction*, pages 91–109. Springer (1999). 20, 48

[54] Jean-Claude Tarby. *Gestion automatique du dialogue homme-machine à partir de spécifications conceptuelles*. Thèse de Doctorat, Université des Sciences Sociales-Toulouse I (1993). 21

[55] Nicholas Halbwachs, Paul Caspi, Pascal Raymond, and Daniel Pilaud. *The synchronous data flow programming language lustre*. Proceedings of the IEEE **79**(9), 1305–1320 (1991). 21, 46

[56] Jean-Raymond Abrial. *Modeling in Event-B : System and Software Engineering*. Cambridge University Press, New York, NY, USA, 1st edition (2010). 21, 40, 84

[57] Alan Dix, Janet E. Finlay, Gregory D. Abowd, and Russell Beale. *Human-Computer Interaction (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1993). 21

[58] D.J. Duke and M.D. Harrison. *Event model of human-system interaction*. Software Engineering Journal **10**(1), 3–12 Jan (1995). 21, 48

[59] Christian Gram and Gilbert Cockton, editors. *Design Principles for Interactive Software*. Chapman & Hall, Ltd., London, UK, UK (1997). 21, 22

[60] Pierre Rochï¿½. *Modï¿½lisation et validation d'interface homme-machine*. Thèse de Doctorat, (1998). Thï¿½se de doctorat dirigï¿½e par Ausbourg, Bruno d'Informatique Toulouse, ENSAE 1998. 21, 46

[61] J. Coutaz and L. Nigay. Les propriï¿½tï¿½s care dans les interfaces multimodales. In *IHM'94*, Lille, France (1994). 24

[62] Joëlle Coutaz, Laurence Nigay, Daniel Salber, Ann Blandford, Jon May, and Richard M Young. Four easy pieces for assessing the usability of multimodal interaction : the care properties. , **95**, pages 115–120 (1995). 25

[63] Laurence Nigay and Joëlle Coutaz. *Multifeature systems : The care properties and their impact on software design*. Intelligence and multimodality in multimedia interfaces (1997). 25

[64] Jullien Bouchet, Laurence Nigay, and Thierry Ganille. Icare software components for rapidly developing multimodal interfaces. In *Proceedings of the 6th International Conference on Multimodal Interfaces*, ICMI '04, pages 251–258, New York, NY, USA (2004). ACM. 25, 27

[65] Nadjet Kamel. Utilisation de smv pour la vérification de propriétés d'ihm multimodales. In *IHM*, pages 219–222 (2004). 25

[66] Y. Bellik and D. Teil. Les types de multimodalitï¿½s. In *Les 4iï¿½mes journï¿½es sur l'ingï¿½nierie des interfaces Homme-Machine, IHM92*, pages 22–28, France (1992). Telecom Paris. 27

[67] Frans Flippo, Allen Krebs, and Ivan Marsic. A framework for rapid development of multimodal interfaces. In *Proceedings of the 5th International Conference on Multimodal Interfaces*, ICMI '03, pages 109–116, New York, NY, USA (2003). ACM. 27

[68] Laurence Nigay and Joëlle Coutaz. A generic platform for addressing the multimodal challenge. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 98–105, New York, NY, USA (1995). ACM Press/Addison-Wesley Publishing Co. 27

[69] Sharon Oviatt, Phil Cohen, Lizhong Wu, Lisbeth Duncan, Bernhard Suhm, Josh Bers, Thomas Holzman, Terry Winograd, James Landay, Jim Larson, et al. *Designing the user interface for multimodal speech and pen-based gesture applications : state-of-the-art systems and future research directions*. Human-computer interaction **15**(4), 263–322 (2000). 27

[70] Carlos Duarte and Luís Carriço. A conceptual framework for developing adaptive multimodal applications. In *Proceedings of the 11th international conference on Intelligent user interfaces*, pages 132–139. ACM (2006). 27

[71] James A Larson, TV Raman, Dave Raggett, Michael Bodell, Michael Johnston, Sunil Kumar, Stephen Potter, and Keith Waters. *W3c multimodal interaction framework*. W3C NOTE **6** (2003). 27

[72] Philip R. Cohen, Michael Johnston, David McGee, Sharon Oviatt, Jay Pittman, Ira Smith, Liang Chen, and Josh Clow. Quickset : Multimodal interaction for distributed applications. In *Proceedings of the Fifth ACM International Conference on Multimedia*, MULTIMEDIA '97, pages 31–40, New York, NY, USA (1997). ACM. 27

[73] Kisub Song and Kyong-Ho Lee. *Generating multimodal user interfaces for web services*. Interact. Comput. **20**(4-5), 480–490 September (2008). 27

[74] Tracy Westeyn, Helene Brashear, Amin Atrash, and Thad Starner. Georgia tech gesture toolkit : Supporting experiments in gesture recognition. In *Proceedings of the 5th International Conference on Multimodal Interfaces*, ICMI '03, pages 85–92, New York, NY, USA (2003). ACM. 27

[75] James Glass, Eugene Weinstein, Scott Cyphers, Joseph Polifroni, Grace Chung, and Mikio Nakano. A framework for developing conversational user interfaces. In RobertJ.K. Jacob, Quentin Limbourg, and Jean Vanderdonckt, editors, *Computer-Aided Design of User Interfaces IV*, pages 349–360. Springer Netherlands (2005). 27

[76] N. Krahnstoever, S. Kettebekov, M. Yeasin, and R. Sharma. A real-time framework for natural multimodal interaction with large screen displays. In *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, ICMI '02, pages 349–, Washington, DC, USA (2002). IEEE Computer Society. 27

[77] Kenneth W. K. Lo, Will W. W. Tang, Grace Ngai, Alvin T. S. Chan, Hong Va Leong, and Stephen C. F. Chan. I*chameleon : A platform for developing multimodal application with comprehensive development cycle. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 1103–1108, New York, NY, USA (2013). ACM. 27

[78] Pierre Dragicevic and Jean-Daniel Fekete. Input device selection and interaction configuration with icon. In *People and Computers XVï¿½Interaction without Frontiers*, pages 543–558. Springer (2001). 27

[79] Marilyn Rose McGee-Lennon, Andrew Ramsay, David McGookin, and Philip Gray. User evaluation of oide : A rapid prototyping platform for multimodal interaction. In *Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '09, pages 237–242, New York, NY, USA (2009). ACM. 27

[80] M. Bordegoni, G. Faconti, M.T. Maybury, T. Rist, P. Trahanias S. Ruggieri, and M. Wilson. *A standard reference model for intelligent multimedia presentation systems*. Computer Standards and Interfaces **6**(4), 477–496 (1997). 28, 55

[81] P. Merloz, S. Lavallee, J. Tonnetti, and L. Pittet. *Image-guided spinal surgery : Technology, operative technique and clinical practice*. Operative techniques in Orthopaedics **10**(1), 56–63 January (2000). 31

[82] N. Bernsen and L. Dybkjaer. Exploring natural interaction in the car. In *International Workshop on Information Presentation and Natural Multimodal Dialogue*, pages 75–79, Verona, Italy (2001). 31

[83] J. Juster and D. Roy. Elvis. Situated speech and gesture understanding for a robotic chandelier. In *Sixth International Conference on Multimodal Interfaces*, pages 90–96. ACM Press (2004). 31

[84] L. P. Nedel, C. Freitas, L. Jacob, and M. Pimenta. Testing the use of egocentric interactive techniques in immersive virtual environments. In *INTERACT 2003 - Ninth IFIP TC13 International Conference on Human-Computer*, pages 471–478. IOS Press (2003). 31

[85] Laurence Nigay. *Conception et Modï¿½lisation Logicielle des Systï¿½mes Interactifs : Application aux Interfaces Multimodales*. Doctorat d'universitï¿½ (phd thesis) Universitï¿½ Joseph Fourier (1994). 31, 48

[86] Steven K. Feiner and Kathleen R. McKeown. *Automating the generation of coordinated multimedia explanations*. Computer **24**(10), 33–41 (1991). 31

[87] Oliviero Stock. *Alfresco : Enjoying the combination of natural language processing and hypermedia for information exploration*. pages 197–224 (1993). 31

[88] M. Fasciano and G. Lapalme. Postgraphe : a system for the generation of statistical graphics and text. In *Workshop on Natural Language Generation*, pages 51–60, Sussex, UK (1996). 32

[89] M. Dalal, S. Feiner, K. McKeown, S. Pan, M. Zhou, T. Hï¿½llerer, J. Shaw, Y. Feng, and J. Fromer. Negotiation for automated generation of temporal multimedia presentations. In *ACM Multimedia'96*, pages 55–64, Boston, USA November (1996). 32

[90] N. Reithinger, J. Alexandersson, T. Becker, A. Blocher, R. Engel, M. Lï¿½ckelt, J. Mï¿½ller, N. Pfleger, P. Poller, M. Streit, and V. Tschernomas. Smartkom : Adaptive and flexible multimodal access to multiple applications. In *ICMI'03*, pages 101–108, Vancouver, British Columbia, Canada November (2003). 32

[91] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, Inc., Orlando, FL, USA, 1st edition (1997). 38

[92] Edmund M Clarke, Orna Grumberg, and Doron Peled. *Model checking*. MIT press (1999). 38

[93] Cem Kaner, Jack L. Falk, and Hung Quoc Nguyen. *Testing Computer Software, Second Edition*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition (1999). 38

[94] Jean-Raymond Abrial and Stefan Hallerstede. *Refinement, decomposition, and instantiation of discrete models : Application to event-b*. Fundam. Inf. **77**(1-2), 1–28 January (2007). 38

[95] Jonathan P Bowen and Michael G Hinchey. *Applications of Formal Methods*. Prentice Hall PTR (1995). 39

[96] Marie-Claude Gaudel. *Précis de génie logiciel*. Masson (1996). 39

[97] J-F Monin. *Comprendre les mï¿½thodes formelles*. Masson (1996). 39, 40

[98] Jeannette Wing. *What is a formal method ?* Carnegie Mellon University (1989). 39, 41

[99] J.M Spivey. *The Z notation : A Reference Manual*. Prentice Hall Int (1988). 39

[100] C. B. Jones. *VDM : une mï¿½thode rigoureuse pour le dï¿½veloppement de logiciels*. Masson (1992). 39

[101] J-R Abrial. *The B-Book*. Cambridge University Press (1996). 39, 79

[102] G. Huet, G. Kahn, and C. Paulin-Mohring. The coq proof assistant : A tutorial. Technical report logical project (2007). 39

[103] M. J. C Gordon and T. F. Melham. *Introduction to HOL, A theorem proving environment for higher order logic*. Cambridge University Press (1993). 39

[104] S. Owre, J. M. Rushby, , and N. Shankar. PVS : A prototype verification system. , **607**, pages 748–752, Saratoga, NY jun (1992). Springer-Verlag. 39

[105] L. C Paulson. The isabelle reference manual. Technical report University of Cambridge, Computer Laboratory (1993). 39

[106] Annie Choquet-Geniet. *Les rï¿½seaux de Petri Un outil de modï¿½lisation*. Dunod (2006). 40

[107] Charles Antony Richard Hoare. , **178**. Prentice-hall Englewood Cliffs (1985). 40, 48

[108] J. Ellsberger, D. Hogrefe, and A. Sarma. *SDL : formal object-oriented language for communicating system*. Prentice Hall PTR (1997). 40

[109] Estelle. : Iso 9074, a formal description technique based on an extended state transition model. Technical report (1997). 40

[110] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press (1999). 41

[111] Hervï¿½ Lehning. *Les ï¿½quations algï¿½briques*. Pï¿½le, Collectif Tangente (2005). 41

[112] Tadao Kasami, Kenichi Taniguchi, Yuji Sugiyama, and Hiroyuki Seki. *Principles of algebraic language asl*. Systems and Computers in Japan **18**(7), 11–20 (1987). 41

[113] Jan De Meer, Rudolf Roth, and Son Vuong. *Introduction to algebraic specifications based on the language act one*. Computer Networks and ISDN Systems **23**(5), 363–392 (1992). 41

[114] Joseph Goguen, Grant Malcolm (eds.), Edited Joseph Goguen, and Grant Malcolm. *Software engineering with obj : algebraic specification in practice*. Jouannaud, Introducing OBJ, in : J. Goguen, G. Malcolm (Eds.), Software (1992). 41

[115] D. Bert, R. Echahed, and J-C. Reynaud. Reference manual of the lpg specification language and environment. Technical report (2000). 41

[116] Peter H. J. Van Eijk, Chris A Vissers, and Michel Diaz. *Formal Description Technique Lotos : Results of the Esprit Sedos Project*. Elsevier Science Inc., New York, NY, USA (1989). 41, 44

[117] R. Milner. **92**, New York, NY, USA (1980). LNCS, Springer-Verlag. 41

[118] C. A. R. Hoare. *An axiomatic basis for computer programming*. **12**(10), 576–580 (1969). 41, 79

[119] Edmund Clarke and Xudong Zhao. Analytica - a theorem prover for mathematica. In *The Mathematica Journal*, pages 761–765. Springer-Verlag (1993). 42

[120] Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. Nusmv 2 : An opensource tool for symbolic model checking. , **2404**, pages 359–364. Springer Berlin Heidelberg (2002). 42

[121] Gerard J Holzmann. , **1003**. Addison-Wesley Reading (2004). 42

[122] Edmund M. Clarke, Orna Grumberg, and David E. Long. *Model checking and abstraction*. ACM Trans. Program. Lang. Syst. **16**(5), 1512–1542 (1994). 43

[123] Fabio Domenico Paterno. *A method for formal specification and verification of interactive systems.* Thèse de Doctorat, University of York (1995). 44

[124] Fabio Paternò and G Faconti. *On the use of lotos to describe graphical interaction*. People and computers pages 155–155 (1992). 44

[125] Panos Markopoulos, Jon Rowson, and Peter Johnson. Dialogue modelling in the framework of an interactor model. In *Pre-conference Proceedings. Design Specification and Verification of Interactive Systems. Namur, Belgium* (1996). 44

[126] Panagiotis Markopoulos. *A compositional model for the formal specification of user interface software*. Thèse de Doctorat, Citeseer (1997). 44

[127] Muffy Calder and Carron Shankland. A symbolic semantics and bisimulation for full lotos. In *Formal Techniques for Networked and Distributed Systems*, pages 185–200. Springer (2002). 44

[128] Hartmut Ehrig, Werner Fey, and Horst Hansen. *ACT ONE : an algebraic specification language with two levels of semantics*. Techn. Univ., Fachbereich Informatik (1983). 44

[129] Peter van Eijk. Tool demonstration : The lotosphere integrated tool environment lite. In *Proceedings of the IFIP TC6/WG6.1 Fourth International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols : Formal Description Techniques, IV*, FORTE '91, pages 471–474, Amsterdam, The Netherlands, The Netherlands (1992). North-Holland Publishing Co. 44

[130] Jean-Claude Fernandez, Hubert Garavel, Laurent Mounier, Anne Rasse, Carlos Rodriguez, and Joseph Sifakis. A toolbox for the verification of lotos programs. In *Proceedings of the 14th international conference on Software engineering*, pages 246–259. ACM (1992). 44

[131] Joelle Coutaz, Fabio Paterno, Giorgio Faconti, and Laurence Nigay. A comparison of approaches for specifying multimodal interactive systems. In *Proceedings of the ERCIM Workshop on Multimodal Human-Computer Interaction*, pages 165–174 (1993). 45

[132] Fabio Paterno and Menica Mezzanotte. *Analysing matis by interactors and actl*. Amodeus Project Document : System Modelling/WP36 (1994). 45

[133] Philippe A Palanque, Rémi Bastide, and Valérie Sengès. Validating interactive system design through the verification of formal task and system models. In *EHCI*, pages 189–212. Citeseer (1995). 45

[134] A. Schyn, D. Navarre, P. Palanque, and L. P. Nedel. Description formelle d'une technique d'interaction multimodale dans une application de rï¿½alitï¿½ virtuelle immersive. In *Proceedings of the 15th French Speaking conference on human-computer interaction (IHM'2003)*, pages 25–28, Caen, France November (2003). 45, 46

[135] Ousmane Sy, Rémi Bastide, Philippe Palanque, D Le, and David Navarre. *Petshop : a case tool for the petri net based specification and prototyping of corba systems*. Petri Nets 2000 page 78 (2000). 45

[136] P. Palanque and A. Schyn. A Model-based for Engineering Multimodal Interactive Systems. In *9th IFIP TC13 International Conference on Human Computer Interaction (Interact'2003)* (2003). 46

[137] D. Navarre, P. Palanque, R. Bastide, A. Schyn, M. Winckler, L.P. Nedel, and C.M.D.S. Freitas. A formal description of multimodal interaction techniques for immersive virtual reality applications. In *INTERACT 2005*, pages 25–28, Roma, Italy September (2005). Lecture Notes in Computer Science, Springer Verlag. 46

[138] Amï¿½lie Schyn, , and Philippe Directeur de thï¿½se Palanque. *Une approche fondï¿½e sur les modï¿½les pour l'ingï¿½nierie des systï¿½mes interactifs multimodaux*. Thèse de Doctorat, Toulouse 3, s.l (2005). Thï¿½se de doctorat : Informatique. 46

[139] Bruno dï¿½Ausbourg. *Using model checking for the automatic validation of user interfaces systems*. Springer (1998). 46

[140] Paula M. Ferguson and David Brennan. *Motif Reference Manual B : For OSF/Motif Release 1.2*. Thomson Learning, 1st edition (1993). 46

[141] Ch Ratel. *Lesar : un outil pour la vï¿½rification d'invariants de programmes lustre*. Thï¿½se, Universitï¿½ Joseph Fourier, Grenoble, France (1992). 46

[142] F. Jourde, L. Nigay, and I. Parissis. Test formel de systï¿½mes interactifs multimodaux : couplage ICARE - Lutess. In *ICSSEA'2006, 19ï¿½me journï¿½es Internationales "gï¿½nie logiciel & Ingï¿½nierie de Systï¿½mes et leurs Applications" Globalisation des services et des systï¿½mes*, Paris, France (2006). 47

[143] J. Bouchet, L. Madani, L. Nigay, C. Oriat, and I. Parissis. Formal testing of multimodal interactive systems. In *DSV-IS2006, the XIII International Workshop on Design, Specification and Verification of interactive systems*. Lecture Notes in Computer Science, Springer-Verlag (2006). 47

[144] L. Du Bousquet, F. Ouabdesselam, J.-L. Richier, and N. Zuanon. Lutess : a specification driven testing environment for synchronous software. In *International Conference of software engineering*, pages 267–276. ACM Press (1999). 47

[145] J. Bouchet, L. Nigay, and T. Ganille. The icare component-based approach for multimodal input interaction : application to real-time military aircraft cockpits. In *The 11th International Conference on Human-Computer Interaction*, Las Vegas, Nevada, USA (2005). Lawrence Erlbaum Associates. 47

[146] Marie-Luce Bourguet. Designing and prototyping multimodal commands. , **3**, pages 717–720. Citeseer (2003). 47

[147] I. MacColl and D. Carrington. Testing MATIS : a case study on specification based testing of interactive systems. In *Formal Aspects of HCI (FAHCI98)*, pages 57–69, Sheffield-Hallam University (1998). 48

[148] Yamine Ait-Ameur. Cooperation of formal methods in an engineering based software development process. In *Proceedings of the Second International Conference on Integrated Formal Methods*, IFM '00, pages 136–155, London, UK, UK (2000). Springer-Verlag. 48

[149] Yamine Aït-Ameur, Mickaël Baron, and Nadjet Kamel. Utilisation de techniques formelles dans la modélisation d'interfaces homme-machine. une expérience comparative entre b et promela/spin. In *6th International Symposium on Programming and Systems ISPS*, pages 57–66 (2003). 48

[150] Francis Jambon. From formal specifications to secure implementations. In *Computer-Aided Design of User Interfaces III*, pages 51–62. Springer (2002). 48

[151] Y. Ait-Ameur, I. Ait-Sadoune, M. Baron, and JM. Mota. Validation et vï¿½rification formelles de systï¿½mes interactifs multi-modaux fondï¿½es sur la preuve. In *18ï¿½ Confi¿½rence Francophone sur l'Interaction Homme-Machine (IHM)*, pages 123–130, Montrï¿½al (2006). 49

[152] Y. Ait-Ameur, I. Ait-Sadoune, and M. Baron. Etude et comparaison de scï¿½narios de dï¿½veloppements formels d'interfaces multi-modales fondï¿½s sur la preuve et le raffinement. In *MOSIM 2006 - 6ï¿½me Confi¿½rence Francophone de Modï¿½lisation et Simulation. Modï¿½lisation, Optimisation et Simulation des Systï¿½mes : Dï¿½fis et Opportunitï¿½s*, Rabat (2006). 49

[153] Gregory D Abowd, Hung-Ming Wang, and Andrew F Monk. A formal technique for automated dialogue development. In *Proceedings of the 1st conference on Designing interactive systems : processes, practices, methods, & techniques*, pages 219–226. ACM (1995). 49

[154] Dan R Olsen, Andrew F Monk, and Martin B Curry. *Algorithms for automatic dialogue analysis using propositional production systems*. Human-Computer Interaction **10**(1), 39–78 (1995). 49

[155] José Campos. Automated deduction and usability reasoning. Phd thesis Department of Computer Science, University of York, Braga, Portugal (1999). 49

[156] Mark Ryan, José Fiadeiro, and Tom Maibaum. Sharing actions and attributes in modal action logic. In *Theoretical Aspects of Computer Software*, pages 569–593. Springer (1991). 49

[157] N. Kamel. Utilisation de SMV pour la vï¿½rification de propriï¿½tï¿½s d'IHM multimodales. In *16ï¿½ Confi¿½rence Francophone sur l'Interaction Homme-Machine (IHM'2004)*, pages 219–222, Namur, Belgique (2004). ACM Press. 49

[158] Cyril Rousseau, Yacine Bellik, and Frï¿½dï¿½ric Vernier. WWHT : Un modï¿½le conceptuel pour la prï¿½sentation multimodale d'information. In *IHM2005*, pages 59–66. ACM (2005). 55

[159] Wolfgang Wahlster. *SmartKom : Foundations of Multimodal Dialogue Systems (Cognitive Technologies)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006). 57

[160] James F. Allen. *Maintaining knowledge about temporal intervals*. Commun. ACM **26**(11), 832–843 November (1983). 60

[161] Frédéric Vernier and Laurence Nigay. Espace de conception pour les interfaces multimodales. In *Colloque sur la multimodalité, Grenoble* (2000). 60

[162] Jean-Raymond Abrial. *Modeling in Event-B : System and Software Engineering*. Cambridge University Press (2010). 79

[163] Edsger-Wybe Dijkstra. *A Discipline of Programming*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1 edition (1977). 79

[164] Jean-Raymond Abrial and Stefan Hallerstede. *Refinement, Decomposition, and Instantiation of Discrete Models : Application to Event-B*. Fundamenta Informaticae **77**, 1–28 (2007). 80, 85

[165] Rodin. *User Manual of the RODIN Platform*, October (2007). http ://deploy-eprints.ecs.soton.ac.uk/11/1/manual-2.3.pdf. 80

[166] Idir Aït-Sadoune. *Modélisation et vérification formelles de compositions de services : Une approche fondée sur le raffinement et la preuve*. Thèse de Doctorat, (2010). 80, 85, 87

[167] Yamine Ait-Ameur, Mickael Baron, Nadjet Kamel, and Jean-Marc Mota. *Encoding a process algebra using the Event B method*. International Journal on Software Tools for Technology Transfer (STTT) **11(Number 3)**, 239–253 (2009). 89, 94, 115

Les interfaces homme-machine (IHM) multimodales offrent à l'utilisateur la possibilité de combiner les modalités d'interaction afin d'augmenter la robustesse et l'utilisabilité de l'interface utilisateur d'un système. Plus particulièrement, en sortie, les IHM multimodales permettent au système de restituer à l'utilisateur, l'information produite par le noyau fonctionnel en combinant sémantiquement plusieurs modalités. Dans l'optique de concevoir de telles interfaces pour des systèmes critiques, nous avons proposé un modèle formel de conception des interfaces multimodales en sortie. Le modèle proposé se décompose en deux modèles : le modèle de fission sémantique qui décrit la décomposition de l'information à restituer en informations élémentaires, et le modèle d'allocation qui spécifie l'allocation des modalités et médias aux informations élémentaires. Nous avons également développé une formalisation B Événementiel détaillée des deux modèles : fission sémantique et allocation. Cette formalisation a été instanciée sur des études de cas puis généralisée dans un processus de développement B Événementiel cadre dans lequel s'inscrivent les modèles de fission sémantique et d'allocation. Cette formalisation a permis de procéder à la vérification de propriétés de sûreté, de vivacité et d'utilisabilité.

Mots clés : Interfaces Homme Machine multimodales, multimodalité en sortie, conception et vérification formelles, B Événementiel, raffinement, preuve de théorème.

Multimodal Human-Computer Interfaces (HCI) offer to users the possibility to combine interaction modalities in order to increase user interface robustness and usability. Specifically, output multimodal HCI allow system to return to the user, the information generated by the functional core by combining semantically different modalities. In order to design such interfaces for critical systems, we proposed a formal model for the design of output multimodal interfaces. The proposed model consists of two models : the semantic fission model describes the decomposition of the information to return into elementary information and the allocation model specifies the allocation of the elementary information with modalities and media. We have also developed a detailed Event B formalization for the two models : semantic fission and allocation. This formalization has been instantiated on case studies and generalized in an Event B development process framework including semantic fission and allocation models. This formalization allows to carry out safety, liveness and usability properties verification.

Keywords : Multimodal Human Computer Interfaces, output multimodality, formal design and verification, Event B, refinement, theorem proving.