Interaction multimodale en entrée : Conception et Prototypage

Marcos Serrano

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE
(ARRÊTÉ MINISTÉRIEL DU 7 AOUT 2006)

Spécialité: informatique

Présentée et soutenue publiquement par

Marcos Serrano

le 30 Juin 2010

Interaction multimodale en entrée : Conception et Prototypage

Thèse dirigée par Laurence NIGAY

JURY

M. CAPLIER Alice	Président
M. MARTIN Jean-Claude	Rapporteur
M. PALANQUE Philippe	Rapporteur
M. LALANNE Denis	Examinateur
M. LECOLINET Eric	Examinateur
M. NIGAY Laurence	Examinateur

Thèse préparée au sein du Laboratoire d'Informatique de Grenoble (LIG)
385, rue de la Bibliothèque, BP 53, 38041 GRENOBLE CEDEX 9
Université de Grenoble

REMERCIEMENTS

Mes premiers remerciements vont à mes parents, qui m'ont toujours aidé à progresser, encouragé à découvrir et à explorer de nouveaux horizons. Sans eux je ne serais pas arrivé là. Gracias.

Je remercie les personnes qui ont participé à ce travail et tout d'abord Laurence Nigay qui m'a ouvert les portes de ce domaine de recherche. Je la remercie pour la liberté et la confiance qu'elle m'a accordées et qui m'ont tant permis de grandir.

Je tiens à remercier Jean-Claude Martin et Philippe Palanque pour avoir accepté de rapporter cette thèse et pour leurs remarques pertinentes. Je voudrais aussi remercier Alice Caplier pour avoir présidé mon Jury de thèse ainsi que Denis Lalanne et Eric Lecolinet pour avoir accepté de faire partie de celui-ci.

Je remercie les collègues du projet OpenInterface avec qui j'ai eu le plaisir de travailler. En particulier, je remercie Michael Ortega et David Juras, collègues de travail et de voyage (grut!).

Je remercie également tous les membres de l'équipe IIHM, pour les nombreux conseils, retours et expériences partagées. Je n'oublierais pas les nombreuses discussions, les jours de grimpe et les sorties en montagne. Merci aussi à Nadine Mandran pour l'aide apportée à l'évaluation de mon travail.

Aux collègues de l'ENSAD et à Samuel Bianchini, avec qui j'ai découvert une autre perspective de l'interactivité.

Merci finalement à Anne-Cécile, pour tout.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

«En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. »

Don Quijote de la Mancha (Miguel de Cervantes, 1605)

Contenu du chapitre

1.1 Constats et besoins

La progression des technologies informatiques dans les 30 dernières années a été fulgurante. Aujourd'hui, les besoins en termes de puissance de calcul et de capacité de stockage sont comblés. L'enjeu majeur dans l'utilisation de l'ordinateur ne réside plus dans ses capacités internes, mais dans l'interaction entre le système et l'utilisateur.

L'Interaction Homme-Machine (IHM) est le domaine de recherche visant à étudier et concevoir cette interaction en s'appuyant sur une mise en relation des connaissances de l'ingénierie en informatique (côté Machine) et de la psychologie ou de l'ergonomie (côté Homme) entre autres. Les travaux présentés dans ce mémoire s'inscrivent dans le domaine de l'IHM et résultent de l'observation d'une série de constats sur l'état actuel des connaissances en IHM.

Constats

L'interaction avec les ordinateurs n'a quasiment pas évolué pendant les 30 dernières années, reposant sur le triplet clavier-souris-écran (interface WIMP). Néanmoins, nous assistons aujourd'hui à une prolifération de nouvelles façons d'inter-

agir avec un ordinateur : en effet l'apparition de nouveaux capteurs et leur miniaturisation a permis d'étendre les capacités d'interaction avec les ordinateurs. Le triplet clavier-souris-écran traditionnel a par exemple été remplacé par le triplet "clavier virtuel"-"souris multitouch"-"écran tactile" dans la suite de machines iPhone, MacBook et iPad de Apple [Apple, 2010].

Pour souligner la rapidité de ce phénomène de multiplication et d'innovation en termes de moyens d'interagir avec une machine, nous considérons les jeux vidéo. En trois ans (le temps d'une thèse), nous sommes passés de l'utilisation des manettes filaires sur la Playstation, à l'utilisation de gestes captés grâce à un dispositif sans fils sur la Wii, en attendant d'interagir avec les gestes du corps sur la future Xbox Natal (Figure 1.1).

FIGURE 1.1 – Évolution des interfaces des consoles de jeux vidéo de 2006 à 2009.

Evidemment, la prolifération des moyens d'interaction n'est pas uniquement positive. En effet, la multiplicité des possibilités d'interaction peut se voir comme un progrès en terme de souplesse et d'utilisabilité mais s'accompagne aussi d'une complexité accrue que ce soit en conception ou en développement. En faisant l'analogie entre moyens d'interaction et langues, nous illustrons de façon métaphorique cette complexité en considérant le mythe de la Tour de Babel. Comme représenté de nombreuses fois dans l'histoire de l'art (Figure 1.2), selon le mythe de la Tour de Babel, les hommes tentent de construire une tour allant jusqu'au ciel. Afin de les punir pour leur orgueil, Dieu multiplie les langues afin que les hommes ne se comprennent plus. Ne pouvant plus se parler entre eux, les hommes cessent la construction.

Ainsi, la multiplication des moyens d'interaction soulève des nouveaux problèmes et besoins.

Besoins

L'apparition rapide de nouveaux moyens d'interaction soulève des problèmes de conception et de développement. Ainsi, les modèles d'interaction tels que la ma-

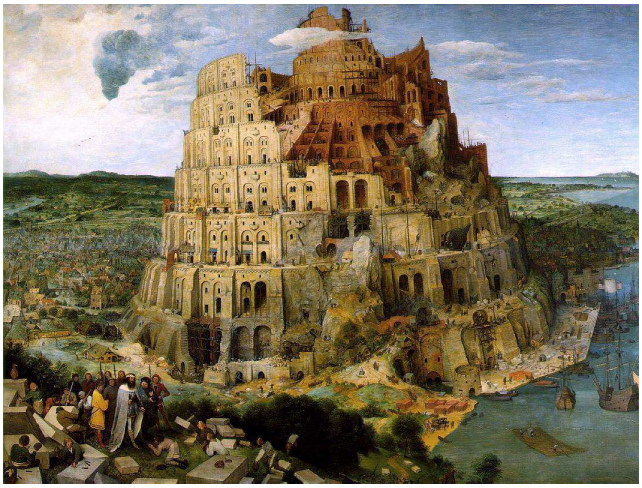


FIGURE 1.2 – La Tour de Babel vue par Pieter Bruegel l'Ancien au XVIe siècle.

nipulation directe [Shneiderman, 1987], permettant de décrire des interfaces graphiques de type WIMP, atteignent leur limite. De plus, les outils de développement comme les boîtes à outils, basés sur ces modèles n'intègrent pas les nouveaux moyens d'interaction disponibles aujourd'hui.

Ainsi, nous avons besoin de nouveaux modèles et outils qui permettent de concevoir et développer des interfaces qui utilisent et intègrent les nombreux moyens d'interaction disponibles. L'objectif de notre travail en IHM consiste ainsi à essayer que la construction d'interfaces ne devienne pas une vraie *Tour de Babel*.

1.2 Portée des travaux

Nos travaux ont trait à la conception de systèmes interactifs. En particulier nous nous intéressons aux **systèmes multimodaux**, systèmes informatiques mettant en jeu plusieurs modalités d'interaction. Afin de mieux appréhender cette introduction à nos travaux, nous donnons une première définition des termes *modalité* et *multimodalité*. Une définition plus complète sera donnée dans le chapitre 2 du mémoire.

Considérons un système interactif composé d'un noyau fonctionnel et d'une interface entre ce noyau et l'utilisateur. Une modalité d'interaction représente alors un médiateur matériel et logiciel entre l'utilisateur et le noyau fonctionnel. Nous distinguons alors les modalités en entrée, de l'utilisateur vers le système, des modalités en sortie, du système vers l'utilisateur (Figure 1.3).

A partir de cette définition, nous pouvons décrire la multimodalité comme la multiplicité des modalités d'interaction, en entrée ou en sortie, dans un système interactif.

Dans le contexte de systèmes multimodaux ainsi définis, nos travaux visent à définir des méthodes et des outils pour la conception de tels systèmes interactifs. En particulier, nous visons le **prototypage d'interfaces multimodales en entrée**, une étape importante dans la conception de systèmes interactifs.

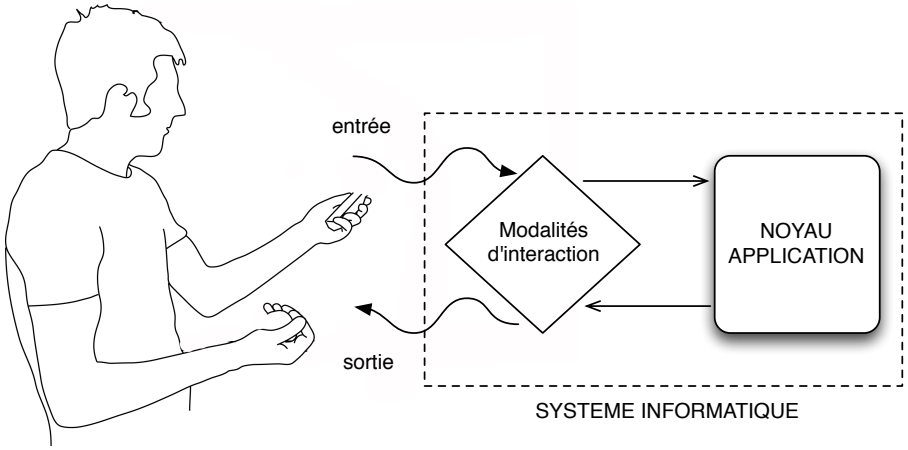


FIGURE 1.3 – Schéma simplifié d'un système interactif.

Un prototype, selon la définition du CNRTL[1], désigne un premier modèle réel d'un objet ou d'une machine, établi afin de le mettre au point avant d'entreprendre la fabrication en série. En interaction homme-machine, un prototype est un premièr modèle d'un système interactif [Greenberg et Buxton, 2008].

Certains cycles de vie du logiciel, comme IntuiSign [Schlienger et Chatty, 2007], explicitent cette étape de prototypage. Dans nos travaux, nous situons le prototypage d'interfaces multimodales dans le cycle de vie de la Figure 1.4 qui allie un cycle en spirale avec un cycle en V plus classique.

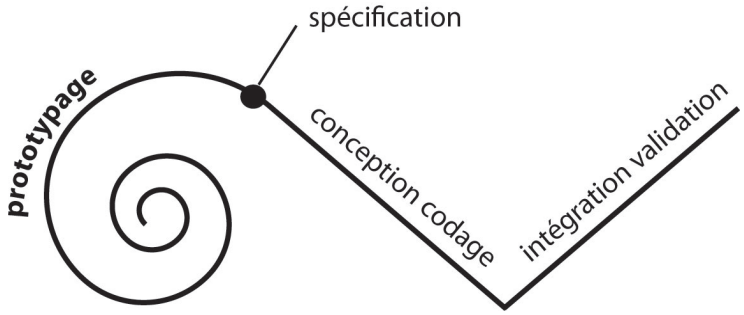


FIGURE 1.4 – Cycle de vie en spirale puis en v.

1.3 Contributions et démarche

Dans ce mémoire, nous présentons une approche à composants pour la conception et le prototypage fonctionnel ou non fonctionnel d'interfaces multimodales en entrée. Cette approche est basée sur un modèle de la multimodalité. Nos contributions sont conceptuelles et logicielles.

[1]CNRTL : Centre National de Ressources Textuelles et Lexicales [CNRTL-CNRS, 2009]

1.3.1 Modèle conceptuel

Notre contribution conceptuelle consiste en un espace de caractérisation de composants logiciels pour la multimodalité. Cet espace permet de caractériser les composants logiciels au sein d'un assemblage de composants visant à décrire une interaction multimodale. Il offre au concepteur un cadre conceptuel afin de lui permettre de décrire précisément des alternatives de conception.

1.3.2 Outil logiciel

Notre contribution logicielle consiste en un outil logiciel de prototypage basé sur une approche à composants qui implémente notre espace de caractérisation (notre contribution conceptuelle). Avec cet outil, nous avons implémenté plusieurs prototypes multimodaux, qu'ils soient fonctionnels ou en partie simulés par un compère humain selon une approche Magicien d'Oz.

1.3.3 Démarche

Nos contributions à la fois conceptuelles et logicielles sont le résultat de la démarche scientifique suivante :

- L'identification de caractéristiques d'une modalité d'interaction et de la multimodalité motivées par les sciences cognitives et le génie logiciel,
- l'organisation des caractéristiques en un modèle de conception basé composants logiciels permettant d'expliciter la nature de l'interaction multimodale, et de raisonner sur des choix de conception,
- l'implémentation de notre modèle de conception au sein d'un outil logiciel pour le prototypage.

Aussi la structure en chapitres de ce mémoire reflète notre démarche de recherche. En particulier, la structure du mémoire explicite les deux facettes, conceptuelle et logicielle, de nos travaux centrés sur la conception d'interfaces multimodales en entrée.

1.4 Organisation du mémoire

Ce mémoire est composé de deux parties précédées de **deux chapitres introductifs**. Après cette introduction (Chapitre 1), le Chapitre 2 a pour objectif d'introduire l'interaction multimodale. Nous situons l'interaction multimodale à travers l'histoire de l'IHM. Ensuite nous figeons la terminologie utilisée dans ce mémoire. Nous illustrons ces concepts à travers différents exemples d'applications multimodales qui couvrent le spectre des champs d'application de ce type d'interfaces. Finalement, nous introduisons l'espace problème que nous traitons dans nos travaux.

La première partie est dédiée à la conception. Elle contient trois chapitres.

- Le Chapitre 3 fait le point sur les modèles de conception de l'interaction multimodale, les modèles de conception logicielle pour l'interaction et enfin les modèles de conception logicielle spécifiques à l'interaction multimodale.

- Le Chapitre 4 présente notre contribution à la conception de l'interaction multimodale, un modèle conceptuel pour le prototypage d'interfaces multimodales.
- Le Chapitre 5 présente l'évaluation analytique et empirique de notre modèle.

La deuxième partie est consacrée à la réalisation logicielle de prototypes multimodaux. Elle est composée de trois chapitres.

- Le Chapitre 6 présente l'état de l'art des outils pour le prototypage d'interfaces multimodales.
- Le Chapitre 7 introduit l'outil de prototypage que nous avons développé et qui concrétise au sein d'un outil logiciel notre modèle de conception du chapitre4.
- Le Chapitre 8 présente l'évaluation empirique de l'outil ainsi que la validation expérimentale de l'outil par la réalisation de prototypes multimodaux.

La Figure 1.5 reprend la structure du mémoire.

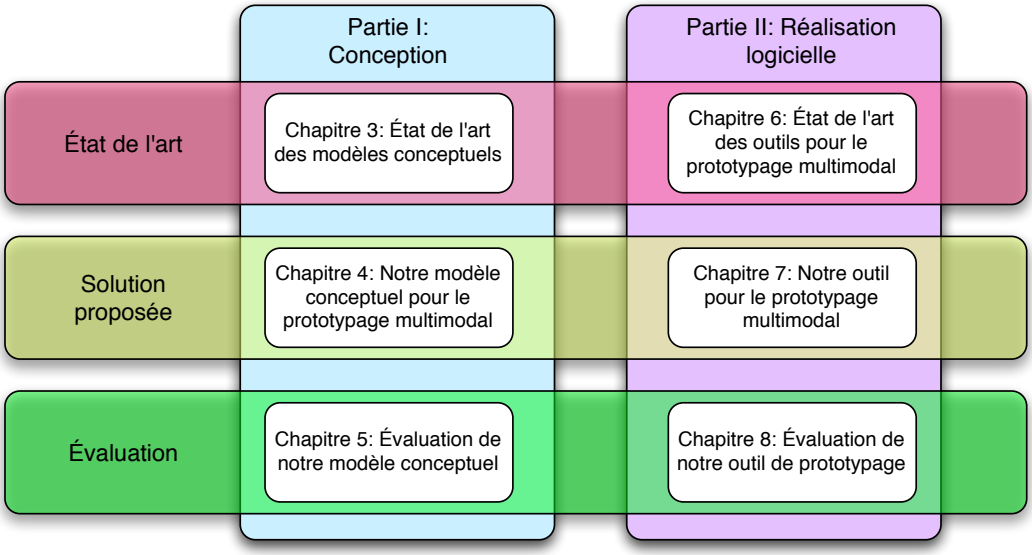


FIGURE 1.5 – Structure du mémoire.

1.5 Collaborations

Les travaux décrits dans ce mémoire sont le fruit de collaborations. En effet une partie de ces travaux s'inscrit dans le projet européen OpenInterface dédié à l'interaction multimodale (FP6-35182, www.oi-project.org) où j'ai eu la responsabilité d'une tâche complète (Workpackage) en relation directe avec mes travaux : la tâche sur l'outil graphique pour le prototypage d'interfaces multimodales.

Ainsi pendant mes travaux de thèse, j'ai eu la chance de collaborer avec de nombreux partenaires européens. En particulier, la collaboration avec l'Université Catholique de Louvain (UCL) a permis de développer un outil complet de prototypage s'appuyant sur un noyau à composants développés par UCL. De plus la collaboration avec les chercheurs en psychologie et ergonomie du Frauhofer Institute of Technology (FIT) et l'Université de Glasgow a permis de mieux cerner les caractéristiques des modalités d'interaction d'un point de vue utilisateur mais aussi de mener des évaluations expérimentales de mon outil.

Enfin de nombreux partenaires ont utilisé mon outil l'enrichissant de nouvelles modalités d'interaction et ont développé plusieurs prototypes multimodaux selon mon approche, que ce soient sur ordinateurs de bureau, dispositifs mobiles ou tables augmentées. Le projet OpenInterface a été donc été une opportunité unique pour que mon travail soit exploité par d'autres concepteurs que ceux de mon équipe de recherche.

Chapitre 2

Les interfaces multimodales

« Where the sea meets the land, life has blossomed into a myriad of unique forms in the turbulence of water, sand, and wind. At another seashore between the land of atoms and the sea of bits, we are now facing the challenge of reconciling our dual citizenships in the physical and digital worlds. Windows to the digital world are confined to flat square screens and pixels, or "painted bits." Unfortunately, one can not feel and confirm the virtual existence of this digital information through one's body.»

Iroshi Ishii

Contenu du chapitre

Nos travaux concernent la conception et le développement de prototypes multimodaux. Les interfaces multimodales se caractérisent par la multiplication des modes d'interaction pour interagir avec un ordinateur. Avant de présenter nos résultats conceptuels et logiciels, nous définissons dans ce chapitre tous les concepts relatifs aux interfaces multimodales et utilisés dans ce mémoire.

Tout d'abord nous situons les interfaces multimodales par rapport à l'histoire du domaine de l'Interaction Homme-Machine (IHM). Ensuite nous introduisons la terminologie utilisée en énonçant notre définition d'une interface multimodale. Nous illustrons la terminologie adoptée par des exemples concrets d'interfaces multimodales. Finalement, nous menons une première réflexion sur la nature des problèmes et des limitations identifiés pour la conception et le prototypage d'interfaces multimodales.

2.1 Un peu d'histoire

Le domaine de l'Interaction Homme-Machine (IHM) est relativement jeune. Les premiers pas vers les interfaces graphiques datent des années 60. Parmi les avancées majeures de ce domaine, figurent la manipulation directe d'objects graphiques, la souris, les fenêtres graphiques, l'édition de texte, l'hypertexte ou la reconnaissance de gestes.

Dans cette section, nous présentons brièvement ces différentes technologies qui composent les points de repère dans l'histoire de l'IHM. Ensuite nous exposons les faits marquants qui jalonnent l'histoire de l'interaction multimodale.

2.1.1 Points de repère dans l'histoire de l'IHM

Les points de repère de l'histoire de l'IHM correspondent à différentes technologies remontant à 1960 avec l'apparition de la manipulation directe d'objets graphiques. L'histoire de l'IHM a été décrite entre autre par Brad A. Myers [Myers, 1998], dont nous tirons les principales informations présentées ici.

Manipulation directe d'objets graphiques

Sketchpad, interface développée par Ivan Sutherland en 1962, permettait pour la première fois la manipulation directe d'objets graphiques (Figure 2.1). Sketchpad était composé d'un écran cathodique et d'un crayon optique qui permettait l'édition graphique de dessins techniques. Sketchpad est considéré comme la première interface graphique à manipulation directe.

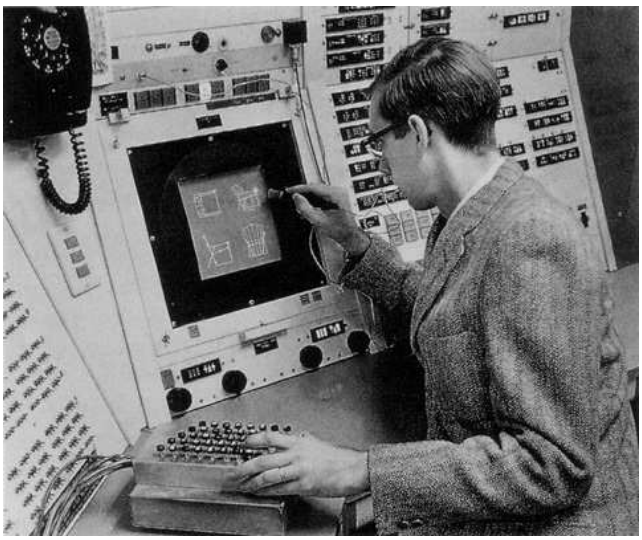


FIGURE 2.1 – Ivan Sutherland et la console Sketchpad (1962).

La souris

Deux ans plus tard, en 1964, Doug Engelbart invente la souris. L'objectif initial est de remplacer le crayon optique. La souris sera présentée publiquement en 1968

au SRI, à Stanford. La première souris est dotée d'un système à deux roues. Ce mécanisme ne sera pas retenu par la suite et laissera place à une boule puis au laser. Engelbart présente également un clavier à accords qui permet la spécification de données en composant des accords avec les doigts (Figure 2.2).

FIGURE 2.2 – Système NLS avec clavier à accords et souris.

Fenêtres graphiques

Des fenêtres graphiques avaient déjà été montrées dans le projet NLS de Engelbart en 1968. Néanmoins, les premières fenêtres graphiques superposées (multi-fenêtrage) apparaissent avec le système Smalltalk au Xerox PARC. Elles sont peu après introduites dans le système InterLisp. Le standard international X Window, reposant sur le multi-fenêtrage, fut développé au MIT en 1984.

Édition de texte

En 1962, Engelbart proposa un éditeur de texte avec quelques commandes habituelles aujourd'hui : recherche, remplacement, défilement, copie, etc. L'édition d'un texte avec la souris est également issue du projet NLS et fut présentée en 1968. Le logiciel Bravo, développé en 1974 au Xerox Parc, fut le premier éditeur de texte WYSIWIG (What You See Is What You Get).

Hypertexte

Un système hypertexte est un système dans lequel des documents sont liés entre eux par des hyperliens. Le concept vient du MEMEX (MEMory EXtender), une idée conçue en 1945 par Vanevar Bush, ingénieur et chercheur au MIT. Le terme hypertexte a été inventé par Ted Nelson, sociologue américain, en 1965. Le système NLS de Engelbart utilisait déjà la notion d'hypertexte en 1965 avec une liste d'articles liés (NLS Journal). Tim Berners-Lee reposa sur la notion d'hypertexte pour créer le World Wide Web en 1990 au CERN. Le premier navigateur web, Mosaic, fut développé à l'Université d'Illinois en 1992.

Reconnaissance de parole

La reconnaissance de parole naît en 1951 avec le détecteur de phonèmes présenté par S.P. Smith. L'année suivante est construite une machine capable de reconnaître

les dix premiers chiffres analogiquement. En 1961, J. Dreyfus invente le *phonétographe*, un appareil analogique capable d'identifier des phonèmes. L'appareil offre de très bons résultats mais l'utilisateur doit adapter sa diction pour que le système fonctionne.

En 1971 est lancé le premier système commercial, appelé "Voice Command System", capable de reconnaître 24 mots isolés. Parmi les dates phare des décennies suivantes, nous pouvons citer 1994 comme l'année où IBM lance le premier système de reconnaissance vocale sur PC, et 1997 comme l'année où IBM et la société Dragon proposent deux solutions de reconnaissance vocale continue (dans les systèmes précédents, il fallait marquer un temps de pause entre les mots).

Reconnaissance de gestes

La reconnaissance de geste est un domaine de l'informatique visant à permettre l'interaction gestuelle avec les ordinateurs. «L'interaction gestuelle est un ensemble de techniques d'interaction reposant sur l'interprétation de la dynamique des actions de l'utilisateur» [Baudel, 1995].

Le premier système de reconnaissance de gestes, utilisant un crayon optique, fut développé en 1963 par Teitelman [Teitelman, 1963]. La reconnaissance des gestes est ensuite intégrée dans les premières tablettes, comme dans la tablette Rand [Davis et Ellis, 1964].

La reconnaissance de gestes ne sera utilisé commercialement que à partir de 1991 avec l'apparition du système PenPoint [Carr, 1991]. Dans ce système, la dynamique du geste est analysée afin de générer la sémantique d'une commande. Par exemple, un tracé en forme de croix génère une commande *effacer* (Figure 2.3).

Tap	Sélectionner
Press-hold	Début déplacement
Tap-hold	Début dépl. (copie)
Flick (4 directions)	Navigation
Cross out	Effacer
Scratch out	Supprimer
Circle	Éditer
Check	Options
Caret	Insérer
Brackets	Sélection d'objet, ajustement
Pigtail	Supprimer Caractère
Down-right	Insérer un espace

FIGURE 2.3 – Les commandes gestuelles de PenPoint. Illustration extraite de [Carr, 1991].

La recherche en reconnaissance de gestes se dirige par la suite vers des nouveaux moyens de détection, comme la vision par ordinateur [Davis, 1994], puis vers la détection de gestes 3D [Heap et Hogg, 1996].

2.1.2 Histoire de l'interaction multimodale

Face à la variété des moyens d'interaction, il est difficile d'établir une histoire précise de l'interaction multimodale. En effet, la plupart des systèmes présentés précédemment sont multimodaux. Le terme "interaction multimodale" n'existait pourtant pas à l'époque.

On considère généralement le système "Put that there", présenté par Richard Bolt en 1980 [Bolt, 1980], comme le premier exemple d'interface multimodale. Dans ce système, l'utilisateur pouvait employer le geste combiné à la parole pour réaliser certaines tâches. Néanmoins, le terme sera utilisé pour la première fois par Bolt dans son article "The Integrated Multi-Modal Interface" en 1987 [Bolt, 1987].

Initialement l'interaction multimodale visait une interaction naturelle en se reposant sur la communication humaine, par nature multimodale, et en particulier la combinaison de la parole avec des gestes de type déictique. Il a fallu attendre les années 90 pour que les travaux sur la multimodalité aillent au-délà de la parole plus geste avec par exemple les travaux de Chatty sur l'interaction bi-manuelle ou les travaux de Bérard sur une caméra couplée à une souris [Bérard, 1999].

La première conférence internationale sur les interfaces multimodales, ICMI (International Conference on Multimodal Interfaces) a eu lieu en 1996 à Beijing. On peut donc établir cette date comme l'acceptation par la communauté scientifique de l'interaction multimodale en tant qu'axe de recherche à part entière.

2.2 Terminologie

Les termes *multimodal* et *multimodalité* sont des néologismes et n'existent pas dans le Dictionnaire de l'Académie Française. Il est donc intéressant d'analyser autant l'origine linguistique, en anglais, du terme original *multimodal*, que les morphèmes dont sont composé les termes créés en français.

Selon le Oxford Dictionnary [Abate et Jewell, 2001], *multimodal* est un adjectif qui caractérise ce qui possède différents modes d'activité ou occurrences[1]. Le terme *mode* est un nom qui désigne une manière d'expression ou de réalisation. En particulier, appliqué à l'informatique, un mode est une façon d'utiliser un système[2].

En français on a donc créé les termes *multimodal* (adjectif) et *multimodalité* (substantif) à partir des mots anglais *multimodal* et *multimodality*. Le terme multimodalité, d'un point de vue linguistique, est composé d'un préfixe, *multi*, suivi du radical *modalité* :

- **multi**, selon le dictionnaire du CNRTL [CNRTL-CNRS, 2009], est un préfixe qui indique la multiplicité.
- **modalité**, selon le dictionnaire du CNRTL, est une forme particulière d'une pensée, d'une organisation ou comme la manière dont se fait une action (modalité de paiement).

Face à ce flou terminologique, il convenait d'adopter une définition pour nos travaux.

[1] «**MULTIMODAL** (also multimode) adj. Characterized by several different modes of activity or occurence.» [Abate et Jewell, 2001]

[2] «**MODE** noun. A way or manner in which something occurs or is experienced, expressed, or done. Computing : a way of operating or using a system.» [Abate et Jewell, 2001]

2.3 Modalité et interaction multimodale : définitions adoptées

Considérons un système interactif composé d'un noyau fonctionnel et d'une interface entre ce noyau et l'utilisateur. Une modalité d'interaction représente un médiateur logiciel et matériel entre l'utilisateur et le système (Figure 2.4). Nous distinguons les modalités en entrée, de l'utilisateur vers le système, des modalités en sortie, du système vers l'utilisateur.

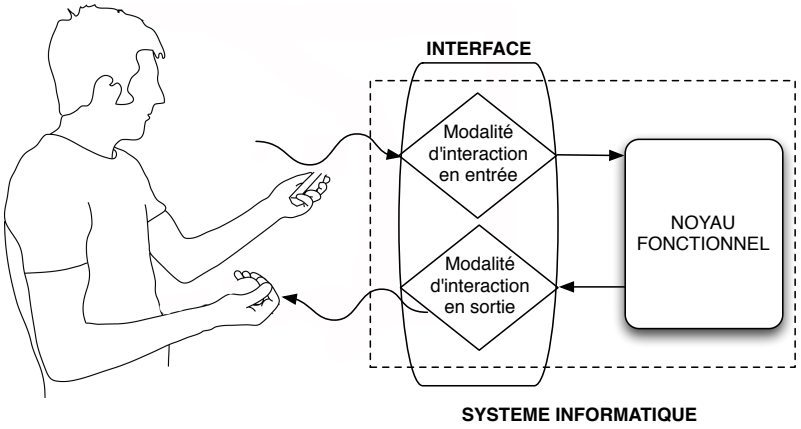


FIGURE 2.4 – Système interactif= utilisateur + système informatique

Modalité d'interaction

Prenant en compte à la fois le point de vue utilisateur et système, nous adoptons la définition d'une modalité d'interaction proposée par [Nigay et Coutaz, 1996]. Les auteurs distinguent deux niveaux d'abstraction, le niveau physique et le niveau de représentation d'une modalité. Une modalité d'interaction est alors définie par le couple <dispositif physique, langage d'interaction> :

- **un dispositif physique** désigne un dispositif d'entrée/sortie, tel que la souris, le clavier ou l'écran.
- **un langage d'interaction** est un système conventionnel structuré de signes qui assure une fonction de communication.

Par exemple, une modalité de localisation peut être décrite par le couple <GPS, localisation dans un référentiel>.

Multimodalité

À partir de la définition d'une modalité d'interaction, nous déduisons qu'un système interactif est multimodal lorsqu'il possède plusieurs modalités en entrée ou en sortie. Notons qu'un système peut être multimodal même s'il ne possède qu'un seul dispositif physique. En effet, si un même dispositif physique est couplé à deux langages d'interaction différents, cela définit deux modalités d'interaction séparées

(Tableau 2.1). Par exemple, nous pouvons coupler un microphone à une reconnaissance de parole pour composer une modalité <microphone, langage pseudo-naturel> comme dans le système *Voice as Sound* [Igarashi et Hughes, 2001], où l'utilisateur utilise un langage pseudo-naturel pour contrôler le déplacement d'éléments graphiques. Nous pouvons aussi utiliser l'intensité du son pour composer une modalité <microphone, intensité son> comme dans le système *Vocal Joystick* [Harada *et al.*, 2006], où l'utilisateur contrôle un curseur grâce à l'intensité de sa voix.

Exemple modalité	**Dispositif**	**Langage d'interaction**
Voice as Sound [Igarashi et Hughes, 2001]	Microphone	Langage pseudo-naturel
Voice Joystick [Harada *et al.*, 2006]	Microphone	Intensité son

TABLE 2.1 – Deux modalités différentes utilisant le même dispositif.

2.4 Exemples de systèmes interactifs multimodaux

Nous présentons des exemples d'interfaces multimodales pour illustrer la terminologie adoptée et les combinaisons de modalités possibles. Nous classons les exemples selon deux aspects complémentaires différents : les modalités d'interaction impliquées et les domaines d'application visés.

2.4.1 Modalités d'interaction utilisées en contexte multimodal

Soulignant la variété des possibilités, nous présentons des modalités d'interaction souvent utilisées dans des systèmes multimodaux, en commençant par la combinaison classique qui associe geste et parole.

Geste et parole

Le système Put-That-There, de Richard Bolt [Bolt, 1980], connu pour être le premier système multimodal, associe des commandes vocales à des techniques de pointage. Afin de déplacer un objet, l'utilisateur pointe du doigt sur l'objet et prononce "put that" (mets ça), puis désigne la destination et prononce "there" (là) (Figure 2.5 et Tableau 2.2).

Modalités	**Dispositif**	**Langage d'interaction**
Modalité 1	Microphone	Langage pseudo-naturel
Modalité 2	Capteur magnétique	Gestes 3D de pointage

TABLE 2.2 – Modalités utilisées dans le système Put-That-There.

Cette combinaison de modalités a été largement reprise dans différents systèmes multimodaux et reste l'exemple classique d'interaction multimodale. Les systèmes varient souvent dans la façon de capter le geste, en 2D [Oviatt *et al.*, 2000] ou en 3D [Corradini *et al.*, 2002].

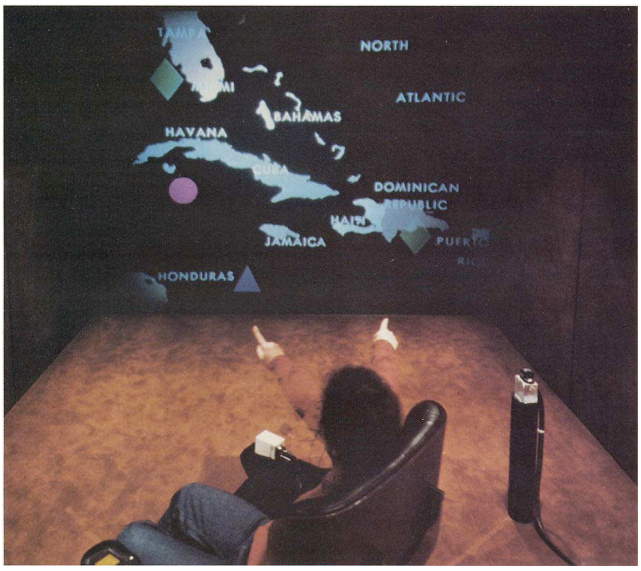


FIGURE 2.5 – Système Put-That-There de Richard Bolt. Illustration extraite de [Bolt, 1980].

Interaction multi-doigt et bi-manuelle

Dans cette section nous considérons deux techniques d'interaction, l'interaction multi-doigt et l'interaction bi-manuelle, en tant que modalités d'interaction individuelles (ces techniques d'interaction peuvent parfois être considérées comme des interactions multimodales). Nous présentons ensuite des exemples de combinaison de ces modalités afin de définir des interfaces multimodales.

L'interaction multi-doigt [Wu et Balakrishnan, 2003] consiste à utiliser plusieurs doigts d'une même main pour réaliser des tâches interactives, à différence des techniques d'interaction mono-doigt. L'interaction multi-doigt permet d'augmenter considérablement les capacités interactives d'un système. Nous retrouvons un exemple de système multi-doigt dans [Malik *et al.*, 2005]. Ce système permet à un utilisateur d'interagir avec une surface de projection, depuis sa table en utilisant des gestes avec la main. Ce gestes sont basés sur des positions et des gestes des doigts (Figure 2.6), captés grâce à une caméra (Tableau 2.3).

Dispositif	**Langage d'interaction**
Caméra	Reconnaissance des doigts

TABLE 2.3 – Modalité d'interaction multi-doigt.

L'interaction bi-manuelle consiste à utiliser les deux mains pour effectuer des tâches interactives. Cette utilisation des deux mains peut avoir lieu selon différents aspects temporels, de dépendance et de continuité [Bailly *et al.*, 2005]. L'interaction bi-manuelle tire profit de l'utilisation naturelle par l'être humain des deux mains pour effectuer des tâches. En 1986, Buxton et Myers mettent en évidence que l'interaction à deux mains permet de réaliser certaines tâches plus rapidement, comme la recherche et sélection de mots dans un texte [Buxton et Myers, 1986].

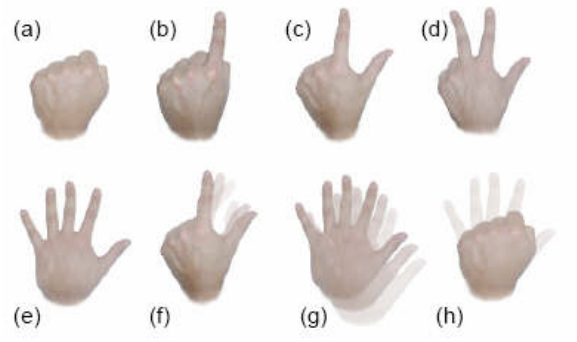


FIGURE 2.6 – Positions (a, b, c, d, e) et gestes (f, g, h) des doigts reconnus dans une interaction multi-doigt pour une surface de projection. a) Poing fermé b) Pointage simple c) Double pointage d) Triple pointage e) Main ouverte f) Geste de pinçage g) Geste de déplacement h) Geste de fermeture de la main. Illustration extraite de [Malik *et al.*, 2005].

Peu apès, Richard Bolt étudie la pertinence d'utiliser l'interaction bi-manuelle de façon conjointe avec des commandes vocales dans des systèmes multimodaux. Il propose un prototype multimodal qui utilise l'interaction bi-manuelle et la reconnaissance vocale [Bolt et Herranz, 1992] (Figure 2.7 et Tableau 2.4). Le système permet à l'utilisateur de manipuler des objets virtuels en 3D avec des gestes et des commandes vocales. Par exemple, l'utilisateur peut faire tourner un objet en bougeant les deux mains de façon synchrone ou en disant «tourne l'objet» (Figure 2.7).

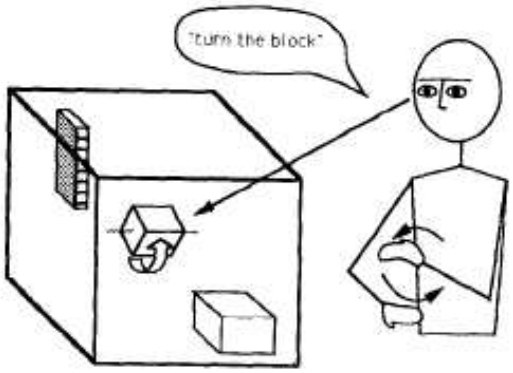


FIGURE 2.7 – Manipulation d'objets 3D avec le geste et la voix. Illustration extraite de [Bolt et Herranz, 1992].

Modalités	**Dispositif**	**Langage d'interaction**
Modalité 1	Microphone	Langage pseudo-naturel
Modalité 2	Capteur	Gestes bi-manuels

TABLE 2.4 – Modalités d'interaction dans le système de manipulation d'objets 3D [Bolt et Herranz, 1992].

L'interaction multi-doigt et l'interaction bi-manuelle ont également été combinées afin de créer des interfaces multimodales. Le système RoomPlaner [Wu et Balakrishnan, 2003] utilise une interaction bi-manuellet multi-doigt afin de permettre aux utilisateurs de collaborer sur une table tactile multi-utilisateurs. Les utilisateurs peuvent utiliser les doigts d'une seule main afin de réaliser différentes commandes mais aussi des gestes avec les deux mains (Figure 2.8 et Tableau 2.5).

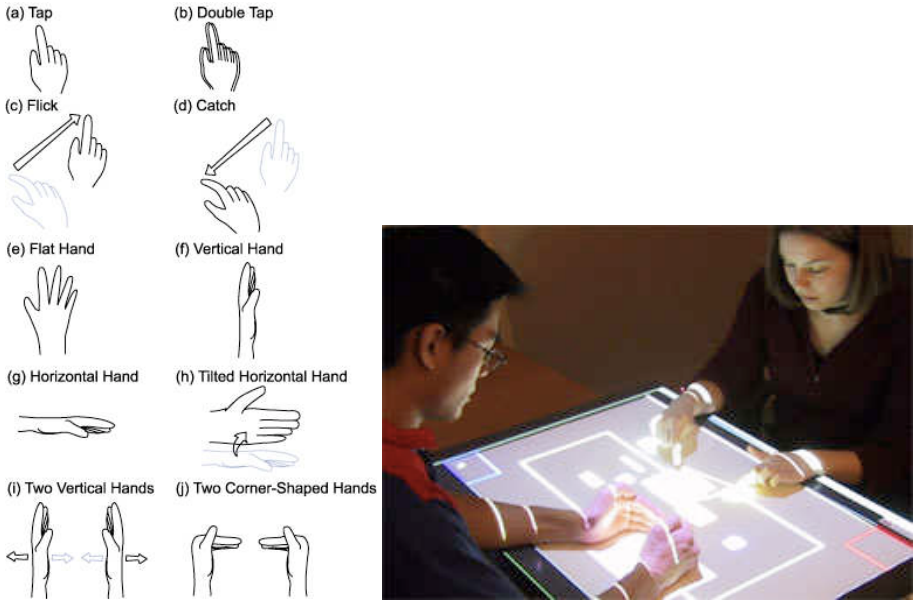


FIGURE 2.8 – Interaction multimodale multi-doigt et multi-utilisateur dans le système RoomPlaner. Illustration extraite de [Wu et Balakrishnan, 2003].

Modalités	**Dispositif**	**Langage d'interaction**
Modalité 1	Caméra	Reconnaissance des doigts
Modalité 2	Caméra	Gestes bi-manuels

TABLE 2.5 – Modalités d'interaction dans le système RoomPlaner [Wu et Balakrishnan, 2003].

Suivi du regard

Le suivi du regard[3] est une technique qui consiste à capturer le regard de l'utilisateur à l'aide de dispositifs de vision, idéalement non-intrusifs [Collet, 1999]. L'idée d'utiliser le suivi du regard en tant que mode d'interaction fut introduit par Richard Bolt en 1982 [4] [Bolt, 1982]. Cette technique permet entre autre aux utilisateurs à motricité réduite d'interagir avec des systèmes informatiques en utilisant le mouvement de la pupille (Tableau 2.6).

En libérant les autres sens humains, le suivi du regard se prête bien à son utilisation composée dans des systèmes multimodaux. Des chercheurs de Queen Univer-

[3] gaze tracking, eye movement tracking
[4] «Think of it as "eyes as output".» [Bolt, 1982]

Dispositif	Langage d'interaction
Dispositif de captation du regard	Mouvement de la pupille

TABLE 2.6 – Modalité suivi du regard.

sity (Kingston, Ontario) ont exploré l'utilisation du suivi du regard pour contrôler des jeux vidéos existants [Smith et Graham, 2006]. Ils ont ainsi testé le contrôle de Quake, un jeu de style *shooter*, de façon multimodale avec le regard pour contrôler la vue du personnage, le clavier pour se déplacer et la souris pour utiliser leur arme (Figure 2.9 et Tableau 2.7).

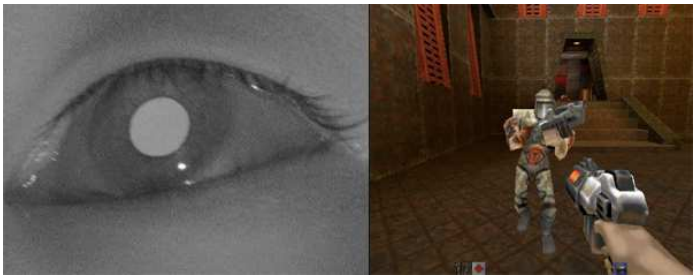


FIGURE 2.9 – Jeu commercial Quake contrôlé avec le regard, le clavier et la souris [Smith et Graham, 2006].

Modalités	Dispositif	Langage d'interaction
Modalité 1	Dispositif de captation du regard	Mouvement de la pupille
Modalité 2	Clavier	Commandes de déplacement
Modalité 3	Souris	Commandes d'attaque

TABLE 2.7 – Modalités utilisées dans le jeu Quake contrôlé de façon multimodale.

Interaction cerveau-machine

Les interfaces cerveau-machine[5] [Millán, 2006] visent à utiliser l'activité cérébrale de façon consciente et spontanée à travers la modulation des ondes cérébrales pour contrôler des tâches interactives. Afin de mesurer l'activité cérébrale, nommée electro-encéphalographie (EEG), des électrodes sont placés sur le cuir chevelu (Figure 2.10 et Tableau 2.8).

Dispositif	Langage d'interaction
EEG	Déplacement vertical

TABLE 2.8 – Modalité interaction cerveau-machine.

[5]Brain-Computer Interfaces (BCI)

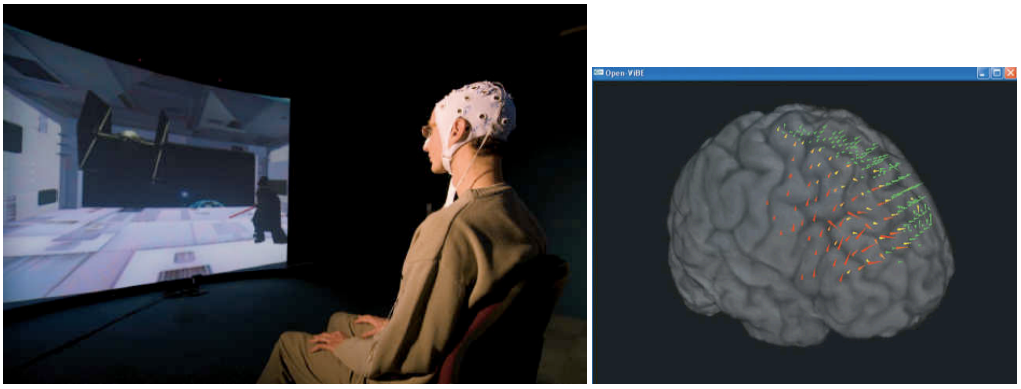


FIGURE 2.10 – Système d'interaction cerveau-machine "Use the force", où l'utilisateur peut déplacer verticalement des objets virtuels avec sa pensée.

L'interaction cerveau-machine est souvent appliquée aux domaines de la réalité virtuelle et des jeux afin de contrôler les personnages ou de déclencher certaines commandes [Lécuyer *et al.*, 2008]. Cette modalité d'interaction ne se présente pas comme une substitution des autres modalités, mais plutôt comme un complément ou extension. Les interfaces cerveau-machine sont donc ainsi utilisées de façon multimodale avec d'autres modalités d'interaction. Un exemple de système multimodal qui exploite une interface cerveau-machine comme une modalité d'interaction est présenté dans [Vilimek et Zander, 2009]. Le système utilise la détection du regard afin de sélectionner un objet et des données EEG pour simuler un click souris (Tableau 2.9).

Modalités	**Dispositif**	**Langage d'interaction**
Modalité 1	EEG	Séléction d'un objet
Modalité 2	Dispositif de captation du regard	Manipulation directe

TABLE 2.9 – Modalités utilisées dans le jeu Quake contrôlé de façon multimodale.

Localisation et Géolocalisation

L'interaction basée sur la localisation consiste à exploiter l'emplacement de l'utilisateur, par exemple sa distance par rapport à l'ordinateur, pour activer des tâches interactives. La géolocalisation est un type particulier de localisation qui consiste à positionner une personne sur un plan à l'aide de ses coordonnées géographiques. La géolocalisation se réalise grâce au Système de Positionnement Global (Global Positionning System ou GPS) (Tableau 2.10).

Dispositif	**Langage d'interaction**
GPS	Localisation dans un référentiel

TABLE 2.10 – Modalité géolocalisation.

Souvent combiné à l'orientation de l'utilisateur, la localisation est souvent utilisée dans les systèmes de réalité augmentée mobile. Nearest Tube est un exemple d'application multimodale sur iPhone qui exploite la position de l'utilisateur (Figure 2.11 et Tableau 2.11). Nearest Tube permet à l'utilisateur de chercher les stations de métro à proximité, à partir de sa position géographique combinée avec l'orientation du dispositif. Ainsi, l'utilisateur peut déplacer le iPhone autour de lui afin d'afficher les informations associées à sa position géographique de façon augmentée sur l'écran. En se déplaçant, l'utilisateur change les informations obtenues.

FIGURE 2.11 – Nearest Tube, application multimodale utilisant le geste et la géo-localisation pour afficher des informations augmentées.

Modalités	**Dispositif**	**Langage d'interaction**
Modalité 1	GPS	Localisation
Modalité 2	Accéléromètre	Orientation

TABLE 2.11 – Modalités utilisées dans Nearest Tube.

Interfaces tangibles

Les interfaces tangibles désignent des interfaces où l'interaction repose sur la manipulation d'objets physiques [Coutrix *et al.*, 2005]. Même si nous pouvons considérer que toute interface est tangible (manipulation de la souris, du clavier), le terme d'interface tangible s'appliqueaux cas dépassant le cadre classique souris-clavier-écran. Par exemple, dans le système Sandscape [Ishii *et al.*, 2004] les utilisateurs manipulent du sable qui modifie en temps réel l'affichage projeté sur le sable, qui représente la morphologie du bac à sable (la hauteur, en différentes couleurs) (Tableau 2.12 et Figure 2.12).

Dans [Cohen et McGee, 2004] la complémentarité entre la manipulation d'un objet tangible et des modalités linguistiques (modalité orale ou écrite) est souli-

Dispositif	Langage d'interaction
Caméra	Forme du sable

TABLE 2.12 – Modalité d'interaction tangible.

FIGURE 2.12 – Des utilisateurs manipulant le sable afin d'interagir avec le système Sandscape.

gnée. Une modalité linguistique est exploitée pour annoter un objet physique [6]. Par exemple, dans le système NISMap [Cohen et McGee, 2004], un système multimodal tangible de notation de cartes dans un contexte militaire, l'utilisateur peut interagir avec la voix ou avec l'écriture en utilisant un stylo numérique sur des cartes imprimées sur des feuilles tramées (Figure 2.13 et Tableau 2.13). L'objectif de ce système est de garder la méthode traditionnelle et tangible de travail des membres de l'armée, qui travaillent de façon collaborative sur des cartes annotées, avec les capacités d'interaction de la multimodalité.

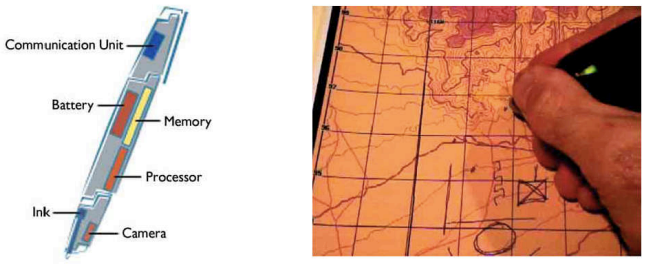


FIGURE 2.13 – Interaction multimodale tangible dans le système militaire NISMap. L'utilisateur peut utiliser la voix et l'écriture afin d'annoter numériquement des cartes papier tramées. Illustration extraite de [Cohen et McGee, 2004].

[6] «Although tangible systems allow users to manipulate physical objects, they typically do not interpret any annotations that accompany those objects. On the other hand, multimodal user interfaces (MMUIs) can analyze users' verbal and written information, but they typically do not acquire information from objects situated in the real world.» [Cohen et McGee, 2004]

Modalités	Dispositif	Langage d'interaction
Modalité 1	Microphone	Langage pseudo-naturel
Modalité 2	Stylet numérique	Annotation sur carte papier

TABLE 2.13 – Modalités utilisées dans le système NISMap.

2.4.2 Domaines d'application de la multimodalité

Systèmes adaptés aux personnes handicapées

Sharon Oviatt affirme dans son article «10 mythes sur l'interaction multimodale» [Oviatt, 1999] que l'avantage des interfaces multimodales réside moins dans l'efficacité apportée que dans l'adaptation à différents types d'utilisateurs, comme par exemple les handicapés[7]. La multimodalité en entrée, grâce à l'utilisation de modalités telles que le suivi du regard ou l'interaction cerveau-machine, permet aux handicapés moteurs d'interagir avec un ordinateur. De plus, la multimodalité en sortie exploite plusieurs modalités autres que visuelles pour des handicapés visuels. Par exemple dans [Saarinen *et al.*, 2005] est présenté un système multimodal composé d'un retour haptique et sonore pour enfants malvoyants (Figure 2.14 et Tableau 2.14).

FIGURE 2.14 – Système multimodal pour enfants malvoyants. Illustration extraite de [Saarinen *et al.*, 2005].

Modalités	Dispositif	Langage d'interaction
Modalité 1	Joystick	Langage haptique
Modalité 2	Haut parleurs	Langage sonore

TABLE 2.14 – Modalités utilisées dans un système multimodal en sortie pour enfant malvoyants.

[7] «the flexibility of a multimodal interface can accommodate a wide range of users, tasks, and environments—including users who are temporarily or permanently handicapped» [Oviatt, 1999]

Systèmes critiques

Les systèmes critiques englobent les différents systèmes dont une panne provoquerait des conséquences dramatiques. Les systèmes de transport, de production d'énergie ou de santé sont des exemples de systèmes critiques. Un aspect spécialement important de ces systèmes réside dans la définition de l'interface qui permet de les contrôler. Différents facteurs, comme la croissante complexité des interfaces[Palanque *et al.*, 2007], augmente la probabilité d'erreurs de manipulation de l'utilisateur. Les interfaces multimodales sont particulièrement adaptées aux systèmes critiques grâce à la possibilité d'utiliser des modalités de façon équivalente ou redondante [Cohen et McGee, 2004].

D'un côté, l'interaction multimodale permet, par l'utilisation équivalente de modalités en entrée, de changer de modalité en cas de panne d'un dispositif[8]. Des travaux ont par exemple été menés sur les interfaces des cockpits des avions (Figure 2.15) afin d'offrir des architectures logicielles capables de gérer la panne d'un dispositif en entrée ou en sortie [Navarre *et al.*, 2008]. Par exemple, un clavier logiciel peut remplacer le clavier physique par défaut en cas de panne[9] (Figure 2.15 et Tableau 2.15).

FIGURE 2.15 – Interface de cockpit d'avion utilisant des modalités en entrée redondantes, comme plusieurs claviers, afin de contourner les pannes de dispositifs. Illustration extraite de [Barboni *et al.*, 2007].

D'un autre côté, l'interaction multimodale permet également, par l'utilisation redondante de modalités en entrée, d'éviter les fausses manipulations ou les commandes erronées. Un exemple d'application multimodale utilisant des modalités de façon redondante pour un système militaire est présenté dans [Bouchet *et al.*, 2004]. Il s'agit d'une interface multimodale pour le système FACET, un simulateur de vol du Rafale, avion de chasse français (Figure 2.16). Le système

[8] «In systems offering standard input device combination such as mouse + keyboard, it is possible to handle one input device failure by providing redundancy in the use of the device.» [Navarre *et al.*, 2008]

[9] «For instance a soft keyboard [...] can provide an efficient palliative for a keyboard failure» [Navarre *et al.*, 2008]

Modalités	Dispositif	Langage d'interaction
Modalité 1	Clavier physique	Langage clavier
Modalité 2	Caméra	Langage clavier

TABLE 2.15 – Modalités redondantes utilisées dans un cockpit d'avion afin de contourner la panne d'un dispositif.

permet au pilote de réaliser plusieurs tâches de façon multimodale en utilisant le Hotas (Hans on Throttle and Stick), la direction du casque et la voix. Par exemple, le pilote peut marquer un point au sol en utilisant la direction de son casque couplée à une commande vocale confirmée par l'appui d'un bouton sur le Hotas (Tableau 2.16).

FIGURE 2.16 – FACET, simulateur multimodal de vol du Rafale. Illustration extraite de [Bouchet *et al.*, 2004].

Modalités	Dispositif	Langage d'interaction
Modalité 1	Microphone	Langage pseudo-naturel
Modalité 2	Casque	Pointage objet
Modalité 3	Hotas	Commande confirmation

TABLE 2.16 – Modalités utilisées dans FACET.

Systèmes de réalité augmentée

La réalité augmentée a été initialement définie par les techniques consistant à superposer au monde réel des informations générées par ordinateur [Mackay, 1998]. Pour cela, les systèmes de réalité augmentée utilisent souvent des casques de réalité virtuelle (*video see-through*) ou des casques semi-translucides (*optic see-through*). Les systèmes de réalité augmentée, en sortant l'utilisateur de l'environnement classique clavier-souris-écran, utilisent souvent des interfaces multimodales afin d'interagir dans le monde ainsi augmenté.

Dans [Olwal *et al.*, 2003] est présenté un système de réalité augmentée multimodal noté Senseshapes (Figure 2.17). Le système focalise sur la sélection d'objets virtuels 3D juxtaposés au monde réel et affichés dans un casque. Dans Senseshapes,

FIGURE 2.17 – Interaction multimodale dans Senseshapes, un système de réalité augmentée. Illustration extraite de [Olwal *et al.*, 2003].

la solution proposée consiste à utiliser le geste avec des rayons de sélection (le cône sur la Figure 2.17) afin de pointer un objet, couplé avec la voix afin de le sélectionner (Tableau 2.17). Le système utilise un gant pour capturer les gestes qui déterminent la forme du rayon de sélection (à droite sur la Figure 2.17). Le système de fusion multimodale utilisé dans SenseShapes permet de d'éliminer l'ambigüité produite par les modalités gestuelles et vocales [10].

Modalités	**Dispositif**	**Langage d'interaction**
Modalité 1	Microphone	Langage pseudo-naturel
Modalité 2	Gant	Pointage objet

TABLE 2.17 – Modalités utilisées dans Senseshapes.

Systèmes sur supports mobiles

L'interaction multimodale sur supports mobiles vise des modalités d'interaction plus adaptées au support mobile que le clavier-souris-écran traditionnel. La plupart des terminaux mobiles ont aujourd'hui un écran tactile, voir multi-touch. Les dispositifs mobiles offrent également des commandes vocales, en plus des boutons physiques. Le meilleur exemple d'interaction multimodale sur support mobile est le iPhone.

Le iPhone est le premier terminal mobile multitouch à avoir bénéficié d'un large succès commercial. L'interface du iPhone est composée d'un écran multitouch et de trois boutons latéraux et un bouton frontal. Le iPhone contient également un accéléromètre qui permet de détecter l'orientation du terminal. Les applications sur iPhone utilisent autant les capacités tactiles de son interface que l'accéléromètre. Par exemple, le jeu multimodal Crash Kart (Figure 2.18) permet au joueur

[10] «Mutual disambiguation» [Kaiser *et al.*, 2003]

de contrôler un véhicule en inclinant le iPhone. Le joueur utilise également l'écran tactile pour activer différentes options du jeu ou pour freiner le véhicule (Tableau 2.18).

FIGURE 2.18 – Interaction multimodale dans le jeu Crash Kart pour iPhone

Modalités	Dispositif	Langage d'interaction
Modalité 1	Écran tactile	Commandes tactiles
Modalité 2	Accéléromètre	Déplacement latéral

TABLE 2.18 – Modalités utilisées dans le jeu Crash Kart.

Jeux vidéo

Les nouvelles consoles de jeux vidéos dirigent leurs innovations non plus vers la qualité des graphismes ou la rapidité du rendu, mais vers les nouvelles interfaces de contrôle. Les manettes de jeu sont passées de filaires à sans-fils. Elles intègrent désormais des capteurs tels que des accéléromètres ou une caméra infra-rouge (IR) qui permettent de détecter la position de la manette dans l'espace.

Nintendo avec la Wii a été un précurseur pour l'interaction multimodale dans les jeux vidéos en introduisant de nouveaux capteurs pour les jeux vidéos. La Wiimote, manette de la Wii intégrant plusieurs capteurs, et la Wii Fit, dispositif qui permet de capter le poids et la position d'un joueur, sont utilisés de façon multimodale dans la suite Wii Fit Plus, qui intègre différents jeux. La Figure 2.19 illustre l'utilisation complémentaire de la manette et de la Wii Fit dans le jeu Boules Maboules afin de contrôler les trois plateformes qui interfèrent sur le parcours de la boule vers les tuyaux en bas de l'écran (Tableau 2.19).

FIGURE 2.19 – Utilisation multimodale de la Wiimote et de la Wii Fit dans le jeu Boules Maboules afin de contrôler les différentes plateformes du jeu.

Modalités	Dispositif	Langage d'interaction
Modalité 1	Wiimote	Inclinaison
Modalité 2	Wii Fit	Inclinaison

TABLE 2.19 – Modalités utilisées dans le jeu Boules Maboules.

2.5 Prendre du recul

2.5.1 Constat : explosion des possibilités

Les exemples précédents, par la diversité des modalités et domaines d'application, soulignent que l'espace des possibles en termes de modalités d'interaction est de plus en plus vaste.

Multiplicité des modalités d'interaction

L'interaction multimodale ne se limite plus à la combinaison de la parole et du geste, comme le préconisait Sharon Oviatt [11] [1999]. L'apparition de nouvelles modalités d'interaction multiplie les possibilités de combinaison.

Variété des domaines d'application

Cette multiplicité s'unit à la variété des domaines d'application où l'interaction multimodale démontre sa pertinence, comme nous l'avons vu dans les exemples cités préalablement. Des prototypes de recherche aux applications commerciales comme les jeux, la multimodalité sous différentes formes est aujourd'hui utilisée pour des tâches interactives diverses.

[11] «the belief that speech is primary risks underexploiting the valuable roles to be played by other modes in next-generation multimodal architectures.» [Oviatt, 1999]

Diversité des plates-formes logicielles et physiques

Nous avons montré comment l'interaction multimodale se manifeste aujourd'hui non seulement sur les ordinateurs de bureau, mais également sur des dispositifs mobiles, sur des systèmes embarqués (comme un avion de chasse) ou sur des consoles de jeux vidéo. Ainsi, l'interaction multimodale se développe sur des plates-formes possédant différentes caractéristiques logicielles et physiques. La multimodalité vise alors à tirer profit pour chaque plate-forme des possibilités en termes de modalités d'interaction : les accéléromètres et écrans tactiles pour les plates-formes mobiles, le casque et le Hotas dans l'avion de chasse ou les nouvelles manettes dans les jeux vidéo.

Ainsi nous constatons que la multimodalité joue un rôle important pour des plates-formes variées, impliquant des contextes d'usage très différents, que ce soit dans la rue, dans une voiture ou dans la salle à manger. Le contexte d'usage est donc un paramètre important à prendre en compte lors de la conception d'interfaces multimodales.

Utilisateurs

Cette diversité de modalités, de domaines d'application et de plates-formes implique un large spectre d'utilisateurs. Ainsi, l'interaction multimodale se révèle adaptée pour palier certains handicaps des utilisateurs qu'ils soient permanents ou temporaires. De plus, l'interaction multimodale est appropriée non seulement pour des applications professionnelles, incluant les systèmes critiques mais aussi des applications grand public comme celle sur téléphones mobiles ou les jeux vidéos.

2.5.2 Positionnement des travaux

Nous constatons que l'espace des possibilités en termes de modalités d'interaction et de leurs combinaisons est de plus en plus vaste et que les dimensions de conception à prendre en compte, que ce soient celles liées à l'utilisateur, au domaine d'application ou au contexte d'usage, sont multiples. Face à ce constat, il est donc important de pouvoir explorer efficacement l'espace des possibilités afin de prendre en compte les multiples facettes de la conception d'interfaces multimodales. Pour un système à concevoir, de nombreuses modalités et formes de combinaison de modalités doivent pouvoir être considérés et testés rapidement avec des utilisateurs finaux.

Or un frein important à l'exploration des possibilités est l'absence d'outils conceptuels et/ou logiciels. En particulier l'absence d'outil logiciel implique un coût élevé de développement des interfaces multimodales, qui n'autorise pas le développement de plusieurs solutions à tester avec les utilisateurs. En effet le développement d'interfaces multimodales est complexe, souvent fait de façon ad hoc, spécifique aux modalités d'interaction considérées et dépendant du domaine d'application, de l'utilisateur, de la plate-forme logicielle ou matérielle, empêchant la réutilisation même partielle du code développé.

Dans ce contexte, nos travaux ont pour objectif la définition d'outils conceptuels et logiciels pour faciliter l'exploration des possibilités en termes d'interaction mul-

timodale en entrée, de l'utilisateur vers le système. Nous centrons donc nos travaux sur la phase de conception d'interaction multimodale en visant :

- un modèle conceptuel qui permet la description des alternatives de conception,
- un outil logiciel de prototypage rapide des alternatives de conception décrites selon notre modèle.

Pour l'exploration efficace de l'espace des possibilités, nous adoptons donc une approche itérative centrée utilisateur reposant sur le prototypage. Le prototypage rapide d'interfaces existe depuis longtemps, comme en témoigne l'outil Druid proposé en 1990 [Singh *et al.*, 1990] et qui visait la création rapide d'interfaces pour tester des alternatives de conception[12].

Le prototypage représente une solution pour l'exploration efficace de l'espace de conception [Lim *et al.*, 2008] et s'inscrit directement dans une démarche de conception itérative centrée utilisateur [Jones *et al.*, 2007]. Rappelons que les trois caractéristiques principales de la conception centrée utilisateur comme définies dans la norme ISO 13407 sont :

- la prise en compte des utilisateurs en amont du cycle de développement par l'analyse de leurs besoins ;
- la participation active des utilisateurs dans la conception à la fois par l'évaluation des solutions de conception et par l'identification de leurs besoins ;
- la démarche de conception itérative.

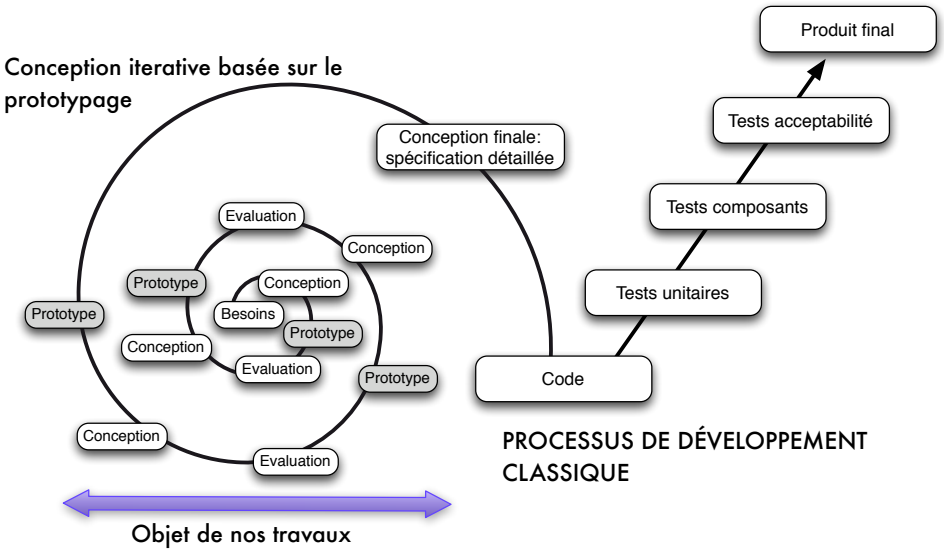


FIGURE 2.20 – Démarche itérative de prototypage centrée utilisateur.

[12] «Druid facilitates the rapid creation and testing of design alternatives for interfaces» [Singh *et al.*, 1990]

Au sein d'une telle démarche de conception itérative centrée utilisateur, le prototypage rapide prend tout son sens pour permettre des itérations courtes comprenant [la conception, le prototypage et l'évaluation]. A la Figure 2.20, inspirée de [Schlienger et Chatty, 2007], nous articulons cette étape de conception itérative basée sur le prototypage pour obtenir une spécification complète et détaillée de l'interaction multimodale, avec les étapes de développement d'un cycle de vie classique en Génie Logiciel.

2.6 Synthèse

Dans ce chapitre nous avons introduit les principales notions liées à l'interaction multimodale et la terminologie adoptée dans nos travaux. Nous avons illustré le domaine par des exemples concrets d'interfaces multimodales selon deux axes : les modalités d'interaction utilisées et les domaines d'application.

Ces deux axes permettent de souligner la grande variété des modalités d'interaction existantes aujourd'hui et des possibilités en termes d'applications pour l'interaction multimodale.

Face à cette diversité, nous avons conclu sur le besoin d'outils conceptuels et logiciels pour favoriser l'exploration efficace de l'espace des possibilités lors de la conception itérative centrée utilisateur.

Ceci constitue notre objet de recherche, que nous traitons dans les chapitres suivants selon sa facette conceptuelle et logicielle.

Première partie

Conception

Chapitre 3

Etat de l'art des modèles conceptuels

«Le poète ne doit avoir qu'un modèle, la nature »

Victor Hugo, écrivain français (1802-1885).

Dans les deux premiers chapitres, nous avons introduit la portée de nos travaux, l'interaction multimodale et la terminologie utilisée dans ce mémoire.

Nous consacrons la partie 1 de ce mémoire aux modèles conceptuels et concepts clefs de la multimodalité pour décrire des solutions d'interaction. Dans ce chapitre nous faisons le point sur les modèles de conception existants. Ensuite, dans le chapitre 4, nous présentons notre contribution à la conception d'interfaces multimodales puis, dans le chapitre 5, nous présentons l'évaluation de notre modèle.

Contenu du chapitre

3.1 Introduction

Selon [Beaudouin-Lafon, 2004], il existe deux niveaux conceptuels dans l'analyse et la conception de l'interaction : les **paradigmes d'interaction** et les **modèles d'interaction**. Les paradigmes d'interaction offrent une vision haut-niveau du phénomène de l'interaction, tandis que les modèles d'interaction permettent des descriptions opérationnelles du fonctionnement de l'interaction.

Trois paradigmes d'interaction principaux sont identifiés[1] et étudiés par des domaines de recherche différents :

- **Machine-outil** : dans ce type de paradigme, la machine est perçue comme une extension des possibilités humaines, un outil très sophistiqué. Le domaine de l'interaction homme-machine étudie ce premier paradigme ;

- **Machine-partenaire** : dans ce type de paradigme, l'homme communique avec la machine grâce à des moyens de communications anthropomorphiques, comme la parole. Le domaine de l'intelligence artificielle étudie ce deuxième paradigme ;

- **Machine-médium** : dans ce type de paradigme, la machine est un médium qui permet aux hommes de communiquer entre eux. Le domaine de la collaboration assistée par ordinateur[2] examine ce paradigme.

Notre étude se limite donc au premier paradigme. Dans ce contexte, le but d'un modèle d'interaction est de fournir un outil pour guider les concepteurs, développeurs et même les utilisateurs dans la création de systèmes interactifs [Beaudouin-Lafon, 2004]. Les modèles peuvent alors avoir différentes formes, des guides de conception haut-niveau, comme le modèle de la manipulation directe [Shneiderman, 1987], aux règles de conception précises, comme les Apple Human Interface Guidelines [Computer, 1987].

Dans ce chapitre, nous présentons d'abord les modèles de conception dédiés à l'interaction multimodale (Figure 3.1). Ces derniers concernent à la fois les modalités d'interaction mais aussi leurs compositions.

Dans une deuxième partie, nous étudions les modèles de conception logicielle. Nous y distinguons les modèles d'architecture logicielle des modèles de programmation pour l'interaction.

Finalement, nous présentons les modèles de conception logicielle pour l'interaction multimodale. Deux approches de conception logicielle pour la multimodalité sont analysées : le modèle des configurations d'entrée et le modèle ICARE.

Dans la synthèse qui conclut ce chapitre, nous centrons la discussion sur les modèles pour le développement rapide de prototypes multimodaux. Cette synthèse motive notre modèle, que nous présentons au chapitre suivant.

[1] «computer-as-tool, computer-as-partner, and computer-as-medium»[Beaudouin-Lafon, 2004]
[2] Computer-Supported Cooperative Work

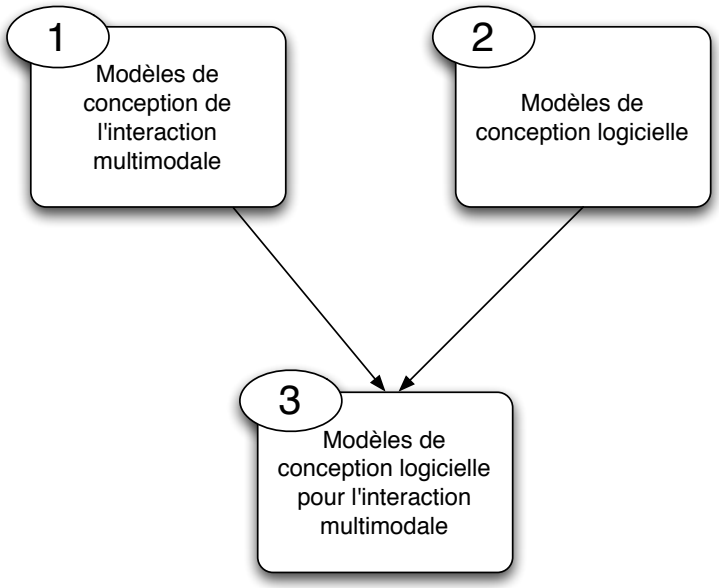


FIGURE 3.1 – Organisation du chapitre.

3.2 Modèles de conception de l'interaction multimodale

3.2.1 Introduction

Dans cette section, nous présentons les modèles pertinents pour la conception d'interaction multimodale en entrée. Ces modèles concernent les différentes étapes au sein du processus de transformation des actions de l'utilisateur aux tâches interactives.

Une part des travaux sur l'interaction en entrée est dédiée aux dispositifs en entrée. Ces modèles caractérisent les dispositifs physiques à deux niveaux d'abstraction : le niveau physique avec les actions utilisateur capturées par le dispositif et le niveau logique avec les tâches interactives élémentaires réalisables avec le dispositif.

Au-delà des dispositifs, d'autres modèles décrivent l'interaction. Nous commençons notre revue de ces modèles par le modèle fondateur de la manipulation directe. Puis nous considérons le modèle de l'interaction instrumentale.

Ces modèles considèrent une modalité d'interaction pour réaliser une tâche. La multimodalité impliquant plusieurs modalités d'interaction, il convient aussi d'étudier les relations que peuvent entretenir les modalités d'interaction dans la chaîne d'abstraction des données captées par un dispositif jusqu'aux tâches. Ainsi, nous présentons des modèles qui organisent dans des cadres cohérents les éléments de conception liés aux relations et compositions de modalités.

3.2.2 Modèles de dispositifs physiques

Taxonomie de Buxton

La taxonomie de Buxton [Buxton, 1983] isole les caractéristiques principales des dispositifs physiques en entrée. Buxton précise que sa taxonomie ne permet de

décrire que les dispositifs continus contrôlés à la main[3]. Ainsi les dispositifs à fonctionnement discret comme des boutons ne sont pas considérés. De plus seuls les dispositifs actionnés par la main sont décrits dans ce modèle : ainsi un microphone ou une caméra ne sont pas traités.

La taxonomie est construite sur deux dimensions (Figure 3.2). La première dimension décrit la propriété captée par le dispositif. Buxton identifie trois propriétés pouvant être captées par un dispositif : le mouvement, la position et la pression. Les propriétés de mouvement et position sont sous-divisées en deux sous-propriétés qui décrivent le fonctionnement du dispositif : mécanique ou tactile (la propriété de pression était forcément mécanique).

Le deuxième axe décrit le nombre de dimensions du dispositif en entrée. Des dispositifs de 1, 2 et 3 dimensions sont distingués. Buxton illustre sa taxonomie avec des dispositifs existants. Par exemple, la souris est un dispositif à 2 dimensions qui capte le mouvement de façon mécanique (Figure 3.2).

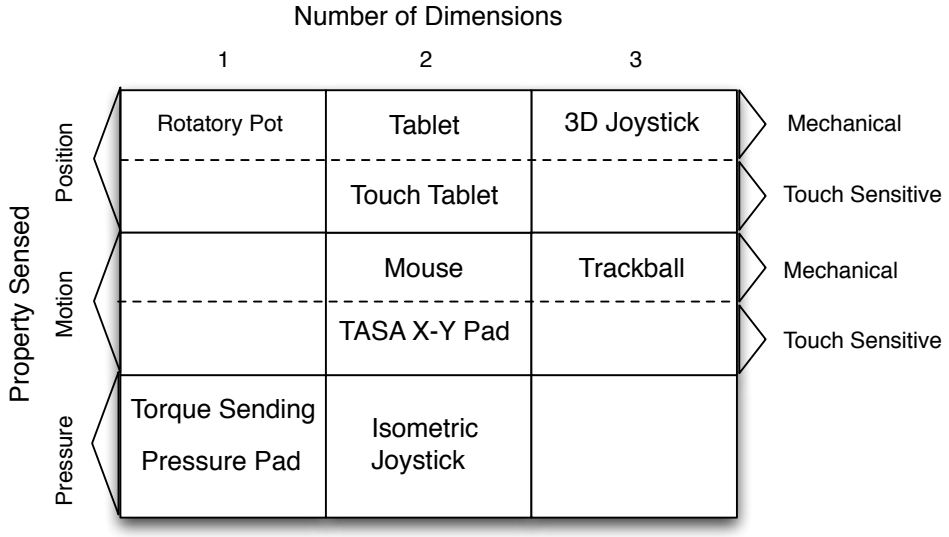


FIGURE 3.2 – Taxonomie de Buxton. Illustration extraite de [Buxton, 1983].

Espace de conception de Card, Mackinlay et Robertson

L'espace de conception de Card, Mackinlay et Robertson [Card *et al.*, 1991] étend les concepts de Buxton. La description d'un dispositif est basée sur un vocabulaire de mouvement et des opérateurs de composition.

Le *vocabulaire de mouvement* permet de décrire le dispositif par un sextuplet <M, In, Out, R, S, W> :

- **M** est un opérateur de manipulation, basé sur les concepts de Buxton. Ainsi, l'opérateur de manipulation est une combinaison des propriétés physiques <linéaire/rotation>, <absolu/relatif> et <position/force>.

- **In** est le domaine des valeurs en entrée possibles.

[3] «To begin with, the tableau deals only with continuous hand-controlled devices.»[Buxton, 1983]

- **Out** est le domaine des valeurs en sortie possibles.
- **R** est la fonction de résolution qui définit la correspondance entre les domaines In et Out.
- **S** représente l'état actuel du dispositif.
- **W** représente le fonctionnement du dispositif.

La Figure 3.3 illustre la description d'un bouton de volume d'une radio selon ces six propriétés.

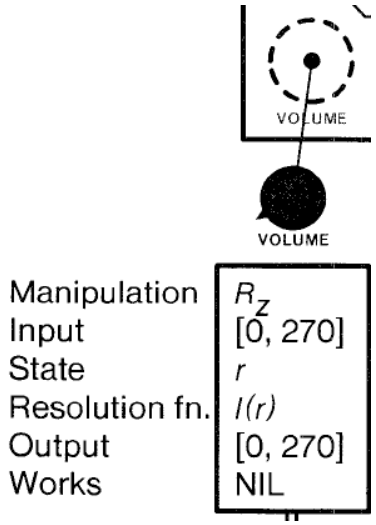


FIGURE 3.3 – Analyse des propriétés du bouton de volume d'une radio selon la taxonomie de Card. Illustration extraite de [Card *et al.*, 1991].

Les *opérateurs de composition* permettent ensuite de décrire les relations entre différents capteurs physiques. Trois types de composition sont identifiés : composition de fusion, composition de disposition et composition de connexion[4]. Pour illustrer ces trois opérateurs, et comme pour la taxonomie de Buxton, considérons l'example d'une souris. La Figure 3.4 illustre la décomposition de la souris selon ces trois opérateurs. La fusion décrit la combinaison de deux capteurs afin de former une seule sortie combinée : dans le cas de la souris, les deux axes de défilement unidimensionnel X et Y sont fusionnés afin de former une sortie (X,Y). La disposition décrit la mise en place de plusieurs capteurs dans un même espace physique : dans le cas de la souris, les boutons et les deux axes XY sont regroupés pour former un seul objet physique (la souris). La connexion a lieu lorsque la sortie d'un dispositif est liée à l'entrée d'un autre dispositif : dans le cas de la souris, la sortie (X,Y) est connectée à l'écran afin de fournir les coordonnées dans le repère de ce dernier.

La conjugaison du vocabulaire de mouvement avec les opérateurs de composition forme un espace de conception pour les dispositifs en entrée, illustré à la Figure 3.5. Les cercles représentent une propriété captée par un dispositif. Les traits continus représentent une composition de fusion, les traits discontinus une composition de disposition et les traits fléchés une composition de connexion. Nous observons

[4] «Merger, layout and connect composition» [Card *et al.*, 1991]

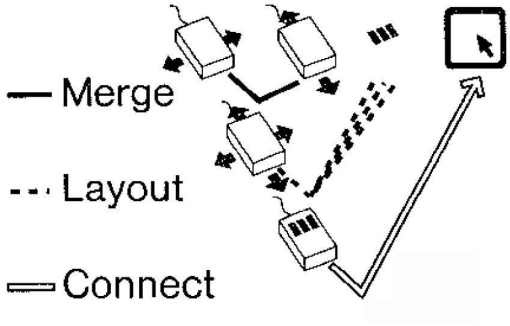


FIGURE 3.4 – Décomposition de la souris selon les opérateurs de composition de Card. Illustration extraite de [Card *et al.*, 1991].

ainsi que la souris est représenté par deux cercles reliés par un trait continu (les XY) et par un troisième cercle avec le chiffre 3 connecté par une ligne discontinue (les trois boutons de la souris). La connexion avec l'écran n'est pas représentée à la Figure 3.5. Les cercles Volume et Sélection ainsi que la flèche marquée Station représentent les données de la radio présentée à la Figure 3.3.

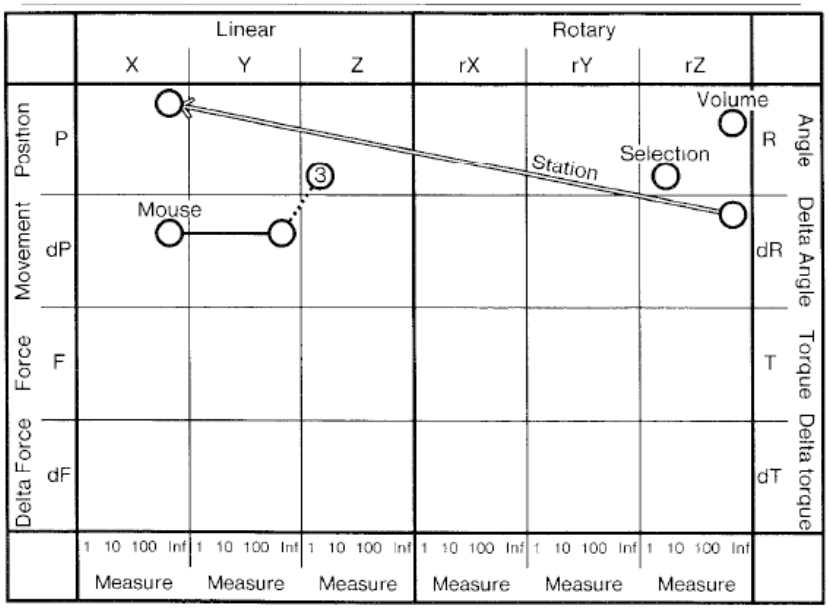


FIGURE 3.5 – Espace de conception des dispositifs en entrée de Card. Illustration extraite de [Card *et al.*, 1991].

Cet espace de conception permet donc de décrire les dispositifs d'interaction en entrée de façon claire et systématique à l'aide d'une notation simple. L'avantage de cet espace est de permettre la prospection de nouveaux dispositifs et l'évaluation du pouvoir d'expression des dispositifs au regard de la tâche donnée.

3.2.3 Modèles de dispositifs logiques

Tandis que les deux modèles précédents décrivent un dispositif à partir des actions captées de l'utilisateur, les modèles de dispositifs logiques adoptent un point de vue complémentaire en étudiant les tâches élémentaires réalisables avec les dispositifs (Figure 3.6).

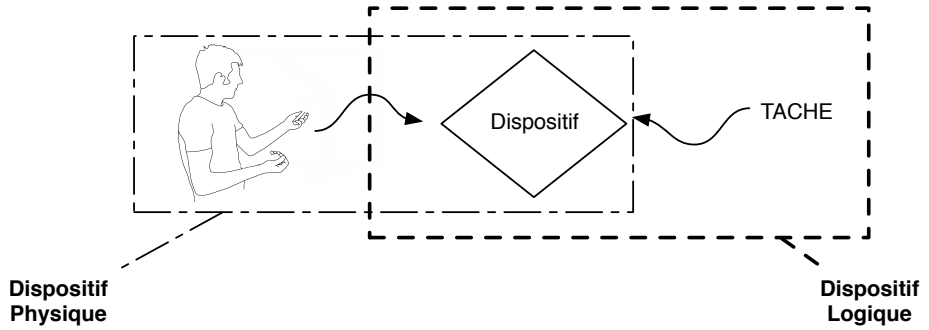


FIGURE 3.6 – Dispositif logique et dispositif physique.

Modèles de GKS et PHIGS

À la fin des années 70, les applications graphiques n'étaient pas portables d'une machine à une autre. Motivée par ce constat, le International Standard Organisation (ISO) a proposé le Graphical Kernel System (GKS) [Eckert *et al.*, 1979], un modèle de standard graphique qui traite les différents périphériques.

Le modèle GKS introduit la notion de dispositif logique : un dispositif logique est composé d'un dispositif physique et d'une tâche graphique élémentaire. Six classes de dispositifs logiques en entrée sont définis par GKS, permettant de normaliser le contenu d'une page graphique :

- *Locator* : position de type (x, y).
- *Stroke* : série de points de type $(x, y)^n$.
- *Valuator* : valeur réelle ou entière.
- *Choice* : entier représentant une sélection parmi un ensemble d'alternatives.
- *Pick* : identifiant d'un objet présent à l'écran.
- *String* : chaîne de caractères.

Ainsi, chaque classe en entrée peut être concrétisée en utilisant différents dispositifs physiques. Par exemple, le Locator peut être implémenté en utilisant une souris ou une tablette. Le but de ce modèle est de garantir la portabilité des applications et leur indépendance vis-à-vis des dispositifs physiques. Ce modèle a donné lieu à deux extensions 3D : GKS-3D [Puk et McConnell, 1986] et PHIGS [Hewitt, 1984], puis PHIGS+ [Van Dam, 1988] (Figure 3.7).

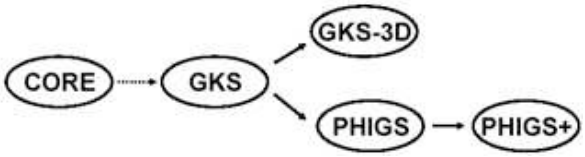


FIGURE 3.7 – Évolution des standards graphiques.

Tâches d'interaction de Foley

Foley propose une approche centrée sur la tâche d'interaction graphique [Foley *et al.*, 1984]. Afin de classer un dispositif, Foley propose d'utiliser la tâche d'interaction que ce dispositif permet de réaliser. Les tâches d'interaction de Foley sont au nombre de six :

- **Selection** : sélection d'un objet ;
- **Position** : positionnement d'un objet sur 1, 2 ou 3 dimensions ;
- **Orient** : orientation d'un objet sur 1, 2 ou 3 dimensions ;
- **Ink** : dessin d'une ligne ;
- **Text** : saisie d'un texte ;
- **Value** : spécification d'une valeur scalaire.

Les tâches d'interaction de Foley permettent de guider le choix des dispositifs en fonction des tâches à réaliser. Ainsi, elles permettent de définir la relation entre les techniques d'interaction, la tâche d'interaction réalisée et le dispositif physique utilisé (Figure 3.8). Les techniques d'interaction, à différence des tâches d'interaction, sont dépendantes des propriétés physiques des dispositifs. Le tableau illustré à la Figure 3.8 met en relation la tâche d'interaction Selection, techniques d'interaction et dispositifs physiques. Par exemple, la tâche Sélection peut se réaliser avec la technique de sélection directe ("Direct pick" sur la Figure 3.8) en utilisant un stylet ou une tablette tactile.

Tandis que certaines tâches d'interaction de Foley sont en correspondance directe avec les classes de GKS, comme Select avec Pick, ou Position avec Locator (Figure 3.9), d'autres tâches sont de plus haut niveau d'abstraction, comme la tâche Orient qui serait décrite par une suite de tâches en GKS (Pick Stroke)*.

Paradigme des dispositifs en cascade

Le paradigme des dispositifs en cascade, introduit par Pierre Dragicevic [Dragicevic, 2004] utilise la notion de connexion logicielle pour étendre la connexion physique des dispositifs en entrée. Ce modèle se construit autour de deux entités : les dispositifs physiques d'entrée et l'application. Un système interactif est modélisé comme une connexion de dispositifs physiques à l'application, dans ce qui est appelé la métaphore de la connexion logicielle (Figure 3.10). Les

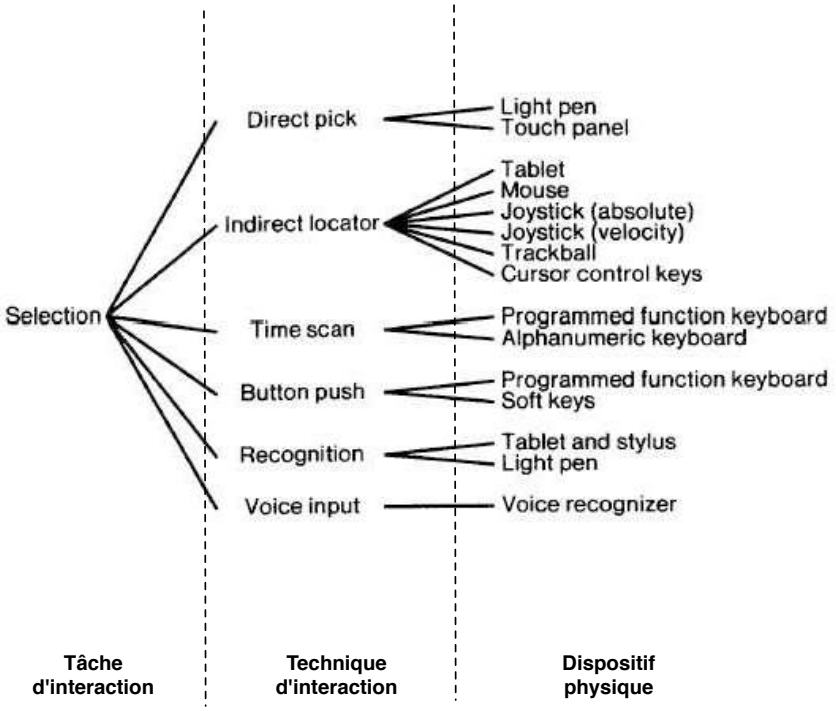


FIGURE 3.8 – Relation entre la tâche d'interaction Selection, certaines techniques d'interaction et les dispositifs physiques. Illustration extraite de [Foley *et al.*, 1984].

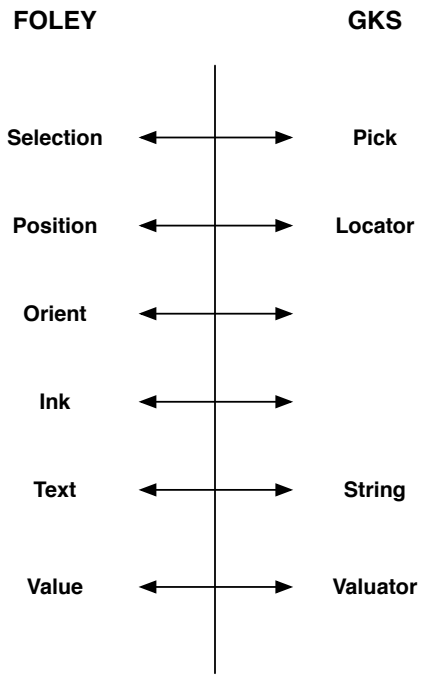


FIGURE 3.9 – Comparaison entre les tâches d'interaction de Foley et les classes GKS. Les tâches Orient et Ink, de plus haut niveau, n'ont pas d'équivalent sous GKS.

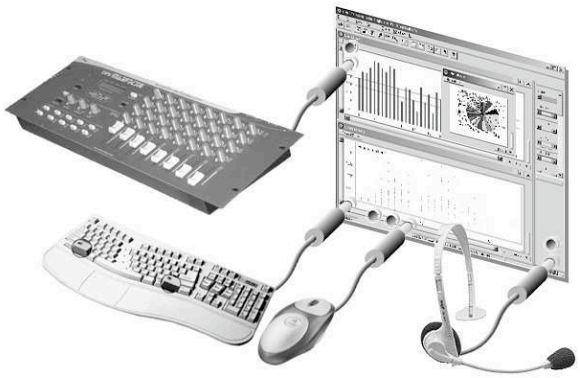


FIGURE 3.10 – Métaphore de la connexion logicielle. Illustration extraite de [Dragicevic, 2004].

points d'entrée d'une application correspondent aux données fournies par l'utilisateur. Ces points d'entrée peuvent être décrits par exemple par des tâches élémentaires comme celles de GKS ou Foley. Chaque point d'entrée définit une destination possible pour un dispositif physique d'entrée.

Afin de connecter ces dispositifs aux points d'entrée, il peut s'avérer nécessaire d'effectuer des opérations sur les données fournies par les dispositifs. Des adaptateurs sont utilisés à cette fin, lorsque les données émises par le dispositif sont incompatibles avec celles attendues par les points d'entrée.

Une cascade de dispositifs est une sérialisation d'adaptateurs connectés à un dispositif d'entrée. La cascade représente la série de traitement sur les données en entrée. La Figure 3.11 illustre ce concept à travers le mécanisme de gestion du clavier.

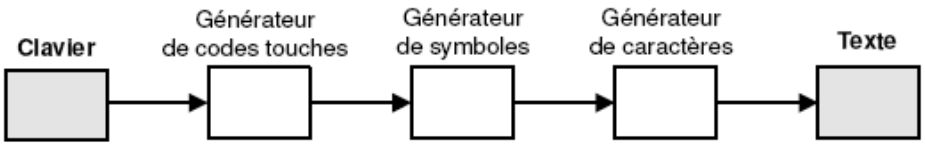


FIGURE 3.11 – Cascade de dispositifs entre le clavier et un éditeur de texte. Illustration extraite de [Dragicevic, 2004].

Par rapport aux classes de GKS et aux tâches de Foley, l'apport de ce paradigme réside dans la modélisation de la chaîne d'abstraction des données fournies par le dispositif jusqu'à l'application. Au sein de cette modélisation en une suite de dispositifs logiques, nous pouvons retrouver les classes GKS ou les tâches de Foley.

Ce paradigme a été concrétisé dans le modèle des configurations d'entrée, que nous présentons à la section 3.4.2.

3.2.4 Modèles d'interaction

Dans cette section, nous étudions un paradigme d'interaction, le paradigme de la manipulation directe, et un modèle d'interaction, le modèle de l'interaction instru-

mentale.

Manipulation directe

La manipulation directe [Shneiderman, 1987] [Shneiderman, 1997] désigne le fait de manipuler des objets graphiques sur l'écran grâce à un dispositif de pointage. Elle a été démontrée pour la première fois par Ivan Sutherland dans Sketchpad (cf. chapitre 2) en 1963 [Myers, 1998] (Figure 3.12).

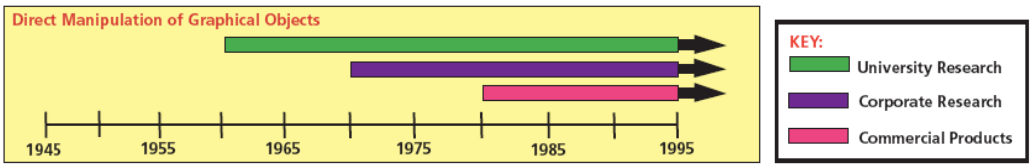


FIGURE 3.12 – Histoire de la manipulation directe d'objets graphiques. Illustration extraite de [Myers, 1998].

Le terme a été introduit par Ben Shneiderman en 1982. Shneiderman identifie les différents composants et les apports de la manipulation directe. Ce paradigme d'interaction peut se résumer en trois points principaux :

- La **représentation continue** des objets graphiques ;
- L'utilisation d'**actions physiques** pour manipuler les objets graphiques au lieu de l'utilisation d'une syntaxe complexe ;
- La **rapidité et réversibilité** des actions physiques sur les objets graphiques concernés.

Nous retrouvons aujourd'hui des exemples de manipulation directe dans la plupart des sytèmes interactifs. Par exemple, la Figure 3.13 illustre l'utilisation de la manipulation directe pour faire défiler des listes sur le iPhone [Apple, 2010]. L'utilisateur utilise le geste avec le doigt pour faire défiler verticalement la liste, qui réagit aux mouvements du doigt.

FIGURE 3.13 – Défilement de listes par manipulation directe sur l'iPhone. Illustration extraite de [Apple, 2010].

La manipulation directe, en réduisant la distance entre l'image mentale de la tâche à réaliser et la tâche effectivement accomplie, permet de rendre les interfaces plus utilisables. D'autres avantages de ce type d'interfaces sont l'apprentissage rapide, la réduction des messages d'erreur et la possibilité de défaire une action.

Interaction instrumentale

Tandis que la manipulation directe est un paradigme d'interaction, l'interaction instrumentale [Beaudouin-Lafon, 1997] est un modèle d'interaction. Il a été introduit par Michel Beaudoin-Lafon pour les applications graphiques basées sur la manipulation directe. Ce modèle définit un espace de conception pour les interfaces graphiques et facilite la définition de nouvelles techniques d'interaction. Pour cela, le modèle propose la notion d'instrument d'interaction.

Un instrument d'interaction est un objet qui joue le rôle de médiateur entre les actions de l'utilisateur et les objets de l'application. Un instrument est composé d'une partie physique et d'une partie logique (Figure 3.14). L'interaction se fait donc à deux niveaux : entre l'utilisateur et l'instrument physique, et entre l'instrument logique et l'objet de l'application. La Figure 3.14 illustre un exemple : l'utilisateur manipule un instrument composé d'une partie physique, la souris, et d'une partie logique, la barre de défilement. Cet instrument sert à faire défiler un objet édité, par exemple un texte.

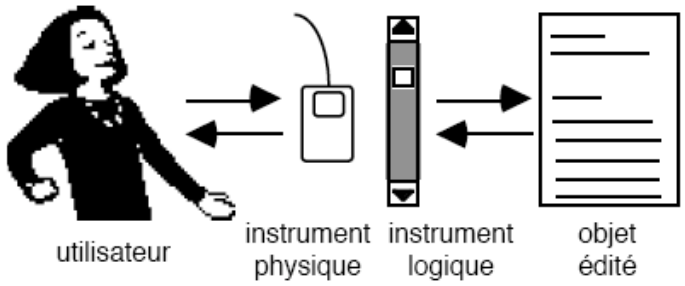


FIGURE 3.14 – Instrument selon le modèle d'interaction instrumentale. Illustration extraite de [Beaudouin-Lafon, 1997].

Le retour d'information vers l'utilisateur se fait à trois niveaux : retour de l'action sur l'instrument physique, retour visuel via la représentation de l'instrument logique et le retour de l'effet de l'interaction sur l'objet édité. Dans l'exemple de la Figure 3.14, le retour de l'information sur la souris est proprioceptif[5] via la position de la souris, le retour d'information de l'instrument logique est la position de l'ascenseur dans la barre de défilement et le retour d'information de l'objet édité est le défilement du document.

Le modèle de l'interaction instrumentale étend les principes du paradigme de la manipulation directe en introduisant la notion d'instrument d'interaction qui permet de guider les concepteurs lors de la conception d'interfaces graphiques. Nous positionnons les modèles de dispositifs (physiques ou logiques) des sections 3.2.2 et 3.2.3 précédents au niveau de l'instrument physique.

[5]Propre aux muscles, ligaments, os. [CNRTL-CNRS, 2009]

3.2.5 Modèles pour la multimodalité

Les modèles présentés précédemment considèrent, soit un dispositif, soit toute l'interaction, mais ne traitent pas de la multiplicité des modalités, objet de ce paragraphe. Nous présentons maintenant des modèles pour la multimodalité, commençant par des dimensions de conception générales, (espace MSM), puis focalisant sur les relations entre plusieurs modalités (CARE et TYCOON et l'espace de composition de Vernier).

L'espace MSM

L'espace Multi-Sensori-Moteur (MSM) [Nigay et Coutaz, 1996] identifie six dimensions pour caractériser un système interactif multimodal (Figure 3.15) :

- **Fusion/Fission** : Décrit la combinaison de plusieurs unités d'information pour former de nouvelles unités. La fission correspond au processus inverse.
- **Sens** : Spécifie s'il s'agit de dispositifs en entrée ou de dispositifs en sortie.
- **Nombre de dispositifs de même sens** : Décrit le nombre de dispositifs en entrée ou en sortie.
- **Niveau d'abstraction** : Exprime le degré de transformation appliquée aux informations reçues ou émises par les dispositifs.
- **Contexte** : Comprend un ensemble de variables d'état utilisées par les processus interne pour contrôler l'acquisition ou la restitution.
- **Parallélisme** : Décrit le parallélisme perceptible à l'interface selon trois niveaux de granularité (action physique, tâche et grappe de tâches).

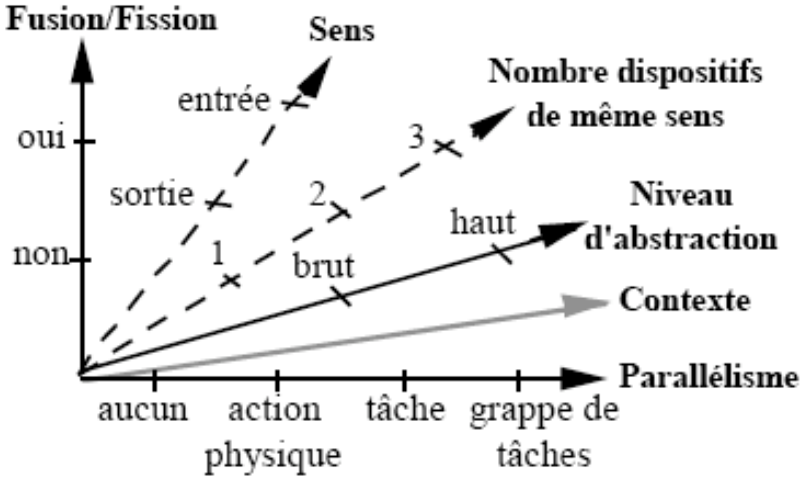


FIGURE 3.15 – Le référentiel MSM. Illustration extraite de [Nigay et Coutaz, 1996].

Nous nous intéressons ici à l'utilisation de MSM pour la caractérisation de l'interaction en entrée. Le couplage de la dimension de Fusion/Fission avec celle de

Parallélisme permet d'identifier différents cas d'usage en entrée des modalités illustrées à la Figure 3.16.

La ligne du bas de la Figure 3.16, intitulé "Ni Fusion ni fission", représente l'absence de fusion et de fission. La case 5 représente l'usage exclusif d'une modalité, c'est-à-dire lorsque l'utilisateur emploie au plus une modalité à la fois. L'exemple de la manipulation directe de la Figure 3.13 illustre l'usage exclusif du geste avec le doigt pour faire défiler une liste. La case 6 représente l'usage concurrent de deux modalités, qui ne donne lieu ni à une fusion ni à une fission. Par exemple, dans le jeu multimodal Crash Kart, présenté au chapitre 2 section 2.4, le joueur peut incliner le iPhone pour diriger la voiture et utiliser l'écran pour activer des options du jeu. Les deux modalités correspondent à des tâches indépendantes menées concurremment sans redondance. L'usage concurrent avec redondance a lieu lorsque les deux modalités utilisées en parallèle couvrent le même sens. Par exemple, le joueur incline le iPhone pour diriger la voiture et utilise un bouton à l'écran en même temps pour diriger la voiture. Le système peut alors éliminer le superflu ou, en milieu bruité, tirer partie comme information de complémentarité.

La ligne du centre de la Figure 3.16, intitulé "Fusion", comprend deux cas correspondant à une production séquentielle (case 3) ou simultanée (case 4) de plusieurs informations donnant lieu à une fusion pour faire sens. L'interaction multimodale du système *Put-That-There*, présenté au chapitre 2 section 2.4, correspond à ce dernier cas d'usage de deux modalités (case 4) : l'utilisateur pointe du doigt sur un objet (modalité 1) puis prononce "put that" (modalité 2) afin de sélectionner l'objet. L'utilisateur va ensuite répéter l'usage combiné en pointant la destination et en prononçant "there". L'information de pointage est fusionnée avec celle issue de la reconnaissance vocale afin d'obtenir une tâche complète (identification puis déplacement d'un objet).

La ligne du haut de la Figure 3.16 représente l'usage exclusif d'une modalité, comme la case 5, mais cette fois donnant lieu à une fission d'information. Le résultat de cette opération de fission conduit à plusieurs tâches distinctes. Par exemple, la commande vocale "dessine un cercle dans une nouvelle fenêtre" fait référence à deux tâches : la création d'une figure géométrique (le cercle) et la création d'une nouvelle fenêtre. L'énoncé selon une seule modalité doit donc être décomposé en deux tâches distinctes pour l'application.

Des exemples ci-dessus de différents cas d'usage des modalités en entrée, nous remarquons l'absence de choix lorsqu'une modalité est utilisée de façon exclusive (case 5 de la Figure 3.16) et l'émergence de sens (tâche pour l'application) par fusion/fission. Par exemple, l'usage complémentaire de plusieurs modalités de la ligne 2 de la Figure 3.16. Ces différents cas d'usage des modalités se retrouvent dans les propriétés CARE et l'espace Tycoon présentés dans les deux paragraphes suivants.

CARE

Les propriétés CARE (Complémentarité, Assignation, Redondance et Equivalence) [Coutaz et Nigay, 1994] définissent des relations entre plusieurs modalités d'interaction. Rappelons qu'une modalité d'interaction est un couple <dispositif physique, langage d'interaction> (cf. section 2.3 du chapitre 2). Un dispositif physique est un transducteur de propriétés physiques, alors qu'un langage d'interaction est

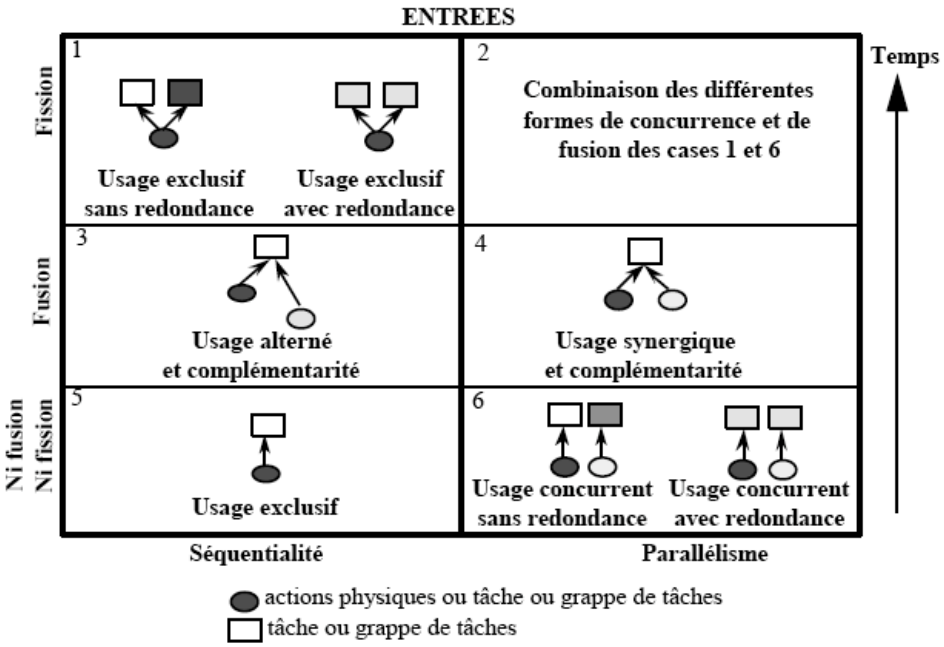


FIGURE 3.16 – Différents cas d'usage en entrée de modalités dans le référentiel MSM. Illustration extraite de [Nigay et Coutaz, 1996].

un système conventionnel qui assure une fonction de communication entre l'utilisateur et le système informatique. Les propriétés CARE décrivent les relations possibles entre ces entités et les tâches de l'application, comme schématisé à la Figure 3.17.

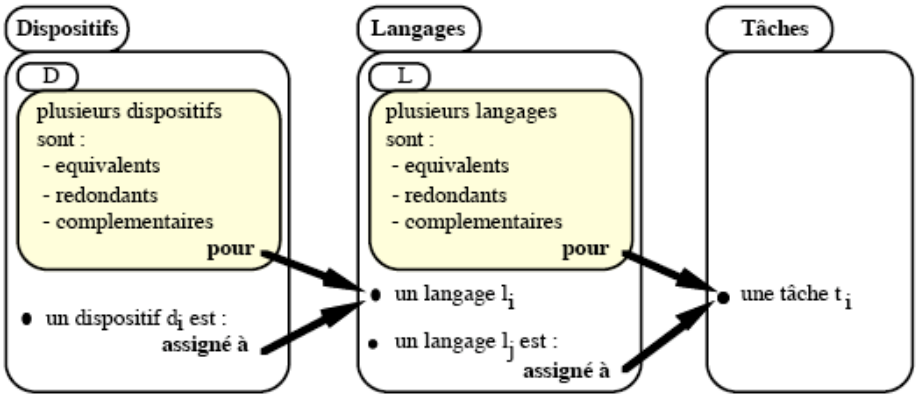


FIGURE 3.17 – Les relations CARE entre dispositifs, langages d'interaction et tâches. Illustration extraite de [Coutaz et Nigay, 1994].

La définition des propriétés CARE (Figure 3.18) repose sur les notions d'État, But, Modalité et Relation temporelle :

- **État**. Noté s, l'état est un ensemble de propriétés qui peuvent être mesurées à un instant donné.

- **But**. Noté g, le but correspond à l'état que l'agent veut atteindre. Un agent est une entité (utilisateur, système ou composant) capable de réaliser des actions.
- **Modalité**. Notée m, une modalité est composée par un dispositif et un langage d'interaction qui permet d'atteindre un but.
- **Relation temporelle**. Notée TR, la relation temporelle caractérise l'utilisation temporelle d'un ensemble de modalités. L'utilisation des modalités peut avoir lieu simultanément ou à l'intérieur d'une fenêtre temporelle, notée TW.

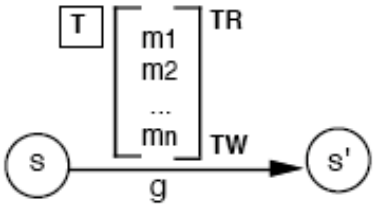


FIGURE 3.18 – Notation pour exprimer les propriétés CARE.

Les propriétés CARE ainsi définies permettent de décrire des relations entre les dispositifs et les langages d'une part, et les langages et les tâches d'autre part. L'équivalence et l'assignation caractérisent la portée des choix en matière de langage et de dispositif. La redondance et la complémentarité qualifient la combinaison des choix. Par exemple, deux dispositifs sont équivalents pour un langage donné si le langage peut être utilisé au moyen de l'un quelconque des dispositifs. La souris et la tablette sont deux dispositifs équivalents pour le langage *Pointage en 2D*. Deux langages sont équivalents pour une tâche donnée si cette tâche peut être réalisée au moyen de l'un quelconque des langages. Par exemple le système de manipulation d'objets 3D présenté au chapitre 2 section 2.4 (Figure 2.7) illustre l'usage équivalent du langage *geste* et du langage *commande vocale* pour la tâche de manipulation d'un objet en 3D.

Les propriétés CARE fournissent un cadre conceptuel pour définir les relations entre modalités en entrée. Combinées à d'autres modèles, comme MSM ou Tycoon que nous présentons dans le paragraphe suivant, les propriétés CARE ont fortement contribué à la conception d'interfaces multimodales [Serrano et Nigay, 2009].

Tycoon

Tycoon [Martin, 1999] est une typologie qui permet la description, l'évaluation et la spécification des coopérations entre agents humains et logiciels. Une coopération est un échange d'information réalisé afin d'atteindre un but commun. Tycoon définit six types de coopération possibles entre agents : equivalence, transfert, spécialisation, redondance, complémentarité et concurrence (Figure 3.19).

Tycoon enrichit CARE en considérant le transfert. Une coopération de type Transfert entre deux agents est définie par une fonction de correspondance entre la sortie du premier agent et l'entrée du deuxième agent. Lorsque plusieurs agents coopèrent par transfert, cela signifie que l'information produite par un agent est utilisée par un autre agent.

	Improving Recognition	Fast interaction	Adaptation to changes	...
Equivalence				
Transfer				
Specialization				
Redundancy				
Complementarity				
Concurrency				

FIGURE 3.19 – Typologie proposée par Tycoon. Illustration extraite de [Martin, 1999].

De plus l'espace Tycoon identifie un ensemble de buts de coopération (partie supérieure du tableau à la Figure 3.19) qui décrivent les raisons pour lesquelles les agents échangent de l'information et coopèrent. Par exemple une coopération de type Equivalence peut avoir pour but l'adaptation aux changements (3ème colonne à la Figure 3.19).

Espace de composition

L'importance de la temporalité (comme l'axe parallélisme dans l'espace MSM et l'opérateur TR dans CARE) nous amène à considérer les différents types de relations temporelles possibles dans l'usage de plusieurs modalités. Les propriétés d'Allen [Allen, 1983] décrivent les relations existantes entre deux intervalles temporels t et s.

Allen identifie 5 relations temporelles entre t et s : t < s, t = s, t overlaps s, t meets s et t during s. Ces 5 relations ont été illustrées graphiquement par Frédéric Vernier [Vernier, 2001] (Figure 3.20)

Ces relations d'Allen ont été reprises par Frédéric Vernier pour définir un espace de composition des modalités. Les cinq relations d'Allen définissent des schémas de composition (colonnes de la Figure 3.21). De plus les relations temporelles d'Allen sont étendus à d'autres types de relations entre modalités définissant ainsi des aspects de composition (lignes de la Figure 3.21).

Les cinq aspects de composition considèrent les relations spatio-temporelles lors de l'usage de plusieurs modalités mais aussi le niveaux dispositif, langage et tâche dans l'interaction multimodale :

- **Temporelle** : cet aspect caractérise les enchaînements possibles de modalités.
- **Spatiale** : cet aspect reflète la possibilité de placer les modalités à des endroits spécifiques de l'environnement de l'utilisateur.

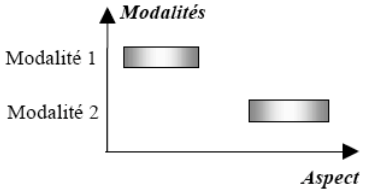


Figure 12a : Composition de modalités éloignées.

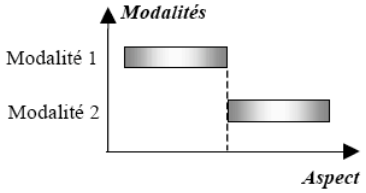
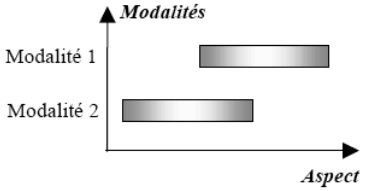


Figure 12b : Composition de modalités ave un point de contact.



igure 12c : Composition de modalités avec une intersection non vide.

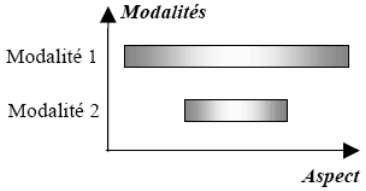


Figure 12d : Composition de modalités don l'une est plus étendue et englobe l'autre.

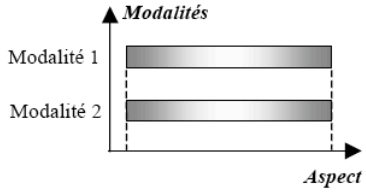


FIGURE 3.20 – Cinq relations d'Allen. Illustration extraite de [Vernier, 2001].

- **Articulatoire** : plus adapté à l'interaction en sortie, cet aspect décrit la composition des flux d'informations des différents dispositifs.
- **Syntaxique** : cet aspect considère les relations entre les langages d'interaction utilisés.
- **Sémantique** : cet aspect décrit la relation entre la sémantique des informations véhiculées par les modalités pour une tâche donnée.

Cet espace de composition unifie et complète MSM, CARE et Tycoon en identifiant vingt cinq relations entre modalités.

3.2.6 Conclusion

Pour analyser les modèles de conception de l'interaction multimodale nous avons adopté une approche ascendante, en considérant d'abord les modèles de dispositifs physiques et logiques (les éléments de base) puis les modèles d'interaction incluant les dispositifs. Nous avons enfin considéré la multiplicité des modalités au sein de l'interaction (Figure 3.22).

Nous considérons maintenant les modèles de conception logicielle pour décrire le logiciel qui réalise les interactions décrites par les modèles de conception de l'interaction de cette section 3.2.

Schémas de composition

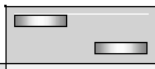
Aspects de composition \ Composition					
Temporelle	Anachronique	Séquentielle	Concomitante	Coïncidente	Parallèle / Simultanée
Spatiale	Disjointe	Adjacente	Intersectée	Imbriquée	Recouvrance
Articulatoire	Indépendance	Fissionnée	Fissionnée + Dupliquée	Partiellement Dupliquée	Dupliquée
Syntaxique	Différente	Complétion	Divergence	Extension	Jumelage
Sémantique	Concurrente	Complé-mentaire	Complémentaire + Redondante	Partiellement Redondante	Totalement Redondante

FIGURE 3.21 – Application des schémas de composition aux aspects de composition. Illustration extraite de [Vernier, 2001].

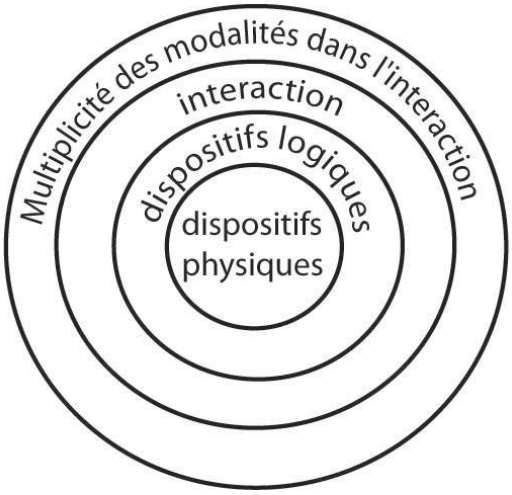


FIGURE 3.22 – Synthèse des modèles présentés à la section 3.2 du chapitre.

3.3 Modèles de conception logicielle

Dans cette section, nous présentons d'abord les modèles d'architecture logicielle pour l'interaction homme-machine. Ces modèles définissent des décompositions fonctionnelles des systèmes interactifs. Ensuite nous introduisons les modèles de conception logicielle pour la réutilisation. Ces derniers permettent d'implémenter des architectures et de créer des systèmes modulaires et aux fonctionnalités réutilisables. Puis nous présentons les modèles de programmation pour l'interaction et finalement les modèles de programmation pour non-programmeurs.

3.3.1 Modèles de programmation de l'interaction homme-machine

Introduction

Afin de décrire les différents modèles de programmation de l'interaction, nous adoptons la typologie proposée par Appert [Appert *et al.*, 2009]. Cette typologie considère trois modèles de programmation principaux : les approches à fonctions de rappel, les modèles à automates et les modèles à flot de données.

Bien que le modèle des fonctions de rappel est orthogonal aux deux autres (en effet, un modèle à automates peut être programmé avec des fonctions de rappel), il est pertinent de l'analyser séparemment car de nombreuses boîtes à outils pour l'interaction sont construites selon ce modèle. Dans cette section, nous présentons ces trois modèles et analysons leur pertinence pour la programmation d'interfaces multimodales.

Approche à fonctions de rappel

L'approche à base de fonctions de rappel[6] a été une des premières approches à être appliquée à la programmation d'interfaces graphiques. Une fonction de rappel est une fonction qui est passée en argument à une autre fonction. Cette dernière peut alors faire appel à cette fonction de rappel à n'importe quel moment. Ce mécanisme est souvent utilisé dans les boîtes à outils, comme Java Swing ou Gtk, pour programmer des écouteurs d'événements : les différents objets graphiques de l'interface (boutons, barre de défilement, etc..) sont assignés à des fonctions de rappel. Lorsque l'utilisateur agit sur un objet graphique, celui-ci fait appel à la fonction de rappel correspondante, qui va déclencher la réaction du système (Figure 3.23).

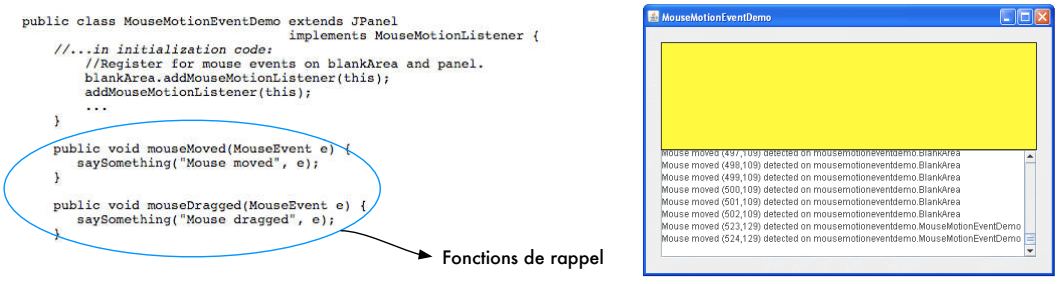


FIGURE 3.23 – Fonctions de rappel assignées à différentes actions de la souris sur un objet graphique (ici un panel) dans Java Swing .

Dans les boîtes à outils graphiques, cela implique d'avoir une fonction de rappel par objet graphique et par type d'événements utilisateur. Par exemple à la Figure 3.23, deux fonctions de rappel sont définies pour un objet graphique de type zone de dessin (JPanel) : l'une est associée à un mouvement de la souris, l'autre à un mouvement de la souris alors que le bouton de la souris est maintenu appuyé.

Les systèmes interactifs construits avec ce mécanisme sont donc souvent difficiles à modifier et maintenir en raison du grand nombre de fonctions de rappel existantes [Myers, 1991].

[6] Callbacks

Modèles à automates

Le modèle à automates déterministes finis (DFA) est à la base des différentes approches à états et transitions. Le modèle général repose sur des états et des transitions qui permettent de passer d'un état à un autre. Il existe différentes approches, telles que les machines à états, les machines à états hiérarchiques et les réseaux de Petri.

Dans un système interactif programmé avec une machine à états [Myers *et al.*, 1995], des actions utilisateur peuvent être associées aux transitions. Les machines à états ont été utilisées pour modéliser des interactions multimodales en entrée [Bourguet, 2003].

Par exemple la Figure 3.24 montre une interaction multimodale modélisée avec une machine à états. Les états sont représentés par des cercles et les actions de l'utilisateur sont associées aux transitions. La souris et des commandes vocales sont utilisées pour réaliser la tâche de déplacement d'un objet graphique. Ainsi, les transitions entre les états du système sont associés à des interactions en entrée comme la commande vocale "move" ou l'appuie d'un bouton de la souris. La machine à états permet de modéliser facilement l'enchaînement des actions de l'utilisateur. Nous observons dans cet exemple que le nombre d'états dans ce type d'automates tend à augmenter assez rapidement, surtout lorsque nous utilisons plusieurs modalités.

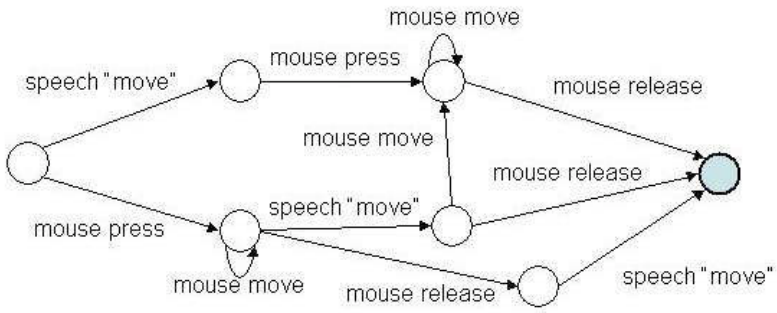


FIGURE 3.24 – Machine à états modélisant une interaction multimodale avec la souris et la parole. Illustration extraite de [Bourguet, 2003].

Les limitations des machines à états, en particulier en ce qui concerne l'explosion de la complexité de l'automate lorsque le nombre d'états augmente, a conduit aux machines à états hiérarchiques. Dans ce type de machines, chaque état peut être composé lui-même d'une machine. On parle alors de sous-états de la machine. La Figure 3.25 illustre une machine à états implémentant le comportement d'un bouton (au centre sur la Figure 3.25) et la machine à états hiérarchique raffinant le comportement du bouton précédent (à droite sur la Figure 3.25). La boîte à outils hsmTk [Blanch et Beaudouin-Lafon, 2006] permet d'implémenter des interfaces en utilisant ce type de machines à états.

Les réseaux de Petri [Petri, 1962], généralisation des machines à états, constituent une technique de description formelle de la dynamique des systèmes. Les

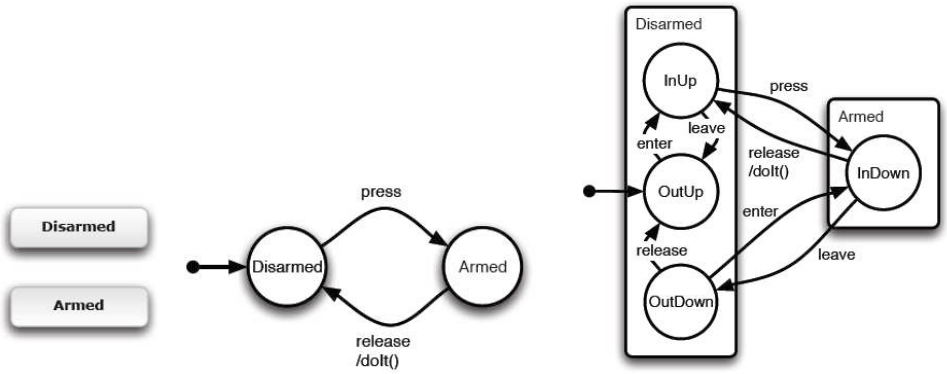


FIGURE 3.25 – Bouton possédant deux états (Disarmed-Armed, à gauche), machine à état implémentant le comportement du bouton (au centre) et machine à états hiérarchiques raffinant le comportement du bouton précédent (à droite). Illustration extraite de [Blanch, 2005].

réseaux de Petri considèrent quatre types d'éléments : les places, les transitions, les arcs et les jetons. Les places sont les variables d'état du système, les transitions sont des opérateurs de changement d'état, les jetons d'une place caractérisent sa valeur et les arcs permettent de relier les places et les transitions.

Les réseaux de Petri ont été exploités pour la spécification, conception et validation formelle de systèmes interactifs [Palanque *et al.*, 1993]. Pour l'interaction multimodale, Hinckley a utilisé des réseaux de Petri pour modéliser une technique d'interaction bi-manuelle [Hinckley *et al.*, 1998]. La Figure 3.26 illustre un réseau de Petri complet décrivant une technique d'interaction bi-manuelle utilisant deux surfaces tactiles (un Palm et un Touchmouse, une souris modifiée afin de détecter le toucher) comme deux modalités d'interaction. Les transitions sont symbolisées par un label faisant référence à l'action de l'utilisateur (toucher sur le Palm, *PalmTouch*, ou sur le Touchmouse, *Touch*). Par exemple, la place T0p contient un jeton tant qu'il n'y a pas de toucher sur le Palm. Dès que le contact a lieu, le jeton est retiré de T0p et déposé dans 1p. Ce jeton porte les valeurs dx, dy et dR qui correspondent au toucher sur la surface du Palm.

Comme pour l'approche à machine à états, le nombre d'états du réseau de Petri a tendance à exploser lorsque l'on veut modéliser une interaction complexe comme l'interaction multimodale. Ainsi il est nécessaire d'adopter un mécanisme de structuration afin de structurer et réduire l'explosion combinatoire du nombre d'états. Par exemple dans [Schyn, 2005], une approche basée sur des réseaux de Petri à objets (ICO pour Objets Coopératifs Interactifs) est étudiée pour l'interaction multimodale. ICO utilise une approche orientée objets afin de décrire les aspects statiques ou structurels du système interactif et les réseaux de Petri afin de décrire les aspects dynamiques ou comportementaux du système. ICO a par exemple été utilisé pour décrire l'interface utilisateur au sein de la spécification ARINC 611, norme utilisée pour définir le système de cockpit des avions [Barboni *et al.*, 2007].

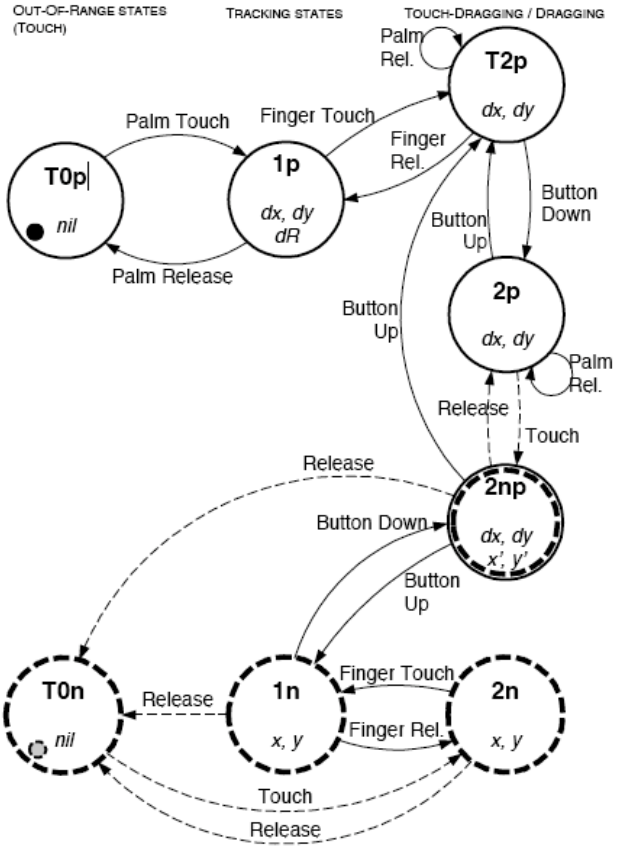


FIGURE 3.26 – Réseau de Petri complet décrivant une technique d'interaction bi-manuelle. Illustration extraite de [Hinckley *et al.*, 1998].

Modèle à flot de données

Le modèle à flot de données repose sur la notion de propagation de changements de valeur. Ces modèles sont souvent utilisés dans les systèmes à temps-réel, tels que le traitement d'images, le contrôle de processus industriels, les systèmes embarqués ou les pilotes de dispositifs.

Les systèmes qui implémentent ce modèle sont appelés systèmes réactifs. Selon Berry [Berry, 1989], les systèmes réactifs sont définis par le maintien constant d'une interaction avec leur environnement , à une vitesse déterminée par l'environnement et non pas par le programme lui-même. Ces systèmes reposent sur deux propriétés :

- **Synchronisme parfait**
- **Déterminisme** : un programme est déterministe si son comportement dépend uniquement de ses entrées. Un système déterministe est donc complètement prédictible.

Les premiers langages de programmation dédiés à ce type de modèles furent Esterel [Berry, 1989], Lustre [Halbwachs *et al.*, 1991] et Signal [Gautier et Guernic, 1987]. Le modèle des dispositifs en cascade présenté à la

section 3.2.3 repose sur un modèle à flot de données. Ces modèles ont également servit de fondements pour la validation formelle de systèmes interactifs multimodaux, comme dans la méthode formelle de test présentée par Bouchet [Bouchet *et al.*, 2008] basée sur l'environnement de test Lutess, implémenté en Lustre.

Nous venons de décrire les trois types de modèles de programmation de l'interaction (fonctions de rappel, automates et flots de données) et leurs utilisations pour la programmation de l'interaction multimodale. Ces modèles sont dédiés à des programmeurs et nous étudions dans la section suivante les modèles envisageables pour des non-programmeurs.

3.3.2 Modèles de programmation de l'interaction pour non-programmeurs

Définition

La programmation par des non-programmeurs est un domaine de recherche qui vise à fournir des modèles et des outils de programmation utilisables par des personnes non expertes en programmation.

Motivations

Dans nos travaux, nous ne visons pas la création d'une approche utilisable par des non-programmeurs. Nous considérons que le prototypage d'interfaces multimodales est réalisé par des concepteurs et des développeurs. Nous nous intéressons néanmoins aux approches pour non-programmeurs car par leurs caractéristiques de simplification ou d'abstraction de la programmation, ces approches peuvent faciliter le développement rapide de prototypes multimodaux. En nous référant aux dimensions cognitives de Green [Green, 1989], qui caractérisent les notations de programmation, nous estimons que les approches pour non-programmeurs peuvent contribuer à diminuer la *viscosité* et le *coût mental* lors de la construction rapide d'un prototype multimodal :

- **Viscosité** : La viscosité représente le nombre d'actions nécessaires pour développer un programme. En reposant sur des sous-ensembles des langages de programmation, les approches pour non-programmeurs réduisent la viscosité lors de la programmation. La réduction de la viscosité permet d'accélérer le développement d'un prototype.

- **Coût mental** : Le coût mental représente l'effort mental de programmation. Certaines opérations de programmation exigent un coût mental supérieur, comme les pointeurs en C. Les approches pour non-programmeurs proposent une approche simplifiée de la programmation à haut niveau d'abstraction. Ces approches ont pour objectif de réduire le coût mental de la programmation, permettant ainsi l'accélération de la programmation donc du développement de prototypes.

Brad Myers introduisait en 1986 deux types de programmation pour non-programmeurs : la programmation visuelle et la programmation par l'exemple [Myers, 1986]. La même année il introduisait la notion de la programmation par démonstration [Myers et Buxton, 1986]. Ces trois types de programmation ne sont

pas distincts : par exemple un langage de programmation peut être visuel et par démonstration.

Nous présentons ci-après les principales caractéristiques des trois modèles principaux de programmation pour non-programmeurs : la programmation sur exemple, la programmation par démonstration et la programmation visuelle.

Programmation sur exemple et programmation par démonstration

La Programmation sur Exemple[7] (PsE) [Myers, 1986] [Sanou *et al.*, 2006] permet au développeur de créer un programme en spécifiant un exemple de données que le système doit traiter. Ces données peuvent correspondre à des données en entrée/sortie [Shaw *et al.*, 1975], ou à des traces d'exécution du programme [Biermann et Krishnaswamy, 1976]. Le terme de Programmation par Démonstration[8](PpD) [Myers et Buxton, 1986] a ensuite remplacé celui de PsE. Le principe reste le même : l'utilisateur spécifie un exemple qui est analysé et généralisé par le système. Ce type de programmation vise surtout les domaines d'applications graphiques interactives, comme les jeux, les interfaces de dessin ou les animations. La PpD est basée sur une analogie avec l'apprentissage scientifique où l'enseignement est basé sur des exemples de problèmes concrets. Comme souligné par Myers [1986], il est plus simple de traiter des problèmes concrets que des problèmes abstraits.

Les exemples de systèmes de PpD sont nombreux. Nous citons certains des premiers systèmes de PpD créés, parmi lesquels figurent Tinker [Lieberman, 1982], SmallStar [Halbert, 1993] ou Péridot [Myers et Buxton, 1986] . La Figure 3.27 illustre le système de programmation graphique par démonstration Mondrian [Lieberman, 1993]. L'utilisateur peut définir des nouvelles commandes graphiques en créant des "dominos", composés de deux parties : l'entrée et la sortie correspondante. L'utilisateur peut également associer un nom de commande à chaque domino.

Programmation visuelle

Selon [Myers, 1986], la programmation visuelle comprend toute programmation qui permet à l'utilisateur de spécifier un programme par un plan bi-dimensionnel. La programmation textuelle n'est pas considérée bi-dimensionnelle car le compilateur traite le programme comme une longue chaîne uni-dimensionnelle.

Un exemple de langage de programmation visuelle est Quartz Composer, qui permet de créer des rendus graphiques par assemblage visuel de nodes (Figure 3.28). Des nodes de type *Interaction* permettent également d'utiliser les entrées de l'utilisateur afin de créer des rendus graphiques interactifs.

La programmation visuelle permet de tirer complètement profit du processus mental humain, plus adapté aux tâches multi-dimensionnelles qu'aux tâches linéaires [Clarisse et Chang, 1986]. Les fondements psychologiques des avantages de la programmation visuelle ont été abordés dans la littérature [Smith, 1975].

L'intérêt de la programmation visuelle pour le prototypage rapide est identifié très tôt [Schmucker, 1996]. La réduction du temps de développement par la sim-

[7]Programming by Example
[8]Programming by Demonstration

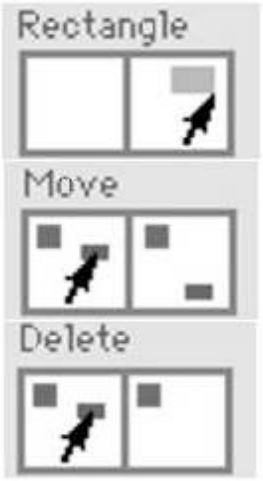


FIGURE 3.27 – Exemple de commandes créées par un utilisateur dans le système de programmation graphique par démonstration Mondrian.

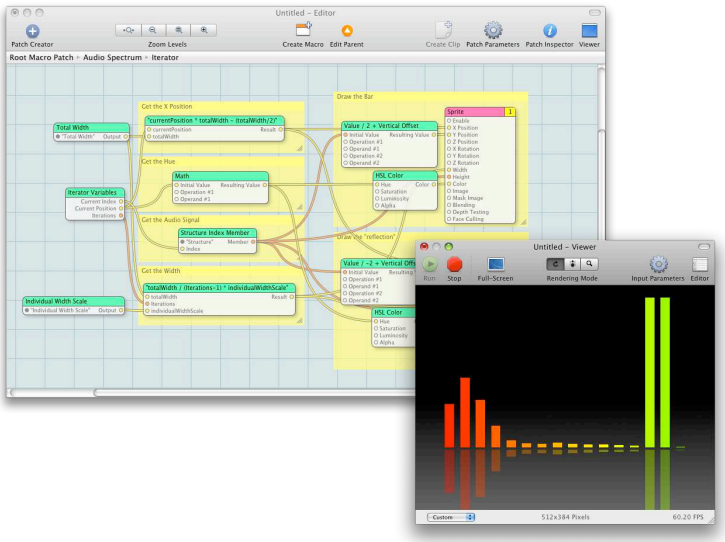


FIGURE 3.28 – Interface de Quartz Composer, un langage de programmation visuelle pour la création de rendus graphiques.

plification du langage (induite du haut-niveau d'abstraction des langages visuels) permet en effet de développer rapidement des prototypes interactifs.

Il existe un lien fort entre la programmation visuelle et le modèle de programmation à flot de données (section 3.3.1). En effet, la programmation visuelle de flots de données (Data Flow Visual Programming ou DFVP) réduit la distance sémantique entre le modèle conceptuel et l'implémentation [Johnston *et al.*, 2004] jusqu'à supprimer cette distance [Baroth et Hartsough, 1995]. Cette approche permet donc le prototypage rapide[9].

3.3.3 Modèles de conception logicielle par réutilisation

Les modèles de conception logicielle par réutilisation offrent aux développeurs les mécanismes pour implémenter des systèmes de façon modulaire et réutilisable. Nous présentons trois approches principales : approche à agents, approche à composants et approche à services.

Approche à agents

L'approche à agents [Genesereth et Ketchpel, 1994] est une des premières solutions pour la réutilisabilité au sein des architectures logicielles. L'approche à agents structure une application en un ensemble d'agents logiciels capables de communiquer entre eux en échangeant des messages dans un langage de *communication agent*. L'approche est née du constat suivant : une grande variété d'applications existaient, mais celles-ci n'étaient pas capables d'inter-opérer.

Une des premières questions soulevées par cette approche est celle du processus de transformation des applications existantes en agents. Ce processus est dit *agentification* en anglais. Trois approches sont alors identifiées afin de réaliser cette agentification (Figure 3.29) La première consiste à créer un transducteur qui réalise la médiation entre l'application et les autres agents (à gauche sur la Figure 3.29). La deuxième approche consiste à modifier le code de l'application en rajoutant un code de wrapper qui va permettre de communiquer directement avec les autres agents (au centre sur la Figure 3.29). La troisième approche consiste à réécrire l'application afin d'avoir un agent complètement conforme aux spécifications agent (à droite sur la Figure 3.29).

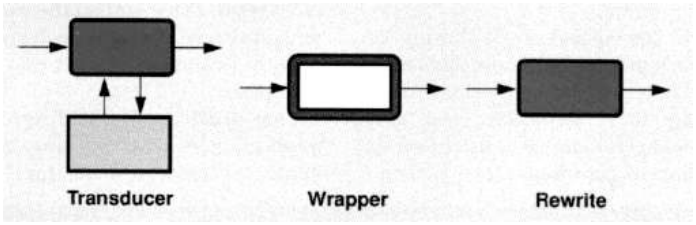


FIGURE 3.29 – *Agentification* d'une application existante. Illustration extraite de [Genesereth et Ketchpel, 1994]

La deuxième question soulevée par l'approche à agents est celle de la communication entre agents. Deux approches principales sont considérées. Dans la pre-

[9] «DFVPLs tend to significantly blur the distinctions between the requirements, design, coding, and testing phases of the software lifecycle. This blurring offers the opportunity for rapid prototyping.» [Johnston *et al.*, 2004]

mière approche, les agents peuvent communiquer directement. Cette approche est intéressante lorsque le nombre d'agents n'est pas trop important. Le principal problème est le coût de déploiement de cette solution à grande échelle, étant donné que les agents doivent partager des contrats ou des spécifications afin de pouvoir communiquer entre eux. La deuxième approche propose un système fédéré qui fait appel à des facilitateurs, plus approprié lorsque le nombre d'agents est important. Dans ce cas, les agents vont "fédérer" certaines ressources, comme les contrats, au sein des facilitateurs, qui gèrent la communication entre agents. La Figure 3.30 montre un système à agents composé de trois machines, une avec trois agents et les deux autres avec deux agents chacune.

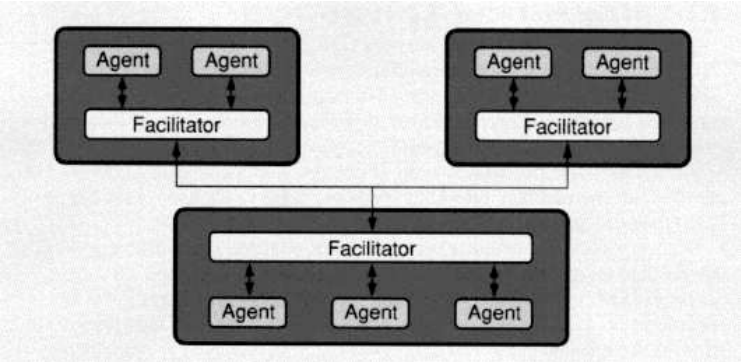


FIGURE 3.30 – Système à agents. Illustration extraite de [Genesereth et Ketchpel, 1994].

Nous avons déjà montré à la section 3.3.4 de ce chapitre des exemples d'architectures pour la conception logicielle de systèmes interactifs basées sur le modèle à agents, comme PAC, MVC ou PAC-Amodeus.

Le modèle à agents a également été utilisé dans le cadre de l'interaction multimodale pour concevoir des architectures logicielles et pour implémenter des plateformes de développement multi-agents. Nous citons l'architecture multi-agents Open Agent Architecture (OAA) [Martin *et al.*, 1999], où plusieurs agents sont définis et implémentent les différentes modalités d'interaction du système interactif. Par exemple, OAA a été utilisé dans Quickset [Cohen *et al.*, 1997], un outil pour le développement de systèmes multimodaux. Dans Quickset un facilitateur gère les informations venant des différentes modalités d'interaction afin de définir un système multimodal. Nous présentons Quickset en détail au chapitre 6.

Approche à composants

Les termes *composant* et *composition* possèdent d'innombrables définitions. Nous reprenons ici les définitions, largement acceptées, de Szyperski [Szyperski, 2002] et [Szyperski, 2003].

Selon ces définitions, un composant est «une unité de composition uniquement associée à des interfaces spécifiées de manière contractuelle et à des dépendances de contexte explicites. Un composant logiciel peut être déployé de manière indépendante et sujet à composition par des tiers» [Szyperski, 2002]. Une composition (ou assemblage) est l'application d'un opérateur de composition à des composants.

Alors que l'approche à agents a été conçue pour résoudre les aspects d'interopérabilité entre applications, l'approche à composants propose des solutions aux problémes propres au cycle de vie des systèmes informatiques. Ainsi les objectifs principaux de l'approche à composants sont d'assurer la **réutilisation**, le **déploiement** et la **dynamicité** des systèmes.

Approche à services

Nous reprenons la définition de service de Bieber [Bieber et Carpenter, 2001] : un service est un comportement défini contractuellement qui peut être implémenté et fourni par un composant quelconque pour être utilisé par un autre composant ; l'interaction entre les deux composants s'appuie uniquement sur le contrat[10].

L'approche à services permet la découverte dynamique de services au moment de l'exécution. Cette approche se révèle pertinente dans le cadre des systèmes distribués ou dans celui de l'intelligence ambiante, où différents dispositifs distribués doivent pouvoir communiquer entre eux sans forcément se connaître, ou être découverts dynamiquement.

L'approche à services est maintenant combinée à une approche à composants. Par exemple le modèle à composants orientés services iPOJO [Escoffier *et al.*, 2007] combine l'approche à services et la programmation par composants afin d'introduire la dynamicité au sein d'un modèle à composants.

3.3.4 Modèles d'architecture logicielle pour l'interaction homme-machine

Un aspect incontournable

Le choix de l'architecture logicielle est central dans la conception logicielle de systèmes interactifs. Avec les progrès technologiques et la croissante complexité de la partie interactive des systèmes logiciels au début des années 90, l'architecture logicielle pour l'IHM a donné lieu à plusieurs travaux [Coutaz, 1993].

Le degré de complexité des systèmes interactifs atteint aujourd'hui un tel niveau que même le développement de prototypes interactifs ne peut ignorer ces considérations architecturales. Pour une approche de prototypage rapide d'interfaces multimodales, deux caractéristiques de l'architecture sont particulièrement importantes : l'indépendance de la partie interactive du noyau de l'application et la réutilisation des modules logiciels de la partie interactive.

L'indépendance interface-noyau permet idéalement de tester différents prototypes d'interaction sans modifier le code du noyau. La réutilisation logicielle permet quant à elle d'accélerer le cycle de vie du prototypage par la réutilisation de solutions déjà existantes.

Modèle de Seeheim

Le modèle de Seeheim [Pfaff et Hagen, 1985] est un des premiers patrons d'architecture logicielle à structurer l'interface homme-machine au sein d'un système interactif. Ce modèle est issu d'un atelier de travail sur le thème *User Interface Management Service* organisé dans la ville de Seeheim (Allemagne) en 1985.

[10] «A service is a contractually defined behavior that can be implemented and provided by any component for use by any component, based solely on the contract.»[Bieber et Carpenter, 2001]

Le modèle de Seeheim décompose le code de l'interface en trois parties principales (Figure 3.31) :

- La **Présentation** gère l'interaction avec l'utilisateur. Elle interprète les actions de l'utilisateur (interaction en entrée) et génère l'affichage (interaction en sortie).
- Le **Contrôleur de dialogue** gère le séquencement des entrées et des sorties.
- L'**Interface du noyau fonctionnel** gère la communication entre les entrées/sorties et l'application.

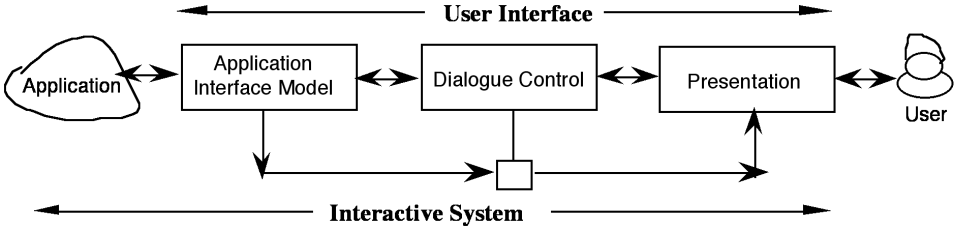


FIGURE 3.31 – Modèle de Seeheim. Illustration extraite de [Coutaz, 1993].

Le modèle de Seeheim a permis pour la première fois de séparer l'interface du noyau fonctionnel (noté Application dans le modèle Seeheim). Cette séparation permet entre autre de modifier l'interface sans que cela ait un impact sur le noyau fonctionnel.

Modèle Arch

Le modèle Arch [UIMS, 1992] étend la décomposition fonctionnelle du modèle Seeheim et est aussi appelé "Seeheim-revisited". En particulier, Arch reprend les notions de noyau fonctionnel, de contrôleur de dialogue et de présentation. L'objectif du modèle Arch était de proposer un modèle d'architecture qui soit indépendant des évolutions technologiques.

Le nom du modèle répond à sa forme en arche (Figure 3.32) . Les différents composants de l'arche sont les suivants :

- Le **noyau fonctionnel** implémente les concepts du domaine.
- L'**adaptateur de domaine** ajuste les différences de modélisation entre le noyau fonctionnel et le contrôleur de dialogue.
- Le **contrôleur de dialogue** gère l'enchaînement des tâches ainsi que le lien entre les couches adjacentes : la couche de présentation et l'adaptateur de domaine.
- Le **composant de présentation** gère l'interaction avec l'utilisateur. Il interprète les actions de l'utilisateur (interaction en entrée) et génère l'affichage (interaction en sortie).
- Le **composant d'interaction** gère le contact direct avec l'utilisateur et est souvent implémenté au moyen d'une boîte à outils.

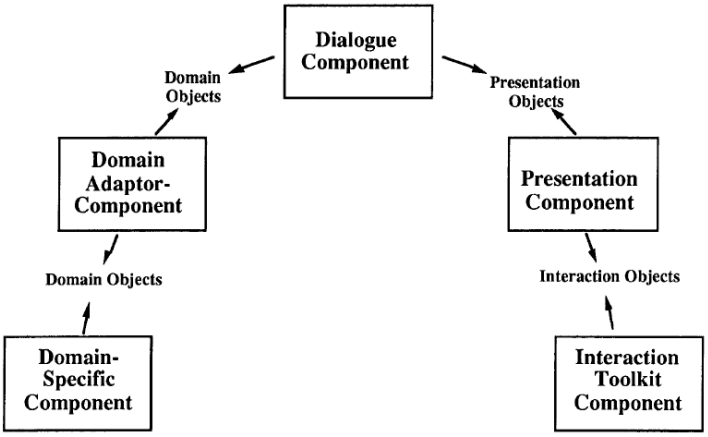


FIGURE 3.32 – Modèle Arch. Illustration extraite de [UIMS, 1992].

L'importance relative des cinq composants de Arch varie en fonction du cas à traiter. Cette variation reposant sur le métamodèle Slinky permet de définir plusieurs modèles d'architecture Arch.

Métamodèle Slinky

Le métamodèle Slinky [UIMS, 1992] exprime métaphoriquement la migration des fonctions dans le modèle Arch. Le nom du métamodèle provient de ce jouet en forme de ressort qui se déplace dynamiquement une fois mis en mouvement (Figure 3.33).

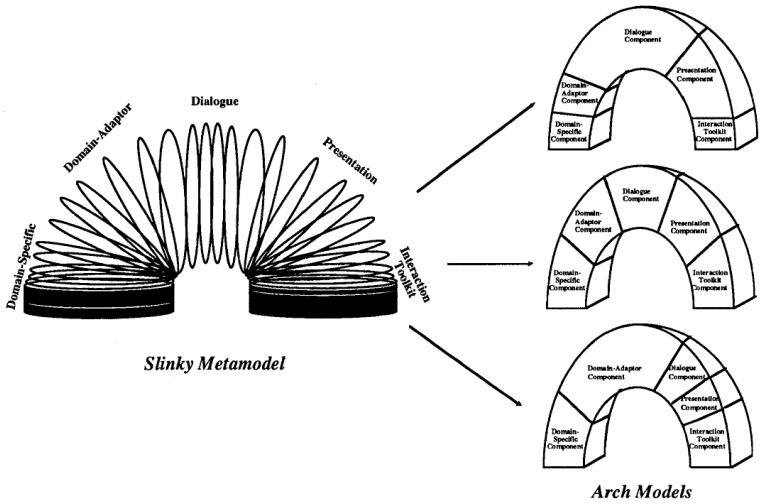


FIGURE 3.33 – Métamodèle Slinky. Illustration extraite de [UIMS, 1992].

Slinky permet la définition de différentes instances du modèle Arch, selon la manière dont les fonctions migrent (Figure 3.33). Cela permet d'adapter l'architecture

aux critères de conception du cas à traiter, comme la performance, la réutilisation de code, le coût de développement, la complexité de la spécification...

Modèle MVC

Le modèle MVC (Modèle-Vue-Contrôleur) [Krasner et Pope, 1988] fut conçu au Xerox Parc à la fin des années 70 pour le système Smalltalk [Goldberg et Robson, 1983]. L'objectif de ce modèle était de rendre explicite le modèle des données et de permettre à l'utilisateur d'inspecter et modifier ces données.

Ce modèle considère trois composants différents (Figure 3.34) :

- Le **Modèle** représente le modèle de données de l'application.
- La **Vue** représente l'interface avec l'utilisateur.
- Le **Contrôleur** gère la communication et la synchronisation entre la vue et le modèle.

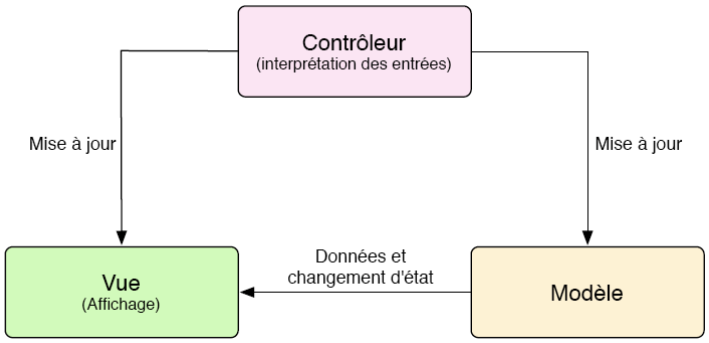


FIGURE 3.34 – Modèle MVC tel que défini par Xerox Parc.

Le modèle MVC a été largement utilisé dans la création de systèmes interactifs. De plus, MVC est aussi utilisé pour structurer des éléments graphiques dans des boîtes à outils comme Swing [Tucker, 2000] ou Cocoa [Apple, 2010]. Par exemple, les composants de la boîte à outils Swing de Java, comme les listes (JList), les tables (JTable) ou les arbres (JTree), utilisent MVC afin de séparer les données de la représentation [Tucker, 2000].

La création de systèmes interactifs avec MVC peut s'avérer lourde à programmer, en particulier à cause de la grande quantité de classes à implémenter. Cela devient encore plus évident dans le cadre d'applications complexes [Martin *et al.*, 2005].

Néanmoins le modèle MVC a été largement appliqué et a donné lieu à différentes variations. La Figure 3.35 illustre l'adaptation du modèle MVC réalisée par Apple pour Cocoa. Cette implémentation donne plus de poids au contrôleur et sépare d'avantages la vue et le modèle.

Cette solution architecturale (MVC dans Cocoa) est en fait beaucoup plus proche du modèle PAC présenté dans le paragraphe suivant que de MVC. De nombreux

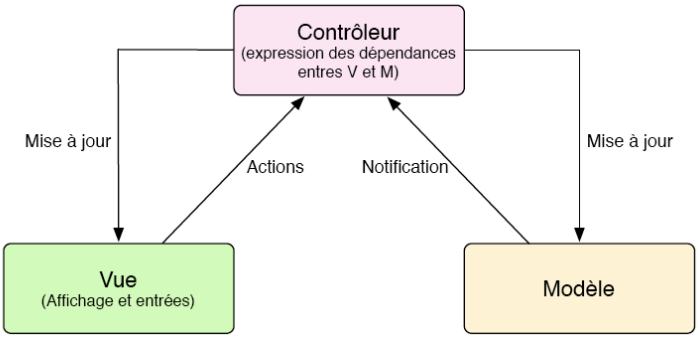


FIGURE 3.35 – Modèle MVC utilisé par Apple dans Cocoa.

blogs sur le www discutent cette architecture présentée comme issue du modèle MVC et qui est plus proche du modèle PAC.

Modèle PAC

Comme MVC, le modèle PAC (Presentation-Abstraction-Contrôle) [Coutaz, 1987] est basé sur la notion d'agent. Un agent dans PAC est composé de trois facettes :

- La **Présentation** définit la syntaxe concrète de l'agent, c'est-à-dire l'entrée et la sortie de l'agent.
- L'**Abstraction** représente la sémantique de l'agent, c'est-à-dire les fonctions que l'agent peut réaliser (les tâches).
- Le **Contrôle** gère la cohérence entre la Présentation et l'Abstraction.

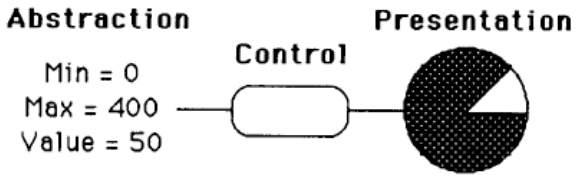


FIGURE 3.36 – Objet interactif dans le modèle PAC. Illustration extraite de [Coutaz, 1987].

La Figure 3.36 illustre les trois facettes d'un agent implémentant un objet interactif simple. La Présentation de cet objet est composé d'une interface en sortie, un camembert, et d'une interface en entrée, l'interaction directe avec la souris sur le camembert. L'Abstraction de cet objet est une valeur entière comprise entre deux limites, 0 et 400 (dans l'exemple de la Figure 3.36, valeur égale à 50). Le Contrôle maintient la cohérence entre la Présentation et l'Abstraction. Par exemple, si l'utilisateur modifie la taille d'une part du camembert, le Contrôle va demander la mise à jour de la valeur de l'entier dans l'Abstraction.

Une application interactive dans PAC est alors composée de plusieurs agents (Figure 3.37). Les agents sont connectés entre eux à travers des Contrôles.

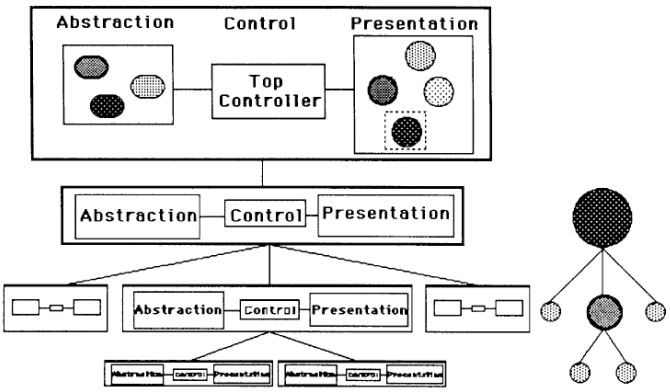


FIGURE 3.37 – Application interactive structurée selon le modèle PAC. Illustration extraite de [Coutaz, 1987].

Modèle PAC-Amodeus

Le modèle PAC-Amodeus [Nigay et Coutaz, 1991] étend le modèle Arch en affinant la facette Contrôle par une hiérarchie d'agents PAC (Figure 3.38).

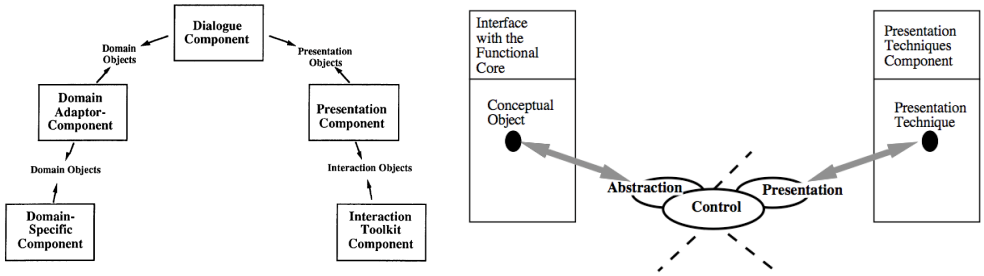


FIGURE 3.38 – Modèle Arch (à gauche) et PAC-Amodeus (à droite). Illustration extraite de [Nigay et Coutaz, 1991].

Le contrôleur de dialogue, souvent sous-décrit dans les autres modèles bien que central comme en témoigne sa position dans le modèle Arch, est composé ici d'un ensemble d'agents collaborant afin de gérer les correspondances entres les tâches du noyau fonctionnel et l'interaction.

3.3.5 Conclusion

Comme pour les modèles de conception de l'interaction (section 3.2), nous avons opté pour une approche ascendante dans l'analyse des modèles de conception logicielle. Pour cela nous avons d'abord étudié les modèles de programmation de l'interaction, puis les modèles de programmation pour la réutilisation qui reposent sur une structuration du code en briques logicielles élémentaires que ce soient des agents, des composants ou des services. Nous finissons par les modèles d'architecture logicielle qui fournissent une décomposition fonctionnelle à haut niveau d'abstraction d'un système interactif (Figure 3.39).

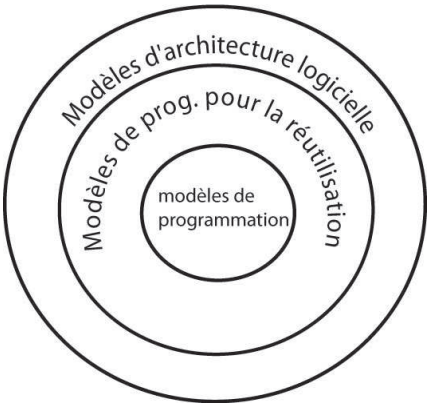


FIGURE 3.39 – Synthèse des modèles présentés à la section 3.3 du chapitre.

3.4 Modèles de conception logicielle de l'interaction multimodale

Comme schématisé à la Figure 3.1, les modèles de conception logicielle de l'interaction multimodale sont issus du couplage de modèles conceptuels pour l'interaction multimodale avec de modèles de conception logicielle afin d'offrir des solutions conceptuelles pour le développement des systèmes multimodaux. Nous présentons les deux modèles qui reposent explicitement sur des modèles conceptuels de l'interaction multimodale : le modèle des configurations d'entrée et le modèle ICARE.

3.4.1 Introduction

Ces deux modèles présentent des caractéristiques communes. À plus haut niveau d'abstraction (section 3.3.4), ces deux modèles concernent tous deux la partie logicielle qui est dédiée à l'interaction. En nous situant par rapport à Arch, ces deux modèles ne concernent que les composants de présentation et d'interaction (Figure 3.40). À un niveau d'analyse plus fine (section 3.3.3), ces deux modèles reposent sur une approche à composants. Enfin pour ces deux modèles, l'approche de programmation visuelle par flot de données (section 3.3.1) est adoptée. Leurs différences résident principalement sur le modèle conceptuel pour l'interaction sous-jacent (section 3.2).

3.4.2 Modèle des configurations d'entrée

Le modèle des configurations d'entrée [Dragicevic, 2004] est une concrétisation du paradigme des dispositifs en cascade (Section 3.2.3). Le modèle de conception logicielle correspondant est basé sur quatre notions principales : les configurations d'entrée, les dispositifs, les slots et les connexions.

Une configuration d'entrée (Figure 3.41) est un ensemble de dispositifs système et de dispositifs d'application reliés par des dispositifs utilitaires. Un dispositif est une boîte noire qui interagit avec son environnement par l'intermédiaire de slots, ou canaux typés. Il existe des slots d'entrée et des slots de sortie des dispositifs.

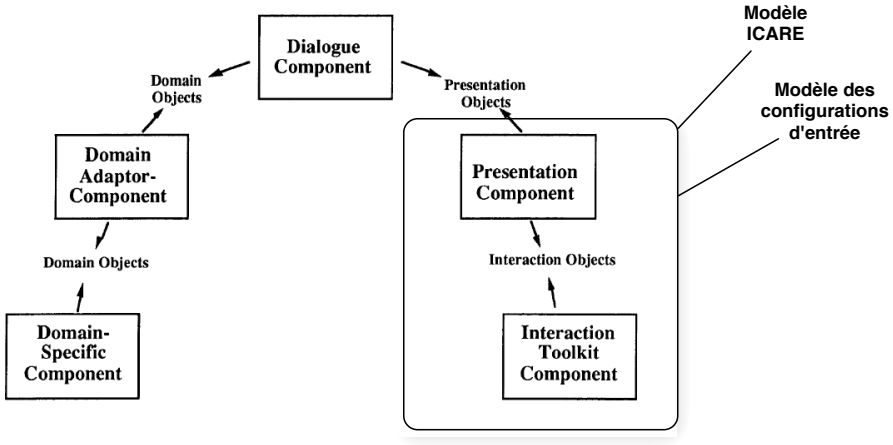


FIGURE 3.40 – Portée du modèle des configurations d'entrée et du modèle ICARE par rapport à Arch.

Une connexion est un lien reliant un slot de sortie d'un dispositif à un slot d'entrée d'un autre dispositif.

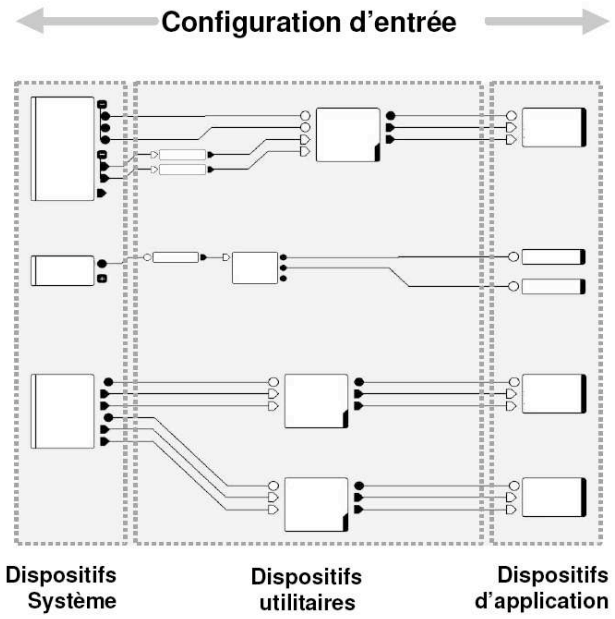


FIGURE 3.41 – Une configuration d'entrée. Illustration extraite de [Dragicevic, 2004].

La Figure 3.42 illustre un exemple simple de configuration d'entrée, avec un dispositif système *souris*, deux dispositifs utilitaires qui transforment les *(dx, dy)* de la souris en *(x, y)* et un dispositif d'application *freehand*, qui est une application de dessin. Le slot *left* du composant *souris*, correspondant aux événements du bouton gauche de la souris, est connecté au slot *use* du composant *freehand* afin de déclencher l'action de dessiner lorsque l'utilisateur appuie sur le bouton. Les deux composants *sum* permettent quand à eux de transformer les positions relatives *(dx,*

dy) en positions absolues *(x, y)*.

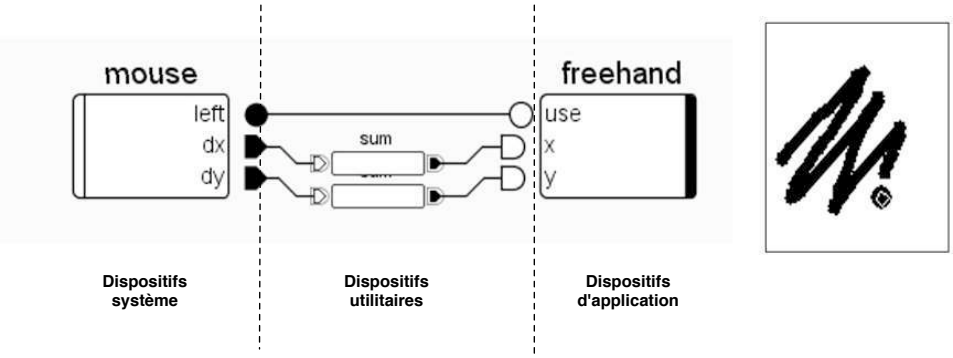


FIGURE 3.42 – Exemple de configuration d'entrée simple. Illustration extraite de [Dragicevic, 2004].

Ce modèle ne fixe pas le nombre de transformations (c'est-à-dire le nombre de dispositifs utilitaires). Néanmoins, il n'explicite pas la composition de modalités, qui sera traduite à bas niveau par exemple par des opérateurs arithmétiques sur des données provenant de deux flots de données, comme le composant mathématique d'addition illustré à la Figure 3.43.

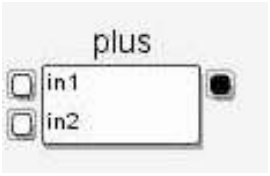


FIGURE 3.43 – Composant dispositif utilitaire dans Icon réalisant une opération d'addition. Illustration extraite de [Dragicevic, 2004].

3.4.3 Modèle ICARE

Comme le modèle des configurations d'entrée, le modèle ICARE [Bouchet, 2006] est une approche de programmation graphique à flot de données qui repose sur le modèle de conception de l'interaction CARE et sur les propriétés temporelles de Allen afin de définir des interactions multimodales. Le modèle de conception logicielle correspondant s'articule autour de la notion de modalité d'interaction, définit comme le couple <dispositif physique, langage d'interaction>, et de la notion de combinaison de modalités selon CARE.

Ce modèle définit deux types de composants : les composants élémentaires et les composants de composition. Les composants élémentaires servent à définir des modalités d'interaction et peuvent être du type Dispositif ou du type Langage d'interaction. Les composants de composition sont basés sur les propriétés CARE (Figure 3.44).

Une interaction multimodale est définie graphiquement par l'assemblage de plusieurs composants. La Figure 3.44 illustre un exemple de composition Equivalente de deux modalités en entrée, une modalité de manipulation directe et une

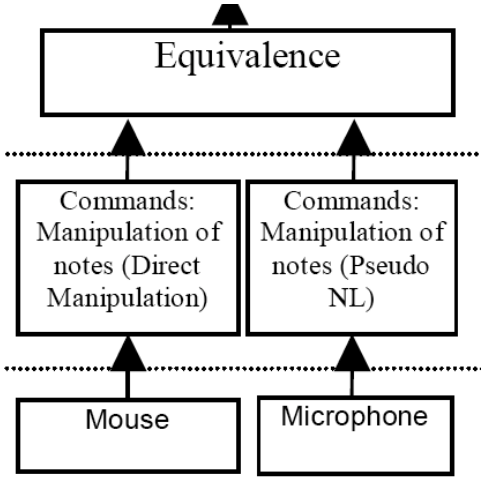


FIGURE 3.44 – Equivalence de deux modalités dans le modèle ICARE.Illustration extraite de [Bouchet *et al.*, 2004].

modalité de parole. La modalité de manipulation directe est réalisée avec un composant dispositif Souris (*Mouse*) et un composant langage Commande Manipulation de notes. La modalité de parole est réalisée avec un composant dispositif Microphone et un composant langage Commande Manipulation de notes.

Contrairement au modèle des configurations d'entrée, ce modèle explicite les compositions de modalités en considérant CARE. Néanmoins, la définition simple d'une modalité d'interaction par le couple <dispositif, langage> ne correspond pas à la réalité des opérations qui sont opérées sur les données des dispositifs en entrée. Comme l'illustre Pierre Dragicevic dans son modèle des dispositifs en cascade, montré à la section 3.2.3, les opérations sur les données en entrée correspondent souvent à un ensemble de traitements en série. Or le modèle ICARE modélise cela par une seule phase de traitement sémantique, réalisé par le composant Langage d'interaction.

L'approche ICARE reste donc difficile à appliquer en raison de la granularité de description des modalités. En effet, au niveau de la conception de l'interaction, les deux niveaux dispositif et langage sont adéquats mais pour la conception logicielle, en vue du prototypage rapide, il convient d'affiner ce niveau langage pour pouvoir définir les briques de base réutilisables. En effet, la pratique avec ICARE a montré que les langages d'intération se révèlent très peu réutilisables, ce qui implique un frein au développement rapide d'interfaces multimodales.

3.5 Synthèse

Dans ce chapitre nous avons présenté un ensemble d'approches conceptuelles pour la conception de l'interaction multimodale et la conception logicielle. Nous avons présenté ces approches selon trois axes pertinents de notre étude : les modèles de conception de l'interaction multimodale, les modèles de conception logicielle et enfin les modèles de conception logicielle pour l'interaction multimodale.

En effet, notre démarche scientifique en vue de la définition d'un modèle de conception logicielle pour l'interaction multimodale est de partir de modèles de

conception **de l'interaction multimodale** à instancier selon un modèle **de conception logicielle**.

Les modèles de conception de l'interaction multimodale englobent les modèles de dispositifs physiques et logiques ainsi que les modèles d'interaction et de composition de modalités. Les modèles de dispositifs nous permettent, à travers les taxonomies de dispositifs logiques, comme les **tâches de Foley** (section 3.2.3), et de dispositifs physiques, comme la **taxonomie de Buxton** (section 3.2.2), de séparer les tâches interactives des dispositifs physiques en entrée. Les modèles d'interaction en entrée offrent une approche pour concevoir les interfaces en entrée et pour lier les dispositifs physiques aux tâches, comme le **paradigme des dispositifs en cascade** (section 3.2.3). Les différents **modèles de composition** permettent de combiner ces approches selon différents aspects : temporel, spatial, articulatoire, syntaxique et sémantique.

Les modèles de conception logicielle englobent les modèles de programmation pour l'interaction et pour les non-programmeurs et les modèles d'architecture logicielle . Les modèles de programmation permettent de réfléchir à la façon dont est programmée l'interaction. En ce qui concerne la logique utilisée pour implémenter l'interaction, deux grands courants coexistent : les approches à automates et les modèles à flot de données. Alors que les approches à automates sont pertinentes pour la description de l'enchaînement des actions et des aspects temporels de la multimodalité, les modèles réactifs permettent de mieux centrer la conception logicielle sur la fonction d'abstractions des données issues des dispositifs. Nous avons vu que l'utilisation d'une **approche par programmation visuelle à flot de données** (section 3.3.2) est pertinente pour le développement rapide. Les modèles de conception logicielle par réutilisation nous permettent de construire des briques logicielles réutilisables (section 3.3.3), à travers différentes approches : **à agents, à composants ou à services**. Elles nous permettent également de séparer la partie interactive du noyau fonctionnel. Pour cela, le modèle **Arch** (section 3.3.4) reste une référence dans la construction de systèmes interactifs.

Aussi il n'est pas étonnant que les deux approches de conception logicielle de l'inter'action multimodale retenues reposent sur une approche par programmation visuelle à flot de données. Les deux modèles, le **modèle des configurations d'entrée** (section 3.4.2) et le **modèle ICARE** (section 3.4.3), diffèrent par le modèle de conception de l'interaction multimodale sous-jacente. Pour la conception logicielle, ces deux modèles présentent des limitations qui ont motivé la définition de notre modèle de conception logicielle, présentée au chapitre suivant.

Chapitre 4

Notre modèle conceptuel pour le prototypage d'interfaces multimodales

« 1 REDUCE The simplest way to achieve simplicity is through thoughtful reduction.

2 ORGANIZE Organization makes a system of many appear fewer.

(...)

8 TRUST In simplicity we trust.

9 FAILURE Some things can never be made simple.»

John Maeda. The Laws of Simplicity. The MIT Press (2006).

Contenu du chapitre

Dans ce chapitre, nous présentons notre modèle conceptuel pour le prototypage d'interfaces multimodales. Notre modèle est basé sur une approche à composants. Motivé par un pouvoir descriptif, comparatif et génératif [Beaudouin-Lafon, 2004] élevé, notre modèle définit des caractéristiques des composants pour la multimodalité. Ce modèle est un outil conceptuel pour l'exploration rapide de solutions d'interaction multimodale.

Nous rappelons d'abord les motivations qui nous ont conduit à établir ce modèle. Puis nous définissons la portée de notre modèle avant d'en présenter une vue globale. Nous détaillons ensuite les notions de base de notre modèle : le composant, le port, la connexion et l'assemblage.

Nous illustrons enfin notre modèle en considérant une interaction multimodale du type Put-That-There [Bolt, 1980], présenté à la section 2.4 du chapitre 2.

4.1 Motivations

Le (dés)équilibre des modèles

Comme le souligne Beaudouin-Lafon [2004], la qualité d'un modèle repose sur l'équilibre de ses trois pouvoirs : le pouvoir descriptif, le pouvoir comparatif et le pouvoir génératif[1]. Le pouvoir descriptif désigne la capacité du modèle à décrire un ensemble significatif de solutions existantes. Le pouvoir comparatif désigne l'aptitude d'un modèle à permettre la comparaison de différentes alternatives de conception. Le pouvoir génératif exprime la faculté d'un modèle à favoriser la conception de nouvelles solutions d'interfaces.

Ainsi, des modèles à haut-niveau d'abstraction, comme le paradigme de la manipulation directe (section 3.2.4 du chapitre 3), offrent un fort pouvoir descriptif et comparatif, mais un faible pouvoir génératif. Des modèles à bas-niveau d'abstraction, comme les règles de style du Apple Human Interface Guidelines [Computer, 1987], offrent un faible pouvoir descriptif et comparatif mais un fort pouvoir génératif.

Notre motivation à définir un modèle pour la multimodalité repose sur le constat suivant : des deux modèles existants pour la multimodalité (section 3.4 du chapitre 3), nous constatons que l'un offre un bon pouvoir génératif, mais un faible pouvoir descriptif et comparatif, et que l'autre présente un fort pouvoir descriptif et comparatif, mais un faible pouvoir génératif.

Analyse des deux modèles de référence

Les deux modèles conceptuels pour la conception logicielle d'interfaces multimodales présentés au chapitre précédent, Icon [Dragicevic et Fekete, 2004] et Icare [Bouchet et Nigay, 2004], sont basés sur une approche à composants. Comme le montre le Tableau 4.1, ces deux modèles correspondent aux deux cas, haut/bas-niveau, que nous avons cités à la section précédente.

[1] «High-level models tend to have good descriptive power but poor evaluative and generative power. Low-level models tend to have poor descriptive and evaluative power, but higher generative power. A good interaction model must strike a balance between generality (for descriptive power), concreteness (for evaluative power) and openness (for generative power).» [Beaudouin-Lafon, 2004]

Modèle	Pouvoir descriptif	Pouvoir génératif	Pouvoir comparatif
Modèle Icon	Faible	Fort	Faible
Modèle Icare	Fort	Faible	Fort

TABLE 4.1 – Résumé des pouvoirs des modèles conceptuels Icon et Icare.

Le modèle Icon répond au profil d'un modèle bas-niveau. Les interfaces des composants sont définies de façon ad hoc, ce qui ne facilite pas la compatibilité des composants et donc leur réutilisation. De même, l'assemblage de composants est faiblement structuré : entre les dispositifs en entrée et les tâches interactives, Icon ne considère qu'un seul niveau. De plus, le modèle n'explicite pas la composition d'entrées afin de définir une interaction multimodale. Cela se traduit par un modèle à grand pouvoir générateur, mais faible pouvoir descriptif et comparatif.

Le modèle Icare offre un cadre conceptuel haut-niveau pour la conception d'interfaces multimodales. Les modalités d'interaction sont représentées de manière simplifiée, de même que les données venant des dispositifs en entrée (utilisation d'événements génériques simples appelés Icare Events). Malgré son pouvoir descriptif et comparatif élevé, le modèle Icare a un faible pouvoir générateur.

Notre solution : une approche mixte

Nous proposons une solution qui allie les avantages des deux modèles grâce à une approche mixte. Notre objectif est de donc définir un modèle qui offre à la fois un fort pouvoir descriptif et comparatif ainsi qu'un fort pouvoir génératif.

Dans les sections suivantes nous présentons notre modèle ainsi que les différents types de composants qui le constituent. Nous soulignons, tout au long de l'exposé, les caractéristiques de notre modèle au regard de celles de Icon et Icare.

4.2 Vue globale de notre modèle

4.2.1 Portée de notre modèle

Nous visons un modèle conceptuel pour l'***exploration rapide de solutions d'interaction multimodale en entrée***, comme le montre la Figure 4.1. Nous focalisons sur l'interaction multimodale en entrée. Nous avons considéré quelques scénarios sur l'interaction multimodale en sortie [Rousseau, 2006] montrant l'applicabilité de notre approche au cas de la multimodalité en sortie. Néanmoins, l'extension de nos travaux au cas de la multimodalité en sortie constitue une de nos perspectives. Par exemple, la fission multimodale, plus fréquente en sortie, doit être étudiée plus en détail.

Pour mieux appréhender la portée de notre modèle conceptuel pour l'interaction multimodale en entrée, nous situons notre approche au sein d'une décomposition fonctionnelle canonique d'un système interactif, en considérant le modèle d'architecture Arch, présenté au chapitre 3 section 3.3.4. La Figure 4.2 souligne que nos travaux concernent que la partie Interaction d'un système interactif. Ainsi au sein de Arch, notre approche couvre les deux composants *Présentation* et *Interaction*. Notre point de contact avec un système interactif est donc les tâches élémentaires

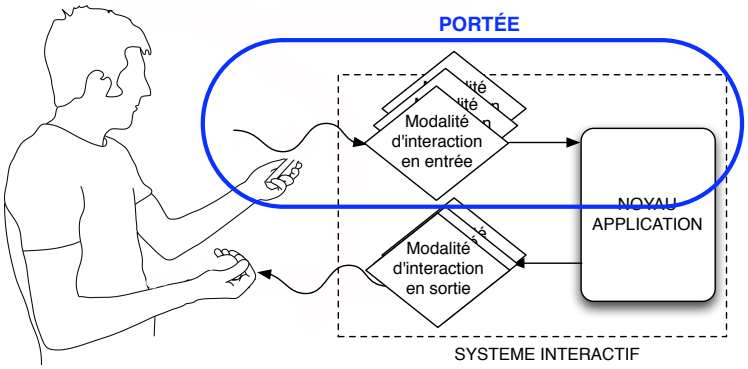


FIGURE 4.1 – Portée de notre modèle par rapport au sens de l'interaction.

que le système interactif est capable de gérer au sein de son composant *Contrôleur de dialogue*.

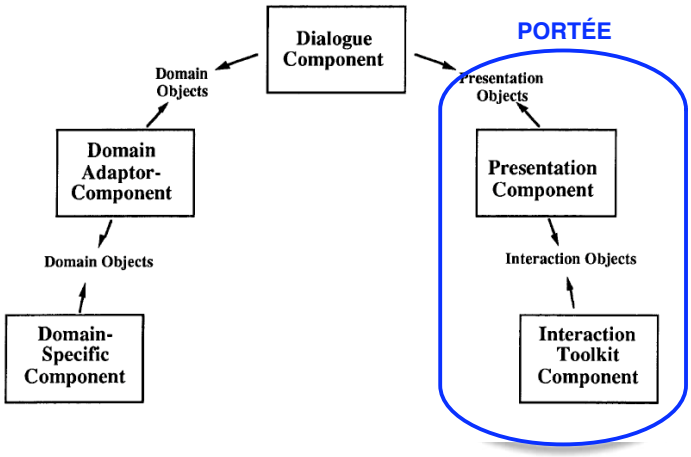


FIGURE 4.2 – Portée de notre modèle au sein d'un système interactif construit selon le modèle Arch.

4.2.2 Eléments de base de notre modèle conceptuel

Nous définissons les éléments de base de notre modèle conceptuel : le composant, le port, la connexion et l'assemblage de composants.

Le composant

Nous nous intéressons au composant en tant qu'unité logicielle. Bien que le composant soit au centre de notre approche, nous ne souhaitons pas figer au sein d'un modèle conceptuel les aspects d'implémentation du composant, tels que l'instanciation, le comportement à l'exécution ou la communication entre composants. Nous adoptons une définition conceptuelle d'un composant logiciel pour pouvoir envisager l'implémentation selon différentes approches existantes, telles que JavaBeans [EJB, 2009], Corba Component Model [Corba, 2002] ou Component Object Model

(COM) [COM, 2009]. Nous souhaitons aussi que notre approche puisse également être appliquée à des solutions modulaires autres que celles basées à composants, telles que les approches à agents (section 3.3.3) ou les approches à services (section 3.3.3). Par exemple, notre modèle conceptuel pourrait s'implémenter avec Open Agent Architecture [Cohen *et al.*, 1998], une plate-forme à agents, ou avec OSGi [Tavares et Valente, 2008], une plate-forme à services.

Aussi, nous reprenons la définition largement adoptée d'un composant fournie par Clemens Szyperski [2003] : **Un composant est une unité logicielle de calcul, interagissant avec son environnement uniquement au moyen de ports** (Figure 4.3).

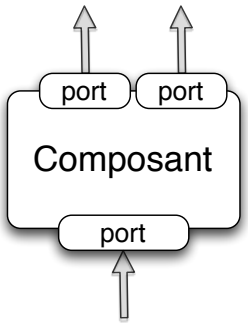


FIGURE 4.3 – Un composant, unité logicielle de notre modèle conceptuel.

Le port

Un port représente une interface d'un composant. Un port est défini par un nom, un sens (entrée ou sortie) et par une opération. Nous utilisons le terme "opération" afin de différencier l'"opération" d'un port de la "fonction de calcul" d'un composant, qui définit son comportement (un composant étant une unité logicielle de calcul). Nous considérons qu'un port ne contient qu'une seule opération. Une opération est définie par un nom et une liste de paramètres. Chaque paramètre est déterminé par un nom et un type (entier, booléen, etc.). Un composant peut avoir plusieurs ports en entrée et en sortie.

Nous illustrons cette définition d'un port avec un composant logiciel dont la fonction de calcul consiste à fournir la position x et y de la souris. Le composant correspondant Souris (Figure 4.4) a un port en sortie dont l'opération est *position* avec comme paramètres deux entiers x et y. Le composant possède également des ports en entrée, l'un pour démarrer, l'autre pour arrêter sa fonction de calcul (ports *start* et *stop* sans paramètre à la Figure 4.4).

La connexion

Une connexion est un lien entre un port en sortie d'un composant et un port en entrée d'un autre composant. La connexion doit spécifier la correspondance entre les paramètres du port en sortie et les paramètres du port en entrée.

Pour illustrer une connexion entre deux composants, nous considérons un composant qui implémente un jeu de pilotage d'hélicoptère où l'utilisateur doit diriger la hauteur de l'engin afin d'éviter les obstacles. Le composant du jeu possède un

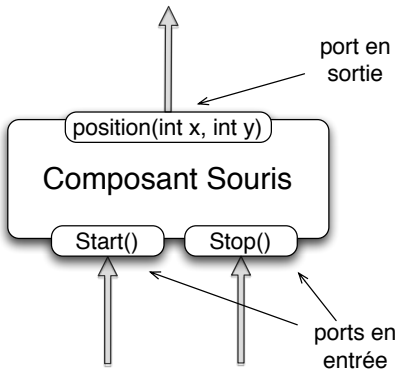


FIGURE 4.4 – Exemple : ports en entrée et en sortie du composant Souris.

port en entrée qui permet de spécifier la hauteur de l'hélicoptère, composé avec un seul paramètre *h*. La connexion entre le port en sortie du composant Souris et le port en entrée du composant Jeu doit spécifier quel paramètre parmi les deux paramètres du port en sortie (x et y) sera utilisé pour définir la valeur du paramètre h.

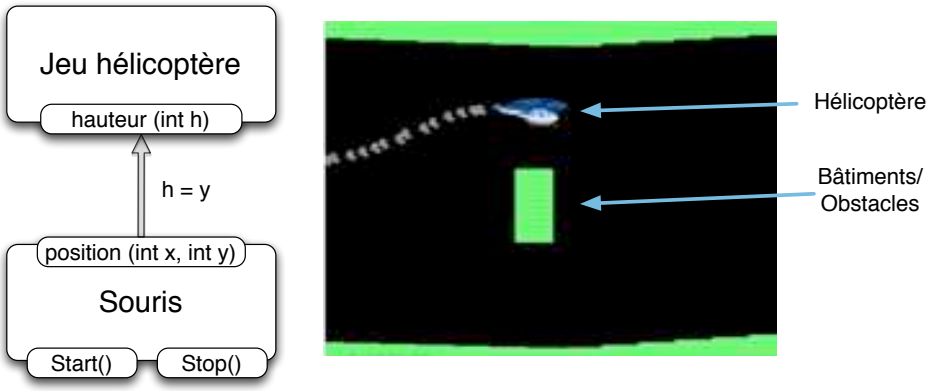


FIGURE 4.5 – Exemple : connexion entre un port en entrée et un port en sortie.

L'assemblage de composants

Un assemblage de composants est un ensemble de composants connectés à travers leurs ports en entrée/sortie. Un assemblage se définit donc par une liste des composants et par une liste des connexions. Un assemblage de composants définit la partie interactive du système multimodal, comme illustré à la Figure 4.6.

4.2.3 Principes de conception liés à notre modèle conceptuel

Notre modèle conceptuel repose sur les notions de base de composant, connexion et assemblage de composants. Pour augmenter le pouvoir génératif de notre modèle conceptuel (sa capacité à produire différentes solutions d'interaction multimodale), nous appliquons les trois principes de conception énoncés par Beaudouin-Lafon

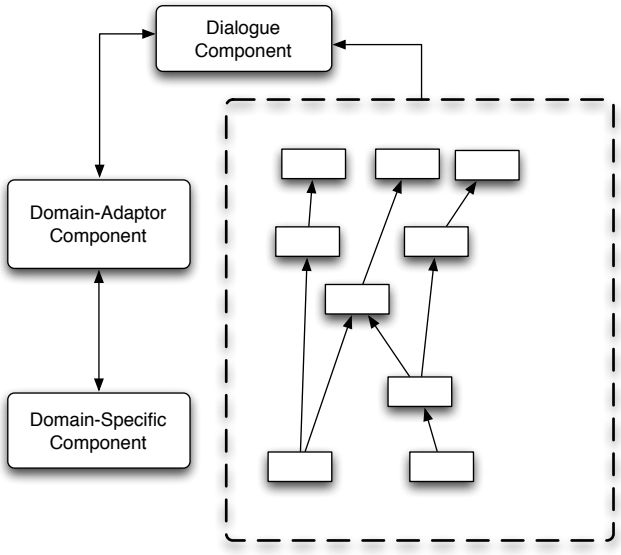


FIGURE 4.6 – Assemblage de composants au sein d'un système interactif construit selon le modèle Arch.

[2004]. Ces trois principes de conception sont appliqués au cas des interfaces graphiques [Beaudouin-Lafon et Mackay, 2000].

- La **réification** : représente la façon dont les concepts sont convertis en objets. Par exemple, dans l'interaction instrumentale, les commandes sont réifiées en instruments d'interaction.

- Le **polymorphisme** : permet d'appliquer une même commande à différents objets. Par exemple, les commandes copier et coller peuvent être appliquées à différents objets graphiques.

- La **réutilisation** : consiste à réutiliser des commandes en entrée antérieures dans un nouveau contexte. Par exemple, la commande coller permet de réutiliser le résultat d'une commande en entrée antérieure.

Nous nous approprions ces trois principes pour les appliquer à notre modèle conceptuel afin d'en augmenter son pouvoir génératif (Figure 4.7).

Au sein de notre modèle, la **réification** désigne la façon dont les concepts sont convertis en composants. La réification est directement liée à la structuration de l'assemblage de composants. La **réutilisation** consiste à réutiliser un composant donné dans différents assemblages de composants .

Pour la réutilisation, nous définissons deux axes orthogonaux qui caractérisent le degré de réutilisabilité d'un composant (axes réutilisation à la Figure 4.7), que nous détaillons à la section 4.4. Pour la réification, nous définissons une typologie des composants d'un assemblage (axe structural à la Figure 4.7), que nous décrivons dans la section suivante.

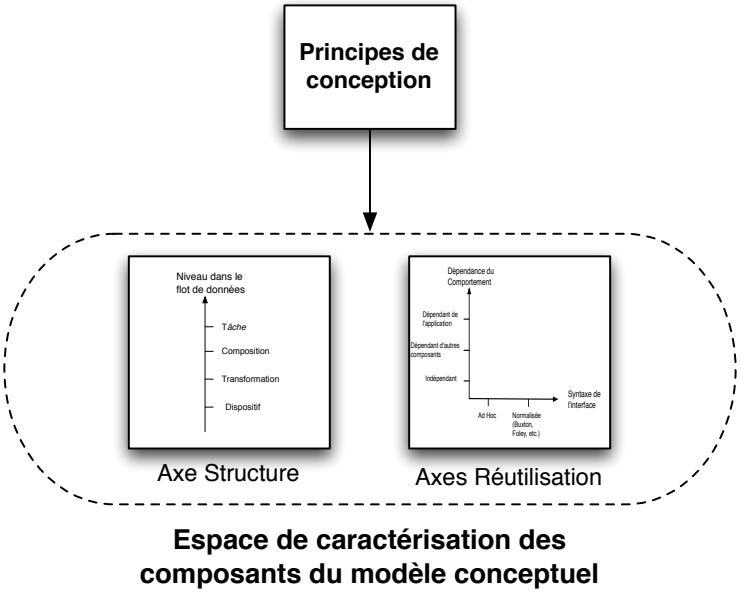


FIGURE 4.7 – Principes de conception de notre modèle conceptuel.

4.3 Structuration d'un assemblage de composants

Pour définir la structuration d'un assemblage, nous introduisons différents types de composants. Pour cela, nous identifions quatre niveaux dans le flot de données des dispositifs aux tâches : dispositif, transformation, composition et tâche. Ces niveaux constituent la *réification* des concepts du domaine de la multimodalité en des composants logiciels. La Figure 4.8 illustre ces niveaux en considérant un assemblage de composants au sein du modèle Arch.

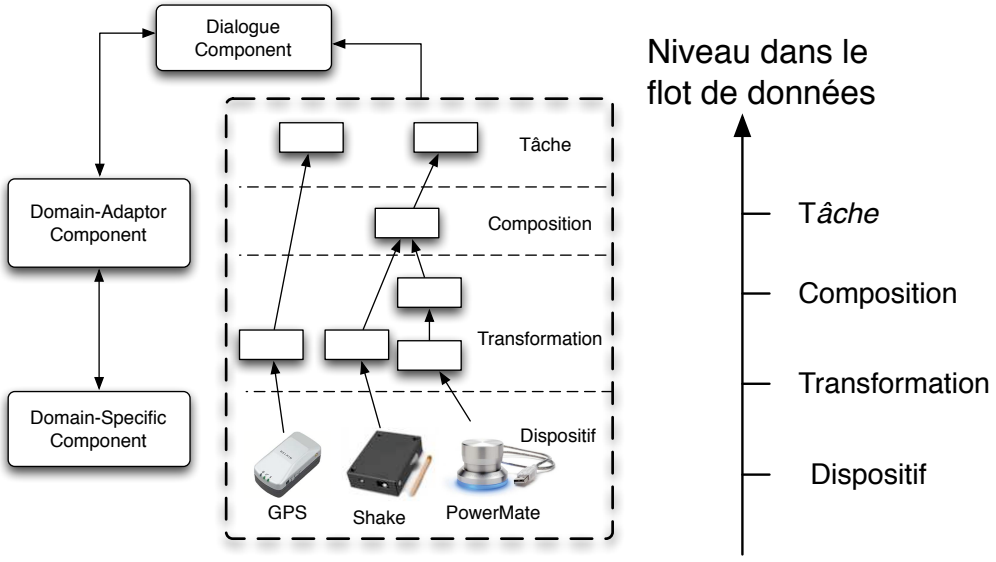


FIGURE 4.8 – Niveaux dans le flot de données des dispositifs en entrée vers les tâches, au sein d'une architecture Arch.

Nous justifions cette typologie des composants du modèle avant de détailler chaque type de composants.

4.3.1 Motivations

La structuration de l'assemblage des composants est essentielle à la conception rapide d'interfaces multimodales. Elle permet d'organiser les idées de conception, de séparer les problématiques selon différents niveaux d'abstraction et favorise aussi la réutilisation de composants, au sein d'une classe de composants. Nous justifions notre typologie en considérant les deux modèles existants Icon et Icare.

Limitations des deux modèles existants

Icon est basé sur des assemblages peu structurés, des dispositifs aux tâches. Cela a pour effet négatif de ne pas fournir au concepteur un guide de conception pour construire des assemblages. En conséquence les assemblages dans Icon sont souvent complexes et déstructurés, comme illustré à la Figure 4.9.

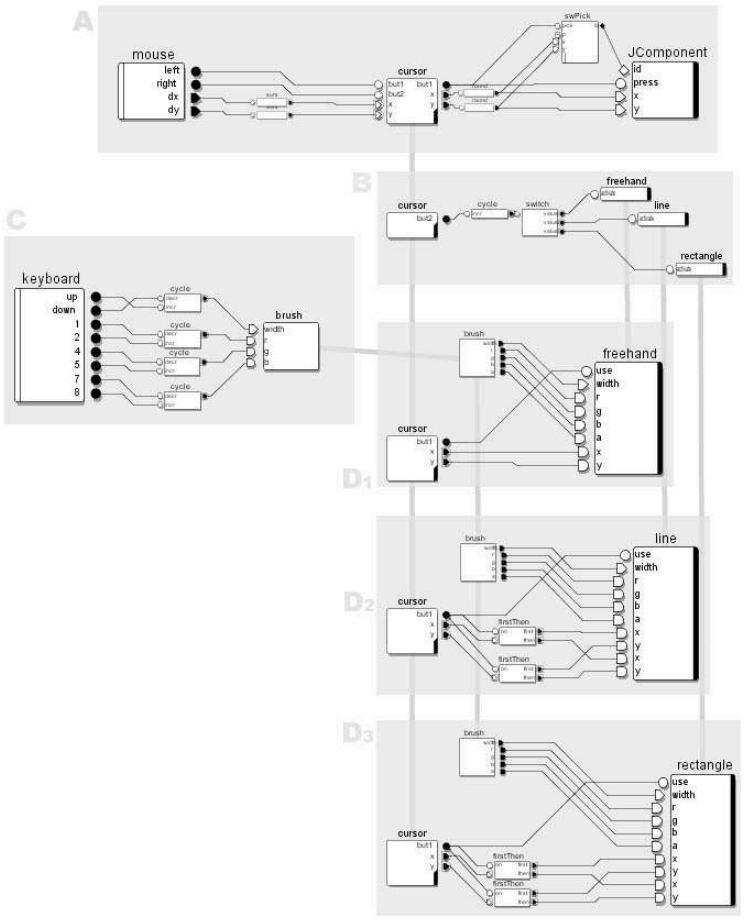


FIGURE 4.9 – Exemples d'assemblages complexes et peu structurés dans Icon. Illustration extraite de [Dragicevic, 2004].

Le modèle Icare propose un assemblage plus structuré, ce qui favorise le pouvoir génératif du modèle. La structuration d'un assemblage Icare repose sur trois

niveaux : Dispositif, Langage d'Interaction et Composition (section 3.4.3 du chapitre 3). Néanmoins, le niveau du Langage d'Interaction dans le modèle Icare propose une granularité de trop haut-niveau d'abstraction, ce qui limite le pouvoir génératif du modèle ainsi que la réutilisabilité des composants Langage d'Interaction. L'exemple de la Figure 4.10 illustre cet aspect : le composant Langage d'Interaction fournit la localisation et l'orientation de l'avion à partir des données du dispositif de localisation. Ce composant Langage d'Interaction est de trop haut niveau d'abstraction et fortement dépendant du dispositif, ce qui rend le composant peu réutilisable dans d'autres contextes.

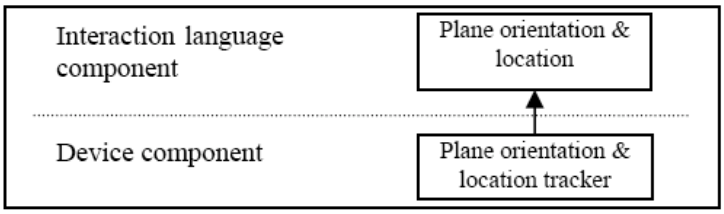


FIGURE 4.10 – Exemple de composant Langage d'interaction de haut niveau d'abstraction dans Icare. Illustration extraite de [Bouchet *et al.*, 2004].

La Figure 4.11 illustre de façon schématique les deux types de structures, Icon et Icare. Les flèches représentent le flux de données. Le flux de données dans Icare est structuré, comprenant deux flux qui fusionnent au niveau des composants de composition, alors que le flux de données dans Icon est plus linéaire et la fusion des flux de données n'est pas explicitée.

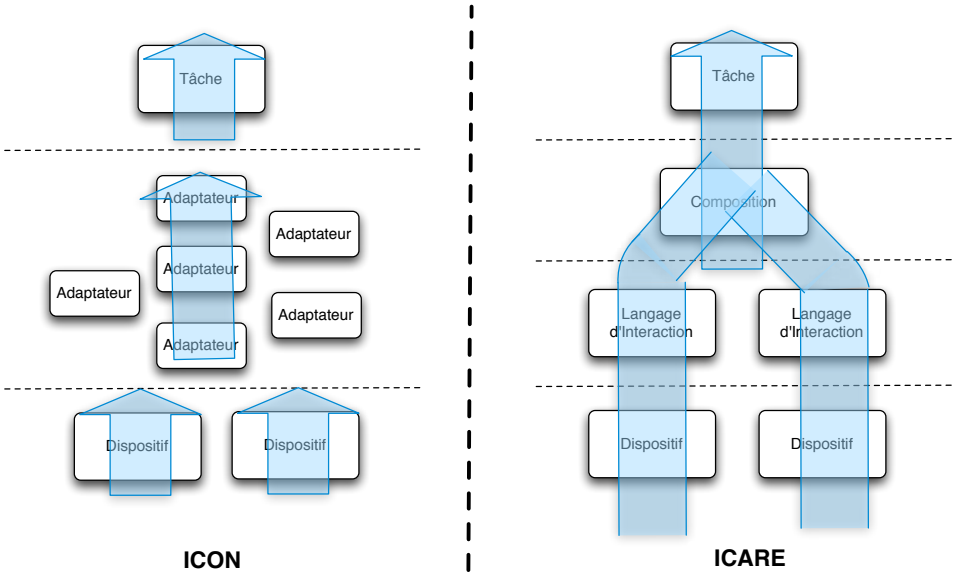


FIGURE 4.11 – Structure du flux de données dans les assemblages de composants Icare et Icon.

Structure mixte de notre approche

Nous adoptons une structure mixte, qui intègre la structuration globale de Icare et le modèle en cascade de Icon. Cette approche permet d'augmenter le pouvoir de génération de notre modèle grâce à l'utilisation du modèle en cascade de Icon, tout en conservant le pouvoir expressif de Icare. Notons que dans notre structure nous permettons la fusion de données au niveau Transformation. Ce type de fusion correspond à une fusion à bas niveau d'abstraction, comme par exemple la fusion de deux flux vidéo afin de générer un flux composé.

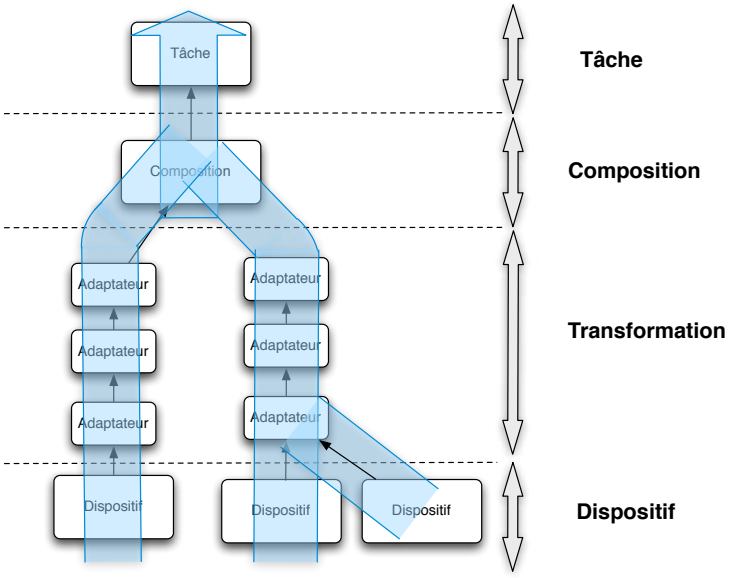


FIGURE 4.12 – Notre modèle : une structure mixte pour l'assemblage de composants.

Ayant justifié notre typologie au regard de Icon et Icare, nous détaillons maintenant les différents types de composants : dispositif, transformation, composition et tâche.

4.3.2 Niveau Dispositif

Le premier niveau dans le flot de données des dispositifs en entrée vers les tâches correspond au Niveau Dispositif. Ce niveau inclut les différents composants qui définissent les dispositifs en entrée, que nous appelons composants Dispositif. Nous identifions deux types de composants Dispositif : les composant Dispositif Physique et les composants Dispositif Logiciel.

Un composant Dispositif Physique représente un dispositif physique donné et permet de récupérer les données bas-niveau que ce dispositif peut fournir. Les dispositifs physiques en entrée incluent tous les dispositifs, qu'ils nécessitent une action explicite ou implicite de l'utilisateur, comme la souris, le GPS, le PowerMate (bouton de contrôle rotatif) ou le Shake (un boîtier sans fil contenant plusieurs capteurs dont un accéléromètre [Williamson *et al.*, 2007]) (Figure 4.13). Nous définissons les caractéristiques de ces composants Dispositif en exploitant les taxonomies existantes, comme celle de Buxton [1983] ou celle de Card [1991], présentées à la

section 3.2.2. Ainsi, pour chaque dispositif nous identifions la propriété capturée (position, pression ou mouvement), le nombre de dimensions et le domaine des valeurs entre autres.

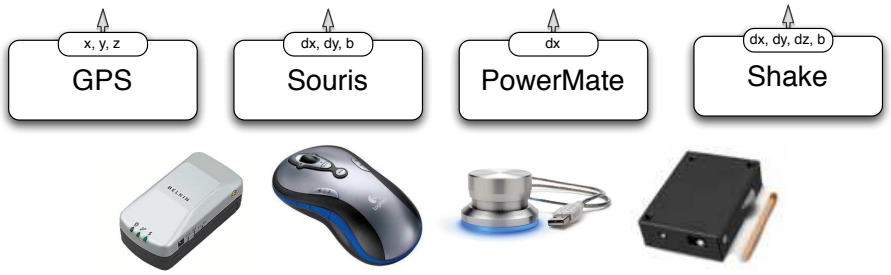


FIGURE 4.13 – Différents dispositifs physiques en entrée. De gauche à droite : gps, souris, powermate et shake.

Un composant Dispositif Logiciel est un composant qui représente une fonctionnalité du type <dispositif physique, traitement logiciel>, lorsque dispositif physique et traitement logiciel sont imbriqués. Des exemples de tels dispositifs concernent l'exploitation d'images et de sons dans l'interaction en entrée (Figure 4.14). Nous motivons ce type de dispositif selon deux aspects complémentaires : l'un pragmatique, l'autre lié aux performances. Premièrement, les solutions basées sur ces dispositifs sont souvent monolithiques, utilisant une caméra donnée, et ne permettant pas de séparer dispositif et traitement. Deuxièmement, l'information captée par ces dispositifs est complexe et lourde à traiter en raison de sa taille (matrice de pixels) et de la fréquence de captation. Dans une approche à composants et pour des raisons de performance, il convient de limiter la taille des événements échangés entre composants.

FIGURE 4.14 – Composant Dispositif logique FingerTracker, composé par une caméra et un algorithme de reconnaissance des doigts [Letessier et Bérard, 2004].

4.3.3 Niveau Transformation

Le niveau suivant dans le flot de données des dispositifs vers les tâches est le niveau Transformation. Ce niveau traite et transforme les données obtenues

des dispositifs. Ce niveau comprend des composants de Transformation. Le niveau Transformation est décrit en utilisant le modèle des dispositifs en cascade [Dragicevic, 2004], présenté à la section 3.2.3. En effet, nous considérons que ce modèle offre la bonne granularité à ce niveau de l'assemblage car le traitement des données en provenance des dispositifs en entrée n'est que rarement une opération monolithique.

Notre approche n'impose néanmoins pas de granularité spécifique pour les composants de ce niveau. Des composants de bas-niveau d'abstraction, tels que des opérations mathématiques, des filtres ou des opérations de mise en correspondance offrent un bon pouvoir de génération à notre modèle. Des composants de plus haut-niveau d'abstraction, dépendants ou non d'un ou plusieurs dispositifs ou de la sémantique de l'application interactive, permettent de traiter des problèmes spécifiques, même si ces composants seront moins réutilisables, comme les composants Langage d'Interaction d'Icare.

Parmi les composants du niveau Transformation, certains peuvent opérer des fusions de flots de données en provenance des dispositifs comme nous illustré à la Figure 4.12. Par exemple, le flux de deux dispositifs Caméra peuvent être fusionnés au niveau Transformation afin de générer un flux combiné (dont la taille correspond à l'addition des tailles des deux flux) qui sera ensuite envoyé vers d'autres composants de Transformation. Cette fusion de flots de données à bas-niveau d'abstraction relève de traitements du signal. Ce type de fusion est différent de la composition multimodale opérée au niveau Composition, que nous présentons dans la section suivante.

4.3.4 Niveau Composition

Le niveau Composition permet de réaliser la fusion des données venant des différentes modalités d'interaction afin de définir des commandes multimodales. Afin de définir des composants de composition réutilisables, nous utilisons les propriétés CARE [Coutaz et Nigay, 1994], présentées à la section 3.2.5 et déjà utilisées dans l'approche Icare.

Ainsi, comme dans Icare, nous définissons quatre composants à partir des propriétés CARE : le composant Complémentarité, le composant Redondance, le composant Equivalence et le composant Redondance-Equivalence. L'assignation (le A dans CARE) n'est pas traduit en composant car il équivaut à une connexion unique entre deux composants.

Chaque composant réalise la fusion de deux modalités en entrée et possède donc deux ports en entrée. Nous considérons les événements selon une modalité en entrée soit comme instantanés (qui ont lieu à un instant t) soit comme ayant une durée (comme le temps d'énoncé oral d'une commande). Les composants de fusion analysent le moment d'arrivée des événements en entrée afin de générer l'événement composé en sortie. Pour définir le traitement de ces composants au regard des aspects temporels, nous nous appuyons sur les propriétés d'Allen [Allen, 1983], introduites à la section 3.2.5 et schématisées à la Figure 4.15. Selon le type de composition et le moment d'arrivée du deuxième événement, le composant de composition réalise ou non la fusion des données.

L'interaction multimodale peut donc être classée en deux grandes catégories temporelles : les deux modalités sont utilisées de façon séparée temporellement

(cas A sur la Figure 4.15) ou bien elles se recouvrent temporellement (cas B sur la Figure 4.15).

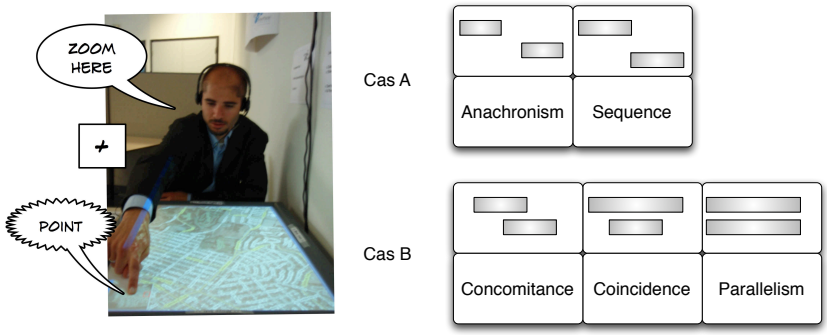


FIGURE 4.15 – Une même interaction multimodale selon deux cas de temporalité : le cas A représente une séparation temporelle dans l'usage des modalités (anachronisme ou séquence) ; le cas B représente un recouvrement temporel dans l'usage des modalités sous trois formes : concomitance, coïncidence ou parallèlisme.

Pour définir un composant de composition, il convient donc de définir à la fois le type de composition (CARE) mais aussi la relation temporelle entre deux événements. Ainsi, le composant Redondance attend que les événements des deux ports en entrée arrivent de façon séquentielle (cas A sur la Figure 4.15) ou de façon concomitante, coïncidente ou parallèle (cas B sur la Figure 4.15) afin de générer l'événement en sortie, qui équivaut au premier événement reçu. Le composant Equivalence envoie tout événement reçu, peu importe sa temporalité. Le composant Complémentarité attend que deux événements arrivent de façon séquentielle, concomitante, coïncidente ou parallèle pour générer un événement en sortie composé des deux événements en entrée. Finalement, le composant Redondance-Equivalence renvoie tout, comme le composant Equivalence, mais dans le cas où les deux événements sont redondants, il ne renvoie qu'un seul.

Le résultat de la fusion dépend donc de l'utilisation dans le temps des modalités d'interaction par l'utilisateur mais aussi du type de fusion choisi lors de la conception.

4.3.5 Niveau Tâche

Le niveau Tâche constitue le lien avec le reste de l'application interactive (Figure 4.8). Chaque assemblage de composants permet de définir une tâche interactive. En nous référant au modèle Arch (Figures 4.6 et 4.8), nous soulignons qu'un assemblage de composants ne gère pas l'enchaînement des tâches, géré par le contrôleur de dialogue.

Nous identifions trois types de composants tâche :

- Le composant Tâche qui réalise le lien avec le reste du système interactif à travers un contrôleur de dialogue (externe à l'assemblage). La tâche correspond alors à une commande élémentaire que peut gérer une application.
- Le composant Tâche qui simule une entrée d'interaction, généralement une commande souris ou clavier. Nous utilisons alors des modalités équivalentes à

des actions souris et clavier. Ce cas permet de manipuler des applications déjà existantes avec des nouvelles modalités d'interaction sans modifier le code de l'application.

- Le composant Tâche qui constitue lui-même une application interactive. Ce type de composant est réalisé pour implémenter des applications simples sous la forme d'un composant. La Figure 4.16 illustre un composant Tâche dans Icon qui implémente une application de dessin à main levée.

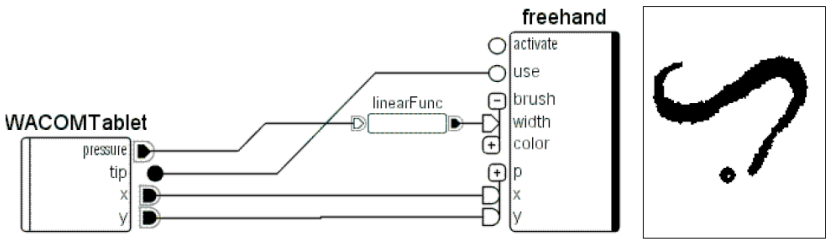


FIGURE 4.16 – Composant Tâche qui implémente une application de dessin dans Icon. Illustration extraite de [Dragicevic et Fekete, 2004].

4.3.6 Synthèse

Notre modèle définit quatre types de composants au sein d'un assemblage. L'assemblage résultant allie les deux modèles Icon et Icare. Notre modèle mixte permet donc un fort pouvoir descriptif issu de Icare et un fort pouvoir génératif de Icon.

Cette typologie des composants définit le principe de réification au sein de notre modèle (section 4.2.3). Nous étudions maintenant notre deuxième principe de conception, la réutilisabilité d'un composant.

4.4 Réutilisation d'un composant de notre modèle

Nous présentons d'abord nos motivations pour caractériser le degré de réutilisabilité des composants avant de présenter les axes qui composent cette caractérisation.

4.4.1 Motivations

Un des principaux enjeux de l'utilisation d'une approche à composants repose sur la réutilisation de composants déjà existants, c'est-à-dire sur sa capacité d'intégration. La capacité d'intégration des composants concourt à faciliter l'exploration rapide de différentes solutions d'interaction et donc de l'espace de conception.

Dans le contexte de l'interaction multimodale, nous constatons une très grande hétérogénéité des solutions et donc des composants, ces derniers étant définis par des équipes aux compétences différentes (par exemple spécialiste de la parole, spécialiste de la localisation) voir par des domaines de recherche distincts (domaine du traitement du signal pour la vison par ordinateur, domaine de la reconnaissance de parole ou encore domaine de l'IHM). La principale conséquence est que les composants manipulés présentent des interfaces et des comportements variés.

Deux approches d'intégration sont alors possibles : forcer les composants à répondre à une spécification commune ou bien adopter une approche ouverte capable d'intégrer des composants variés. La première solution permet de simplifier l'assemblage des composants mais rend plus difficile l'intégration de nouveaux composants. La deuxième solution va complexifier l'assemblage des composants mais permet d'intégrer facilement n'importe quel composant (Figure 4.17).

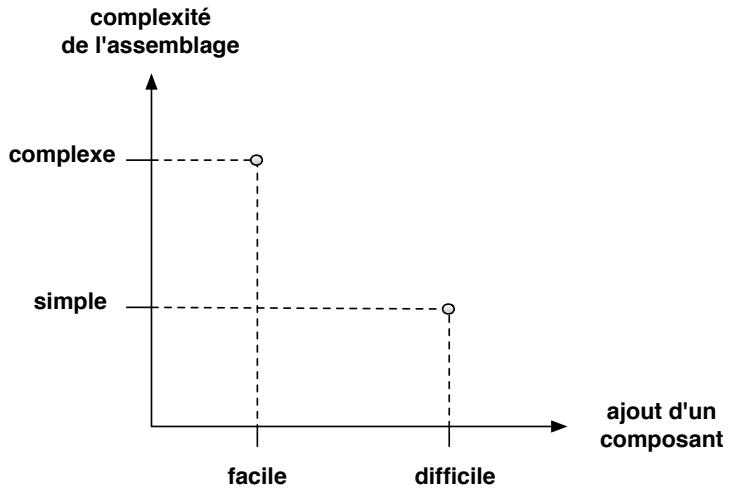


FIGURE 4.17 – Relation entre la complexité d'assemblage des composants et la facilité d'ajout des composants dans une approche à composants.

Dans notre modèle, nous adoptons la deuxième solution afin de permettre d'exploiter de nombreux composants existants pour explorer des techniques d'interaction innovantes. Néanmoins nous caractérisons le degré de réutilisabilité d'un composant afin d'aider le concepteur lors de l'assemblage de composants.

Nous introduisons dans la section suivante la définition que nous avons adoptée afin de caractériser la réutilisabilité d'un composant.

4.4.2 Définition de la réutilisabillité d'un composant

Définir la réutilisabilité d'un composant consiste à définir les aspects qui rendent ce composant compatible avec d'autres composants dans un assemblage. Différents aspects [Beugnard *et al.*, 1999] interviennent dans la compatibilité d'un composant avec d'autres et donc en définissent sa réutilisabilité :

- **Aspect comportemental** : Dans une séquence d'opérations réalisées par différents composants dans un assemblage, il est important de déterminer la dépendance du traitement d'un composant avec un autre ou avec un élément extérieur à l'assemblage.

- **Aspect syntaxique** : L'aspect syntaxique concerne l'interface du composant. La compatibilité syntaxique entre opérations offertes d'un composant et celles requises d'un autre composant consiste à déterminer si les types des paramètres de ces opérations sont dans une relation de sous-typage.

Ainsi, nous définissons deux axes pour caractériser le degré de réutilisation des composants : un axe qui décrit la dépendance du comportement du composant et un axe qui décrit la forme syntaxique de l'interface du composant (Figure 4.18).

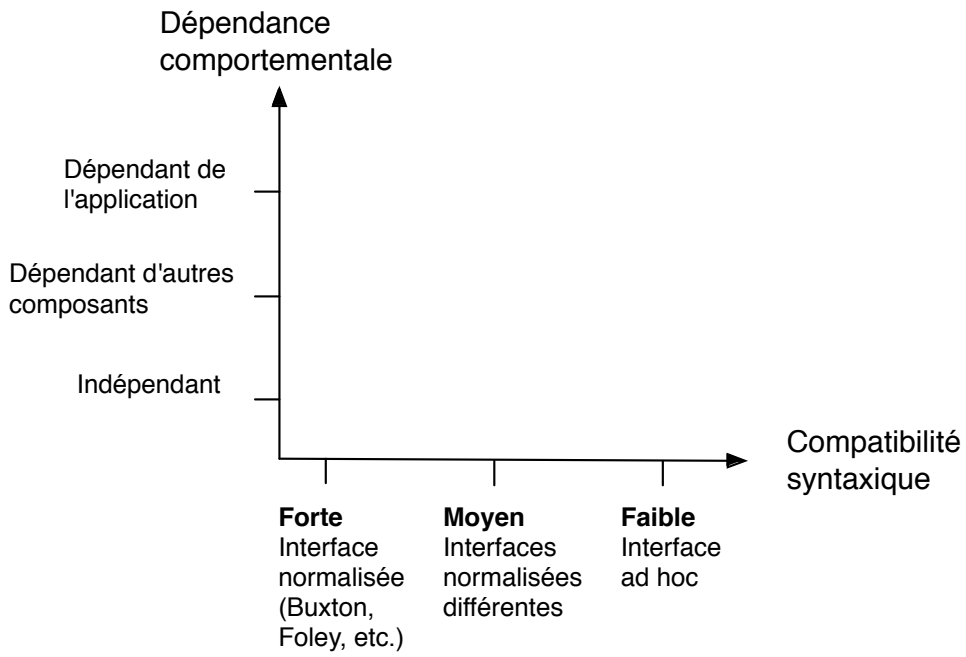


FIGURE 4.18 – Dimensions de caractérisation du degré de réutilisabilité des composants de notre modèle.

4.4.3 Axe Dépendance comportementale

En Génie Logiciel, les dépendances d'un composant indiquent ses services requis. Ainsi, un composant avec des dépendances ne peut pas fonctionner sans les services requis. Un composant qui possède une dépendance comportementale est donc conçu pour être utilisé dans le contexte de cette dépendance.

Dans notre modèle, la dépendance envers un tiers indique un lien entre la logique du comportement du composant et un tiers. La dépendance comportementale représente donc le lien entre la fonction de calcul du composant (son comportement) et le reste du système interactif.

Au sein de notre modèle, cela nous conduit à définir trois valeurs sur l'axe Dépendance comportementale (Figure 4.18) :

- dépendant de l'application (extérieur à l'assemblage), comme par exemple le composant A sur la Figure 4.19 ;
- dépendant d'un autre composant de l'assemblage, comme le composant B sur la Figure 4.19 ;
- indépendant.

Nous illustrons les trois cas identifiés sur l'axe Dépendance Comportementale. Ces trois cas caractérisent le degré de réutilisation d'un composant dans un contexte (application, assemblage) donné.

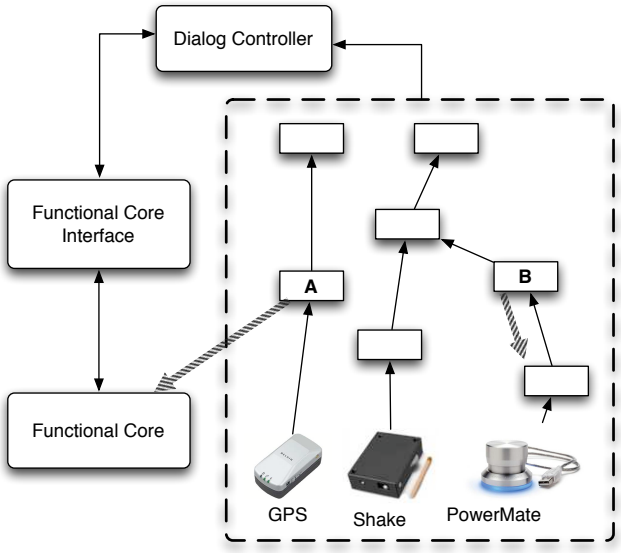


FIGURE 4.19 – Illustration de la dépendance : composant dépendant de l'application (A) et composant dépendant d'un autre composant (B).

Dépendant de l'application

Les composants dépendants des applications implémentent des fonctionnalités liées à l'espace problème d'une application donnée ou d'un type d'applications donné. La Figure 4.20 illustre un composant Icon qui modifie le focus dans une application Swing. Ce composant est donc dépendant des applications Swing.

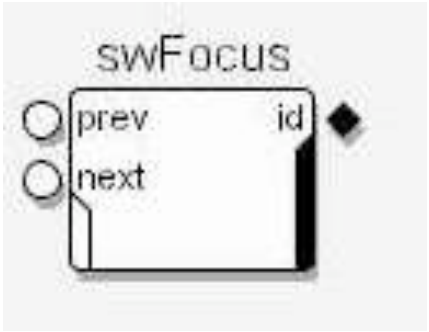


FIGURE 4.20 – Composant dépendant de l'application (qui modifie le focus dans une application Swing) dans Icon. Illustration extraite de [Dragicevic, 2004].

Dépendant d'un autre composant

Le comportement d'un composant peut dépendre d'autres composants au sein de l'assemblage. Cela est par exemple le cas avec les composants de Transformations, dépendants de composants Dispositif. Les composants dépendants des dispositifs en entrée implémentent des fonctionnalités liées à un dispositif donné ou à un ensemble de dispositifs donnés. Par exemple, un composant qui implémente une reconnaissance de gestes à partir de données en trois dimensions d'un accéléromètre

est un composant dépendant des dispositifs de type accéléromètre. Ce composant ne fonctionnerait pas avec un autre type de dispositif, même ceux fournissant des données en trois dimensions comme les gps, étant donné que la reconnaissance de gestes réalisée est prévue pour des valeurs fournies par un accéléromètre lorsqu'il est manipulé par un utilisateur.

La Figure 4.21 illustre un exemple avec un assemblage Icare où un composant Langage d'Interaction implémente une opération spécifique pour un dispositif Hotas[2]. Ce composant de transformation analyse les données du Hotas afin de générer des commandes de pilotage. Ce composant est dépendant du dispositif car il est spécifiquement prévu pour traiter les données du Hotas afin de générer les commandes.

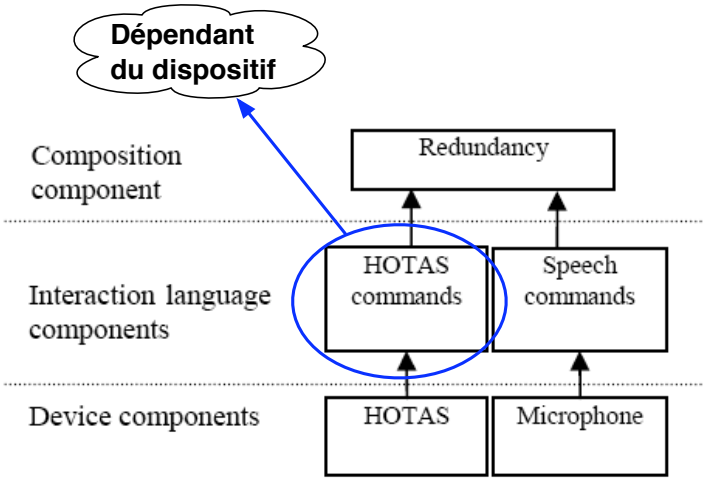


FIGURE 4.21 – Assemblage Icare avec un composant dépendant d'un autre composant. Assemblage extrait de [Bouchet *et al.*, 2004].

Indépendant

Un composant indépendant implémente une opération qui n'est pas liée ni aux applications ni aux autres composants. Les composants de composition CARE (section 3.2.5 du chapitre 3) sont des exemples de composants indépendants : leur comportement (la fusion temporelle des données en entrée) ne dépend pas d'une application donnée ou d'autres composants dans l'assemblage. La fusion est réalisée avec des données en entrée provenant d'un couple quelconque de composants. D'autres exemples de composants indépendants incluent les composants de transformation qui effectuent des traitements bas-niveau, comme des opérations mathématiques, telles que le filtrage. La Figure 4.22 illustre ce type de composants en considérant des composants Icon.

4.4.4 Axe Compatibilité syntaxique

La syntaxe de l'interface des composants est un élément important pour assurer la compatibilité syntaxique des composants. Nous identifions deux types pour les

[2]Hands On Throttle-And-Stick (Hotas), traduit par Main sur manche et manette, est le dispositif de contrôle des avions de combat, où le pilote tient le manche avec la main droite et une manette avec la main gauche.

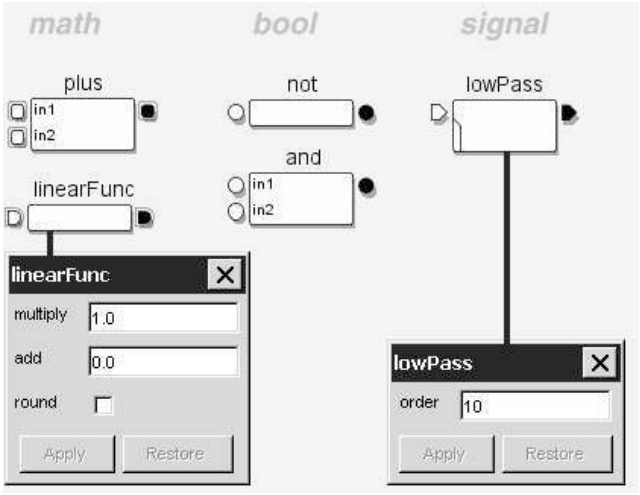


FIGURE 4.22 – Composants de traitement indépendants dans Icon : opérations élémentaires mathématiques, booléennes et de traitement du signal. Illustration extraite de [Dragicevic, 2004].

interfaces des composants de notre modèle : ad hoc et normalisée. Une interface d'un composant qui a une *syntaxe ad hoc* est constituée d'un ensemble de paramètres avec des types de données, sans respecter aucune convention. Dans ce cas, le composant est faiblement compatible avec les autres composants. Pour pouvoir le connecter à un autre composant, un composant de Transformation devra sans doute être rajouté dans l'assemblage. Ce composant Transformation sera alors dépendant du composant à l'interface ad hoc (valeur *dépendant d'un autre composant* sur l'axe Dépendance Comportementale). En effet, son traitement sera dépendant de l'interface de l'autre composant. Au contraire, normaliser la syntaxe de l'interface des composants de notre modèle permet de facilement les connecter au sein d'un assemblage.

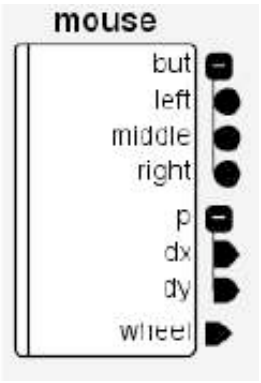


FIGURE 4.23 – Exemple de composant Dispositif Souris avec une interface dont la syntaxe est ad hoc. Illustration extraite de [Dragicevic et Fekete, 2004].

Ces deux types d'interface (ad hoc et normalisée) permettent l'identification de

différents degrés de compatibilité syntaxique entre composants[3]. Ainsi, deux composants avec la même syntaxe normalisée seront totalement compatibles syntaxiquement. Deux composants avec des syntaxes normalisées différentes (par exemple une syntaxe basée sur la taxonomie de Buxton [1983] et une syntaxe basée sur la taxonomie de Foley [1984]) seront faiblement compatibles, mais des mécanismes de traduction syntaxique automatique sont envisageables afin de les rendre compatibles. Un composant ad hoc sera faiblement compatible syntaxiquement avec n'importe quel autre type de composant. Ainsi, la connexion de deux composants totalement compatibles ou compatibles par traduction peut être automatisée. Par contre, la connexion de composants faiblement compatibles nécessite l'intervention du concepteur de l'assemblage pour spécifier la correspondance des paramètres.

Proposition de syntaxes normalisées

Pour proposer une normalisation des interfaces des composants de notre modèle, nous nous appuyons sur des modèles reconnus en IHM. Ainsi, nous proposons de normaliser l'interface des composants Dispositif en exploitant les traits caractéristiques de la taxonomie des dispositifs de Buxton [1983] et l'interface des composants Tâche en nous reposant sur la taxonomie des tâches interactives de Foley [1984].

Interface normalisée basée sur la taxonomie de Buxton

L'utilisation de la taxonomie de Buxton [1983] (section 3.2.2 du chapitre 3) pour définir une syntaxe de l'interface des composants permet de définir l'interface des composants Dispositif. Évidemment, certains composants Transformation dépendants de composants Dispositif peuvent également utiliser ce type de syntaxe.

La taxonomie de Buxton permet de décrire les dispositifs en entrée manipulés à la main[4]. Nous n'avons pas l'intention de définir une syntaxe qui couvre tout l'espace des dispositifs (sorte d'ontologie des dispositifs qui semble inapproprié face aux avancées technologiques) et préférons ainsi utiliser une taxonomie largement acceptée malgré sa limitation en terme de types de dispositifs considérés par l'approche.

L'interface d'un composant Dispositif basée sur la taxonomie de Buxton permet de définir la propriété p captée par le dispositif (Mouvement, Pression ou Position), ainsi que le nombre de dimensions n de cette propriété (1D, 2D, 3D, etc.). Pour chaque dimension de l'interface, est décrit également le domaine de valeurs d. À ces deux informations, nous ajoutons une information temporelle qui permet d'informer de la fréquence f du flot de données de cette interface. Afin de décrire de manière simple cette fréquence, nous considérons deux types de fréquence pour les données : continue ou discrète.

La grammaire résultante pour définir une interface i est la suivante :

i = [p, n, [(n1, d1), (n2, d2),.., (nn, dn)], f]

Une interface à données continues envoie des données à une certaine fréquence, souvent de façon indépendante des actions de l'utilisateur. Ainsi, le composant qui traite ces données devra tenir compte de cette continuité. Un exemple de dispositif fournissant des informations de façon continue est l'accéléromètre.

[3] «plug-compatible» [Batory et O'Malley, 1992]

[4] «hand-controlled devices» [Buxton, 1983]

Une interface à données discrètes n'envoie des données que lorsqu'une action de l'utilisateur est réalisée. Un exemple de dispositif discret est un bouton. Évidemment, il existe des exceptions comme les boutons continus de la Wiimote, ce qui rend la classification des dispositifs très difficile.

La Figure 4.24 illustre l'interface normalisée des composants Dispositif selon la taxonomie de Buxton en considérant le dispositif Souris. Pour ce composant Souris, nous montrons aussi d'autres possibilités de composants non normalisés.

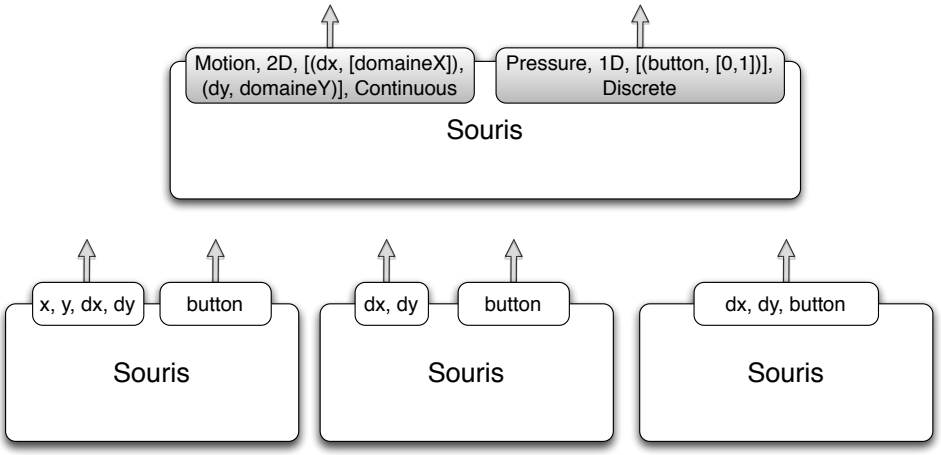


FIGURE 4.24 – Face aux multiples formes des interfaces ad hoc (en blanc), l'interface normalisée (en gris) pour le composant Dispositif Souris.

Syntaxe normalisée basée sur la taxonomie de Foley

Nous définissions une interface propre aux composants Tâche en nous reposant sur la taxonomie de Foley [1984], introduite à la section 3.2.3 du chapitre 3. Rappelons que l'approche avait pour objectif de rendre les tâches indépendantes des dispositifs. Ainsi, l'utilisation de cette taxonomie au niveau des composants Tâche permet de définir des composants hautement réutilisables et indépendants du reste de l'assemblage.

L'interface d'un composant Tâche basée sur la taxonomie de Foley permet de définir le type *t* de la tâche. Ce type peut prendre comme valeur Sélection, Positionnement, Orientation, Cheminement, Quantification ou Saisie de texte. Ces types de tâches d'interaction servent ici à décrire le type d'action attendue par une interface d'un composant tâche. À l'information du type de tâche s'ajoute le nombre de dimensions *n*, le domaine de chaque dimension *d* . Ainsi, une interface *i* décrite selon Foley peut s'écrire :

i = [t, n, [(n1, d1), (n2, d2),.., (nn, dn)]]

La Figure 4.25 illustre l'interface normalisée des composants Tâche selon la taxonomie de Foley en considérant les tâches interactives *Contrôle d'un hélicopère* et *Dessin freehand*, présentées respectivement dans la section 4.2.2 et la section 4.3.5. La tâche *Contrôle d'un hélicopère* peut être définie par une interface normalisée de type Positionnement, à une dimension *h* (la hauteur de l'helicopère). La tâche *Dessin freehand* peut être définie par une interface normalisée de type Cheminement, à deux dimensions *(x,y)*.

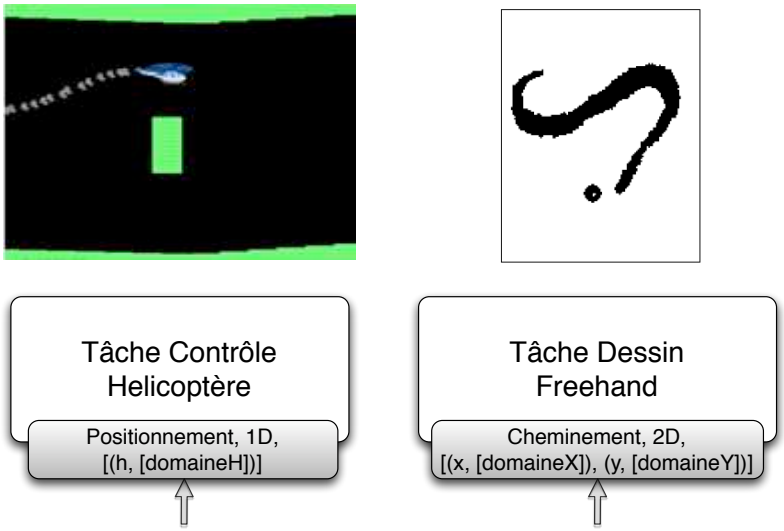


FIGURE 4.25 – Deux exemples de composants Tâche aux interfaces normalisées basées sur la taxonomie de Foley.

4.5 Synthèse du modèle

En synthèse, notre modèle conceptuel pour la définition de solutions d'interfaces multimodales repose sur une approche à composants pour décrire le flot de données par assemblage de composants. Notre modèle offre un fort pouvoir de description et de génération en alliant les avantages des deux autres modèles conceptuels existants Icare et Icon. Pour cela, notre modèle définit une structure de l'assemblage des composants issue des deux modèles Icare et Icon. L'aspect réutilisation d'un composant est essentiel pour explorer efficacement l'espace de possibilités. Pour permettre une plus grande réutilisabilité des composants, nous proposons sans l'imposer des interfaces normalisées basées sur la taxonomie de Buxton [1983] et de Foley [1984].

Nous illustrons maintenant notre modèle conceptuel en considérant un exemple classique de la multimodalité en entrée.

4.6 Illustration

Nous illustrons notre modèle en considérant une interface multimodale du type *Put-That-There* [Bolt, 1980]. Afin d'illustrer les différents aspects de notre modèle, nous décrivons cette interface multimodale avec plusieurs assemblages de composants en considérant des composants aux degrés de réutilisabilité différents.

À des fins d'illustration, nous considérons chaque degré de réutilisabilité des composants, défini selon nos deux axes (Figure 4.18). Ainsi nous proposons un assemblage selon chaque dimension de réutilisabilité identifiée.

4.6.1 Système interactif illustré

Nous utilisons l'exemple du Put-That-There [Bolt, 1980] pour illustrer notre modèle. Nous considérons une version de ce type d'interaction que nous avons implé-

mentée : le Zoom-Here. Cet exemple utilise une table tactile comme dispositif de pointage et un système de reconnaissance vocale pour la commande vocale "zoom here" (Figure 4.26).

FIGURE 4.26 – Système interactif utilisé pour illustrer notre modèle. Le système utilise une table tactile et un système de reconnaissance vocale.

L'application est un navigateur de cartes. Elle utilise, comme repère de la carte, la taille de l'image. Une méthode Zoom permet de centrer la carte sur un point donné (fixé par ses coordonnées dans le repère de la carte et non pas dans le repère de la table). Nous étudions plusieurs assemblages à connecter à cette tâche de zoom (Figure 4.27), qui prend des coordonnées *(x,y,z)* en entrée : les coordonnées *(x,y)* définissent le point de zoom et la coordonnée z définit le niveau de zoom.

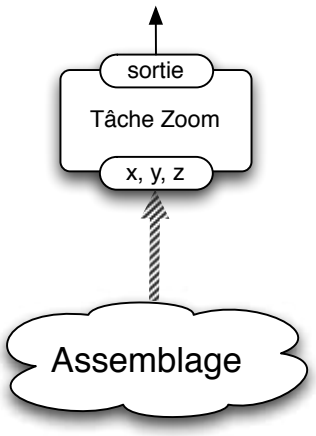


FIGURE 4.27 – Tâche interactive utilisée pour illustrer notre modèle.

4.6.2 Différents assemblages selon la dépendance comportementale

La conception d'un système interactif du type *Put-That-There* avec un assemblage de composants peut prendre de nombreuses formes selon la granularité des composants. Nous illustrons d'abord la dépendance comportementale des composants

avec trois assemblages différents : un assemblage avec des composants indépendants, un assemblage avec des composants dépendants d'autres composants et un assemblage avec des composants dépendants de l'application. Dans cette section, nous simplifions volontairement l'information sur les interfaces des composants afin de faciliter la compréhension des exemples. Les interfaces des composants seront considérées à la section 4.6.3.

Assemblage de composants indépendants

La Figure 4.28 illustre un assemblage de composants indépendants. Le composant Dispositif Table Tactile envoie des données des points de contact avec un identifieur. Si plusieurs doigts sont en contact de la table, chaque doigt a un identifieur différent (le doigt n aura un identifieur *idn*). Le composant envoie les données sous forme d'une chaîne de caractères qui décrit, à un instant t, la position *(x, y)* de tous les doigts en contact de la table : *[(x1, y1, id1), (x2, y2, id2), ... , (xn, yn, idn)]*.

Un premier composant Parser permet ensuite de séparer les descriptions des différents doigts contenus dans la chaîne. Ce composant renvoie donc des chaînes de caractères de la forme *(xn, yn, idn)*. Un deuxième composant 3D Parser (qui prend 3 paramètres en entrée) sépare ensuite les trois paramètres de la chaîne précédente, renvoyant ainsi les trois paramètres *x, y* et *id* de façon séparée. Ces deux composants Parser sont paramétrés à l'aide d'un fichier externe de vocabulaire.

Le composant 2DContrôle joue le rôle d'un interrupteur, bloquant les événements reçus si la condition paramétrable n'est pas satisfaite. Dans l'exemple, le composant 2DContrôle sert à vérifier l'identifieur du doigt détecté (paramètre *id*) afin d'éliminer les détections parasites (lorsque la table envoie des fausses détections ou des détections de contacts involontaires avec d'autres doigts). Ensuite, le composant 2DFoncLinéaire modifie les deux valeurs en entrée selon une opération mathématique paramétrable afin de renvoyer deux événements modifiés. Ce composant sert à transformer les coordonnées données par la table dans le repère de la carte. L'opération mathématique du composant peut être définie par l'utilisateur au lancement du composant. Un composant Filtre permet de filtrer les mots de la reconnaissance vocale afin de générer des commandes, spécifiées dans un fichier externe de vocabulaire.

Au sein de cet assemblage, les composants sont réutilisables d'un point de vue comportemental. De plus, tous les composants sont indépendants de l'application, ce qui permet de réutiliser l'assemblage avec une autre application qui possède une tâche de zoom.

Assemblage de composants dépendants d'autres composants

La Figure 4.29 illustre un assemblage où des composants en gris sont dépendants d'autres composants de l'assemblage. Au niveau Transformation, un composant permet de filtrer les événements de la table afin de récupérer la position d'un seul doigt. Ce composant est dépendant du dispositif étant donné qu'il doit connaître la sémantique utilisée par la table tactile pour coder le contact des doigts. Il intègre donc les opérations correspondantes au parsing et au contrôle de l'identifieur du doigt réalisées par les composants Transformation dans l'exemple précédent.

Avec ce type de composants, certaines opérations sont réalisées de manière plus efficace et adaptée (un seul composant réalisant les opérations de trois composants

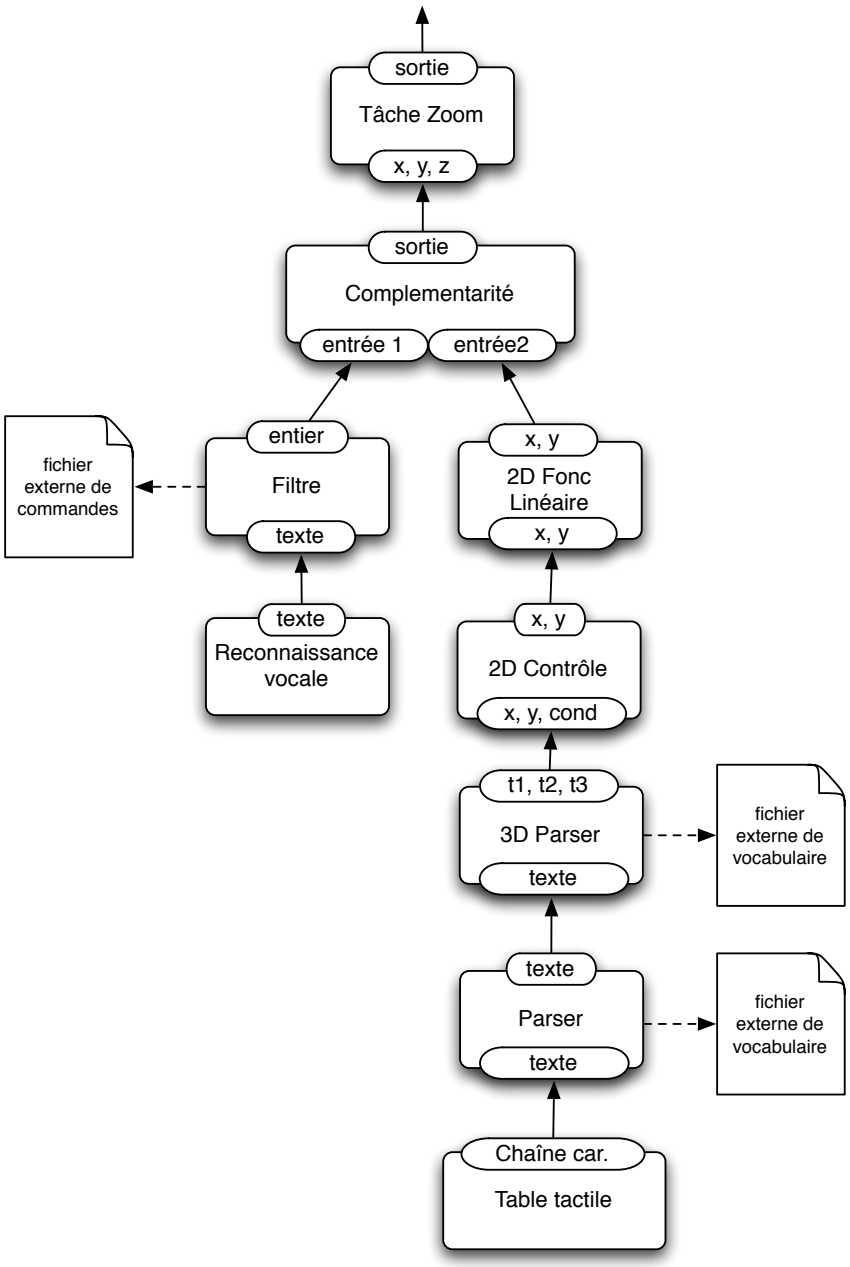


FIGURE 4.28 – Assemblage de composants indépendants.

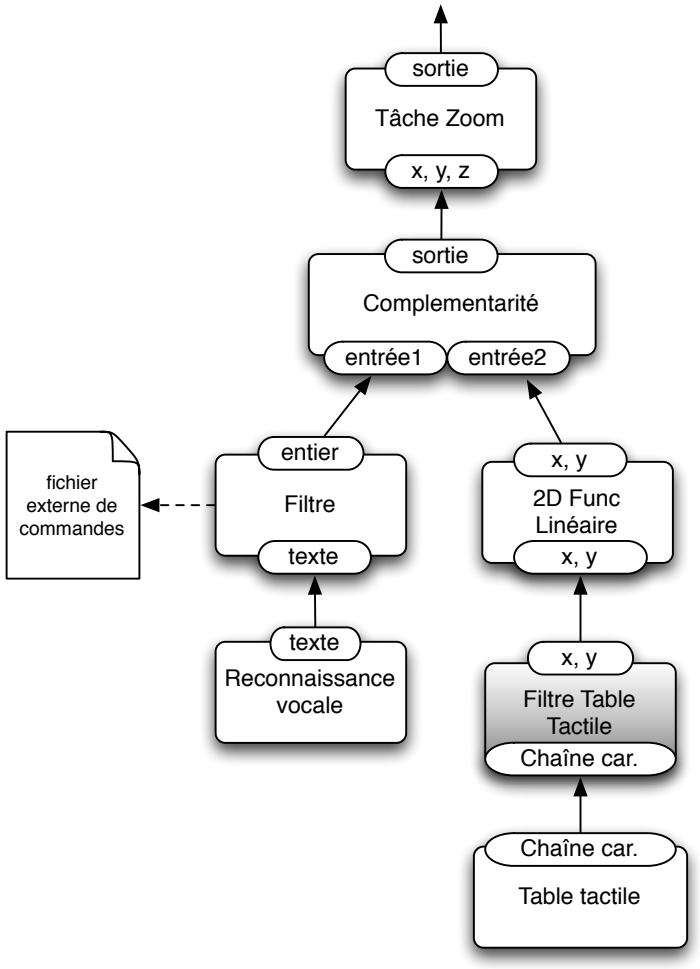


FIGURE 4.29 – Composant dépendant d'autres composants (en gris).

indépendants). Néanmoins, ces composants sont moins réutilisables que ceux de la Figure 4.28.

Assemblage de composants dépendants de l'application

Pour ce troisième assemblage de la Figure 4.30, les composants Transformation et Tâche sont dépendants de l'application. Le composant qui filtre les commandes vocales est un composant qui ne considère que les commandes de cette application. De même, le composant qui filtre les données de la table tactile transforme les points dans le repère de l'application : la table fournit les positions dans une résolution de *(800, 600)* alors que la carte a une résolution de *(1024, 768)*. Ce composant est aussi dépendant du dispositif en entrée.

L'utilisation de composants dépendants de l'application simplifie l'assemblage des composants et réduit le nombre de composants nécessaires afin de transformer les données des dispositifs en entrée dans des données compréhensibles par l'application. Néanmoins, ces composants ne sont pas réutilisables en dehors de cette

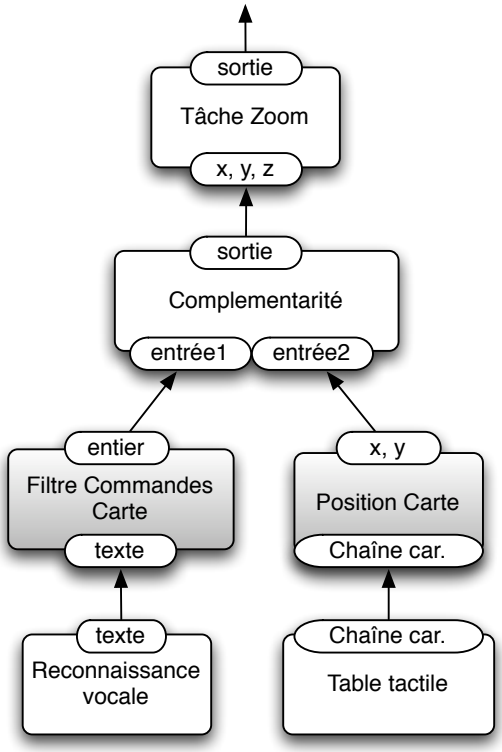


FIGURE 4.30 – Composants dépendants de l'application (en gris).

application. De plus, la simplification de l'assemblage masque l'effort de développement réalisé préalablement afin de créer les composants. Comme ces composants sont très peu réutilisables, le bénéfice d'utiliser ce type de composants dépend du coût de développement et d'intégration des composants dans la technologie utilisée.

4.6.3 Différents assemblages selon la compatibilité syntaxique des interfaces des composants

Assemblage de composants à interfaces ad hoc

Reprenons les assemblages des exemples précédents, dont les interfaces des composants sont ad hoc.

Lorsque les composants sont indépendants, le degré de compatibilité syntaxique entre composants est faible. Nous observons qu'il est nécessaire d'utiliser des composants de Transformation afin d'adapter les données d'une interface au format d'une autre interface, comme le fait le composant 3D Parser à la Figure 4.28.

Lorsque les composants sont dépendants d'autres composants, leurs interfaces sont généralement compatibles syntaxiquement. Ainsi, le composant Transformation Filtre Table Tactile à la Figure 4.29 est totalement compatible avec le composant Dispositif Table. Néanmoins si on utilisait une autre table avec un format d'interface différent, on ne pourrait plus réutiliser le composant Filtre Table Tactile.

Assemblage de composants à interfaces normalisées

La Figure 4.31 reprend l'assemblage de la Figure 4.30 et considère certaines interfaces normalisées selon la taxonomie de Buxton. Dans cet exemple, le concepteur peut directement connecter les composants normalisés : le composant de Transformation 2D Position au composant Dispositif Table Tactile. De plus, la grammaire des interfaces lui indique clairement quelle information est véhiculée par chaque paramètre de l'interface. Notons que ce type d'interface ne permet pas de véhiculer toutes les informations et que certaines sont éliminées, comme l'identifieur du doigt pour la table tactile. En effet, en normalisant les interfaces, nous éliminons des informations propres à certains dispositifs. La normalisation des interfaces des composants ne peut donc être imposée sans réduire le pouvoir génératif du modèle.

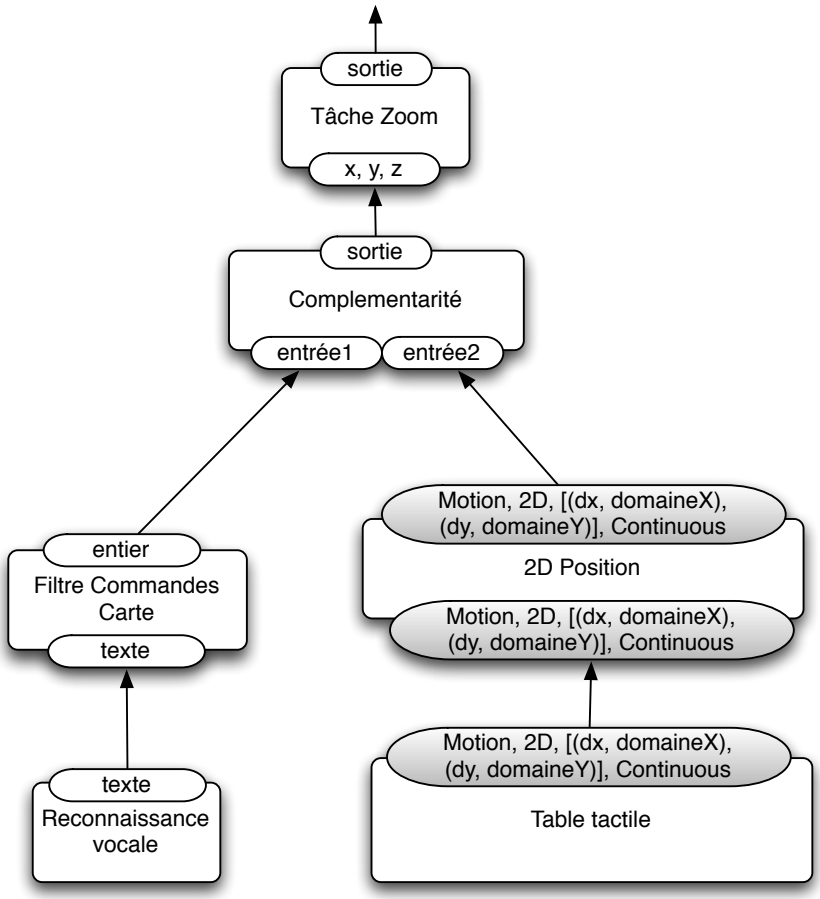


FIGURE 4.31 – Composants avec des interface normalisées (en gris).

4.7 Conclusion

Nous avons présenté et illustré notre modèle conceptuel pour la définition de solutions d'interfaces multimodales en entrée :

- Pour la **réification** nous avons allié les bénéfices des deux modèles conceptuels existants, Icon [Dragicevic et Fekete, 2004] et Icare [Bouchet et Nigay, 2004]. En alliant Icon et Icare, notre modèle conceptuel offre un fort pouvoir descriptif et génératif.

- Pour la **réutilisabilité** des composants, nous avons caractérisé la dépendance comportementale et la compatibilité syntaxique des composants au sein d'un assemblage. En particulier nous proposons des interfaces normalisées des composants en nous reposant sur les taxonomies de Buxton [1983] et de Foley [1984].

Tandis que notre modèle vise à offrir un fort pouvoir descriptif et génératif en alliant les atouts des deux modèles existants Icon et Icare, la réutilisation d'un composant permet de définir rapidement des solutions d'interfaces multimodales. Nous focalisons sur ces deux aspects complémentaires du modèle (pouvoir descriptif/génératif et réutilisation de composants) au chapitre suivant, dédié à l'évaluation de notre modèle.

Chapitre 5

Evaluation de notre modèle conceptuel

*« La politique, c'est l'art de chercher les problèmes, de les trouver, de les **sous-évaluer** et ensuite d'appliquer de manière inadéquate les mauvais remèdes.»*

Groucho Marx, comédien américain (1890-1977)

Contenu du chapitre

Au chapitre précédent, nous avons présenté un modèle conceptuel pour la définition de solutions d'interfaces multimodales en entrée. Avant de nous intéresser à l'outil logiciel qui repose sur ce modèle (Partie 2 de ce mémoire), il convient d'évaluer le modèle indépendamment de tout outil logiciel qui serait basé sur ce modèle.

Notre objectif d'évaluation est d'étudier l'adéquation du modèle pour concevoir rapidement des solutions d'interaction multimodale. Pour cela, nous adoptons deux formes complémentaires d'évaluation, l'une analytique et l'autre empirique.

L'évaluation analytique consiste à analyser notre modèle au regard de critères d'évaluation recensés dans la littérature. Au chapitre 5, nous introduisons d'abord les différents critères existants puis nous présentons l'analyse du modèle.

L'évaluation empirique que nous avons conduit repose sur des entretiens et des exercices réalisés par un panel de sujets. Ce type d'évaluation permet d'obtenir un retour expérimental de l'usage de notre modèle. Nous présentons le protocole expérimental ainsi qu'une analyse des résultats de l'expérience.

Nous commençons ce chapitre en soulignant la difficulté d'évaluation d'un modèle conceptuel pour l'interaction.

5.1 Difficultés d'évaluer un modèle

La recherche en Interaction Homme-Machine définit plusieurs "produits" : techniques d'interaction, outils, systèmes interactifs, modèles, etc. [Newman, 1994]. Selon leur nature, ces "produits" vont être évalués de manière différente. Il existe en particulier une différence entre l'évaluation de techniques d'interaction (et autres produits "simples") et l'évaluation de modèles ou de systèmes interactifs ("produits plus complexes").

Les techniques d'interaction peuvent être évaluées par des évaluations contrôlées selon des métriques. Un exemple de métrique est la performance (rapidité, précision) en considérant des tâches à réaliser. Par exemple, les performances d'une nouvelle technique de pointage 2D peuvent être comparées à celles de la souris [Blanch et Ortega, 2009]. Les modèles et les systèmes, par leur hétérogénéité et complexité, ne se prêtent pas à des évaluations aussi simples[1]. Ainsi, l'évaluation des modèles et des systèmes soulève des problèmes de méthodologie.

Dan Olsen [2007] a analysé les différents problèmes rencontrés lors de l'évaluation de systèmes interactifs que nous réutilisons pour l'évaluation de modèles conceptuels. Cette analyse est considérée aujourd'hui comme une référence dans le domaine[2]. Dan Olsen identifie trois types d'erreurs dans l'évaluation de systèmes interactifs : le piège de l'utilisabilité, l'erreur de l'imperfection fatale et l'héritage du code[3].

Le **piège de l'utilisabilité** désigne la tendance actuelle à utiliser le critère d'utilisabilité comme point central des évaluations. Selon Olsen, ce critère se base sur trois suppositions qui ne sont pas toujours validées. La première consiste à penser que n'importe quel utilisateur possède des connaissances générales qui lui permettent d'utiliser un système interactif. La deuxième consiste à comparer des tâches interactives, alors que celles-ci sont souvent trop complexes pour être comparées. La troisième consiste à réduire le problème en raison des contraintes des test d'utilisabilité, comme le temps d'expérience (1-2 heures en général).

L'**erreur de l'imperfection fatale** représente l'habitude de rejeter un système du moment où une imperfection est découverte dans l'approche. D'après Olsen, la recherche est incapable de produire des systèmes complets qui envisagent toutes les situations possibles. Il est toujours possible de trouver une imperfection, mais celle-ci ne doit pas annihiler le travail effectué.

Finalement, l'**héritage du code** désigne le frein à l'innovation que suppose l'argument du code des applications existantes qu'il faut prendre en compte. Olsen rappelle l'opposition faite aux interfaces graphiques dans les années 70 avec comme argument avancé que tout le code des lignes de commande serait perdu. Le progrès et l'innovation exigent de réécrire les applications existantes.

Olsen propose de centrer l'évaluation de systèmes sur d'autres aspects, comme l'adaptation du système à un STU donné (Situation, Tâche et Utilisateur[4]), l'importance du problème abordé, la nouveauté du problème ou la généralité de la solu-

[1] «Complex systems generally do not yield to simple controlled experimentation.» [Olsen, 2007]

[2] La lecture de l'article [Olsen, 2007] est recommandée aux correcteurs des conférences UIST [UIST, 2009] et CHI [CHI, 2009].

[3] «the usability trap, the fatal flaw fallacy and legacy code» [Olsen, 2007]

[4] Situation, Task and User

tion. Nous présentons ces critères ainsi que deux autres propositions pour évaluer des modèles dans la section suivante.

5.2 Evaluation analytique

5.2.1 Critères d'évaluation

Critères selon Olsen

Comme introduit à la section précédente, nous considérons les critères de Olsen [2007]. Ces critères sont définis pour évaluer un outil ou un système. Nous les déclinons au cas de l'évaluation d'un modèle. Ainsi, nous considérons quatre critères : importance du modèle, pouvoir de combinaison du modèle, passage à l'échelle du modèle et viscosité de la solution conçue selon le modèle,.

Le premier critère, l'**importance du modèle**, est défini en partie par le **STU (Situation, Tâche, Utilisateur)** visé : considérer des utilisateurs cibles, pour un ensemble de tâches donné, dans certaines situations. Le STU constitue ainsi un cadre dans lequel le modèle peut être évalué. Par exemple, on peut se demander si la population visée dans le STU est suffisamment importante. Une population (U) peut être importante pour différentes raisons : taille ou métiers critiques (médecins). Il faut donc reconsidérer l'importance de la population visée par rapport à la société. La même question se pose par rapport à la tâche (T) et à la situation (S). Par exemple, on peut se demander quelle est la fréquence de la situation : un modèle qui répond à une situation habituelle est plus important qu'un modèle qui répond à une situation rare (toujours en fonction de l'importance de T et U évidemment). Parallèlement, il convient de montrer que le problème visé n'a pas déjà été résolu pour le STU visé.

Olsen analyse ensuite le **pouvoir de combinaison**, c'est-à-dire la façon dont l'approche utilise différents composants pour construire la solution. Par exemple, il identifie la combinaison inductive[5], lorsqu'on construit des systèmes complexes à partir de primitives simples.

Ensuite, Olsen cite le **passage à l'échelle**, c'est-à-dire la capacité de l'approche à être utilisée pour des problèmes plus larges. Olsen utilise l'exemple de l'utilisation de machines à états pour décrire le dialogue homme-machine comme approche ne permettant pas le passage à l'échelle. Cette approche est adaptée à des systèmes simples : lorsque le système devient complexe, la multiplication des états de la machine rend l'approche inutilisable.

Finalement, Olsen aborde la **réduction de la viscosité de la solution**. Il existe trois manières de réduire la viscosité selon Olsen : la flexibilité, le pouvoir d'expression et la correspondance expressive[6]. Un modèle est flexible lorsqu'il permet de modifier rapidement la conception d'interfaces. Le pouvoir d'expression est défini lorsque le concepteur peut réaliser plus en exprimant moins. En particulier, une approche a un bon pouvoir d'expression lorsqu'elle réduit le nombre total de choix que le concepteur doit réaliser pour exprimer une solution. La correspondance expressive représente la distance entre l'expression des choix de conception et le problème.

[5] «Inductive Combination»[Olsen, 2007]
[6] «flexibility, expressive leverage and expressive match» [Olsen, 2007]

Les deux critères de pouvoir d'expression et de correspondance expressive sont à mettre en relation avec respectivement le pouvoir de description et le pouvoir d'évaluation identifiés par Beaudouin-Lafon [2004] et présentés à la section 4.1 du chapitre 4. Nous les rappelons à la section suivante.

Pouvoirs d'un modèle d'interaction selon Beaudouin-Lafon [2004]

Beaudouin-Lafon [2004] définit trois propriétés qui permettent d'évaluer un modèle d'interaction : le pouvoir de description, le pouvoir d'évaluation et le pouvoir de génération. Le pouvoir de description désigne la capacité du modèle à décrire un ensemble significatif de solutions existantes. Le pouvoir d'évaluation désigne l'aptitude d'un modèle à permettre la comparaison de différentes alternatives de conception. Le pouvoir de génération exprime la capacité d'un modèle à permettre des nouvelles conceptions d'interfaces.

Dimensions cognitives des notations de Green

Les dimensions cognitives (DC) des notations de Green [1989] [2000] définissent des critères pour l'évaluation des notations des environnements de développement. Treize dimensions cognitives différentes sont identifiées. Green propose d'évaluer la notation selon l'activité générique de développement réalisée [7], dont il détermine six types : incrément, modification, transcription, conception exploratoire, recherche et compréhension.

Dans notre modèle de conception, l'activité cible qui repose sur la notation est la conception exploratoire. Afin d'améliorer la conception exploratoire, Green propose d'utiliser quatre dimensions : il faut assurer une faible viscosité, un faible engagement forcé, une forte visibilité et une forte expressivité du rôle. Nous détaillons ces quatre dimensions :

- **Viscosité** : représente la résistance au changement. Augmente avec le nombre d'actions nécessaires pour accomplir une conception. Ce critère est également cité par Olsen (cf. premier paragraphe de cette section).
- **Engagement forcé** : désigne l'obligation de réaliser les actions de conception dans un certain ordre.
- **Visibilité** : représente la possibilité de voir les éléments de la notation facilement. En effet, certaines notations permettent d'encapsuler des données, ce qui risque de diminuer la visibilité de l'information.
- **Expressivité du rôle** : représente la facilité avec laquelle la fonction d'un élément de la notation est déduite par les concepteurs.

5.2.2 Analyse de notre modèle

L'évaluation menée consiste en une inspection critique de notre modèle conceptuel en nous appuyant sur les critères d'évaluation présentés à la section précédente.

[7] «None of these dimensions is evaluative when considered on its own. Evaluation must always take into account the activities to be supported.»[2000]

Critères de Olsen

Importance du modèle Notre modèle vise un STU précis : un concepteur d'interaction qui souhaite concevoir une interaction multimodale. L'utilisateur (U) est donc le concepteur, la tâche (T) consiste à concevoir une interaction multimodale et la situation (S) est l'exploration de solutions d'interaction lors de la phase de spécifications externes du cycle de développement. Le Tableau 5.1 synthétise le STU visé par notre approche.

Situation (S)	Tâche (T)	Utilisateur (U)
Exploration de solutions d'interaction multimodale	Concevoir une interaction multimodale	Concepteur d'interaction

TABLE 5.1 – STU visé par notre modèle

L'importance de notre modèle se justifie par le fait qu'aucune solution de conception existe aujourd'hui pour ce STU, sauf Icon et Icare. La très grande multiplicité des dispositifs disponibles aujourd'hui nécessite de fournir des outils qui permettent une exploration efficace et rapide des solutions d'interaction multimodale (S). L'importance du modèle se justifie donc par la disponibilité croissante de moyens d'interaction. De plus, comme le souligne Olsen [2007], la conception de systèmes (T) qui intègrent des nouveaux dispositifs est nécessaire pour que des nouveaux dispositifs voient le jour. Le modèle concerne donc un groupe d'utilisateurs (U) en augmentation en raison de la présence habituelle d'interfaces multimodales sur différents supports, comme illustré au chapitre 2.

Pouvoir de combinaison du modèle Notre modèle reposant sur l'utilisation d'assemblages de composants, il possède un fort pouvoir de combinaison. En effet, la conception de solutions d'interaction (systèmes complexes) par assemblage de composants élémentaires (primitives simples) correspond exactement à la combinaison inductive identifiée par Olsen.

Passage à l'échelle du modèle Le passage à l'échelle de notre modèle peut s'avérer difficile. En effet, l'utilisation du modèle pour des systèmes complexes implique l'utilisation de plusieurs assemblages et donc la complexification globale de la conception. Le passage à l'échelle est donc un point faible de notre modèle.

Viscosité du modèle Les trois critères pour réduire la viscosité sont la flexibilité, le pouvoir d'expression et la correspondance expressive [Olsen, 2007]. Notre modèle étant basé sur un flot de données défini par un assemblage de composants, nous pouvons facilement modifier la conception par le remplacement d'un ou plusieurs composants (forte flexibilité).

En ce qui concerne le pouvoir d'expression, celui-ci est augmenté dans notre modèle par la structuration en trois niveaux de l'assemblage, en limitant les choix de conception au niveau concerné. La caractérisation du degré de réutilisation d'un composant augmente aussi le degré d'expression du modèle. Cette caractérisation permet au concepteur de choisir plus facilement le composant à utiliser dans

un contexte donné. Comme le signale Olsen, le pouvoir d'expression augmente lorsque le concepteur voit réduit le nombre de choix à effectuer pour définir une solution donnée[8].

Finalement, en ce qui concerne la correspondance expressive, notre modèle considère une représentation proche mentalement de l'interaction elle-même qu'elle représente. La Figure 5.1 illustre cette correspondance entre une interaction multimodale et sa représentation dans un assemblage de composants.

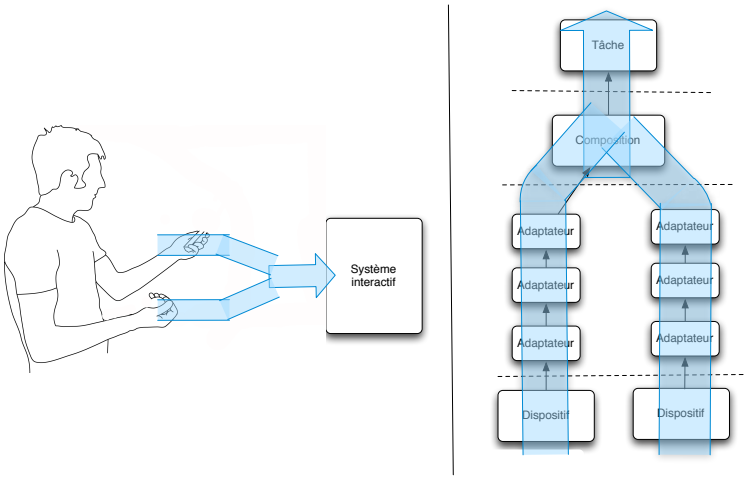


FIGURE 5.1 – Correspondance entre une interaction multimodale et sa représentation selon notre modèle.

Pouvoirs de description, d'évaluation et de génération du modèle de Beaudouin-Lafon

Ces pouvoirs de description, d'évaluation et de génération d'un modèle ont été utilisés tout au long du chapitre précédent pour rationaliser notre modèle.

Nous rappelons en particulier que notre modèle combine les bénéfices des deux approches Icon et Icare en visant un pouvoir descriptif équivalent à Icare et supérieur à Icon, et un pouvoir génératif équivalent à Icon mais supérieur à Icare. De plus au chapitre 4, nous avons considéré plusieurs assemblages avec des composants dont le degré de réutilisabilité était explicitement différent, favorisant ainsi le pouvoir comparatif de notre modèle du point de vue de la réutilisabilité des composants.

En ce qui concerne le pouvoir de génération, nous avons montré à la section 4.1 et à la section 4.3.1 du chapitre 4 en quoi notre approche offre un pouvoir de génération supérieur à celui de Icare, tout en conservant un bon pouvoir de description, contrairement à Icon.

Dimensions cognitives de Green

Nous avons déjà analysé la *viscosité* de notre modèle, un critère aussi considéré par Olsen. En ce qui concerne la *visibilité* du modèle, l'assemblage de composants peut présenter deux problèmes de visibilité : d'un côté, si l'assemblage est trop grand,

[8] «Expressive leverage is achieved when a tool reduces the total number of choices that a designer must make to express a desired solution» [Olsen, 2007]

la visibilité des composants se voit réduite. Ensuite, les composants étant des encapsulations de données, la visibilité de l'information contenue dans le composant peut s'avérer faiblement visible pour le concepteur.

Notre modèle permet de réduire l'*engagement forcé* du concepteur : en effet, le concepteur peut choisir l'ordre dans lequel il va réaliser la conception d'un assemblage. Il peut commencer à haut niveau en identifiant les tâches et la composition de modalités pour finalement choisir les dispositifs, ou bien commencer à bas niveau, par les dispositifs, pour ensuite choisir les transformations puis remonter vers la composition et la tâche.

L'*expressivité du rôle* dans notre modèle est assez fort grâce à la classification en quatre types différents des composants : dispositif, transformation, composition et tâche. Ainsi, chaque composant de notre modèle possède un type et donc un rôle facile à identifier par sa position dans l'assemblage de composants.

Synthèse

La Figure 5.2 reprend les différents points de l'évaluation analytique de notre modèle.

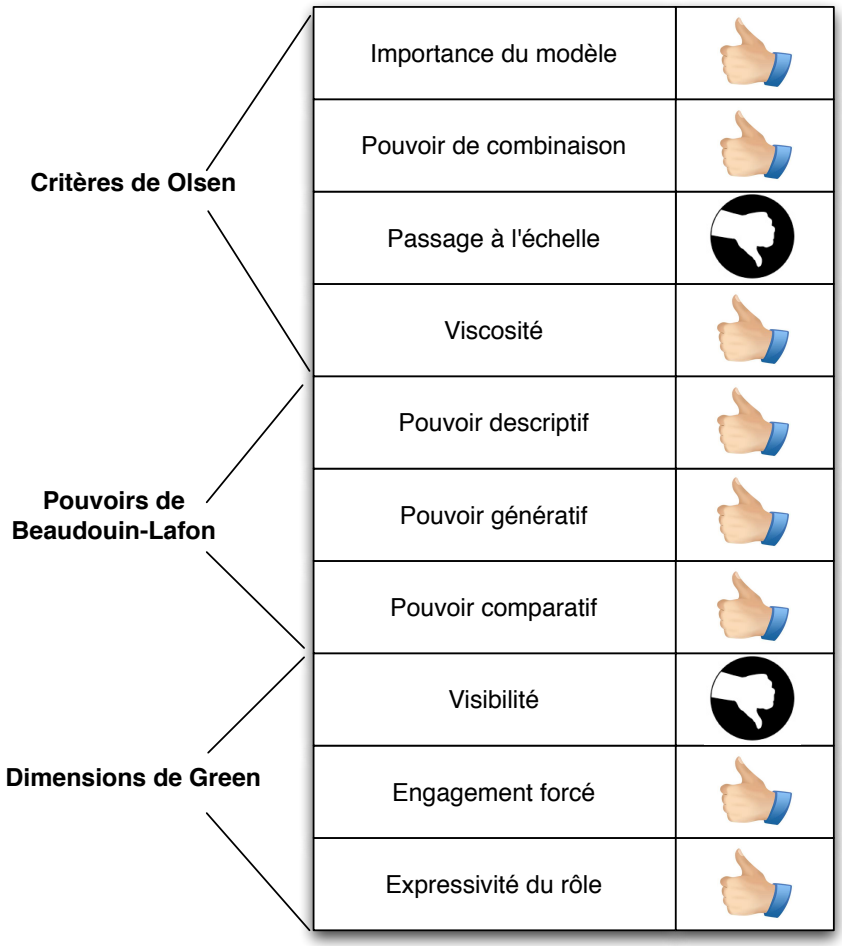


FIGURE 5.2 – Synthèse de l'évaluation analytique.

L'évaluation analytique de notre modèle selon des critères s'accompagne d'une évaluation empirique où les critères sont exploités pour formuler nos hypothèses.

5.3 Évaluation empirique

Afin d'évaluer le modèle de façon empirique[9], nous avons conduit une expérience à Grenoble avec des concepteurs en IHM. Nous avons collaboré dans ce but avec Mme Nadine Mandran, ingénieur en méthodologies des sciences sociales de la plateforme Marvelig. Marvelig [2009], plate-forme d'expérimentations scientifiques du Laboratoire d'Informatique de Grenoble, a pour but, en particulier, de mettre en oeuvre des méthodes d'évaluation de concepts et d'outils.

Nous présentons d'abord l'objectif et le protocole de l'expérience avant de détailler les différents aspects du modèle évalués et les résultats.

5.3.1 Objectif de l'évaluation empirique

Notre évaluation empirique est *qualitative*. Une évaluation qualitative est menée avec un nombre limité de sujets. Elle permet de rendre compte de la diversité des usages. Complémentaire à l'évaluation qualitative, l'évaluation quantitative est réalisée avec un nombre élevé de sujets afin de produire des données statistiques. Ces données permettent de quantifier les usages et les comportements de manière proportionnelle.

L'objectif de notre évaluation qualitative est d'analyser la compréhension du modèle, son usage et sa lisibilité en exploitant les critères utilisés dans notre évaluation analytique.

5.3.2 Protocole de l'évaluation

Nous définissons les paramètres du protocole de notre expérience : profil des sujets, méthodes en sciences humaines et sociales (SHS) utilisées, organisation de l'expérience et données collectées.

Profil des sujets

Les sujets sélectionnés correspondent au profil du STU visé (section 5.2.2) par notre modèle, c'est-à-dire des concepteurs en IHM. Le concepteur en IHM correspond à différents profils : des ingénieurs de recherche, des ergonomes, des étudiants en doctorat, etc. Nous avons réuni un groupe hétérogène qui inclut ces différents profils afin de couvrir le spectre de concepteurs existants dans le monde de la recherche (pas de concepteurs industriels).

Suite à notre appel à participation, huit personnes ont pris part à cette expérience : une femme et sept hommes. Le groupe (Tableau 5.2) est composé de quatre doctorants en IHM, un ingénieur de recherche, un maître de conférences et deux post-doctorants.

Nous avons ensuite crée des binômes de travail, en essayant de former des groupes d'expertise différente. Nous identifions deux types de sujets : des sujets experts, qui possèdent déjà des connaissances liées à certains aspects du modèle

[9] EMPIRIQUE : Qui ne s'appuie que sur l'expérience. [CNRTL-CNRS, 2009]

Nombre	Moyenne d'âge	Sexe	Fonctions
8	23	1 femme, 7 hommes	4 doctorants, 2 post-doctorants, 1 ingénieur de recherche, 1 maître de conférences

TABLE 5.2 – Profil des sujets de l'éxpérience.

(flot de données, interfaces multimodales, etc.) et des sujets novices, sans connaissance préalable des concepts clefs du modèle. Ainsi, nous considérons trois types de binômes : le binôme expert, constitué par deux experts ; le binôme mixte, composé d'un expert et un novice ; et le binôme novice, formé par deux novices. De cette manière quatre binômes sont établis (Tableau 5.3) : un binôme expert, deux binômes mixtes et un binôme novice.

N° du binôme	Profil	Composition
1	Expert	1 doctorant, 1 maître de conférences
2	Mixte	1 ingénieur, 1 doctorant
3	Mixte	1 post-doctorant, 1 doctorant
4	Novice	1 post-doctorant, 1 doctorant

TABLE 5.3 – Profils des binômes.

Méthodes issues des SHS utilisées

Nous avons utilisé deux méthodes des Sciences Humaines et Sociales (SHS) afin de réaliser l'expérience : le **focus group** et le **cultural probe**. Le *focus group* [Krueger et Casey, 2000] est une méthode qualitative qui consiste à organiser des réunions de travail avec un groupe de personnes séléctionnées. Le *cultural probe* [Bernhaupt *et al.*, 2008] consiste à évaluer un produit dans son contexte d'utilisation sans la participation d'évaluateurs, afin de ne pas influencer l'expérience.

Afin de s'approprier le modèle, nous considérons qu'il est nécessaire d'avoir une phase d'apprentissage : il est donc indispensable de disposer de temps pour travailler avec le modèle. C'est pourquoi, le modèle ne peut pas être évalué lors d'un focus group, le temps consacré au travail sur la méthode étant trop court.

Par conséquent, nous avons utilisé le focus group et le cultural probe de façon conjointe. Ainsi, nous recourons aux *cultural probes* afin de mettre les concepteurs dans une situation réelle de conception. Ces cultural probes ayant lieu dans le contexte de travail du sujet, nous pouvons les considérer comme des *work probes*. Une fois les work probes réalisés, nous menons un focus group afin d'analyser le déroulement et les résultats.

Organisation de l'expérience

L'expérience se déroule en trois temps répartis sur une durée d'une semaine : une séance de présentation, un work probe et une séance de focus group.

La première séance est une formation au modèle et une mesure des pratiques de travail des participants : connaissance des interfaces multimodales, compétences en

réalisation de systèmes interactifs et de prototypes, habitudes de modélisation des systèmes interactifs et connaissance de l'approche à flot de données. Cette séance nous a permis d'établir le profil des sujets, de créer les binômes, de présenter le modèle et d'introduire le cahier de charges du work probe.

Le deuxième temps est un work probe, réalisé en binômes sans la participation de l'évaluateur. Les sujets disposent d'une semaine pour le work probe, qu'ils peuvent accomplir en une ou plusieurs séances. Le work probe est composé d'un cahier de charges, composé de différents exercices que les binômes doivent réaliser. Chaque binôme doit réaliser au total six assemblages de composants, représentant six interactions multimodales différentes. Les binômes peuvent poser des questions par mail pendant la réalisation des exercices. Ils doivent également remplir une fiche de suivie qui permet de collecter des informations sur les difficultés rencontrées.

La troisième séance est un focus group qui permet de présenter les résultats du work probe et discuter le travail réalisé. Les sujets remplissent également une grille points forts/points faibles ainsi que des post-it avec des propositions d'amélioration du modèle.

Données collectées

L'expérience a produit un ensemble de données que nous avons analysé à posteriori (donnés en Annexe B). Une partie de ces données est générée par les utilisateurs, comme les résultats des exercices, les fiches de suivi ou les grilles points forts/points faibles. À ces données nous ajoutons des informations collectées par les évaluateurs, comme les enregistrements audio et vidéo du focus group.

L'ensemble des données produites par l'expérience est le suivant :

- Enregistrement audio et vidéo du focus group ;
- Questionnaires des pratiques ;
- Fiches de suivi ;
- Grilles points forts-points faibles ;
- Résultats papier des exercices ;
- Post-it de propositions d'amélioration.

Le Tableau 5.4 synthétise l'organisation de l'expérience et les données produites pour chaque séance.

5.3.3 Résultats de l'évaluation

Nous présentons les principaux résultats de l'évaluation empirique que nous avons menée. Nous organisons cette section selon les hypothèses que nous avons formulées, liées aux critères de notre analyse empirique (section 5.2.1). Nous énonçons l'hypothèse, l'exercice mené afin de la valider puis les conclusions.

Séance	Format	Participants	Durée	Données produites
1	Présentation	Tous	45′	Questionnaires des pratiques
2	Work probe	Binômes	env. 1h30′	Représentation papier des exercices ; Fiches de suivi et d'utilisabilité
3	Focus group	Tous	1h30′	Audio et vidéo ; Grilles points forts-points faibles ; Post-it de propositions d'amélioration

TABLE 5.4 – Synthèse des séances de l'expérience.

Évaluation du pouvoir de description et de génération du modèle

Hypothèse Pour considérer les deux types de pouvoir du modèle, nous avons considéré deux contextes d'usage du modèle : concevoir une interaction multimodale existante (pouvoir de description) et être capable de générer de nouvelles interactions (pouvoir de génération).

Notre hypothèse est donc la suivante :

- Le modèle permet d'exprimer des nouvelles interfaces multimodales (évaluation du pouvoir de génération)
- Le modèle permet de décrire une interaction multimodale existante (évaluation du pouvoir de description).

Exercices Afin d'évaluer l'usage du modèle, nous proposons deux exercices aux sujets. Le premier exercice consiste à imaginer une interaction multimodale puis la décrire selon notre modèle. Le choix d'évaluer le pouvoir de génération avant le pouvoir de description est délibéré et vise à ne pas influencer les sujets dans leur création de nouvelles interactions.

Ensuite, le deuxième exercice consiste à décrire une interaction multimodale existante, définie par une tâche, des dispositifs et des modalités d'interaction précises. Nous avons fourni aux binômes un exemple d'application qui intègre une carte d'une ville projetée sur une table. Nous avons demandé d'abord de penser aux tâches interactives réalisables avec la carte. Ensuite, nous leur avons demandé de spécifier textuellement des interactions multimodales pour réaliser ces tâches. Finalement les binômes ont dû créer les assemblages de composants de trois interactions multimodales.

Résultats Tous les binômes ont réussi à compléter les deux exercices. La définition de l'interaction par un flot de données n'a pas posé de problème, sauf pour le binôme novice, qui n'ayant pas compris la portée du modèle s'est posé la question «de savoir à quel niveau logiciel se placer : interface graphique ou noyau fonctionnel» (binôme 4, novice). Cela n'a pas empêché le binôme de compléter l'exercice correctement. Les sujets ont souligné la simplicité et la facilité d'utilisation du modèle : «Le modèle est satisfaisant pour décrire rapidement et simplement le flux de données» (binôme n°2) ; «Le modèle est très facile à comprendre, à utiliser et à mettre en place» (binôme n°3). Globalement, nous considérons que tous les bi-

nômes comprennent la conception d'interaction multimodale et l'approche à flot de données.

Le pouvoir de génération s'est vu confirmé par la spécification des interactions imaginées : «On arrive facilement à exprimer clairement des interactions envisagées» (binôme n°3). Le pouvoir de description de l'approche est également confirmé : «correspond bien à la description d'interaction multimodale, car les dispositifs émettent des données, les tâches réagissent aux données. L'interaction est alors le routage des flots de données.» (binôme n°1). En ce qui concerne les solutions d'interaction multimodale, nous observons une différence notable selon l'expertise. En effet, à l'opposé des autres binômes, le binôme 1 (Expert) a conçu autant de solutions d'interaction uni-modale impliquant différentes techniques («utilisateur dit "Centrer sur Adresse"», «utilisateur double-tappe sur 1 point pour centrer la carte dessus») que d'interactions multimodales impliquant une combinaison de modalités («utilisateur dit "calculer itinéraire" et appuie successivement sur des points de la carte»). Enfin, la plupart des binômes ont décrit des modalités d'interaction exploitant la voix et le geste.

Parmi les points négatifs, plusieurs binômes ont signalé la limitation de l'approche en ce qui concerne la spécification de la dynamique d'une application : «Difficile de modéliser la dynamique de la structure du flux dans le temps» (binôme n°1) ; «On n'a pas vu comment faire des interactions complexes avec un enchaînement de plusieurs tâches» (binôme n°2).

Évaluation de l'expressivité de la structure des assemblages

Hypothèse Lié au pouvoir génératif et descriptif, nous avons focalisé sur l'expressivité de la structure de l'assemblage en trois niveaux. Nous souhaitons étudier l'expressivité de la structure en niveaux des assemblages de composants (Dispositif, Transformation, Composition et Tâche). Nous évaluons donc le pouvoir d'expression et la correspondance expressive ainsi que la visibilité de la structure des assemblages . Notre hypothèse est donc la suivante :

- La structure permet de mieux organiser les idées de conception (pouvoir d'expression et correspondance expressive) ;
- La structure permet d'expliquer facilement une interaction multimodale conçue (visibilité).

Exercices Les exercices de conception décrits dans les deux sections précédentes permettent d'évaluer l'expressivité de la structure de l'assemblage. Nous avons également proposé deux exercices supplémentaires : un exercice de lecture et un exercice de type "portraits chinois".

Dans l'exercice de lecture, réalisé lors du focus group, nous demandons aux binômes de lire et expliquer à voix haute les assemblages de composants des autres groupes (qu'ils ne connaissent pas préalablement). Les binômes doivent expliquer l'interaction multimodale spécifiée par l'assemblage en donnant autant de détails que possible. Le groupe qui a rédigé l'assemblage confirme ensuite l'exactitude de cette lecture.

Le portrait chinois [Gauthy-Sinéchal et Vandercammen, 2005] est une technique de créativité, consistant à imaginer l'objet étudié non pas tel qu'il est mais comme

quelque chose d'autre, par exemple un animal, un monument, etc. Nous demandons aux sujets de donner des noms d'animal, de plante, de monument et de boisson qui représentent les assemblages. Les sujets doivent préciser la raison de leur choix.

Résultats Dans les exercices de conception, tous les binômes ont organisé leurs assemblages selon les trois niveaux du modèle. Un point notable est la différence de structuration des assemblages entre le binôme novice (n°4) et le binôme expert (n°1). Alors que le binôme novice a réalisé des assemblages relativement simples avec un structure proche du Y inversé (Figure 5.3), le binôme expert a réalisé des assemblages un peu plus complexes, avec des dédoublements de la forme en Y (Figure 5.4). Ce dédoublement avait pour but d'exprimer l'utilisation parallèle de deux compositions de modalités (une pour générer une commande et l'autre pour spécifier une adresse sur la carte). Les assemblages des deux autres binômes étaient proches de ceux du binôme novice. Ainsi, en général la correspondance expressive entre une interaction multimodale et la représentation selon notre modèle est bonne et permet même de représenter des interactions plus complexes comme celle illustrée à la Figure 5.4.

L'exercice de lecture a été réalisé facilement et les sujets n'ont pas identifié de problèmes pour lire les assemblages des autres groupes. Les sujets ont exprimé leur satisfaction en ce qui concerne la structure en trois niveaux des assemblages : «La lecture du modèle est facilitée par la structuration en couches» (binôme n°4) ; «Découpage par niveaux intéressant, cela facilite la compréhension et la décomposition d'une interaction» (binôme n°2). La structuration en couches est «pratique pour isoler les problèmes de chaque niveau» (binôme n°1) ; «la hiérarchie proposée m'a aidée à organiser mes idées» (binôme n°4) ; «permet de bien séparer les préoccupations» (binôme n°1). Nous concluons que la structuration de l'assemblage de composants augmente la visibilité de notre modèle.

Les résultats des portraits chinois sont intéressants car ils permettent d'identifier quelles caractéristiques de l'assemblage ont plus marqué l'esprit des sujets. Ainsi, par exemple, les sujets ont choisi les animaux suivants : l'aigle pour la vue d'ensemble, l'abeille pour l'organisation, le serpent pour le chemin des données, l'oiseau pour la vue d'en haut. Cela souligne la bonne visibilité de notre modèle (oiseau et aigle) ainsi que la forte correspondance expressive entre le problème et la solution (abeille et serpent).

Évaluation de la caractérisation de la réutilisabilité des composants

Hypothèse Nous avons étudié expérimentalement comment caractériser la réutilisabilité des composants. Nous évaluons le pouvoir de combinaison, la visibilité et l'expressivité des rôles de notre caractérisation de la réutilisabilité. Nos hypothèses sont les suivantes :

- Notre description de la réutilisation (axes comportemental et syntaxique) permet de caractériser facilement les composants (visibilité, expressivité des rôles) ;
- Notre description permet de décider quels composants sont réutilisables dans un contexte donné (pouvoir de combinaison) ;

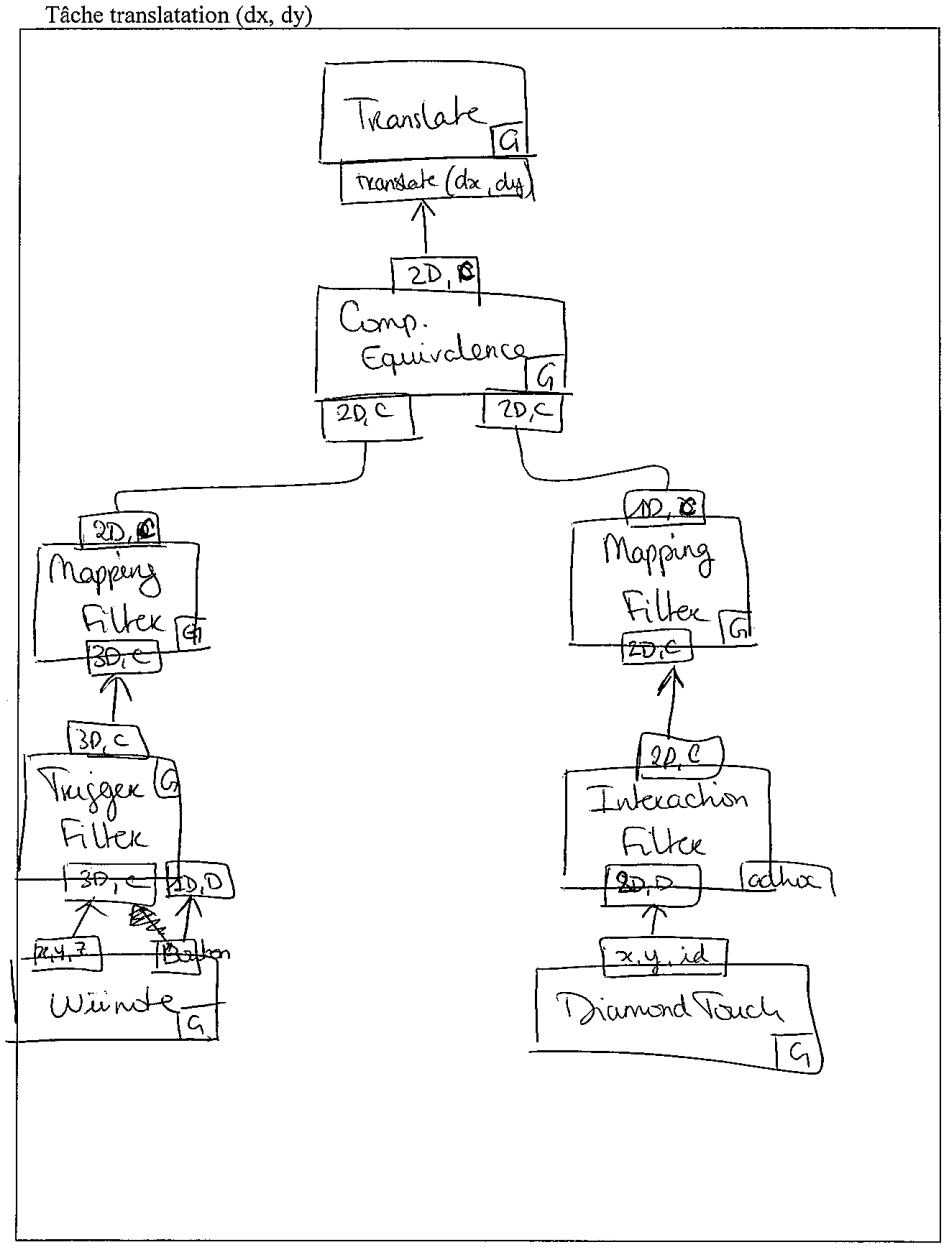


FIGURE 5.3 – Assemblage à structure simple réalisé par le binôme novice (binôme n°4).

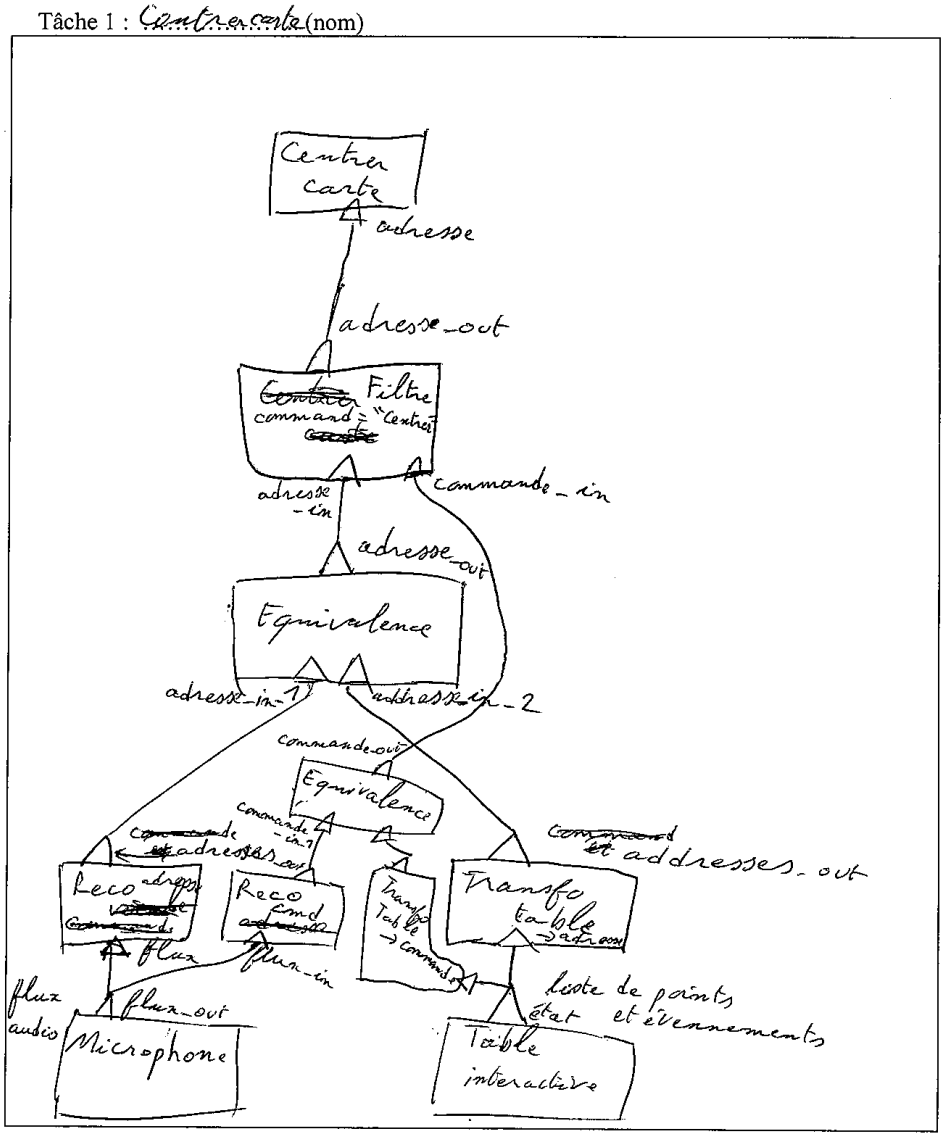


FIGURE 5.4 – Assemblage à structure complexe réalisé par le binôme expert (binôme n°1).

Exercices Nous avons demandé aux sujets de caractériser les composants utilisés dans les assemblages. Nous avons également demandé aux sujets de réutiliser certains composants dans plusieurs assemblages.

Résultats Les sujets ont estimé que la réutilisation des composants est importante : «Permet de savoir ce qui peut être facilement réutilisable du reste» (binôme n°2) ; «permet la réutilisabilité moyennant une configuration par des propriétés» (binôme n°1) ; «gain de temps pour la spécification d'application» (binôme n°4).

Les binômes ont d'ailleurs largement réutilisé des composants dans leurs assemblages : le binôme n°1 a réutilisé 18 composants parmi les 26 utilisés au total ; le binôme n°2 a réutilisé 11 composants parmi les 18 utilisés au total ; le binôme n°3 a réutilisé 22 composants parmi les 41 utilisés au total ; et le binôme n°4 a réutilisé 12 composants parmi les 18 utilisés au total. Parmi les composants réutilisés, 31% correspondent à des composants Dispositif, 47% correspondent à des composants de Transformation et 22% correspondent à des composants de Composition. Les composants Tâche étaient toujours différents étant donné que chaque assemblage définissait une tâche interactive différente.

Les sujets ont souligné que les axes de caractérisation comportementale et syntaxique sont pertinents et permettent d'avantage d'identifier la réutilisabilité des composants : «Bon découpage entre la généricité syntaxique et du comportement» (binôme n°3) ; «la partition entre interface et comportement est pertinente. En effet je trouve nécessaire ce découplage pour que le développeur sache identifier quelle partie d'un composant respecte (ou ne respecte pas) ses besoins.» (binôme n°4).

Nous concluons donc que notre caractérisation de la réutilisabilité augmente la visibilité et l'expressivité des rôles des composants et offre bon pouvoir de combinaison.

5.4 Synthèse

Dans ce chapitre nous avons mené deux types d'évaluation complémentaires de notre modèle conceptuel présenté au chapitre précédent.

De l'évaluation analytique, nous retenons le STU correspondant à notre modèle. Nous avons également identifié les limites de notre modèle en ce qui concerne le passage à l'échelle et la visibilité des grands assemblages de composants.

L'évaluation empirique a permis de valider les éléments fondateurs du modèle : composants, flot de données, structure de l'assemblage en trois niveaux, pour décrire l'interaction multimodale pour une tâche donnée. Enfin une perspective intéressante à notre modèle qui a été proposée pas un des sujets concerne la description de l'enchaînement temporel de l'interaction, par exemple lorsqu'on considère une interaction multimodale qui varie selon les états du système.

Enfin une autre forme d'évaluation de notre modèle est son utilisation par des concepteurs pour concevoir des systèmes interactifs multimodaux, une évaluation par l'usage. Dans le temps imparti d'une thèse, nous ne pouvons obtenir que les contributions soient largement utilisées par la communauté. Néanmoins nos travaux ont été exploités dans le cadre d'un consortium européen et des concepteurs autres que des membres de notre équipe ont appliqué ce modèle à la conception de prototypes multimodaux. Nous en présentons certains dans le chapitre 8 qui ont

été développés à l'aide de l'outil logiciel reposant sur ce modèle qui fait l'objet de la partie 2 du manuscrit.

Deuxième partie

Réalisation logicielle

Chapitre 6

État de l'art des outils pour le prototypage d'interfaces multimodales

«Pay no attention to that man behind the curtain.»

The Wizard of Oz, 1939

Dans la partie 1, nous nous sommes concentrés sur les modèles conceptuels et les concepts clefs de la multimodalité pour décrire des solutions d'interaction. Dans le cadre d'une conception itérative centrée sur l'utilisateur, nous consacrons la partie 2 à la réalisation logicielle de prototypes d'interfaces multimodales. Dans ce chapitre nous étudions les outils existants puis, dans le chapitre 7, nous présentons notre outil qui repose sur notre modèle conceptuel présenté dans la partie 1.

Contenu du chapitre

Dans ce chapitre nous organisons notre revue des outils de prototypage en deux catégories selon le niveau de fonctionnalité du prototype : les outils de développement de prototypes simulés (prototypes Magicien d'Oz) et les outils de développement de prototypes interactifs.

La première section du chapitre motive cette classification des outils de prototypage existants. Nous présentons ensuite les outils existants dans deux sections distinctes correspondant à nos deux classes d'outils.

6.1 Typologie des outils de prototypage

En IHM, il est classique de distinguer plusieurs niveaux dans le prototypage [Buxton, 2007], correspondant à des finalités différentes :

- Le **prototypage basse-fidélité** consiste généralement à produire des prototypes papier-crayon qui permettent d'identifier des aspects fonctionnels dès les premières étapes de conception. Ce type de prototype permet d'explorer l'espace de conception.
- Le **prototypage haute-fidélité** consiste à créer un prototype fonctionnel qui implémente la plupart des fonctionnalités interactives de l'interface finale. Ce type de prototype permet de mener une évaluation expérimentale avec les utilisateurs finaux.

Le niveau de fidélité désigne donc le degré de similitude du prototype par rapport à l'interface finale. Cette classification reste néanmoins très générale : la caractérisation des prototypes selon ces deux niveaux n'est pas facile en raison de la multiplicité des formes de prototypes. McCurdy [McCurdy *et al.*, 2006] propose de définir plus précisément le niveau de fidélité d'un prototype en considérant cinq dimensions :

- Le **niveau de raffinement visuel** : désigne le raffinement de l'interface graphique du prototype. L'aspect visuel d'un prototype peut être défini à bas niveau, par des schémas papier-crayon, ou bien à haut niveau, par des représentations très fidèles des objets graphiques manipulés.
- L'**amplitude des fonctionnalités réalisées** : représente à quel point le prototype couvre l'espace fonctionnel de l'application. Par exemple, pour un prototype d'un distributeur de banque, est-ce que le prototype implémente toutes les opérations possibles (retrait, consultation de solde, etc.) ?
- La **profondeur des fonctionnalités réalisées** : représente, pour chaque fonctionnalité du système, à quel point celle-ci est réalisée dans sa totalité. Par exemple, dans le cas du distributeur de banque, est-ce que l'opération de retrait est totalement réalisée : sélection de la tâche de retrait, insertion du code personnel, spécification du montant à retirer et rendu du ticket d'opération.
- La **richesse de l'interaction** : représente le niveau de fonctionnalité de l'interactivité en entrée ou en sortie. Des prototypes papier sont par exemple moins riches que des prototypes où l'interaction est implémentée.
- La **richesse du modèle de données** : décrit à quel point les données utilisées par le prototype sont représentatives des données de l'application finale.

Nous observons parmi ces cinq dimensions que la plupart concernent avant tout l'interface graphique en sortie. Seule la dimension de richesse de l'interaction considère l'interaction en entrée explicitement. Étant donné que nous nous intéressons

au prototypage d'interaction multimodale en entrée, les aspects décrits par McCurdy ne nous permettent pas de comparer adéquatement l'interactivité réalisable avec les outils existants.

Pour décrire plus précisemment l'interaction en entrée d'un prototype, nous considérons les niveaux de fonctionnalité en entrée définis par Coyette [2007]. Trois niveaux de fonctionalité sont identifiés : le prototype dirigé ou animé, le prototype simulé et le prototype interactif :

- Le **prototype dirigé ou animé** sont des prototypes non interactifs. Le prototype dirigé correspond à un prototype papier crayon où le concepteur dirige les différents étapes de l'interface manuellement, en fonction des actions énoncées par l'utilisateur. Par exemple l'utilisateur énonce "je clique sur ce bouton" et le concepteur montre alors une version papier de l'interface qui serait affichée après l'action sur le bouton. Le prototype animé correspond à une version numérique composée de différents images que l'utilisateur peut parcourir. Par exemple, l'utilisation d'une présentation PowerPoint pour concevoir un site web, où l'utilisateur parcourt les diapositives afin de découvrir les différentes pages du site.

- Le prototype simulé est basé sur la technique du Magicien d'Oz [Kelley, 1984], qui consiste à simuler une interaction en faisant croire au sujet de l'expérience que le système interactif est fonctionnel.

- Finalement, le prototype interactif s'exécute sur un ordinateur et répond aux actions de l'utilisateur. Ce type de prototype implémente donc au moins le composant de Présentation et le composant d'Interaction dans une architecture Arch (chapitre 3 section 3.3.4).

Dans notre approche, nous nous intéressons aux prototypes en tant qu'approximation de l'interface finale et donc nous focalisons sur des prototypes qui permettent à l'utilisateur d'effectivement interagir de manière multimodale avec le système. Par rapport aux types de prototypes énoncés précédemment, cela correspond au prototype simulé et au prototype interactif.

Aussi, nous limitons notre étude des outils de prototypage à ceux qui permettent de créer ces deux types de prototypes, en les présentant en ordre croissant de fonctionnalité. La Figure 6.1 illustre l'organisation des sections du chapitre par rapport au niveau de fonctionnalité des prototypes considérés.

Pour étudier les outils existants selon ces deux niveaux de fonctionnalité, nous dressons une grille d'analyse commune.

6.2 Grille d'analyse des outils existants

Nous analysons les outils de prototypage à trois niveaux d'abstraction différents : à haut-niveau, nous analysons les aspects liés à l'approche conceptuelle de la multimodalité ; au niveau intermédiaire, nous analysons la notation de spécification sur laquelle repose l'outil ; et à bas-niveau, nous considérons des critères généraux liés à l'outil et à ce qu'il est possible de faire avec l'outil.

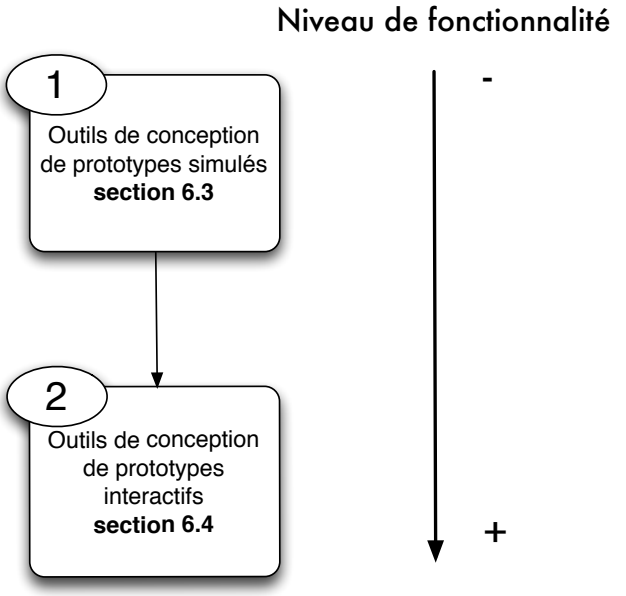


FIGURE 6.1 – Organisation du chapitre.

6.2.1 Critères liés à la multimodalité

À haut-niveau d'abstraction, nous nous intéressons aux critères liés à la multimodalité selon deux facettes. D'abord, nous considérons les modalités en entrée ou en sortie gérées par les outils. Ensuite, nous analysons les fondements conceptuels liés à la multimodalité. Pour cela, nous analysons les outils au regard des modèles conceptuels présentés au chapitre 3.

6.2.2 Critères liés à la notation

Pour chaque outil, nous étudions la notation que les utilisateurs doivent utiliser pour réaliser un prototype. Pour cela, nous utilisons les dimensions cognitives de Green [Green, 1989], introduites au chapitre 5. Ainsi, nous reprenons les dimensions de *viscosité*, *visibilité*, *engagement forcé* et *expressivité du rôle* pour analyser les outils de prototypage.

6.2.3 Critères liés à l'outil

Finalement, nous nous intéressons à caractériser de façon générale l'outil par rapport à sa prise en main et aux possibilités qu'il offre. Pour cela, nous utilisons les critères énoncés par Myers de *seuil*[1] et *plafond*[2] [Myers *et al.*, 2000]. Le seuil représente la difficulté de prise en main d'un outil. Le plafond désigne les possibilités offertes par l'outil (ce que l'on peut réaliser avec l'outil). Tout outil doit idéalement offrir un seuil bas et un plafond élevé. Ceci est particulièrement important pour les outils de prototypage qui doivent permettre de réaliser rapidement des prototypes variés dans le cadre d'une conception itérative centrée sur l'utilisateur.

[1] threshold
[2] ceiling

6.2.4 Grille résultante

Les trois catégories de critères énoncés précédemment permettent d'établir une grille d'analyse des outils de prototypage (Tableau 6.1). La première ligne de cette grille correspond à une description générale de l'outil. Ensuite nous introduisons les critères en ordre décroissant d'abstraction : du haut-niveau d'abstraction vers le bas-niveau d'abstraction.

Carte d'identité	**Information générale** sur l'outil
Critères liés à la multimodalité	**Modalités** en entrée ou en sortie gérées **Modèle conceptuel** sous-jacent (chapitre 3)
Critères liés à la notation [Green, 1989]	**Viscosité** : Résistance au changement **Visibilité** : Possibilité d'identifier facilement les éléments de la notation **Engagement forcé** : Obligation de réaliser les actions de conception dans un certain ordre **Expressivité du rôle** : Facilité avec laquelle la fonction d'un élément de la notation est déduite
Critères liés à l'outil [Myers *et al.*, 2000]	**Seuil** : Difficulté de prise en main d'un outil **Plafond** : Combien peut être achevé avec l'outil (les possibilités de l'outil

TABLE 6.1 – Grille d'analyse des outils de prototypage.

Pour chaque outil étudié, nous résumons leurs points forts et leurs points faibles par rapport à ces critères.

6.3 Prototypes simulés : outils de prototypage

Nous introduisons d'abord la technique du Magicien d'Oz sur laquelle repose un prototype simulé. Nous étudions ensuite les outils pour développer de prototypes simulés. Initialement utilisée pour la reconnaissance de parole, nous constatons que les outils de prototypage de ce type sont souvent dédiés à une seule modalité.

6.3.1 Introduction à la technique du Magicien d'Oz

La technique du Magicien d'Oz [Kelley, 1984] consiste à simuler une interaction en faisant croire au sujet de l'expérience que le système interactif est fonctionnel.

Petite histoire du Magicien d'Oz

La technique du Magicien d'Oz (WoZ en Anglais[3]) fut introduite par J. F. Kelley en 1984 [Kelley, 1984]. Appelée d'abord *Paradigme d'Oz*, elle impliquait la simulation d'une application de reconnaissance vocale. Les utilisateurs croyaient pouvoir parler à la machine comme à un être humain[4], alors que les actions du système étaient en réalité simulées par un opérateur qui interpretait les commandes des utilisateurs. Le nom de l'expérience vient du roman *Le Magicien d'Oz*, où un homme caché derrière un rideau se fait passer pour un puissant magicien.

[3] Wizard of Oz (WoZ)

[4] «Central to the methodology is an experimental simulation which I call the OZ paradigm, in which experimental participants are given the impression that they are interacting with a program that understands English as well as another human would.» [Kelley, 1984]

Initialement utilisée pour simuler des commandes vocales, en raison de la difficulté à implémenter ce genre d'interfaces et aussi pour identifier le lexique et la syntaxe des commandes que l'utilisateur spontanément utiliserait, le Magicien d'Oz a servi par la suite à simuler d'autres techniques d'interaction comme le geste, la localisation ou les émotions.

Utilité en conception

Le magicien, appellé aussi compère, est responsable de la simulation de l'interface dans une expérience du type Magicien d'Oz. Il a donc un rôle d'opérateur pendant le déroulement de l'expérience. Le compère peut généralement remplir deux rôles d'opérateur différents : *contrôleur* ou *superviseur* [Dow *et al.*, 2005]. Un *compère contrôleur* simule complétement une partie du système interactif. Un *compère superviseur* surveille une application interactive fonctionnelle et intervient dans le cas où le système commet une erreur.

Généralement, ces rôles vont évoluer lors de la conception. Dans les premières versions du prototype, le compère aura un rôle de contrôleur. Au fur et à mesure que le prototype s'affine et devient fonctionnel, le compère aura un rôle moins important pour devenir superviseur. Ainsi, la charge de travail du compère et du concepteur s'intervertissent pendant la phase de conception d'une application interactive (Figure 6.2).

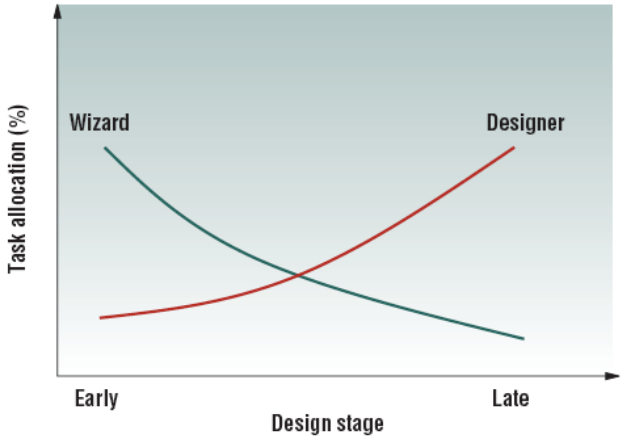


FIGURE 6.2 – Relation entre la charge de travail du compère et celle du concepteur pendant la phase de conception. Illustration extraite de [Li *et al.*, 2007].

La Figure 6.3 illustre comment la technique du Magicien d'Oz permet de simuler à chaque étape de la conception les parties technologiques manquantes, séparant l'implémentation actuelle, représentée par la courbe bleue, de l'implémentation finale souhaitée, représentée par le nuage. Ainsi, dans les premières étapes de la conception (à gauche sur la Figure 6.3), le prototype est faiblement fonctionnel et la distance avec l'implémentation souhaitée est grande. Lorsque la conception avance, la distance diminue (à droite sur la Figure 6.3).

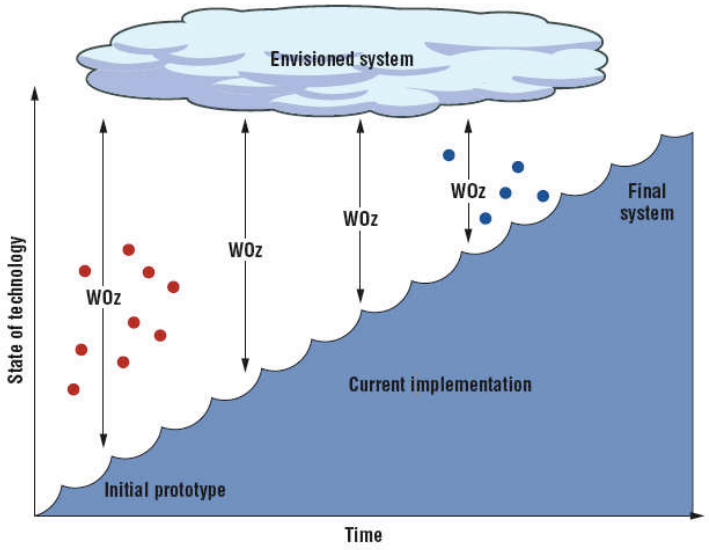


FIGURE 6.3 – Utilité de la technique du Magicien d'Oz dans la construction d'un système interactif. Illustration extraite de [Dow *et al.*, 2005].

Aspects relatifs à la multimodalité

La simulation d'une interaction multimodale par Magicien d'Oz pose un certain nombre de problèmes. Daniel Salber identifie deux difficultés spécifiques à la multimodalité [Salber, 1992] : le pôle compère et la généricité.

Le *pôle compère* représente le besoin d'avoir plusieurs compères pour simuler l'interaction. Selon Salber, une expérience avec une seule modalité peut être réalisée par un seul compère, mais lorsque les modalités se multiplient, le volume d'information à traiter et à simuler devient trop important. L'augmentation de la bande passante d'information rend plus complexe la tâche du compère[5]. Ainsi, le recours à plusieurs compères devient nécessaire. La Figure 6.4 illustre un exemple de configuration multi-compère, où chaque compère simule une modalité d'interaction ou une tâche différente.

La *généricité* est d'après Salber une caractéristique nécessaire dans les expériences Magicien d'Oz pour des interfaces multimodales. En opposition avec les études focalisant sur des modalités précises, une plate-forme Magicien d'Oz pour la multimodalité doit être réutilisable pour n'importe quelle nouvelle modalité. En effet, le coût de développement d'une expérience Magicien d'Oz doit rester limité afin de permettre des itérations rapides dans la démarche de conception.

6.3.2 Outils Magicien d'Oz

Sketchwizard

Sketchwizard [Davis *et al.*, 2007] est un outil magicien d'oz pour le prototypage d'applications basées sur des gestes réalisés avec un stylet. L'objectif de Sketch-

[5] «From the wizard's perspective, multimodality (...) increases the complexity of the task as well as information bandwidth.»[Salber et Coutaz, 1993]

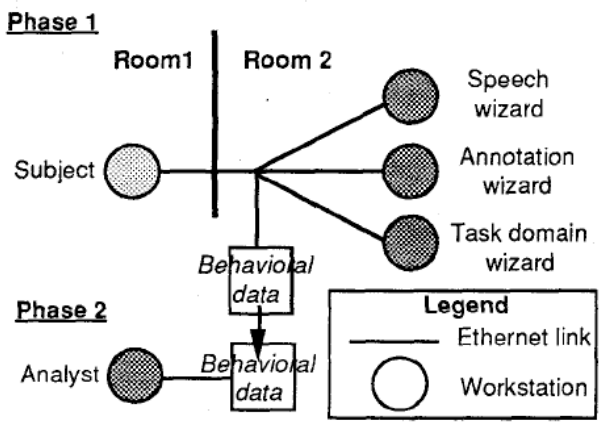


FIGURE 6.4 – Configuration multi-compère dans Neimo. Illustration extraite de [Coutaz *et al.*, 1996].

Wizard est de permettre la simulation de différentes applications basées sur l'utilisation du stylet. L'outil est destiné à des concepteurs sans connaissances de programmation.

L'interface affichée à l'utilisateur est simple et générique, constituée d'un espace de dessin et d'un petit ensemble d'outils de dessins (à gauche sur la Figure 6.5). L'espace de travail du compère (à droite sur la Figure 6.5) est composé de plusieurs éléments : un ensemble d'outils (à gauche), un espace d'édition (au centre de l'interface) et deux boutons pour confirmer les modifications. Le compère peut voir les dessins réalisés par le sujet, les modifier afin de simuler la reconnaissance de gestes et afficher le dessin interprété. Par exemple, le compère peut remplacer un rectangle dessiné à la main par le sujet, par un rectangle vectoriel (à droite sur la Figure 6.5).

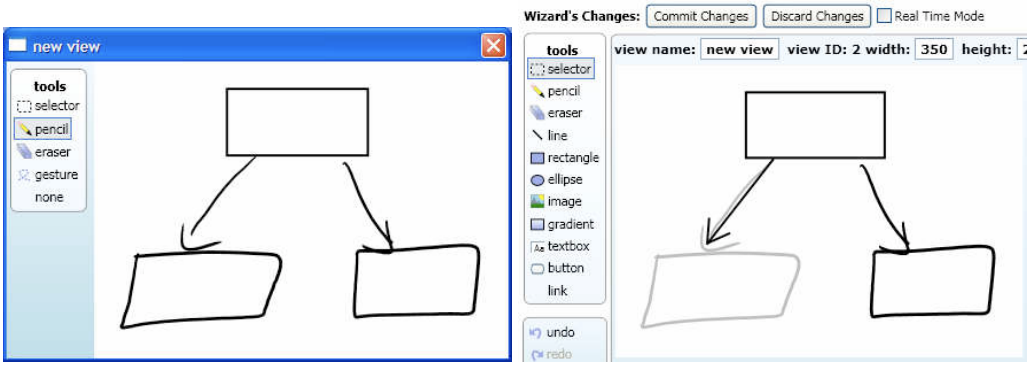


FIGURE 6.5 – Interface du sujet (gauche) et interface du compère (droite) dans une expérience réalisée avec SketchWizard. Illustration extraite de [Davis *et al.*, 2007].

La surimpression du dessin réalisé par l'utilisateur et du dessin réalisé par le compère dote l'outil d'une forte visibilité. L'interface du compère est simple et les outils sont clairement identifiés par des représentations graphiques (sur la partie

gauche de la fenêtre du compère), ce qui augmente l'expressivité du rôle et diminue le seuil dans l'outil. SketchWizard reste toutefois un outil limité dans ses possibilités de simulation : les marking menus, qui reposent sur une interaction gestuelle, ne sont pas considérés par exemple. De même, SketchWizard ne considère pas l'intégration de modules de reconnaissance de gestes afin de réaliser des prototypes semi-simulés. Ainsi, le plafond global de l'outil (ses possibilités) sont limitées. Le Tableau 6.2 résume les caractéristiques de SketchWizard.

ID	Création de prototypes Magicien d'Oz basés sur des gestes réalisés avec un stylet
Multimodalité	Une seule modalité : Geste avec un stylet
Notation	Forte visibilité, forte expressivité du rôle
Outil	Seuil bas, plafond bas

TABLE 6.2 – Fiche-résumé de SketchWizard.

Suede

Suede [Klemmer *et al.*, 2000] est un outil pour le prototypage d'applications à interfaces vocales. Comme SketchWizard, Suede propose une interface de prototypage simple à utiliser afin de permettre aux concepteurs de créer rapidement des prototypes.

Suede est basé sur trois étapes de conception : Conception, Test et Analyse. Dans la phase de conception, les concepteurs créent des conversations-type qui auront lieu entre le système et l'utilisateur, par exemple :

- Système : *Hello, what is your name ?*
- Réponse : *name*
- Système : *What would you like to do ?*
- etc.

Ces conversations sont traduites dans une grammaire, explicitée par un graphe (Figure 6.6).

Ensuite, dans la phase de test, Suede génère une page html à partir du graphe, où les noeuds correspondant aux réponses sont représentés par des hyperliens. Cela permet au compère de jouer des fragments de conversation (les noeuds du graphe) selon les réponses du sujet. La Figure 6.7 illustre un fragment du graphe correspondant à la phase de conception et la page html correspondante générée pour la phase de test. Dans cet exemple, lorsque le sujet répond *Read email* à la question *What would you like to do ?*, posée par le système, le compère click sur le lien *Read email* sur la page html (Figure 6.7). La phase d'analyse permet ensuite de rejouer la vrai discussion maintenue entre le compère et le sujet.

L'interface de Suede permet de facilement visualiser le graphe d'une conversation (Figure 6.7), dotant ainsi l'outil d'une forte visibilité. De plus, les différents éléments de la conversation (questions du système et réponses de l'utilisateur) sont identifiées graphiquement par des couleurs différentes, augmentant ainsi l'expressivité du rôle. La création du graphe se fait par manipulation directe avec la souris,

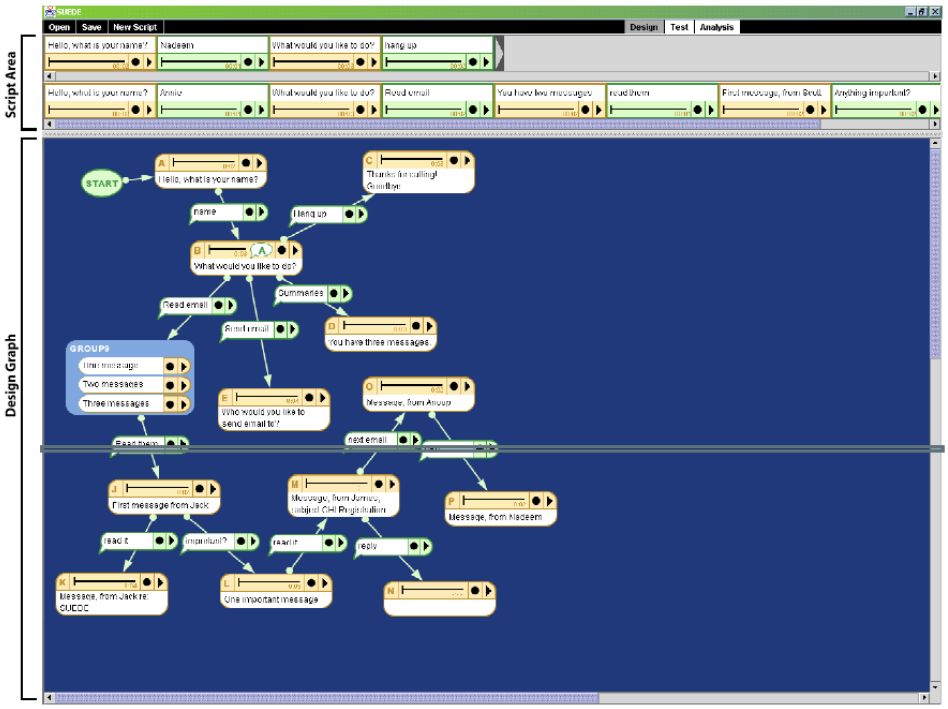


FIGURE 6.6 – Interface correspondant à la phase de conception dans Suede. Illustration extraite de [Klemmer *et al.*, 2000].

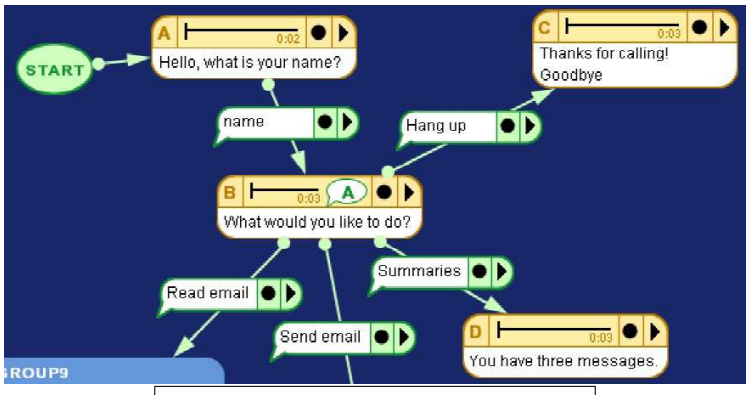


FIGURE 6.7 – Détail de graphe correspondant à la phase de conception (en haut) et page html correspondante générée pour la phase de test (en bas) dans Suede. Illustration extraite de [Klemmer *et al.*, 2000].

en important dans la zone du graphe des éléments (élément question système, élément réponse utilisateur), ce qui rend l'outil assez simple à utiliser (seuil bas). Cet outil reste un outil pour prototyper des interfaces de type langagière basées sur la reconnaissance de la parole. L'outil ne considère donc pas d'autres modalités d'interaction (plafond bas). Le Tableau 6.3 résume les caractéristiques de Suede.

ID	Création de prototypes Magicien d'Oz à interfaces vocales
Multimodalité	Une seule modalité : parole Modèle à automates
Notation	Forte visibilité, forte expressivité du rôle
Outil	Seuil bas, plafond bas

TABLE 6.3 – Fiche-résumé de Suede.

Topiary

Topiary [Li *et al.*, 2004] est un outil pour la construction d'expériences Magicien d'Oz avec simulation de la localisation de l'utilisateur. Topiary permet au concepteur de créer des cartes avec des points d'intérêt (Point of Interest en anglais[6]) et des sujets (par exemple, un gymnase comme POI et Bob comme sujet), d'écrire un scénario d'interaction en lien avec ces POI (par exemple "Bob entre dans le gymnase") puis d'exécuter le système en mode Magicien d'Oz sur un dispositif mobile (à gauche sur la Figure 6.8). Le compère peut alors actualiser la position du sujet sur la carte (au centre et à droite sur la Figure 6.8). Le sujet observe la carte sur PDA et peut utiliser un stylet pour pointer sur des POI (à gauche sur la Figure 6.8). Les liens associés aux POI ont été définis préalablement par le compère, qui ne simule que la localisation de l'utilisateur.

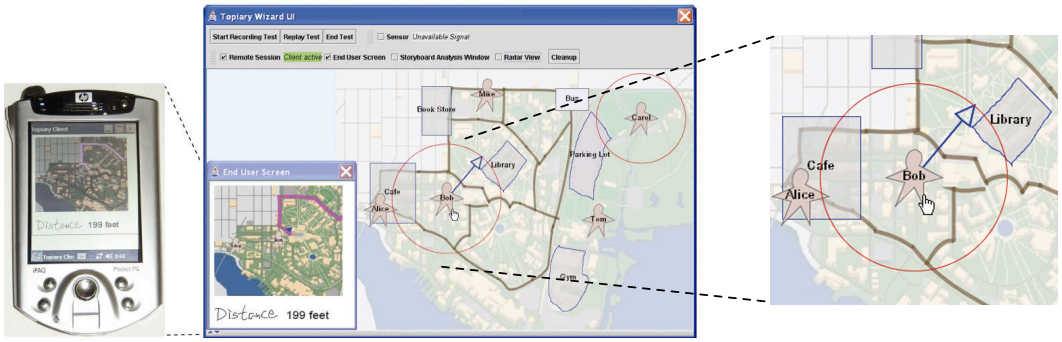


FIGURE 6.8 – Interface de l'utilisateur sur PDA (à gauche), interface compère (au centre) et détail sur l'interaction du compère afin de déplacer la position de l'utilisateur (à droite) dans Topiary. Illustration extraite de [Li *et al.*, 2004].

L'affichage sur l'interface du compère de la fenêtre de l'utilisateur ainsi que la représentation de l'utilisateur par une icône manipulable directement avec la souris dote l'outil d'une forte visibilité et d'un seuil bas. Néanmoins, l'outil est limité

[6]Point of Interest (POI)

ne permettant pas de simuler d'autres modalités d'interaction que la position de l'utilisateur (plafond bas) et est dédié à des systèmes de navigation basés sur une carte. Le Tableau 6.4 résume les caractéristiques de Topiary.

ID	Création de prototypes Magicien d'Oz basés sur la géolocalisation
Multimodalité	Géolocalisation de l'utilisateur et stylet sur PDA
Notation	Forte visibilité
Outil	Seuil bas, plafond bas

TABLE 6.4 – Fiche-résumé de Topiary.

6.4 Prototypes interactifs : outils de prototypage

Dans cette section, nous introduisons des outils de conception de prototypes interactifs. Pour cela, nous étudions à la fois des outils clairement dédiés au prototypage mais aussi des outils de développement d'interfaces multimodales. Nous les organisons selon le modèle de programmation utilisé, en nous reposant sur la taxonomie présentée au chapitre 3. Nous rappelons que cette taxonomie considère d'un côté les modèles de programmation pour l'interaction homme-machine (à fonctions de rappel, à automates et à flot de données) et d'un autre côté les modèles de programmation pour non-programmeurs (programmation sur exemple, par démonstration et visuelle).

Nous considérons ainsi successivement les outils à programmation non-visuelle, à programmation visuelle à automates, à programmation visuelle par démonstration et à programmation visuelle à flot de données.

6.4.1 Outils à programmation non-visuelle

Nous présentons d'abord deux outils de prototypage multimodal à programmation non-visuelle largement utilisés par la communauté de créateurs numériques, Processing et OpenFrameworks.

Processing

Processing [Reas et Fry, 2003] est à la base un environnement de développement d'interfaces graphiques pour le web. Par la suite, Processing a intégré des librairies pour réaliser différents types d'interaction : l'installation par défaut de Processing est accompagnée de librairies pour utiliser les entrées standard (souris, clavier), pour faire de l'analyse vidéo (détection de mouvement, suivi de couleur, distinction de blobs) et audio en temps réel. L'utilisation de capteurs physiques (Phidgets et Arduino) est également possible en installant des librairies additionnelles. Processing est conçu pour des personnes ayant peu d'expérience en programmation.

Processing propose un environnement basé sur Java, avec des commandes propres et une méthodologie simple de programmation basée sur l'utilisation de deux méthodes principales : *setup* et *draw*. La méthode setup permet d'initialiser la taille de la fenêtre graphique principale (commande *size* à la Figure 6.9), la couleur du bord de la fenêtre (commande *stroke* à la Figure 6.9), sa couleur de fond

(commande *background* à la Figure 6.9) ainsi que tous les autres éléments à initialiser selon le cas. La méthode de dessin *draw* est exécutée en boucle indéfiniment et l'utilisateur peut spécifier à l'intérieur de cette méthode le code qui lui permet de réaliser l'interaction en entrée (souris, clavier, caméra, microphone ou capteurs physiques) et/ou en sortie (représentation graphique ou sortie sonore). L'environnement de programmation (IDE) est composé d'une fenêtre d'édition du code (fenêtre du milieu à la Figure 6.9) et d'une fenêtre pour visualiser le résultat graphique (fenêtre à droite à la Figure 6.9).

La Figure 6.9 montre un exemple de programme interactif simple dans Processing. La commande *draw* est composée d'une seule ligne de code, qui dessine une droite entre un point fixe (x,y) et un point défini par la position de la souris (commande *line*). Cette commande *line* illustre bien le type de commandes offertes aux utilisateurs : des abstractions simples de haut niveau.

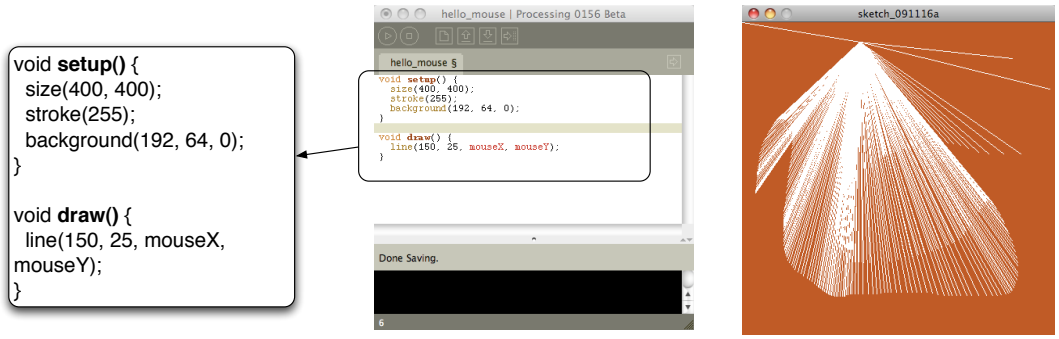


FIGURE 6.9 – Exemple de programme interactif simple dans Processing.

Processing a donné lieu à un autre outil, Arduino, qui permet de programmer des plate-formes de capteurs physiques. Arduino permet de contrôler la plate-forme physique du même nom, en écrivant (output) ou en lisant (input) sur les ports physiques de la plate-forme. La plate-forme contient 12 ports et chaque écriture/lecture d'un port correspond à un signal électronique de 5 V, correspondant à l'intensité de fonctionnement des différents capteurs utilisables avec la plate-forme. La Figure 6.10 montre un exemple d'interface bi-manuelle utilisant Arduino : l'utilisateur joue de l'instrument musical avec des boutons de la main droite et un capteur de pression de la main gauche.

Processing permet donc aux novices ou aux non-programmeurs de réaliser des interactions innovantes facilement (plafond haut). Par contre les possibilités de conception (les librairies) sont faiblement visibles, ce qui freine la conception exploratoire. Aussi le seuil est haut dès que l'on vise des prototypes un peu plus complexes. De plus, lorsque le prototype est complexe, la visibilité du code est très faible étant donné que le IDE est prévu pour des applications simples (faible visibilité). Le Tableau 6.5 reprend les principales caractéristiques de Processing et Arduino.

OpenFrameworks

OpenFrameworks (OF) [Lieberman et Watson, 2009] est une bibliothèque c++ de programmation open source. OF est un projet issu de Processing et de la librairie

FIGURE 6.10 – Interface bi-manuelle développée avec Arduino.

ID	Création d'interfaces graphiques interactives
Multimodalité	Plusieurs modalités en entrée et en sortie : caméra, capteurs physiques, etc.
Notation	Faible visibilité
Outil	Seuil haut, plafond haut

TABLE 6.5 – Fiche-résumé de Processing, Arduino et MobileProcessing.

c++ ACU Toolkit, créée au MIT par Ben Fry (créateur également de Processing). OF offre des méthodes simples qui permettent de créer rapidement et simplement des applications interactives. OF a pour but de permettre à des artistes ou concepteurs d'explorer et expérimenter des nouvelles formes d'interaction.

La méthode de programmation de OF est basée sur le même principe que Processing : une méthode *draw* qui s'exécute en boucle infinie. La principale différence avec Processing réside dans sa performance et dans les librairies existantes, beaucoup plus nombreuses que celles de Processing. En effet, l'utilisateur de OF peut accéder à n'importe quelle librairie c++, comme des librairies pour réaliser de l'analyse vidéo temps réel. L'exemple illustré à la Figure 6.11 utilise une caméra qui détecte des jetons et des gestes des mains afin de permettre d'interagir de façon bimanuelle avec un système de visualisation de photos. OF permet également de développer facilement des applications interactives sur iPhone.

Néanmoins, l'augmentation du pouvoir de génération grâce aux librairies c++ (plafond haut) a son contre-coût dans la difficulté supplémentaire d'apprentissage : même si l'ensemble reste destiné à des programmeurs débutants, la programmation est plus difficile que sur Processing (seuil haut). La notation, basée sur C++, offre moins de visibilité qu'une notation graphique. Le Tableau 6.6 reprend les principales caractéristiques de OpenFrameworks.

Contrairement à Processing et OpenFrameworks, nous présentons deux autres outils destinés à des programmeurs. Ces deux outils sont implémentés de façon conjointe et basés sur des approches différentes : une extension d'une boîte à outils graphique, Java Swing MM, et une approche à agents, Hephais TK [Dumas *et al.*, 2008a].

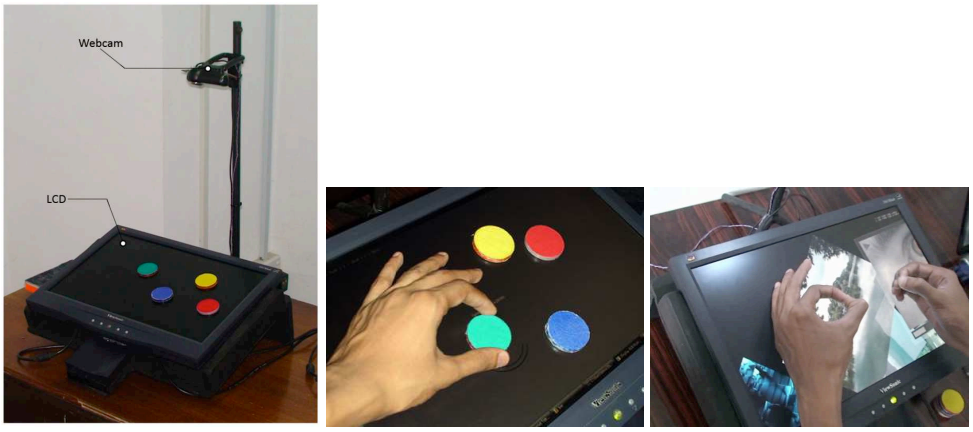


FIGURE 6.11 – iDisplay, un système interactif implémenté avec OpenFrameworks basé sur la manipulation de jetons et des gestes bimanuelles.

ID	Applications graphiques interactives
Multimodalité	Plusieurs modalités en entrée et en sortie : caméra, capteurs physiques, etc.
Notation	faible visibilité
Outil	Seuil haut, plafond haut

TABLE 6.6 – Fiche-résumé de OpenFrameworks.

Java Swing MM

Java Swing MM [Dumas *et al.*, 2008a] est une extension de Java Swing pour la création d'interfaces multimodales. L'objectif principal de cette approche est de proposer un outil pour le développement d'interfaces multimodales en utilisant une plateforme d'interfaces graphiques standard. L'objectif est d'augmenter les widgets de boîtes à outils graphiques, de façon à les rendre multimodales. Ainsi, cet outil de développement peut être utilisé pour du prototypage.

Pour cela, l'architecture repose sur trois composants :

- Le **GUI-Integrator** crée les objets graphiques, sans avoir connaissance des types d'entrées ;
- Le **Media-Integrator** implémente les entrées du système ;
- Le **MMUI-Integrator** étend le composant graphique afin d'intégrer les entrées.

Cette approche est illustrée par un bouton graphique, étendu afin de le rendre multimodal (Figure 6.12) : l'utilisateur peut activer le bouton en utilisant les Phidgets, des manettes ou la voix.

Cette approche est donc peu inter-opérable en dehors de JavaSwing. L'utilisation de l'outil reste difficile pour des programmateurs débutants (seuil haut) et ses possibilités sont améliorées par javaSwing (plafond haut). Le Tableau 6.7 reprend les principales caractéristiques de Java Swing MM.

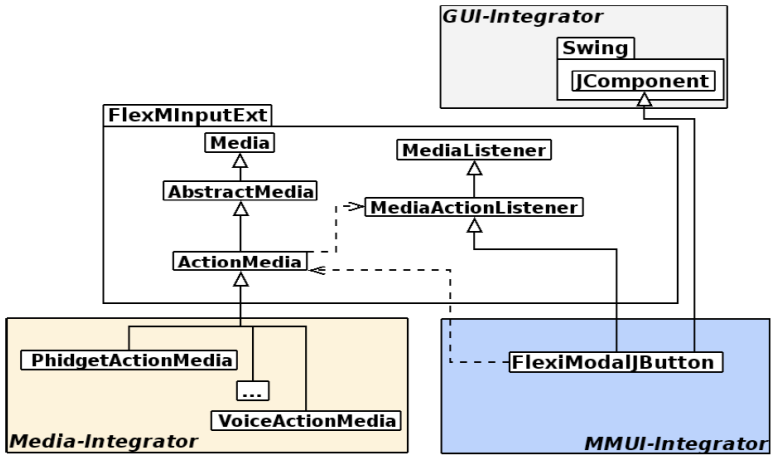


FIGURE 6.12 – Bouton multimodal implémenté sous Java Swing MM. Illustration extraite de [Dumas *et al.*, 2008a] .

ID	Boîte à outils Java
Multimodalité	Phidgets, manettes et reconnaissance vocale
Notation	Inter-opérabilité avec JavaSwing
Outil	Seuil haut, Plafond haut

TABLE 6.7 – Fiche-résumé de Java Swing MM.

Hephais TK

Hephais TK [Dumas *et al.*, 2008b] est un outil pour le développement d'interfaces multimodales basé sur une approche à agents et implémenté en utilisant le langage W3C EMMA [EMMA, 2010]. Le système est composé d'un "facteur" qui centralise les informations venant des agents de "reconnaissance en entrée" (Figure 6.13). Chaque agent de reconnaissance est assigné à un dispositif en entrée. Le facteur va ensuite envoyer les messages reçus en entrée au module de fusion, qui va intégrer les différents messages en entrée. La fusion des entrées dans Hephais TK est définie sous SMUIL (Synchronized Multimodal User Interfaces Modelling Language) [Dumas *et al.*, 2008c]

Les auteurs soulignent que leur solution est trop complexe pour la réalisation d'interactions simples (seuil haut) [7]. Aussi même pour développer un prototype simple, l'exercice sera difficile. L'absence d'un outil graphique pour ajouter des entrées et concevoir l'interaction résulte aussi dans une augmentation de la viscosité. Le Tableau 6.8 reprend les principales caractéristiques de Hephais TK.

[7] «the toolkit is a bulky software, maybe too much bulky for simple use cases» [Dumas *et al.*, 2008b]

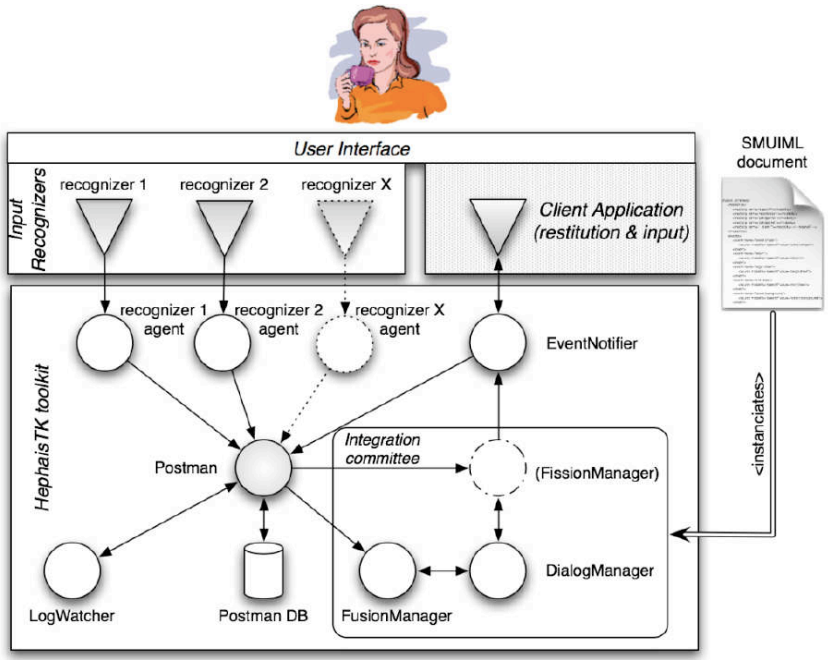


FIGURE 6.13 – Architecture de l'outil Hephais TK. Illustration extraite de [Dumas *et al.*, 2008a].

ID	Outil pour le prototypage multimodal
Multimodalité	Différentes modalités en entrée : parole, table multi-utilisateurs et périphériques matériels. Approche à agents
Notation	Forte viscosité
Outil	Seuil haut

TABLE 6.8 – Fiche-résumé de Hephais TK.

6.4.2 Outils à programmation visuelle à automates

IMBuilder - MEngine

IMBuilder et MEngine [Bourguet, 2002] sont respectivement un outil et une plateforme pour la conception et l'évaluation rapide d'interfaces multimodales.

IMBuilder est un outil graphique pour la spécification d'interfaces multimodales. La spécification d'une interaction multimodale avec cet outil se fait à l'aide d'une machine à états, comme illustré à la Figure 6.14. Comme nous l'avons expliqué à la section 3.3.1 du chapitre 3, au sein des machines à états, les interactions en entrée sont associées aux transitions (par exemple *clique souris*) et les états (les cercles) correspondent aux états du système interactif. La Figure 6.14 illustre un exemple d'application interactive où l'utilisateur peut utiliser la voix ou la souris afin de déplacer un objet.

Les interfaces multimodales spécifiées avec IMBuilder peuvent ensuite être exécutées dans la plateforme multimodale MEngine. MEngine est composé d'un moteur multimodal responsable de l'éxecution des entrées (parole et geste dans

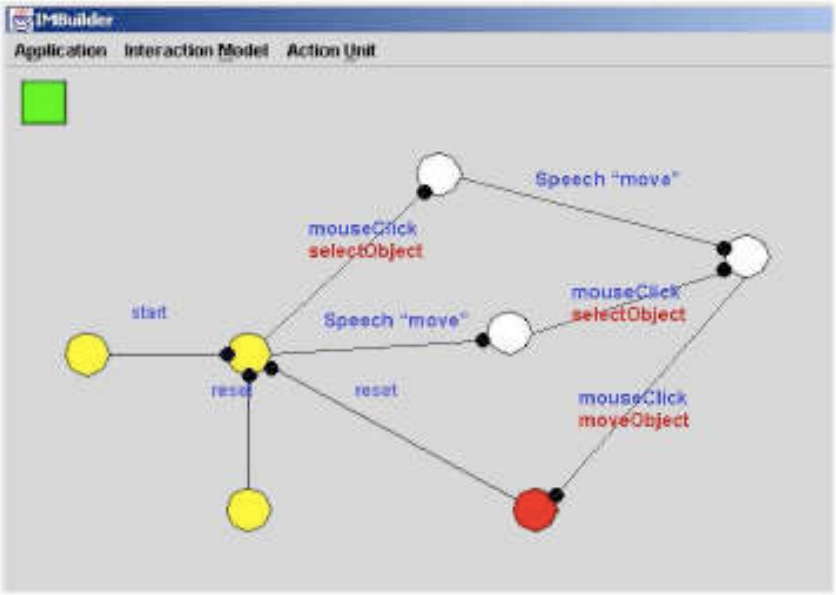


FIGURE 6.14 – Machine à états modélisant une commande multimodale de déplacement d'un objet graphique dans IMBuilder. Illustration extraite de [Bourguet, 2002] .

l'exemple de la Figure 6.15), et de la reconstruction de l'interaction multimodale spécifiée par un fichier xml. Le fichier xml est généré à partir de l'automate spécifié dans IMBuilder.

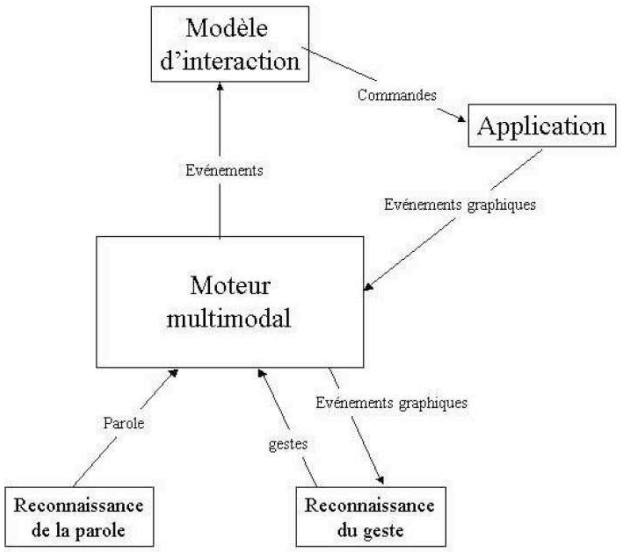


FIGURE 6.15 – Architecture de la plateforme MEngine. Illustration extraite de [Bourguet, 2002] .

L'utilisation d'une machine à états pour développer des interfaces multimodales permet en particulier de décrire la dynamique de l'interaction, c'est-à-dire l'enchaî-

nement des actions de l'utilisateur selon des modalités différentes. La notation est facilement compréhensible et permet de lire rapidement la description de l'interaction (forte visibilité et seuil bas). Néanmoins, la gestion des entrées reste assez pauvre dans l'outil : seuls le geste à travers la souris et la parole étant pris en compte (plafond bas). Cet outil n'est donc pas conçu pour réaliser une exploration rapide de tout l'espace de conception. Le Tableau 6.9 reprend les principales caractéristiques de IMBuilder.

ID	Conception d'interfaces multimodales en entrée
Multimodalité	Gestes avec la souris et parole Modèle à automates
Notation	Forte visibilité
Outil	Seuil bas, plafond bas

TABLE 6.9 – Fiche-résumé de IMBuilder et IMEngine.

Service Counter System (SCS)

Le Service Counter System (SCS) [Dumas *et al.*, 2008a] est une boîte à outils pour la conception et l'implémentation d'interfaces mutimodales basée sur une approche à machine à états. La machine à états permet de définir les états du système interactif ((Figure 6.16). À chaque état est associé un ensemble n de modalités d'interaction, utilisables de façon equivalente (selon la définition CARE, présentée au chapitre 3) par l'utilisateur pour activer la transition suivante. Les modalités intégrées incluent la reconnaissance vocale, les gestes avec la souris et des capteurs physiques.

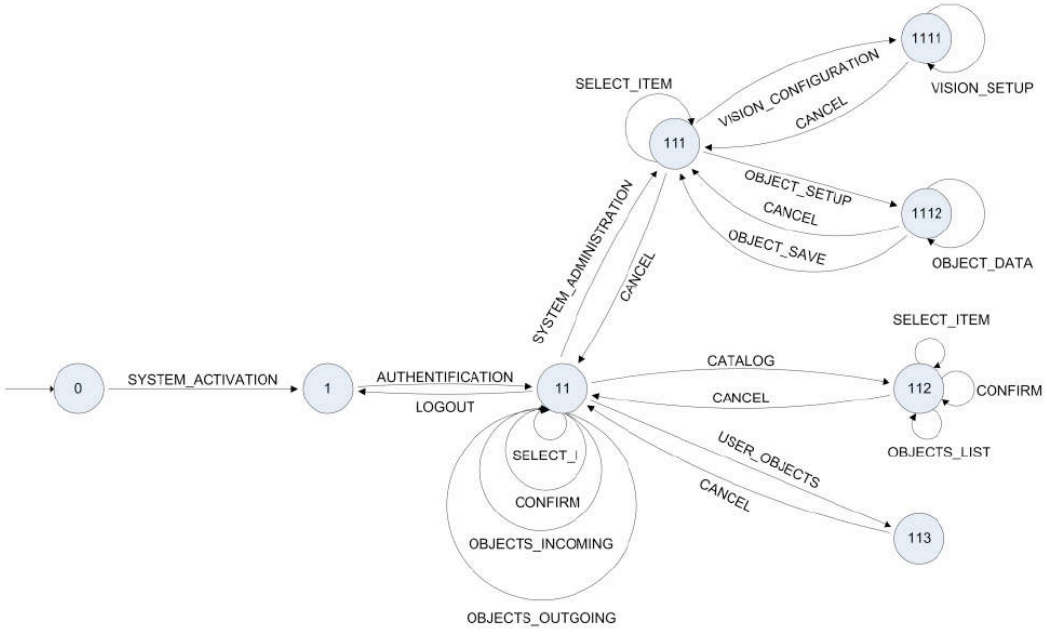


FIGURE 6.16 – Un automate simple modélisant une interaction multimodale dans SCS. Illustration extraite de [Dumas *et al.*, 2008a].

Comme pour l'outil précédent, cet outil présente une forte visibilité. Le Tableau 6.10 résume les principales caractéristiques de SCS.

ID	Conception d'interfaces multimodales en entrée
Multimodalité	Gestes avec la souris, parole et capteurs physiques Modèle à automates, Equivalence (CARE)
Notation	Forte visibilité
Outil	Seuil bas, plafond bas

TABLE 6.10 – Fiche-résumé de SCS.

PetShop

PetShop (Petri nets Workshop) est un environnement de développement d'interfaces multimodales basé sur une approche à réseaux de Petri [Schyn *et al.*, 2003] [Schyn, 2005], introduits à la section 3.3.1 du chapitre 3. L'outil est basé sur l'utilisation des Objets Coopératifs Interactifs (ICO). Le formalisme des ICO est une technique de description formelle permettant la spécification de systèmes multimodaux. Ce formalise utilise les réseaux de Petri pour décrire les aspects comportementaux des systèmes multimodaux. L'environnement PetShop permet l'édition, la vérification et l'exécution de spécifications ICO, supportant ainsi le prototypage rapide de systèmes multimodaux.

L'interface graphique de PetShop (Figure 6.17) contient une zone d'édition (au centre), où le concepteur peut spécifier les réseaux de Petri décrivant le système interactif. L'interface graphique contient deux zones de visualisation des réseaux : un affichage sous forme d'arbre (en bas à gauche) et une représentation matricielle (en bas), ce qui augmente la visibilité de l'outil. De plus, la représentation graphique différenciée des éléments du réseau de Petri (forme géométrique et couleur) augmente l'expressivité du rôle dans l'outil. La notation, basée sur les réseaux de Petri, est plus complexe que la notation d'une machines à états, ce qui augmente le seuil de l'outil. Cette notation permet par contre de représenter autant les aspects structurels ou statiques que les aspects dynamiques ou comportementaux (plafond haut).

Le Tableau 6.11 résume les principales caractéristiques de PetShop.

ID	Développement d'interfaces multimodales
Multimodalité	Souris, clavier et gant - capteur Réseaux de Petri
Notation	Forte visibilité, forte expressivité du rôle
Outil	Seuil haut, plafond haut

TABLE 6.11 – Fiche-résumé de PetShop.

CrossWeaver

CrossWeaver [Sinha et Landay, 2003] est un outil basé sur le prototypage basse fidélité d'applications interactives. CrossWeaver permet de créer des storyboards

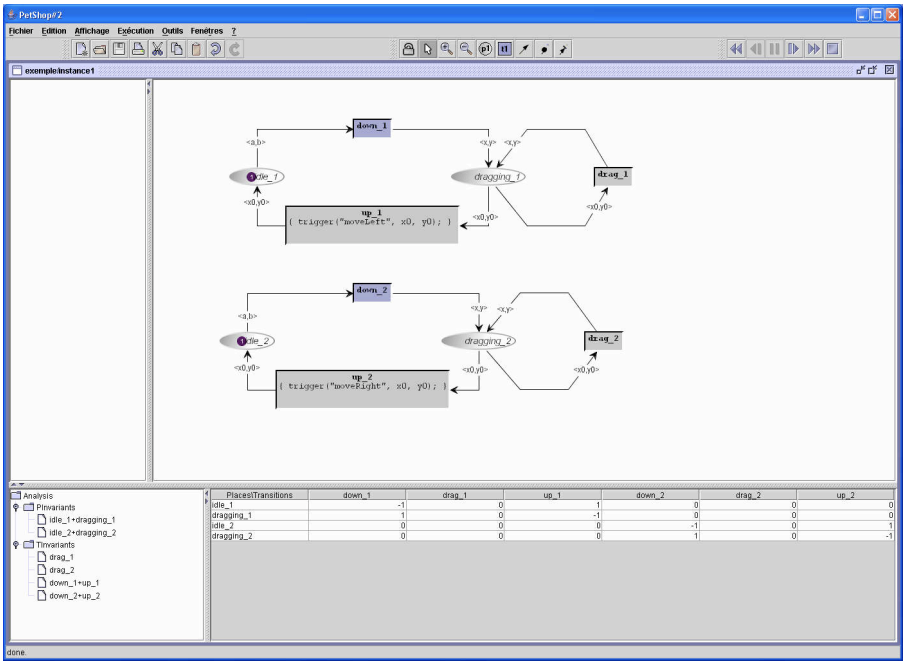


FIGURE 6.17 – Interface graphique de PetsShop. Illustration extraite de [Schyn, 2005].

d'écrans en spécifiant l'interaction multimodale entre les écrans. Ainsi, cet outil n'utilise pas comme les précédents un automate pour programmer l'interaction. Néanmoins l'enchaînement d'écrans peut être assimilé à un automate, chaque écran correspondant à un état de l'automate et les transitions selon des modalités d'interaction à des arcs dans l'automate.

Dans CrossWeaver, le concepteur peut spécifier les modalités d'interaction qui vont permettre de réaliser chaque transition. Les modalités d'interaction sont donc attachées à chaque écran, mais aussi à des objets de l'écran. Quatre modalités d'interaction sont envisagées : le clique souris, le clavier, un geste avec un stylet et la reconnaissance vocale. Le storyboard renseigne également les modalités d'interaction en sortie (graphique ou sonore). La Figure 6.18 illustre l'interface à un instant donné du storyboard.

L'interface simple et la représentation graphique des modalités en entrée et en sortie dotent CrossWeaver d'une forte visibilité et expressivité du rôle ainsi que d'un seuil bas. Cet outil reste toutefois limité dans le nombre de modalités d'interaction utilisables (plafond bas). Le Tableau 6.12 résume les principales caractéristiques de CrossWeaver.

d.tools

D.tools [Hartmann *et al.*, 2006] est un outil pour le prototypage rapide de systèmes utilisant des capteurs physiques. L'outil est centré sur une conception itérative comprenant trois étapes : une première étape de conception qui permet au concepteur de définir le système à travers une machine à états, une deuxième étape de test qui permet de visualiser la trace de l'exécution sur la machine et finalement une phase d'analyse qui permet d'étudier les résultats et les traces du test.

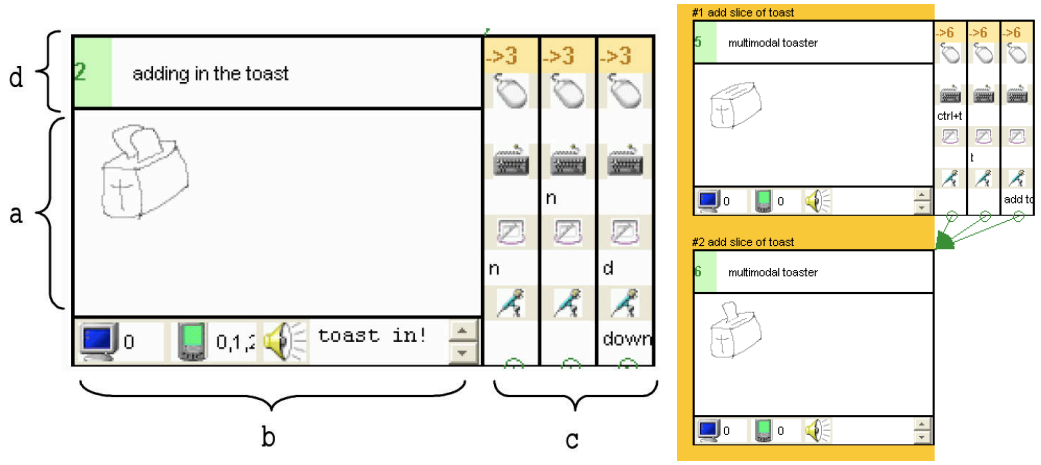


FIGURE 6.18 – À gauche, une scène du storyboard dans CrossWeaver. L'interface contient la fenêtre montrée au sujet (a), les modalités en sortie (b), les modalités en entrée nécessaires pour passer à la scène suivante (c) et un titre (d). À droite, un exemple d'enchaînement de scènes. Illustrations extraites de [Sinha et Landay, 2003].

ID	Conception de prototypes basse-fidélité multimodaux
Multimodalité	Clique souris, clavier, geste avec un stylet et reconnaissance vocale Modèle à automates
Notation	Forte visibilité, forte expressivité du rôle
Outil	Seuil bas, plafond bas

TABLE 6.12 – Fiche-résumé de Crossweaver.

L'interface de la phase de conception de d.tools est composée de l'espace de création du dispositif (zone 1 sur la Figure 6.19), de l'automate représentant les états du prototype (zone 2 sur la Figure 6.19), d'un espace d'édition du code source (zone 3 sur la Figure 6.19) et d'une barre d'outils qui contient les différents types de transitions utilisables (zone 4 sur la Figure 6.19).

L'approche de d.tools inclut également une plate-forme de capteurs physiques qui peut être contrôlée à travers l'éditeur graphique (à droite de la Figure 6.19). Ces capteurs permettent de prototyper différents systèmes interactifs. L'ensemble a été utilisé en particulier pour prototyper des interfaces tangibles sur support mobile (Figure 6.20).

L'utilisabilité de d.tools bénéficie d'une forte expressivité du rôle dans la notation, à travers la représentation graphique des composants. L'outil reste néanmoins limité dans sa capacité d'intégration d'autres dispositifs que les capteurs physiques (plafond bas). Le Tableau 6.13 reprend les principales caractéristiques de d.tools.

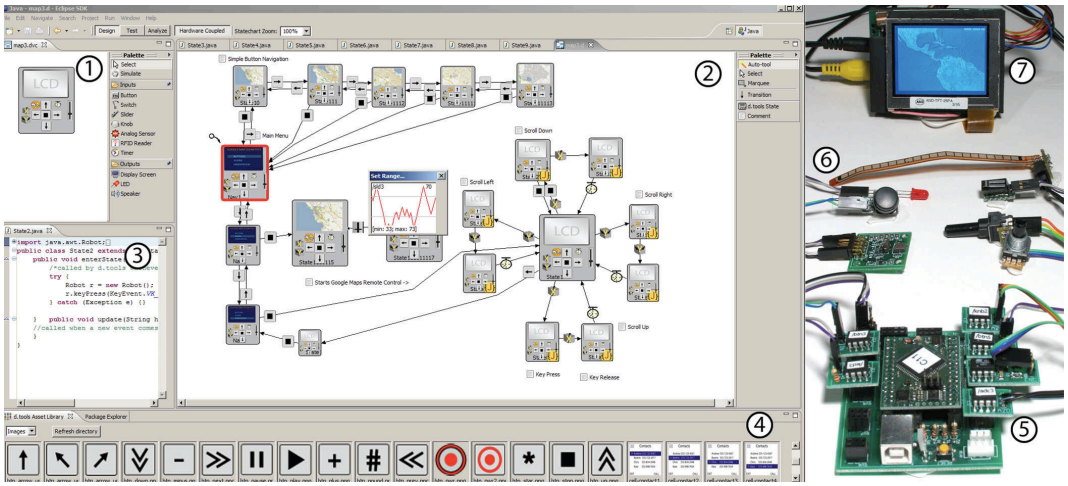


FIGURE 6.19 – Interface graphique de l'outil d.tools. Illustration extraite de [Hartmann *et al.*, 2006].

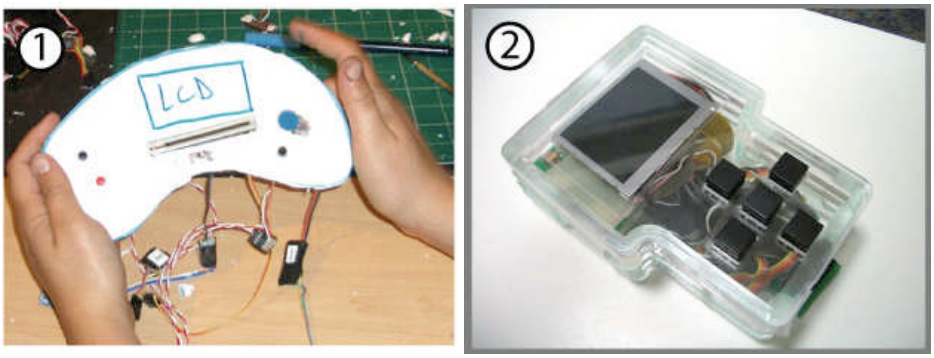


FIGURE 6.20 – Projets réalisés avec d.tools. (1) lecteur de musique ; (2) lecteur multimédia. Illustration extraite de [Hartmann *et al.*, 2006].

ID	outil pour le prototypage rapide de systèmes utilisant des capteurs physiques
Multimodalité	Capteurs physiques : accéléromètre, capteurs de pression et sliders. Modèle à automates
Notation	Forte expressivité du rôle
Outil	Seuil bas, Plafond bas

TABLE 6.13 – Fiche-résumé de d.tools.

6.4.3 Outils à programmation visuelle par démonstration

Exemplar

Exemplar [Hartmann *et al.*, 2007] est un outil pour le prototypage de systèmes interactifs issu de l'outil présenté précédemment, d.tools. Comme d.tools, Exemplar focalise sur l'utilisation de capteurs physiques, tels que des accéléromètres. La principale différence réside dans le modèle de programmation utilisé : alors que dans d.tools le concepteur spécifie l'interaction par une machine à états, dans Exemplar il spécifie l'interaction par démonstration, méthode de programmation que nous avons dèjà présentée à la section 3.3.2 du chapitre 3.

Dans Exemplar le concepteur réalise (*démontre*) les actions ou gestes qui devront être reconnus par le prototype en utilisant des capteurs physiques. La Figure 6.22 illustre des exemples de gestes prototypés avec Exemplar. L'utilisateur peut ensuite voir la trace de sa démonstration et l'annoter afin d'associer des commandes à des actions/gestes particuliers (par exemple à l'inclinaison de la tête vers la droite). Deux types de reconnaissances sont utilisées : une reconnaissance basée sur le seuil, par exemple lorsque l'inclinaison de la tête dépasse un certain angle, et une reconnaissance basée sur des patrons (*patterns* en anglais), par exemple lorsque l'utilisateur réalise deux fois de suite le même geste d'inclinaison de la tête. Ensuite, l'utilisateur peut exécuter le prototype, qui reconnaîtra les actions/gestes prototypés.

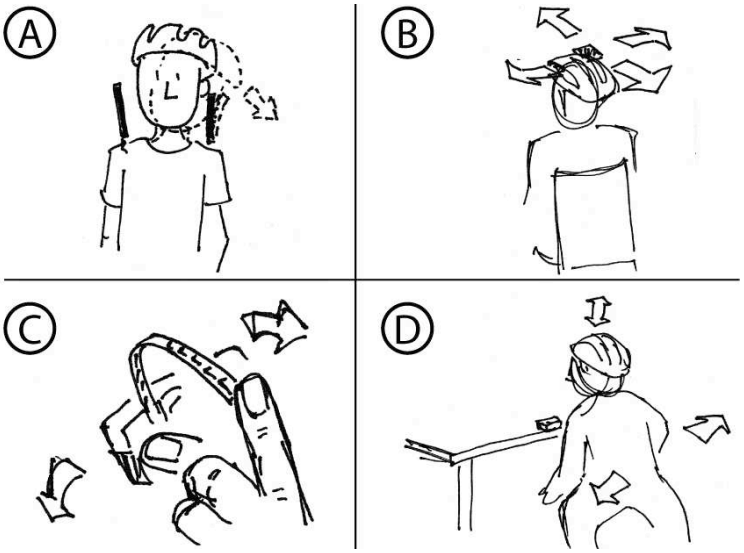


FIGURE 6.21 – Exemples de gestes prototypés avec Exemplar : inclinaison de la tête (A), déplacement de la tête (B), pointage et toucher avec le doigt (C) et déplacement du corps entier (D). Illustration extraite de [Hartmann *et al.*, 2007].

L'interface de Exemplar est composée d'un ensemble de capteurs (zone A à la Figure 6.22) ; d'un ensemble de filtres appliqués au capteur sélectionné (zone B à la Figure 6.22) ; d'un espace de visualisation des donnés du capteur (zone C à la Figure 6.22) ; d'une zone de configuration où l'utilisateur sélectionne le type de reconnaissance à utiliser, seuil ou patron (zone D à la Figure 6.22) ; de la liste des

événements associés au capteur (zone E à la Figure 6.22), représentés également sur un arbre (zone F à la Figure 6.22). Lorsque l'utilisateur sélectionne un évenemment basé su le seuil, celui-ci s'affiche sur la zone de visualisation (zone G à la Figure 6.22).

FIGURE 6.22 – Interface de l'outil Exemplar. Illustration extraite de [Hartmann *et al.*, 2007].

L'évaluation a montré que Exemplar est plus facile à utiliser par des concepteurs que d.tools grâce à son approche de programmation par exemple (seuil bas). La principale limitation de l'approche vient de l'impossibilité d'afficher les données de plusieurs capteurs en même temps, ce qui diminue la visibilité de l'outil. De plus, comme son prédécesseur d.tools, Exemplar ne permet d'utiliser que des capteurs physiques comme entrée, ce qui limite l'espace de conception pour la multimodalité (plafond bas). Le Tableau 6.14 reprend les principales caractéristiques de Exemplar.

ID	Outil pour le prototypage de systèmes interactifs basés sur des capteurs physiques
Multimodalité	Capteurs physiques : accéléromètre, capteurs de pression, température, etc. Programmation par démonstration
Notation	Faible visibilité
Outil	Seuil bas, plafond bas

TABLE 6.14 – Fiche-résumé de Exemplar.

6.4.4 Outils à programmation visuelle à flot de données

vvvv

vvvv [vvvv, 2009] est un outil logiciel pour le traitement temps réel d'images vidéo. vvvv permet également de gérer la manipulation des médias avec différents dispositifs physiques (joysticks, capteurs, etc.).

L'outil est basé sur une approche de programmation visuelle qui vise à fournir un cadre de développement et prototypage rapide. L'utilisateur manipule des composants appelés Nodes. Il existe différentes catégories de nodes : graphique

2d, graphique 3d, dispositif d'interaction en entrée, communication réseau, etc. Par exemple, l'utilisateur peut connecter un composant caméra à plusieurs opérations de traitement vidéo afin de créer une application multimédia (Figure 6.23).

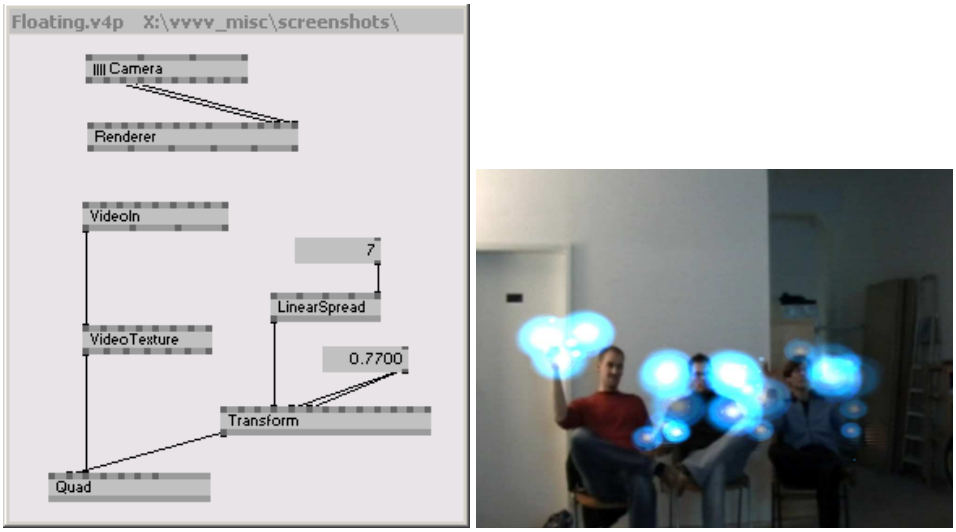


FIGURE 6.23 – Exemple d'assemblage d'opérations et interaction résultante dans vvvv.

Néanmoins, seuls quelques dispositifs sont considérés comme Arduino ou Lemur, une tablette tactile multi-touch [Lemur, 2010], mais le plafond de l'outil reste bas en terme d'interactivité. De plus, la représentation graphiques des nodes, identique pour tous les types de nodes, présente une faible expressivité du rôle. Les exemples d'assemblage qui illustrent l'outil offrent quand même une bonne visibilité globale. Le Tableau 6.15 résume les principales caractéristiques de vvvv.

ID	Traitement temps-réel d'images vidéo
Multimodalité	Plusieurs modalités en entrée : capteurs physiques, tablette tactile. Programmation visuelle à flot de données
Notation	Forte visibilité, faible expressivité du rôle
Outil	Seuil bas, plafond bas

TABLE 6.15 – Fiche-résumé de vvvv.

Max/MSP et Pure Data

Développé à l'Institut de Recherche et Coordination Acoustique/Musique (IRCAM) par Miller Puckette en 1988 sous le nom de The Patcher [Puckette, 1988], Max/MSP est à la base un logiciel de création musicale. Comercialisé par la suite par une autre équipe de développement, Miller Puckette décide de continuer le projet en opensource sous le nom de Pure Data à partir de 1996 [Puckette, 1996].

L'évolution séparée des deux projets a induit quelques petites différences de notation. La principale différence étant que PureData, opensource, bénéficie d'une communauté de développeurs très active qui étend continuellement les capacités du logiciel. Nous nous intéressons donc ici plus à Pure Data qu'à Max/MSP.

Pure Data est un environnement de développement graphique qui permet de réaliser de systèmes musicaux et multimédia temps réel. L'environnement graphique de Pure Data repose sur une décomposition modulaire des éléments programmables (Figure 6.24). Ces éléments, appelés objets, appartiennent à différentes catégories non pré-determinées. En effet, les utilisateurs de PureData sont libres de définir et ajouter des nouveaux objets grâce à une API publique (plafond haut). La liste des objets intégrés est très vaste et inclut entre autre des objets des types suivants :

- Math : implémentent des opérations mathématiques
- Audio Math : implémentent des opérations mathématiques adaptées au traitement audio. Par exemple des transformations de Fourier.
- Gui : permettent de créer des éléments graphiques, comme des boutons ou des barres de défilement.
- Data : permettent de recevoir des données de différents protocoles de communication, comme MIDI (Musical Instrument Digital Interface) ou OSC (Open Sound Control). Ces protocoles permettent d'utiliser différents dispositifs d'interaction comme la plate-forme de capteurs physiques Interface-Z [Interface-Z, 2010].

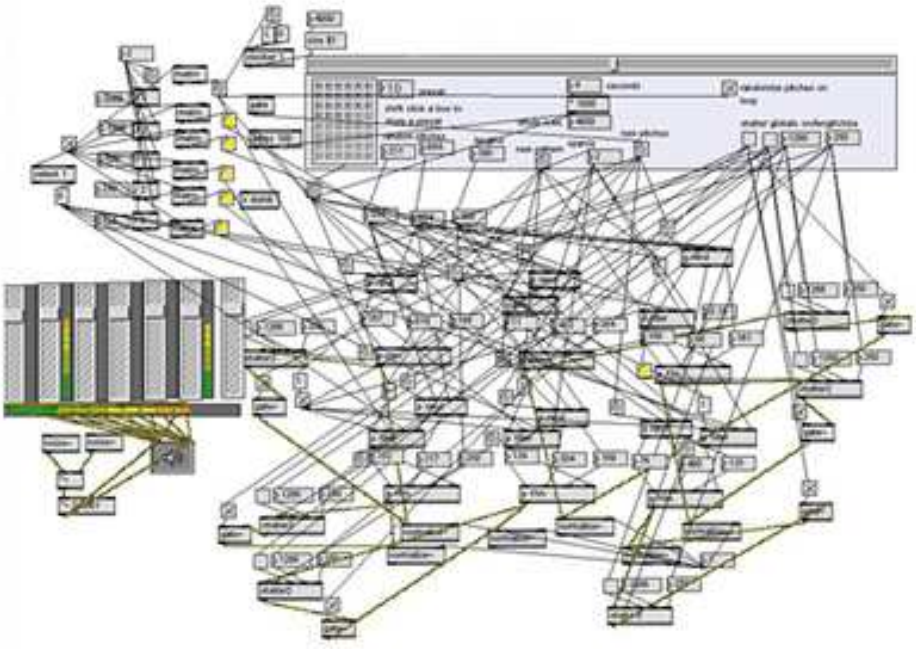


FIGURE 6.24 – Programmation visuelle dans Pure Data.

Comme vvvv, Pure Data ne propose pas une structure précise d'assemblage de composants. Les assemblages ont tendance à devenir très complexes, augmentant

la viscosité et le seuil, et diminuant la visibilité de l'outil (Figure 6.24). Les objets sont des entités paramétrables en temps réel (sans arrêter l'exécution de l'assemblage), ce qui dote PureData d'une grande flexibilité et diminue l'engagement forcé. Le Tableau 6.16 résume les principales caractéristiques de Max/Msp et de PureData.

ID	Créations musicales et multimédia temps réel
Multimodalité	Plusieurs modalités d'interaction utilisant MIDI ou OSC : caméra, capteurs physiques, etc. Programmation visuelle à flots de données
Notation	Forte viscosité, faible visibilité, faible engagement forcé
Outil	Seuil haut, plafond haut

TABLE 6.16 – Fiche-résumé de MaxMsp et PureData.

EyesWeb

EyesWeb [Camurri *et al.*, 2004] est une plate-forme de recherche en interaction multimodale expressive. Les domaines d'application visés par EyesWeb sont les applications multimédia, musicales et de danse temps-réel. La principale interaction gérée par EyesWeb est l'analyse de gestes en temps réel. EyesWeb permet de réaliser quatre types de reconnaissance : couleur, blobs, squelette et posture.

EyesWeb propose un éditeur graphique qui permet d'assembler des composants afin de définir une interaction. Les composants, appelés des Blocks, représentent autant des modules logiciels que physiques (par exemple une caméra, en entrée, ou un écran, en sortie). La plupart des Blocks sont liés à différents traitements et algorithmes de reconnaissance vidéo. La Figure 6.25 illustre un exemple de reconnaissance de mouvement. L'assemblage intègre un Block représentant la caméra en entrée, différents Blocks réalisant les traitements vidéo afin de détecter le mouvement et finalement un Block en sortie (tout à droite de l'assemblage, représenté graphiquement par un écran).

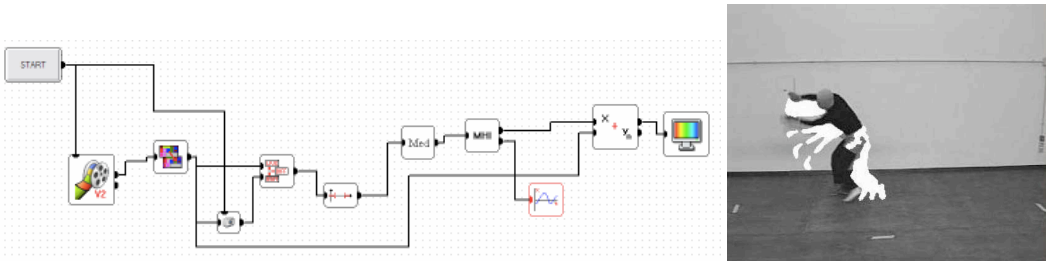


FIGURE 6.25 – Implémentation d'une reconnaissance de mouvement dans un assemblage graphique dans EyesWeb.

EyesWeb reste un outil dédié à l'expérimentation artistique, en particulier autour du mouvement du danseur. Aucun modèle pour construire les assemblages est proposé. Il est par contre intéressant de noter la caractérisation graphique des

composants (par exemple, un écran pour représenter la sortie), qui augmente l'expressivité du rôle des éléments de l'assemblage. Le Tableau 6.17 résume les principales caractéristiques de EyesWeb.

ID	Création d'applications multi-média, musicales et de danse temps-réel
Multimodalité	Analyse de gestes en temps-réel Programmation visuelle à flots de données, paradigme des dispositifs en cascade
Notation	Forte expressivité du rôle, forte visibilité
Outil	Seuil bas, plafond bas

TABLE 6.17 – Fiche-résumé de EyesWeb.

iStuff Mobile

iStuff Mobile [Ballagas *et al.*, 2007] est un outil pour prototyper rapidement des nouvelles interactions sur téléphones mobiles. iStuff est basé sur une interface graphique qui permet d'assembler des composants pour spécifier l'interaction. Les interactions qui peuvent être implémentées avec iStuff sont limitées par les dispositifs d'interaction offerts par le téléphone mobile : accéléromètre, caméra et clavier. Afin de préciser une interaction, iStuff utilise le paradigme des dispositifs en cascade [Dragicevic, 2004].

iStuff est implémenté comme une extension de l'application de Apple, Quartz Composer [Quartz Composer, 2010]. L'outil reprend ainsi les trois types de composants définis dans Quartz : des composants *Provider*, implémentant les dispositifs en entrée, des composants *Processor*, implémentant des opérations de transformation, et des composants *Consumer*, implémentant l'interaction en sortie (affichage sur un écran par exemple). iStuff ajoute à ces trois types de composants un quatrième type, iStuff, afin de communiquer avec les dispositifs mobiles.

La Figure 6.26 illustre un exemple d'assemblage implémentant Sweep, une interaction de pointage de grandes surfaces d'affichage avec un dispositif mobile [Ballagas *et al.*, 2005] basée sur l'utilisation de l'accéléromètre et de la caméra (partie supérieure de la Figure 6.26). La caméra permet de détecter le mouvement et l'accéléromètre est utilisé de façon redondante afin d'améliorer la reconnaissance. Les deux composants à gauche de l'assemblage (partie inférieure de la Figure 6.26) représentent l'accéléromètre (composant SmartItsSensor) et la caméra (composant SweepController). Un composant de fusion (composant Sensor Fusion - JavaScript) permet de générer les donnés de pointage *(x, y)* à partir des données de l'accélèromètre *(Tilt X, Tilt Y, Tilt Z)* et de celles de la caméra *(Motion X, Motion Y)*.

iStuff est donc un outil dédié aux dispositifs mobiles. L'outil présente une bonne modularité et l'expressivité du rôle des composants est soulignée par les couleurs qui différencient les types de composants. Le Tableau 6.18 reprend les principales caractéristiques de iStuff.

Icon

Le Input Configurator Toolkit (Icon) [Dragicevic et Fekete, 2004] est un outil pour l'implémentation d'interfaces à entrées configurables. L'outil repose explicitement

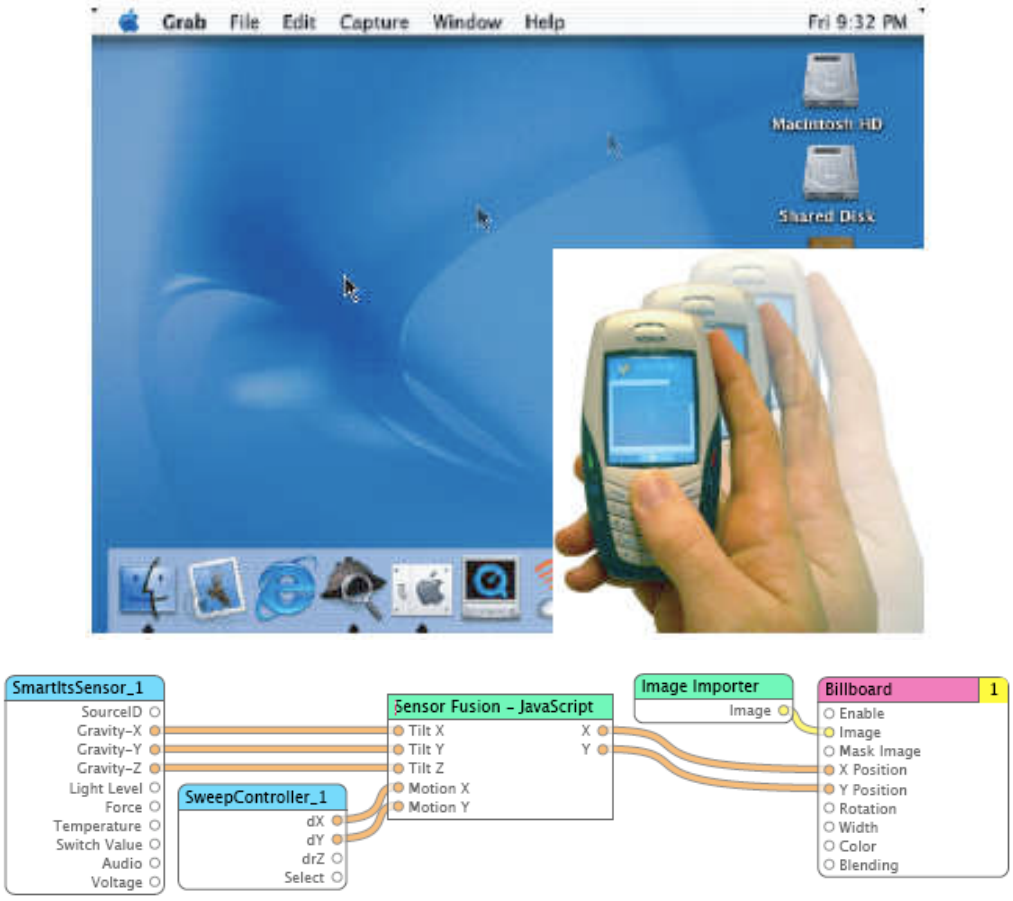


FIGURE 6.26 – Technique d'interaction Sweep basée sur le pointage d'une surface d'affichage avec un dispositif mobile (en haut) et assemblage implémentant l'interaction multimodale avec accéléromètre et caméra (en bas). Illustrations extraites respectivement de [Ballagas *et al.*, 2005] et de [Ballagas *et al.*, 2007].

ID	Outil de conception de prototypes interactifs sur téléphones mobiles
Multimodalité	Clavier, gestes avec accéléromètre, caméra. Programmation visuelle à flot de données, paradigme des dispositifs en cascade.
Notation	Forte expressivité du rôle
Outil	Seuil bas, plafond bas

TABLE 6.18 – Fiche-résumé de iStuff.

sur le modèle des configurations d'entrée [Dragicevic, 2004] (chapitre 3). Icon est implémenté en Java. Les dispositifs intégrés dans l'outil incluent la souris, le clavier, la tablette graphique et des commandes vocales (Figure 6.27).

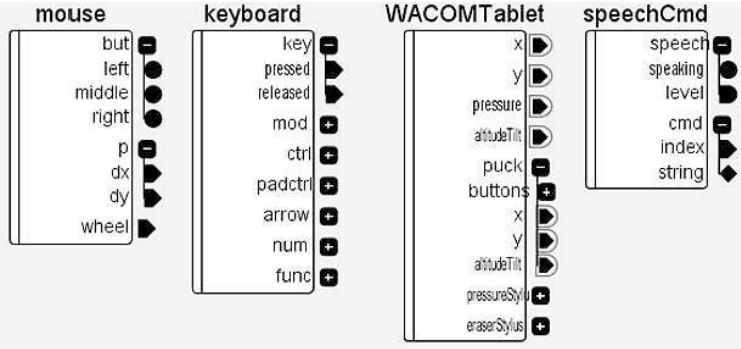


FIGURE 6.27 – Dispositifs implémentés sous Icon. Illustration extraite de [Dragicevic, 2004].

Nous avons analysé en détail le modèle à la base de cet outil dans le chapitre 3 ainsi que ses implications pour la conception dans le chapitre 4. Le Tableau 6.19 reprend les principales caractéristiques de Icon.

ID	Outil de développement d'interfaces multimodales en entrée
Multimodalité	Plusieurs modalités en entrée Programmation visuelle à flot de données, modèle des configurations d'entrée
Notation	Forte viscosité (section 4.1 du chapitre 4)
Outil	Seuil bas, plafond haut

TABLE 6.19 – Fiche-résumé de Icon.

Icare

La plate-forme Interaction-CARE (Icare) [Bouchet *et al.*, 2004] permet de composer des assemblages de composants afin de définir une interaction multimodale en entrée. De cet assemblage de composants, le code de l'interface multimodale est généré. Icare est basé sur le modèle Icare (section 3.4.3 du chapitre 3).

Les composants sont implémentés avec la technologie à composants JavaBeans de Sun Mycrosystems. La communication entre les composants est implémentée par génération d'événements et appels de méthodes. L'assemblage de composants peut se réaliser dans un éditeur graphique (Figure 6.28).

Nous avons largement analysé le modèle Icare ainsi que ses propriétés vis à vis de la conception d'interfaces multimodales au chapitre 4. Le Tableau 6.20 reprend les principales caractéristiques de Icare.

Squidy

Squidy [König *et al.*, 2010] est une librairie d'interaction et un outil graphique pour implémenter des interfaces multimodales. L'objectif de Squidy est de cacher la

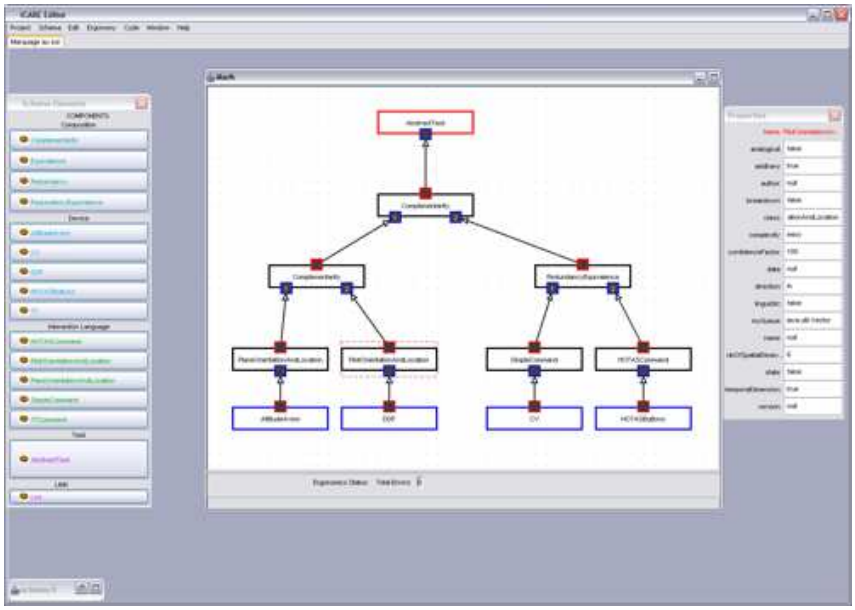


FIGURE 6.28 – Éditeur graphique de Icare.

ID	Conception d'interfaces multimodales en entrée
Multimodalité	Plusieurs modalités en entrée Programmation visuelle à flot de données, modèle ICARE
Notation	Forte visibilité
Outil	Seuil bas, plafond haut

TABLE 6.20 – Fiche-résumé de Icare.

complexité technique de l'implémentation de ce type d'interaction en offrant un langage visuel simple (Figure 6.29). Squidy implémente également un zoom sémantique, au sein de son éditeur graphique, qui permet de passer d'une vue haut-niveau à un vue bas-niveau (code) des composants qui sont assemblés. L'outil est implémenté en Java.

Les composants sont divisés en trois catégories : dispositifs en entrée, filtres et dispositifs en sortie. Le modèle de Squidy est ainsi proche du paradigme des dispositifs en cascade [Dragicevic, 2004]. La Figure 6.29 illustre un exemple d'assemblage implémentant une interaction de pointage avec un laser. Les données du composant Laser (à gauche) sont traitées par un composant de filtrage puis utilisées pour émuler la souris (à droite).

Le principal apport de Squidy par rapport aux outils précédents est l'utilisation du zoom sémantique, qui augmente la visibilité de l'outil. Le Tableau 6.21 reprend les principales caractéristiques de Squidy.

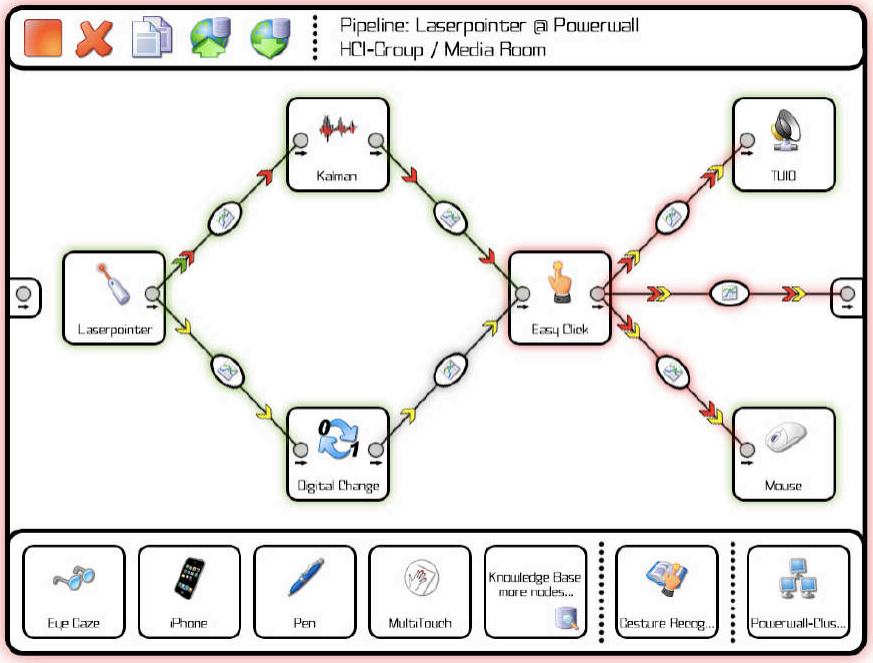


FIGURE 6.29 – Exemple d'assemblage graphique dans Squidy. Illustration extraite de [König *et al.*, 2010].

ID	Outil de conception d'interfaces multimodales
Multimodalité	Plusieurs modalités en entrée : pointeur laser, table tactile Microsoft Surface, iPhone, Phidgets, wii-mote et reconnaissance vocale Programmation visuelle à flot de données, paradigme des dispositifs en cascade
Notation	Forte expressivité du rôle, forte visibilité
Outil	Seuil bas, Plafond haut

TABLE 6.21 – Fiche-résumé de Squidy.

6.5 Conclusion

Dans ce chapitre nous avons analysé les principaux outils de conception de prototypes multimodaux simulés ou interactifs. Nous avons analysé les outils selon un ensemble de critères à différents niveaux d'abstraction. À haut-niveau, nous avons analysé les aspects propres à la multimodalité et au modèle conceptuel sous-jacent. Au niveau intermédiaire, nous avons analysé la notation de spécification sur laquelle repose l'outil en utilisant les dimensions cognitives de Green : viscosité, engagement forcé et expressivité du rôle. À bas-niveau, nous avons considéré des critères généraux liés à l'outil et à ce qu'il est possible de faire avec l'outil, en utilisant les critères de Myers : seuil (la difficulté de prise en main) et plafond (les possibilités de l'outil). Le Tableau 6.22 regroupe les fiches-résumé des outils Magicien d'Oz étudiés selon notre grille d'analyse. Les tableaux suivants regroupent les fiches-résumé des outils pour la création de prototypes interactifs étudiés selon

notre grille d'analyse : outils à programmation non-visuelle (Tableau 6.23), outils à programmation visuelle à automates et par démonstration (Tableau 6.24) et outils à programmation visuelle à flot de données (Tableau 6.25).

Parmi les outils de développement de prototypes simulés, aucun outil ne permet de généraliser la simulation à un ensemble de modalités : tous les outils ont un plafond bas en terme de multimodalité.

Parmi les outils de développement de prototypes interactifs, seuls quelques outils s'appuient sur des modèles conceptuels, ce qui a comme principale conséquence de diminuer les possibilités de génération de l'outil pour l'interaction multimodale (plafond bas). Parmis les outils basés sur un modèle conceptuel, trois sont basés sur le paradigme des dispositifs en cascade (iStuff, Icon et Squidy) et un sur le modèle Icare (Icare). Au chapitre 4, nous avons déjà analysé les avantages et limitations des deux modèles conceptuels, qui nous ont conduits à définir un nouveau modèle mixte.

Enfin, aucun outil ne permet de réaliser à la fois des prototypes simulés et des prototypes interactifs.

À partir de cette analyse des outils existants, nous visons une solution qui permette en même temps de réaliser des prototypes simulés et interactifs. De plus un outil basé sur notre modèle mixte pour spécifier l'interaction permettrait d'obtenir un seuil bas et un plafond haut, afin d'augmenter le pouvoir génératif et descriptif de l'outil. Nous présentons au chapitre suivant cet outil de développement de prototypes simulés et interactifs, basé sur notre modèle conceptuel mixte présenté au chapitre 4.

ID SketchWizard	Simulation des gestes réalisés avec un stylet
Multimodalité	Une seule modalité : Geste avec un stylet
Notation	Visibilité ++, expressivité du rôle ++
Outil	Seuil - -, plafond - -
ID Suede	Simulation d'interfaces vocales
Multimodalité	Une seule modalité : parole
Notation	Visibilité ++, expressivité du rôle ++
Outil	Seuil - -, plafond - -
ID Topiary	Simulation de la géolocalisation
Multimodalité	Géolocalisation de l'utilisateur et stylet sur PDA
Notation	Visibilité ++
Outil	Seuil - -, plafond - -

TABLE 6.22 – Synthèse : Fiches-résumé des outils Magicien d'Oz.

ID Processing	Création d'interfaces graphiques interactives
Multimodalité	Caméra, capteurs physiques, etc.
Notation	Visibilité - -
Outil	Seuil ++, plafond ++
ID OF	Applications graphiques interactives
Multimodalité	Caméra, capteurs physiques, etc.
Notation	Visibilité - -
Outil	Seuil ++, plafond++
ID JavaSwingMM	Boîte à outils Java
Multimodalité	Phidgets, manettes et reconnaissance vocale
Notation	Inter-opérabilité avec JavaSwing
Outil	Seuil ++, Plafond ++
ID HephaisTK	Prototypage multimodal
Multimodalité	Reconnaissance vocale, table multi-utilisateurs et périphériques matériels.
Notation	Viscosité ++
Outil	Seuil ++

TABLE 6.23 – Synthèse : Fiches-résumé des outils de prototypage à programmation non-visuelle.

ID IMBuilder	Conception d'interfaces multimodales en entrée
Multimodalité	Gestes avec la souris et parole Modèle à automates
Notation	Visibilité ++
Outil	Seuil - -, plafond - -
ID SCS	Conception d'interfaces multimodales en entrée
Multimodalité	Gestes avec la souris, parole et capteurs physiques Modèle à automates, Equivalence (CARE)
Notation	Visibilité ++
Outil	Seuil - -, plafond - -
ID PetShop	Développement d'interfaces multimodales
Multimodalité	Souris, clavier et gant - capteur Réseaux de Pétri
Notation	Visibilité ++, expressivité du rôle ++
Outil	Seuil ++, plafond ++
ID CrossWeaver	Conception de prototypes basse-fidélité
Multimodalité	Clique souris, clavier, geste et parole Modèle à automates
Notation	Forte visibilité, forte expressivité du rôle
Outil	Seuil - -, plafond - -
ID d.tools	Prototypage de systèmes utilisant des capteurs
Multimodalité	Accéléromètre, capteurs de pression et sliders Modèle à automates
Notation	Expressivité du rôle ++
Outil	Seuil - -, Plafond - -
ID Exemplar	Prototypage de systèmes utilisant des capteurs
Multimodalité	Accéléromètre, capteurs de pression, temp., etc. Programmation par démonstration
Notation	Faible visibilité
Outil	Seuil bas, plafond bas

TABLE 6.24 – Synthèse : Fiches-résumé des outils de prototypage à programmation visuelle à automates et par démonstration.

ID vvvv	Traitement temps-réel d'images vidéo
Multimodalité	Capteurs physiques, tablette tactile.
Notation	Visibilité ++, expressivité du rôle - -
Outil	Seuil - -, plafond - -
ID Max/Msp	Créations musicales et multimédia temps réel
Multimodalité	Caméra, capteurs physiques, etc.
Notation	Viscosité ++, visibilité - -, engagement forcé - -
Outil	Seuil ++, plafond ++
ID EyesWeb	Création d'applications multi-média, musicales et de danse temps-réel
Multimodalité	Analyse de gestes en temps-réel Paradigme des dispositifs en cascade
Notation	Expressivité du rôle ++, visibilité ++
Outil	Seuil - -, plafond - -
ID iStuff	Conception de prototypes interactifs sur téléphones mobiles
Multimodalité	Clavier, gestes avec accéléromètre, caméra. Paradigme des dispositifs en cascade.
Notation	Expressivité du rôle ++
Outil	Seuil - -, plafond - -
ID Icon	Développement d'interfaces multimodales
Multimodalité	Plusieurs modalités en entrée Modèle des configurations d'entrée
Notation	Viscosité ++
Outil	Seuil - -, plafond ++
ID Icare	Conception d'interfaces multimodales en entrée
Multimodalité	Plusieurs modalités en entrée Modèle ICARE
Notation	Visibilité ++
Outil	Seuil - -, plafond ++
ID Squidy	Conception d'interfaces multimodales
Multimodalité	Pointeur laser, table tactile, iPhone, Phidgets, wii-mote et reconnaissance vocale Paradigme des dispositifs en cascade
Notation	Expressivité du rôle ++, visibilité ++
Outil	Seuil - -, Plafond ++

TABLE 6.25 – Synthèse : Fiches-résumé des outils de prototypage à programmation visuelle à flot de données.

Chapitre 7

Notre outil pour le prototypage multimodal : OpenInterface Interaction Development Environment

«Un outil humain est [...] un objet façonné, transformé, de manière à pouvoir être utilisé commodément et efficacement pour accomplir un certain genre d'action»

Gaston Viaud, Naturaliste et psychologue français (1899-1961)

Contenu du chapitre

Dans ce chapitre, nous présentons l'outil logiciel de prototypage multimodal que nous avons développé. Vis-à-vis de l'état de l'art des outils existants du chapitre précédent, notre outil vise à :

- permettre le développement de prototypes simulés et de prototypes interactifs : nous avons souligné l'absence d'outil qui permette le développement de ces deux types de prototypes qui s'inscrivent pourtant de façon continue dans une démarche itérative de conception centrée utilisateur
- offrir un fort pouvoir descriptif et génératif en alliant les atouts des deux outils Icon et Icare. Pour atteindre cet objectif, notre outil repose explicitement sur notre modèle de conception présenté au chapitre 4 et évalué au chapitre 5. Adopter cette notation au sein de notre outil permet un fort pouvoir descriptif et génératif.

L'outil développé, appelé OIDE (OpenInterface Interaction Development Environment), est un éditeur graphique construit au dessus d'une plate-forme à composants appelée OpenInterface. Le développement de cet outil s'inscrit dans le projet de recherche européen OpenInterface [OpenInterface, 2006]. Aussi, nous commençons par introduire le contexte du développement de cet outil, en présentant le projet européen, puis nous montrons les principales caractéristiques de la plate-forme à composants sous-jacente. Nous détaillons l'architecture de notre outil graphique de conception par rapport à la plate-forme sous-jacente.

Ensuite, nous présentons la concrétisation de notre modèle conceptuel au sein du OIDE à travers la caractérisation des composants et les différents types de composants logiciels implémentés : dispositif, transformation, composition et tâche. Enfin nous présentons les composants implémentés pour le développement de prototypes interactifs et simulés.

7.1 Plate-forme logicielle sous-jacente

7.1.1 Contexte : le projet OpenInterface

Le développement de notre outil s'inscrit dans le cadre d'un projet, OpenInterface [OpenInterface, 2006]. OpenInterface (OI) est un projet européen STREP du framework 6 IST. Le projet OpenInterface est une continuation du réseau d'excellence européen SIMILAR du framework 6 IST. Dans ce réseau, a été développé la plateforme à composants OpenInterface, que nous utilisons comme noyau de notre outil graphique. Partant de ce noyau à composants issu du réseau SIMILAR, nous avons développé notre outil de prototypage dans le cadre du projet OpenInterface.

Le projet OpenInterface avait pour objectif la définition d'un outil logiciel pour le prototypage d'interfaces multimodales, qui s'inscrit dans une démarche de conception itérative centrée utilisateur. Le projet OI implique neuf partenaires académiques et industriels : Université Joseph Fourier Grenoble (France), Université de Glasgow (RU), Fraunhofer Institute for Applied Information technology (Allemagne) et Université Catholique de Louvain (Belgique) comme partenaires académiques ; France Télécom (France), Immersion (France), PhonoClick (Turquie), Arca-

diaDesign (Italie), Multitel ASBL (Belgique) et TXT e-Solutions Spa (Italie) comme partenaires industriels.

Dans ce projet, notre travail a consisté à concevoir et développer l'outil logiciel ainsi qu'à développer et intégrer des composants logiciels pour permettre le prototypage d'interfaces multimodales. Adoptant une démarche opensource, l'outil et les composants présentés dans ce chapitre sont disponibles sur le forge openinterface (https ://forge.openinterface.org/). Son utilisation va donc au-delà des membres du laboratoire et en particulier notre outil a été installé et largement utilisé par les neufs autres partenaires du projet.

L'outil repose explicitement sur notre modèle de conception présenté au chapitre 4. Comme nous l'avons souligné, notre modèle est conceptuel et est indépendant de la technologie à composants utilisée (chapitre 4). Dans le projet OI, nous avons concrétisé notre modèle conceptuel par le développement d'un outil logiciel au dessus de la plate-forme à composants notée OpenInterface. Celle-ci a été développée par l'Université Catholique de Louvain, coordinateur du réseau SIMILAR et partenaire du projet OpenInterface.

7.1.2 Plate-forme à composants OpenInterface

La plate-forme OpenInterface [Lawson *et al.*, 2009] est un intergiciel à composants qui gère l'instanciation et l'exécution de composants logiciels hétérogènes. Par défaut, la plate-forme accepte des composants écrits en Java, c++ et Matlab. La plate-forme est extensible : les utilisateurs peuvent rajouter des nouveaux langages en créant des plugins. Ainsi, des extensions pour Python et pour .Net ont été intégrées.

Par rapport aux technologies à composants existantes (section 3.3.3 chapitre 3), la plate-forme OpenInterface a été conçue pour être une plate-forme facile à installer, peu consommatrice de ressources et pour être flexible. La plate-forme permet l'intégration de composants déjà existants sans imposer un modèle de composants contraignant. Cette approche favorise donc l'intégration de composants hétérogènes implémentant différentes modalités d'interaction et l'exploration de l'espace de conception de la multimodalité.

Description des composants OpenInterface

Les composants dans OpenInterface sont considérés comme des *unités logicielles indépendantes et réutilisables avec des interfaces en entrée/sortie*[1]. Concrètement, les composants sont définis par une API qui explicite les fonctionalités du composant, ainsi que le constructeur du composant. Les fonctionalités d'un composant sont décrites par des facettes (Facets), qui contiennent des Pins, les ports en entrée (Sink) et en sortie (Source et Callback) du composant (Figure 7.1). Cette description est contenue dans un fichier xml selon un langage spécifique, le Component Interface Description Language (CIDL).

L'intégration d'un composant passe par la création de la description CIDL du composant (donnée en Annexe D). Le CIDL informe entre autres du langage du

[1] «a reusable and independent software unit with exported and imported Input/Output interfaces» [Lawson *et al.*, 2009]

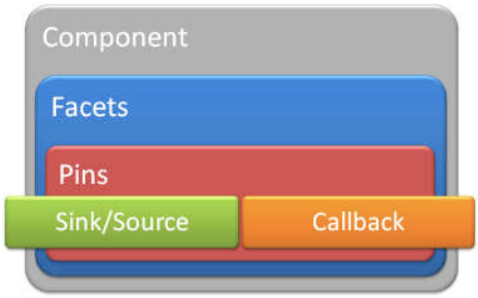


FIGURE 7.1 – Description d'un composant dans OpenInterface. Illustration extraite de [Lawson *et al.*, 2009].

composant (java, c++) et du format du fichier du composant (jar, dll). La plate-forme exploite les descriptions selon le CIDL pour générer des proxies en C++ pour chaque composant. Pour rappel, un proxy est un serveur qui relaye des requêtes entre un client et un serveur. Au sein de la plate-forme OpenInterface, les proxies servent à faire communiquer tous les composants hétérogènes entre eux à travers le coeur de la plate-forme (le kernel), comme illustré à la Figure 7.2.

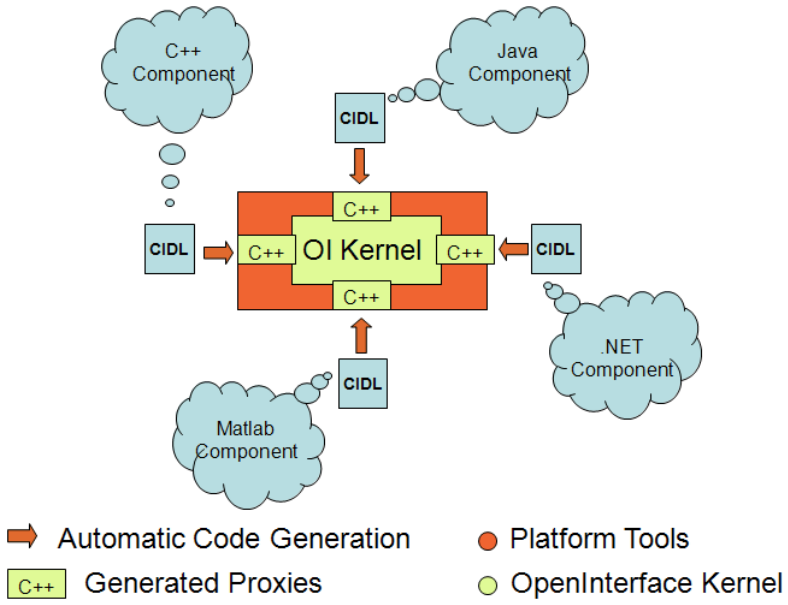


FIGURE 7.2 – Intégration de composants hétérogènes dans OpenInterface. Illustration extraite de [Lawson *et al.*, 2009].

Description des assemblages OpenInterface

La description d'un assemblage dans un fichier xml adhère au langage noté, Pipeline Description and Configuration Language (PDCL) (donné en Annexe E). Les assemblages de composants, nommés *Pipelines* dans OpenInterface, permettent de définir l'interconnexion et la configuration des composants. Une description de Pipeline contient donc la liste des composants utilisés et les connexions réalisées.

La description de l'assemblage permet également de définir les aspects de distribution à l'exécution du composant. Par exemple, un composant peut être exécuté de localement ou à distance (sur une autre machine).

Compilation et exécution

Une fois les composants et l'assemblage décrits par les fichiers CIDL et PDCL respectifs, la plate-forme gère la compilation et l'exécution de l'assemblage. La compilation permet de générer les proxies des composants utilisés. L'exécution va instancier chaque composant et gérer les connexions.

7.2 OIDE, notre environnement de prototypage

Dans cette section, nous présentons notre environnement OIDE (OpenInterface Interaction Development Environment) développé au dessus de la plate-forme à composants de la section précédente. Pour cela, nous décrivons d'abord son architecture et son modèle logiciel. Nous présentons ensuite son interface graphique et l'intégration de composants. Nous finissons en soulignant les deux types d'utilisation de notre outil.

7.2.1 Architecture et modèle logiciel

Notre environnement graphique noté OIDE (OpenInterface Interaction Development Environment) est construit au dessus de la plate-forme à composants OpenInterface. Nous avons appelé l'ensemble de cette architecture la *plate-forme OpenInterface* (OI framework) et nous utilisons le terme *noyau OpenInterface* (kernel OI) pour désigner la plate-forme à composants sous-jacente, présentée à la section précédente (Figure 7.3).

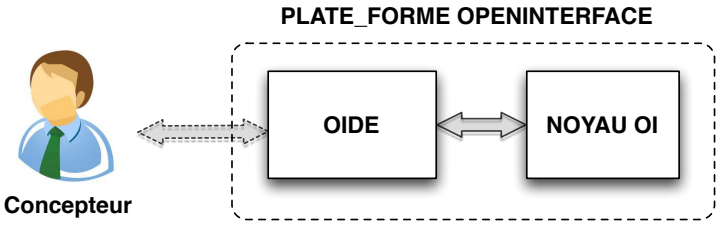


FIGURE 7.3 – Architecture globale de la plate-forme OpenInterface : environnement graphique de prototypage (OIDE) + plate-forme à composants sous-jacente (noyau OI).

L'éditeur a été implémenté en Java afin d'assurer le même niveau de portabilité que le noyau. Nous avons implémenté l'architecture Modèle-Vue-Contrôleur (MVC) (section 3.3.4 du chapitre 3), de la Figure 7.4, afin de séparer l'interface graphique de la description fonctionnelle des composants. Le contrôleur (C) gère la synchronisation entre la vue et le modèle. La vue (V) englobe l'interface graphique de l'éditeur, les différents menus et les aspects graphiques du graphe de l'assemblage de composants. Le modèle (M) inclut notre modèle conceptuel de la multimodalité et le modèle à composants du noyau (Figure 7.4). En effet, afin de clairement

dissocier l'OIDE du noyau OI, le modèle logiciel de l'éditeur considère deux parties : le modèle de la multimodalité et le modèle technique du noyau. Le modèle de la multimodalité implémente notre modèle conceptuel (chapitre 4). Ce modèle est décrit dans un fichier xml nommé MCDL (Multimodal Component Description Language) que nous présentons à la section 7.3.1. Le modèle technique implémente la description technique (CIDL, PDCL) des composants et des assemblages OI.

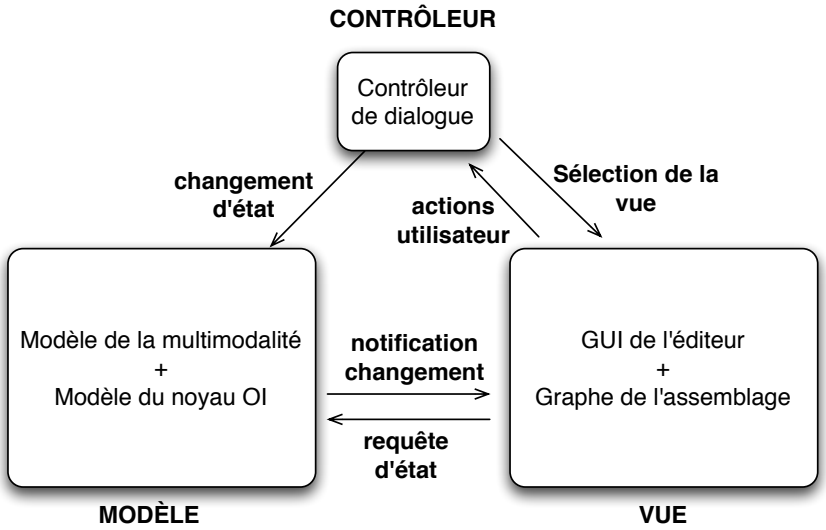


FIGURE 7.4 – Architecture MVC de l'éditeur graphique.

Ainsi, la partie Modèle de notre outil manipule avant tout des composants de type multimodal. Un composant multimodal est composé d'une description selon notre modèle conceptuel dédié à la multimodalité et d'une description technique. La description technique dépend de l'approche à composants sous-jacente utilisée. Dans notre cas, la description technique implémente le modèle logiciel du noyau OI (CIDL + PDCL). La Figure 7.5 schématise le modèle logiciel du OIDE.

7.2.2 Composants logiciels, éléments de base du OIDE

Les composants logiciels décrits selon le modèle introduit ci-dessus constituent les unités élémentaires manipulées par l'OIDE. Dans un contexte de conception rapide de prototypes multimodaux, notre objectif est de définir un processus efficace et flexible pour ajouter des composants logiciels au sein de notre outil de prototypage.

Pour viser une intégration efficace d'un nouveau composant, nous définissons un processus avec un nombre réduit d'étapes (entre une et trois étapes). De plus le processus d'intégration est flexible et certaines étapes sont optionnelles. Ainsi l'OIDE permet la manipulation de composants partiellement décrits.

Le processus complet d'intégration commence avec le code source d'un composant. La Figure 7.6 illustre ce processus avec un composant c++ comme exemple. Il faut ensuite générer la librairie correspondante (dll en c++, jar en java, etc.). Notons qu'il est possible de commencer directement à partir d'une libraire. Puis l'utilisateur doit décrire le composant et son interface dans le langage CIDL afin de le rendre opérationnel au sein du noyau OI. À ce point, le composant est prêt à être

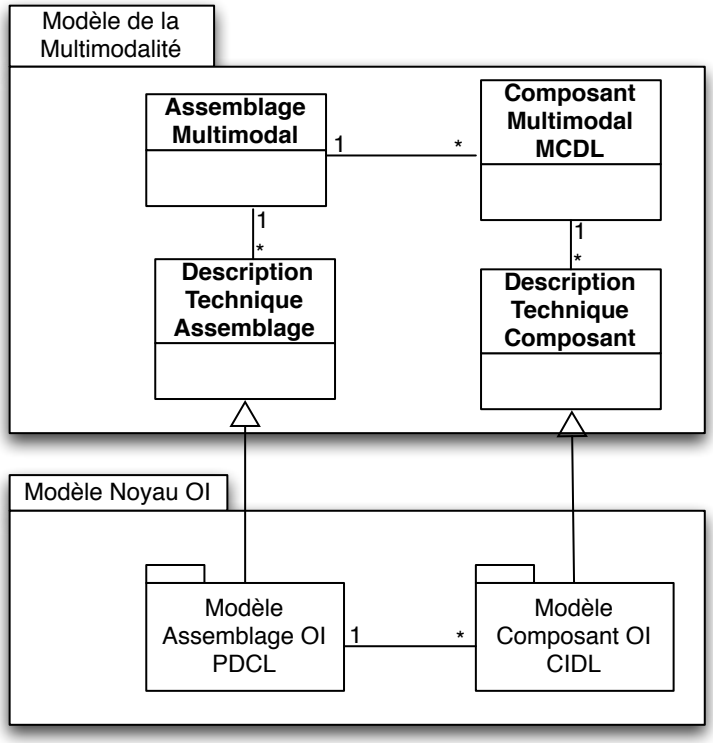


FIGURE 7.5 – Modèle logiciel du OIDE composé d'un modèle multimodal et d'un modèle technique.

utilisé dans l'OIDE, même si la description selon notre modèle de la multimodalité (MCDL) n'est pas réalisée. Cela permet d'intégrer et de tester rapidement des composants techniques.

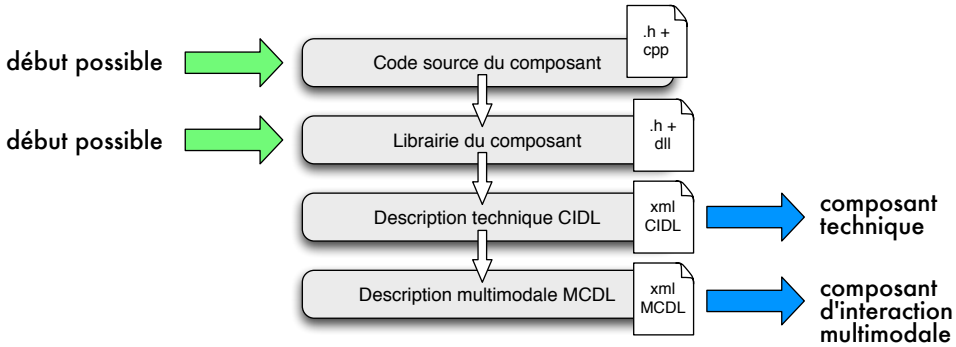


FIGURE 7.6 – Étapes de l'intégration d'un nouveau composant (c++ dans l'exemple) dans la plate-forme. À gauche les départs possibles et à droite les points d'intégration.

Lorsque l'utilisateur veut décrire complètement un composant outre la description CIDL, elle/il doit décrire le composant selon le langage MCDL, que nous présentons en détail à la section 7.3.1. Le composant ainsi décrit peut être déposé en ligne et exploité au sein du OIDE. Nous notons (1) les composants partiellement décrits (uniquement la description CIDL), des composants techniques et (2) les com-

posants complètement décrits (description CIDL et MCDL) des composants d'interaction multimodale. Afin de faciliter la création des descriptions, il est possible d'éditer facilement une description MCDL d'un composant (formulaire).

Afin de gérer les composants ainsi décrits et intégrés à la plate-forme, nous avons établi deux dépôts de composants[2] : un dépôt local et un dépôt en ligne.

Le dépôt local de composants est un stockage localisé sur la même machine que l'installation du OIDE. Ainsi, chaque installation du OIDE possède son propre dépôt local. Le dépôt local est composé de deux stockages distinguant les composants partiellement décrits (composants techniques décrits selon le CIDL), de ceux décrits complétement (composants d'interaction multimodale décrits selon le CIDL et le MCDL). L'OIDE gère donc ces deux types de composants (Figure 7.7). Les composants du dépôt local comprennent les composants installés par défaut avec l'OIDE, des composants que l'utilisateur a développé et intégré à la plate-forme ainsi que des composants que l'utilisateur a téléchargé du dépôt en ligne.

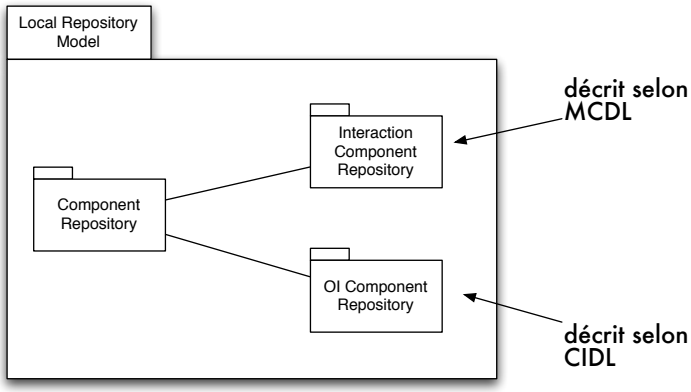


FIGURE 7.7 – Dépôt local de composants divisé en deux : un stockage de composants d'interaction multimodale et un stockage de composants techniques.

L'Université de Glasgow (partenaire du projet OpenInterface) a développé un dépôt de composants en ligne, le *OpenInterface repository*, qui permet d'ajouter ou de télécharger des composants. Le dépôt en ligne exploite la description des composants MCDL reposant sur notre modèle conceptuel (section 7.3.1), ce qui permet entre autre de chercher des composants selon des caractéristiques de notre modèle, comme le type de composant (dispositif, transformation, composition ou tâche) (Figure 7.8).

Le dépôt en ligne est intégré à notre outil : la commande de recherche de composants en ligne du OIDE permet de télécharger et installer automatiquement un composant (qu'il soit technique ou d'interaction multimodale) dans le dépôt local.

7.2.3 Editeur graphique (Vue)

L'interface graphique de l'outil (partie Vue de la Figure 7.4) a été implémentée avec Java Swing, la bibliothèque graphique de Java, afin de bénéficier de sa portabilité sur différents systèmes d'exploitation. L'interface graphique est composée de 4 espaces principaux : la librairie de composants techniques (zone A à la Figure 7.9),

[2]component repository

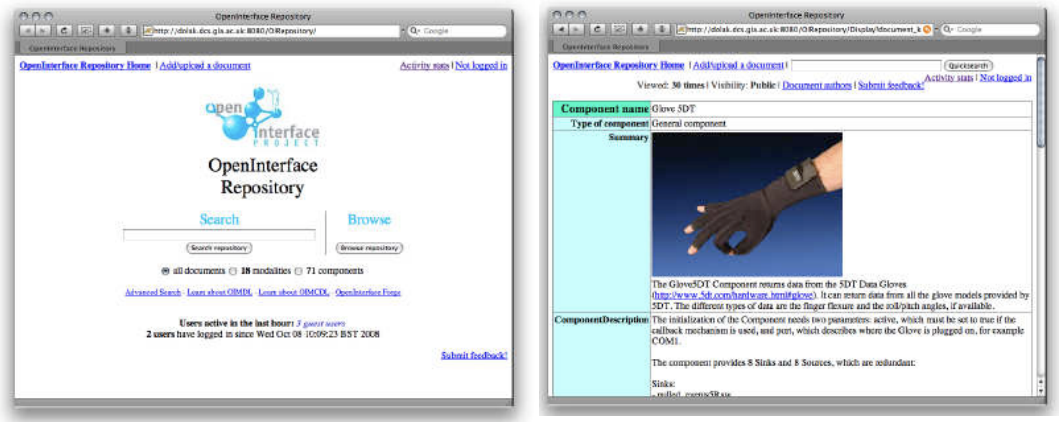


FIGURE 7.8 – Interface web du *OpenInterface repository*.

la librairie de composants d'interaction multimodale (zone B à la Figure 7.9), la librairie d'outils (zone C à la Figure 7.9) et la zone de conception (zone D à la Figure 7.9).

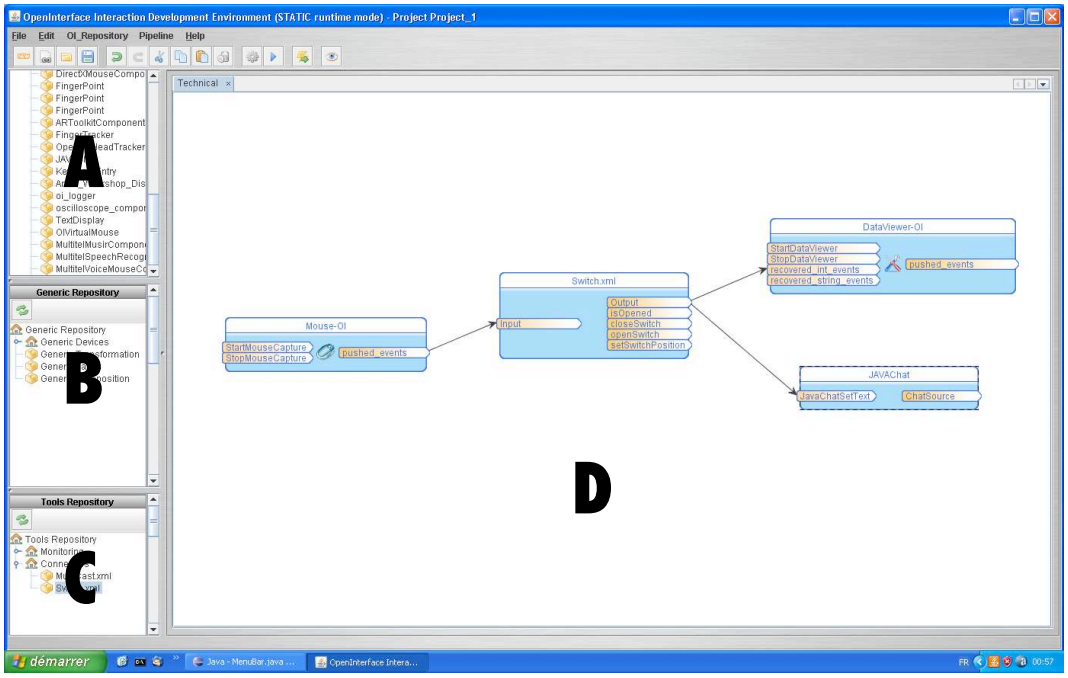


FIGURE 7.9 – Interface graphique du OIDE.

La libraire de composants (A) contient tous les composants techniques disponibles dans le dépôt local. La libraire de composants (B) contient tous les composants d'interaction multimodale disponibles dans le dépôt local. Ainsi, notre outil admet l'utilisation mixte de composants caractérisés et non-caractérisés, dans le but de doter notre outil d'une plus grande flexibilité et d'inter-opérabilité. Comme nous l'avons expliqué, cela permet aussi de rendre plus rapide l'ajout d'un composant par exemple pour le tester au sein d'une interaction multimodale. La librairie

d'outils (C) contient tous les composants utilitaires disponibles dans le dépôt local, comme le composant DataViewer, qui permet de voir textuellement des données numériques ou textuelles. Nous présentons à la section 7.4 les composants utilitaires développés.

Au sein de la barre d'outils de l'interface se trouvent les commandes essentielles de l'OIDE. Quatre boutons rassemblent les principales actions utilisées par le concepteur pendant la création d'une interaction multimodale (Figure 7.10) :

- **Compilation** : permet de compiler l'assemblage afin de générer les proxies des composants utilisés. Cette opération doit seulement se faire si on utilise un composant pour la première fois.
- **Exécution** : permet d'exécuter l'assemblage conçu.
- **Connexion au dépôt de composants distants** : permet de se connecter à la base de composants en ligne, distante pour télécharger des composants.
- **Changement de mode d'exécution** : permet de passer du mode d'exécution statique au mode dynamique. Nous expliquons ces différents modes d'exécution dans la section suivante.

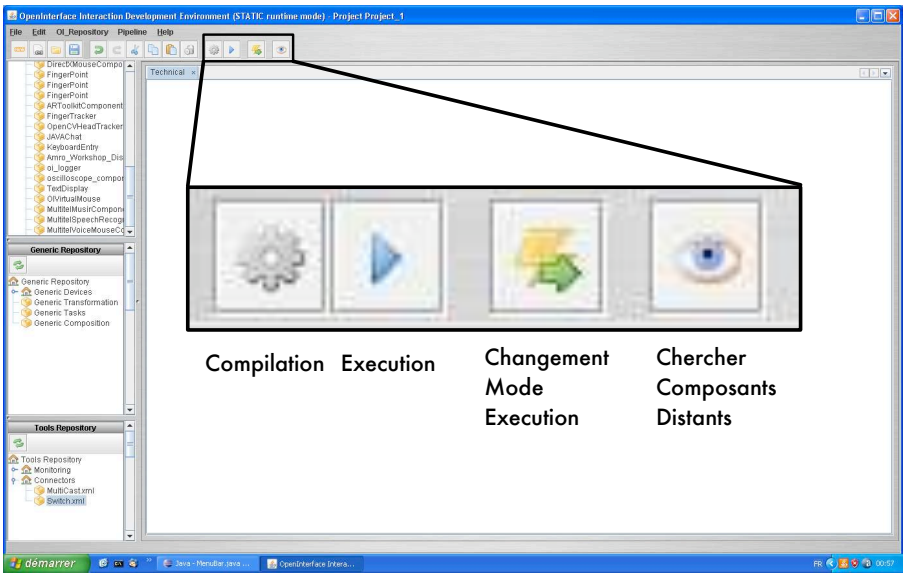


FIGURE 7.10 – Commandes principales du OIDE.

7.2.4 Utilisation du OIDE et dynamicité du prototype développé

Pour développer un prototype multimodal, l'utilisateur assemble des composants dans la zone de conception de l'éditeur (zone B de la Figure 7.9). Nous distinguons deux modes d'utilisation :

- **Mode statique de conception** : dans ce mode, les phases de conception (construction de l'assemblage) et l'exécution sont séquentielles.

- **Mode dynamique de conception** : dans ce mode, l'utilisateur peut modifier l'assemblage en cours d'exécution.

Ces deux modes sont accessibles dans la barre d'outils de l'OIDE (Figure 7.10).

Dans le mode statique de conception, l'utilisateur importe si nécessaire des composants et les assemble. Lorsque l'assemblage est terminé, elle/il démarre l'exécution pour tester l'interaction multimodale ainsi conçue. Pour modifier la conception, l'utilisateur doit arrêter l'exécution et revenir en mode édition de l'assemblage. Ce processus de conception itératif en deux étapes, conception puis exécution, est illustré à la Figure 7.11.

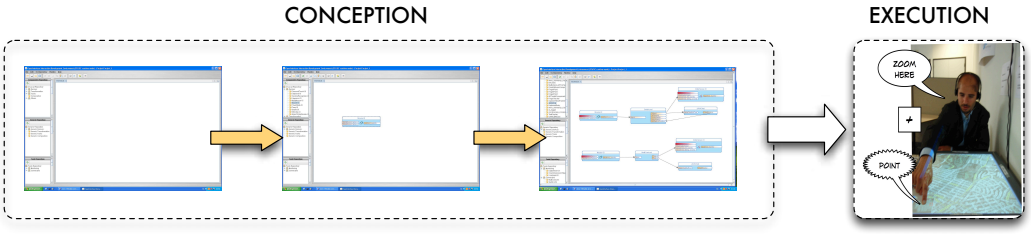


FIGURE 7.11 – Mode statique de conception.

Dans le mode dynamique de conception, les phases de conception et d'exécution sont fusionnés : l'utilisateur peut modifier sa conception pendant l'exécution. Pour cela, chaque composant possède un bouton de démarrage individuel. Ainsi, le concepteur peut décider à quel moment exécuter un certain composant. Cela peut servir par exemple pour ajouter des nouveaux dispositifs sans arrêter l'exécution de l'interaction multimodale.

La Figure 7.12 illustre ce processus de conception, où à chaque étape le concepteur ajoute des nouveaux composants et des nouveaux dispositifs qui sont utilisés par le sujet, utilisateur final de l'interaction multimodale. Ce mode de conception présente néanmoins une limitation technique : il n'est pas possible d'effacer dynamiquement un composant d'un assemblage. En effet, étant donné que les composants sont intégrés dans la plate-forme sans modifier leur code (il suffit de les décrire en CIDL), aucune méthode de destruction n'est à priori présente dans les composants. Cela empêche par exemple d'arrêter et remplacer un dispositif d'interaction pendant l'exécution.

Malgré cette limitation, ce mode de conception se prête bien à des expériences rapides où les choix de conception peuvent changer selon les réactions des sujets (utilisateurs finaux) ou selon le fonctionnement d'un dispositif, tout en assurant une continuité de fonctionnement.

7.3 Concrétisation de notre modèle conceptuel

Notre environnement OIDE repose sur notre modèle conceptuel présenté au chapitre 4. Son évaluation décrite au chapitre 5 a montré son fort pouvoir descriptif et génératif. Aussi adopter ce modèle comme notation de spécification au sein du OIDE permet d'offrir un outil logiciel à fort pouvoir descriptif et génératif, caractéristiques peu présentes dans les outils existants (chapitre 6).

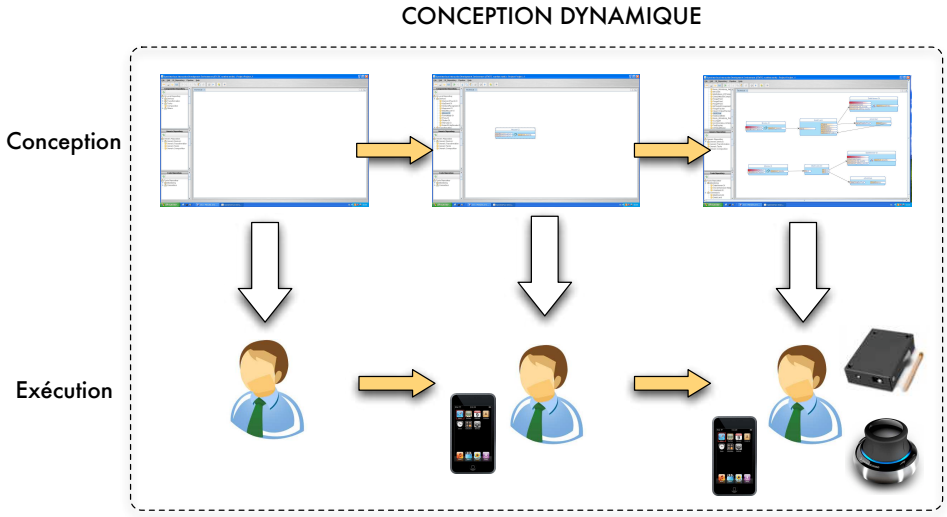


FIGURE 7.12 – Mode dynamique de conception.

Aussi au sein de l'éditeur de la Figure 7.9, nous adoptons une approche de programmation visuelle à flot de données par la construction de l'assemblage de composants. De plus, chaque composant, unité d'assemblage, est décrit selon notre modèle conceptuel à l'origine du MCDL (Multimodal Component Description Language). Enfin, l'assemblage adhère à la structuration issue de notre modèle avec des composants dispositifs, de transformation, de composition et de tâche.

Dans cette section nous détaillons le langage MCDL puis nous étudions les types de composants au sein d'un assemblage.

7.3.1 Caractérisation des composants : Multimodal Component Description Language (MCDL)

Afin de décrire les composants logiciels manipulés selon notre modèle conceptuel, nous avons défini un langage de description xml, le Multimodal Component Description Language (MCDL) (donné en Annexe C). Ce langage de description permet de caractériser le type de composant (Dispositif, Transformation, Composition ou Tâche), ainsi que les caractéristiques de son interface et de son comportement qui définissent son degré de réutilisabilité. Le MCDL est composé de trois sections principales, illustrées à la Figure 7.13.

La première section décrit de façon générale le composant selon notre approche : nom, type (dispositif, transformation, composition, tâche), dépendance du comportement et description textuelle du fonctionnement. Selon le type de composant, des informations sont ajoutées : pour les composants de type dispositif, nous décrivons les caractéristiques physiques du dispositif, comme le poids ou la taille, ainsi que les sens humains en jeu (basés sur la taxonomie de Buxton, présentée au chapitre 3) ; pour les composants de type transformation, nous décrivons la nature de la transformation ; pour les composants de type composition, nous décrivons la com-

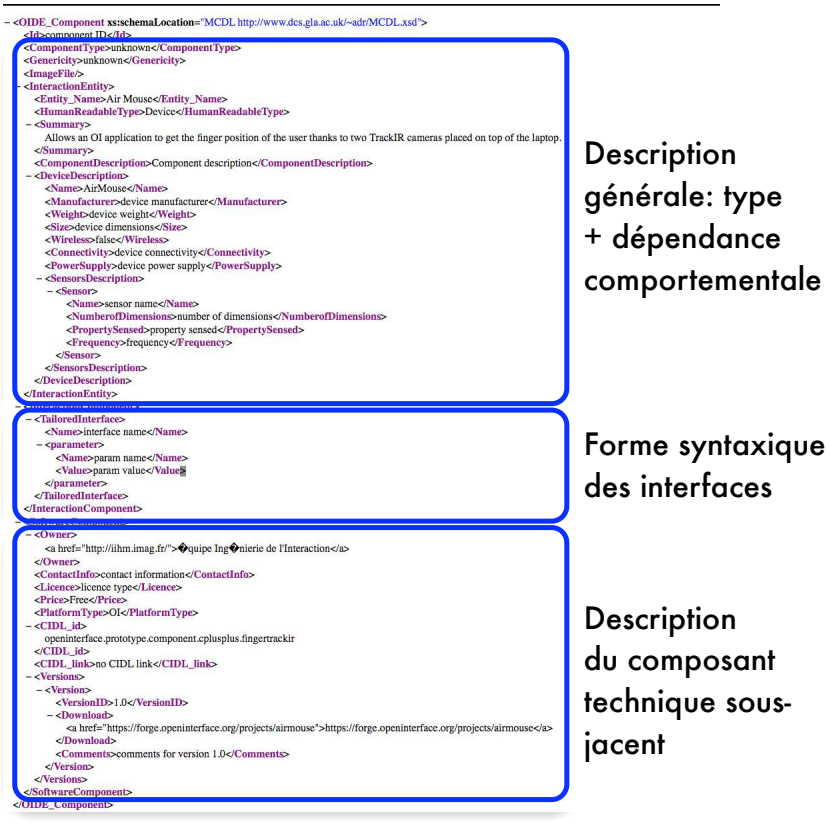


FIGURE 7.13 – Fichier de description du format MCDL d'un composant avec les principales sections de la description.

position CARE utilisée ; pour les composants tâche, nous décrivons la relation avec les tâches de Foley.

La Figure 7.14 illustre le code xml de la première section du composant Wiimote. Le code indique d'abord le nom, le type (dispositif) et la dépendance (indépendant). Le code indique aussi le fabricant, la taille, le poids, la nature sans fils du dispositif, la connectivité (bluetooth), l'alimentation (deux batteries AA) ainsi que les différents capteurs présents dans le dispositif : un accéleromètre, huit boutons et un capteur infra-rouge.

La deuxième section comporte une description de chaque interface du composant. Pour chaque interface nous détaillons la forme syntaxique (ad hoc ou normalisée). Si l'interface est normalisée, alors nous précisons le type de norme et les valeurs.

La Figure 7.15 illustre le code xml de la deuxième section du composant Wiimote. Par souci de clarté, nous montrons seulement quatre interfaces des dix formant la Wiimote. Trois de ces quatre interfaces sont ad hoc (*tailored* en anglais), alors que l'interface correspondante au bouton A est normalisée. Le code décrivant cette interface inclut le type de normalisation (Buxton), la propriété capturée (*pressure*), le nombre de dimensions (1), et pour la dimension son nom, son type et

```
<Entity_Name>Wiimote</Entity_Name>
<Type>Device</Type>
<Dependence>Independent</Dependence>

<DeviceDescription>
    <Name>Wii Remote</Name>
    <Manufacturer>Nintendo</Manufacturer>
    <Size>148mm (L) x 36.2mm (W) x 30.8mm (D)</Size>
    <Wireless>True</Wireless>
    <Connectivity>Bluetooth</Connectivity>
    <PowerSupply>Two AA batteries</PowerSupply>
    <SensorsDescription>
        <Sensor>
            <Name>Accelerometer</Name>
            <NumberofDimensions>3</NumberofDimensions>
        </Sensor>
        <Sensor>
            <Name>Button</Name>
            <Number>8</Number>
        </Sensor>
        <Sensor>
            <Name>IR Sensor</Name>
        </Sensor>
    </SensorsDescription>
```

FIGURE 7.14 – MCDL : Exemple correspondant à la première section de la description MCDL du composant Wiimote.

son domaine (0-1). Finalement, est décrite la fréquence de l'interface du bouton A (continue).

La troisième section décrit le composant technique sous-jacent. Ainsi, nous séparons la description de l'interaction multimodale de la description technique afin de rendre notre approche indépendante d'une implémentation concrète des composants.

La Figure 7.16 illustre le code xml de la troisième section du composant Wiimote. Cette partie décrit d'abord l'auteur du composant, le type de licence et le prix du composant. Ensuite est décrit le type du composant : dans ce cas, il s'agit d'un composant OI, mais d'autres types de composants sont envisageables (JavaBeans, Osgi, etc.). Finalement est indiqué le lien vers la description technique : dans le cas des composants OI, il suffit d'indiquer l'identifieur du fichier CIDL, ce qui permet à l'OIDE de localiser le composant dans le dépôt local.

Les concepteurs utilisant l'OIDE peuvent accéder aux caractéristiques d'un composant qui décrivent la dépendance de son comportement et la syntaxe de son interface afin de savoir si le composant est utilisable dans un certain contexte. Pour cela, l'utilisateur peut faire un click droit sur le composant, affichant une fenêtre contextuelle avec les informations du composant. Ainsi, le concepteur peut choisir d'exploiter ou non ce composant au sein de l'assemblage en cours d'élaboration selon son degré de réutilisabilité ou la forme syntaxique de ses interfaces.

```
<ComponentSpecification>
    <TailoredInterface>
        <Name>Accelerometer</Name>
        <parameter><name>x</name><type>int</type></parameter>
        <parameter><name>y</name><type>int</type></parameter>
        <parameter><name>z</name><type>int</type></parameter>
    </TailoredInterface>

    <NormalizedInterface>
        <Name>Button A</Name>
        <normtype>Buxton</normtype>
        <BuxtonInterface>
            <property>Pressure</property>
            <dim>1<dim>
            <parameter><name>buttonvalue</name><type>int</type><values>0-1</values></parameter>
            <frequency>continuous</frequency>
        </BuxtonInterface>

    </NormalizedInterface>

    <TailoredInterface>
        <Name>Arrow Button</Name>
        <parameter><name>buttonvalue</name><type>int</type></parameter>
    </TailoredInterface>

    <TailoredInterface>
        <Name>IRInput</Name>
        <parameter><name>x</name><type>int</type></parameter>
        <parameter><name>y</name><type>int</type></parameter>
        <parameter><name>z</name><type>int</type></parameter>
    </TailoredInterface>

    ... // other buttons

</ComponentSpecification>
```

FIGURE 7.15 – MCDL : Exemple correspondant à la deuxième section de la description MCDL du composant Wiimote.

```
<ComponentImplementation>
    <Author>University of Grenoble</Author>
    <Licence>BSD</Licence>
    <Price>free</Price>
    <Type>OI</Type>
    <CIDL_id>openinterface.prototype.component.cplusplus.oiwii</CIDL_id>
</ComponentImplementation>
```

FIGURE 7.16 – MCDL : Exemple correspondant à la troisième section de la description MCDL du composant Wiimote.

7.3.2 Structure des assemblages et types des composants

Le concepteur n'est pas contraint à respecter la structuration du flot de données (assemblage de composants dans l'OIDE) issue du modèle. Cette caractéristique permet d'augmenter le pouvoir de génération de l'outil. La partie gauche de la Figure 7.17 illustre un assemblage de test qui consiste en deux composants connectés et qui ne respecte donc pas la structuration du modèle. Dans cet assemblage, le composant de gauche (composant OILightInteraction) correspond au dispositif Cubtile, un dispositif tactile en forme de cube, et le composant de droite est un adaptateur pour manipuler GoogleEarth à partir du Cubtile. La partie droite de la Figure 7.17 illustre un assemblage qui respecte la structuration en Y du modèle. Deux composants Dispositif (à gauche, DiamondTouch et Shake) sont composés de façon complémentaire (composant de composition au centre de l'assemblage) afin de manipuler une carte (composant ImageNavigator, à droite de l'assemblage).

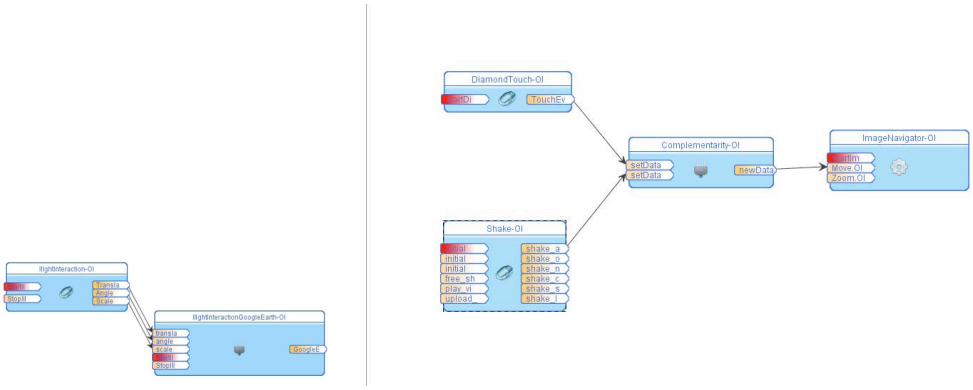


FIGURE 7.17 – Exemple d'assemblage de test ne respectant pas la structuration issue du modèle (à gauche) et d'assemblage respectant la structuration du modèle (à droite).

Néanmoins, la typologie des composants issue de notre modèle est explicitée dans l'OIDE, au niveau de la palette de composants utilisables et au niveau de la zone de conception de l'assemblage.

La typologie des composants dans la palette de composants est explicite (Figure 7.18). Les composants sont classés selon leur types : dispositif, transformation, composition et tâche. Ainsi, le concepteur peut chercher directement un composant d'un type donné.

Au sein de la zone d'assemblage, les types de composants sont également différenciés par un aspect graphique différent. Nous affichons ainsi sur les composants Dispositif une image d'une souris, sur les composants Transformation nous affichons une roue dentée (pour représenter symboliquement le processus de transformation), sur les composants de composition une prise et sur les composants de type utilitaire deux outils croisés (Figure 7.19). Les composants de type tâche sont identifiés par l'absence d'icône. Cette représentation par des icônes permet d'augmenter l'expressivité des rôles des composants et la lisibilité, critères utilisés au chapitre 6 pour analyser les outils existants.

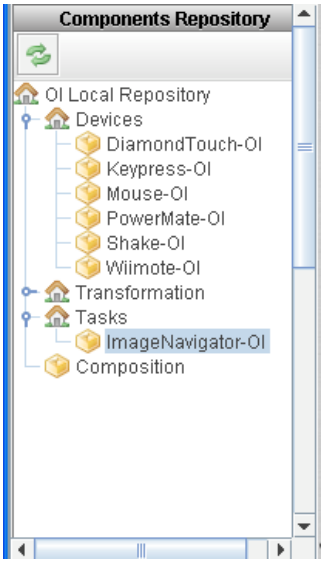


FIGURE 7.18 – Palette de composants du OIDE : typologie des composants disponibles selon leur niveau dans le flot de données.

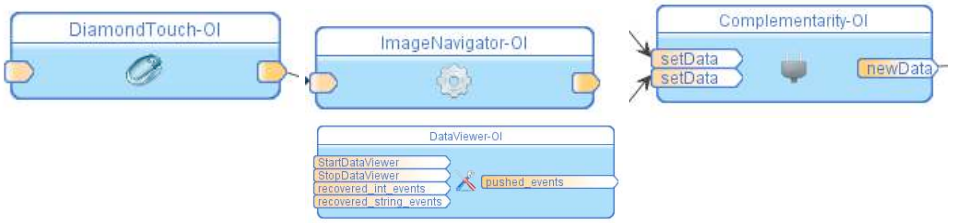


FIGURE 7.19 – Représentation graphique des composants Dispositif, Transformation, Composition et Utilitaire dans le OIDE.

7.4 Composants disponibles pour le développement de prototypes interactifs

Un total de 82 composants ont été intégrés à la plate-forme : 30 composants dispositif, 32 composants de transformation, 3 composants de composition, 7 composants tâche et 10 composants utilitaires. Les composants utilitaires, qui ne sont pas décrits dans notre modèle, correspondent à des composant implémentant des fonctionnalités aidant le concepteur dans la création d'assemblages. Par exemple, le composant DataViewer permet d'afficher sur une fenêtre graphique les données véhiculées par un autre composant. Nous n'avons pas développé tous ces composants, qui sont les résultats du projet européen OpenInterface.

7.4.1 Composants Dispositif

Un grand nombre de composants Dispositif ont été incorporés à la plate-forme. Nous avons commencé par intégrer les dispositifs habituels d'interaction tels que la souris ou le clavier, ainsi que différents types de boutons : un bouton de contrôle rotatif (PowerMate) et un bouton de contrôle à 6 degrés de liberté (SpaceNavigator)

(Figure 7.20).

FIGURE 7.20 – Premiers dispositifs intégrés : la souris et deux boutons de contrôle, PowerMate (au centre) et SpaceNavigator (à droite).

Nous avons également défini des composants liés aux dispositifs commerciaux dédiés aux jeux vidéo, tels que la Wiimote et la Wiifit (manettes de la console Wii de Nintendo), ou une veste à retour d'effort (dispositif en sortie) (Figure 7.21). La Wiimote est un dispositif très riche qui intègre plusieurs boutons, une caméra infrarouge ainsi qu'un accéléromètre, ce qui en fait un dispositif (et donc un composant) très fréquemment utilisé dans nos prototypes multimodaux (gestes langagiers, pointage).

FIGURE 7.21 – Dispositifs commerciaux dédiees aux jeux vidéo : wiimote, wiifit et veste à retour d'effort.

Au-delà des dispositifs commerciaux grand public, nous avons aussi focalisé sur des dispositifs permettant de capturer des gestes. Parmi ces dispositifs se trouvent un gant de données (DataGlove), un boîtier sans fil contenant plusieurs capteurs dont un accéléromètre (Shake), l'iPhone à travers son accéleromètre ou le trackIR, un dispositif qui permet de reconnaître des gestes de la tête (Figure 7.22).

De plus, plusieurs dispositifs capables de capturer le toucher sur une surface ont été intégrés à l'OIDE : deux tables et un cube tactiles (Figure 7.23). La première table, la DiamondTouch [Dietz et Leigh, 2001], est une surface sans affichage capable de détecter plusieurs doigts. La deuxième table, fabriquée par Immersion (partenaire du projet OpenInterface), est une surface retro-projetée qui permet de capturer plusieurs doigts par un système de caméras infrarouges. Le projecteur et les caméras sont occultés sous la table. Le cube tactile, appelé Cubtile et fabriqué

FIGURE 7.22 – Dispositifs de capture de gestes : de gauche à droite, Dataglove, Shake, iPhone et trackIR.

également par Immersion, est un dispositif cubique avec cinq faces tactiles (toutes les faces sauf celle inférieure). Le cube, posé sur un socle, peut être manipulé debout et reconnaît plusieurs doigts en parallèle. Ces trois dispositifs sont particulièrement adaptés à l'interaction bimanuelle. Tandis que les deux tables se prêtent plus à des manipulations en 2D, le cube est particulièrement adapté à l'interaction en 3D.

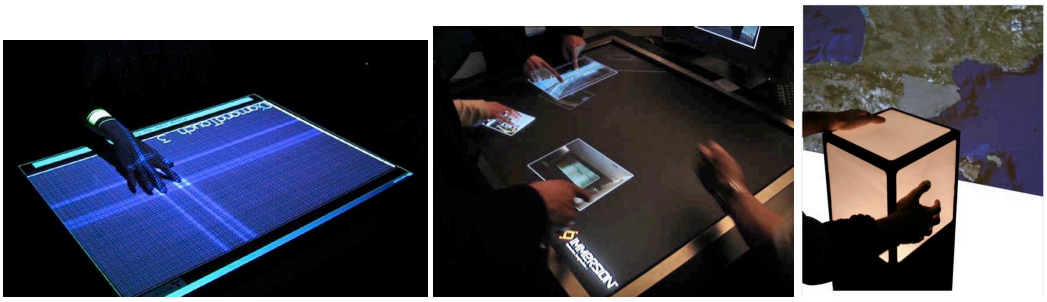


FIGURE 7.23 – Dispositifs tactiles : table DiamondTouch (à gauche), table Immersion (au centre) et cube Cubtile (à droite).

Des composants Dispositif de reconnaissance vocale ont été inclus dans l'OIDE. Des solutions gratuites comme Sphinx [Huang *et al.*, 1993] et des composants propriétaires développés par Multitel (partenaire du projet OI).

Nous avons aussi incorporé des algorithmes de traitement vidéo temps réel, considérés comme des composants Dispositif Logiciel (chapitre 4). Ces composants effectuent différents types de reconnaissances : reconnaissance du mouvement de la tête (head tracking), reconnaissance des doigts (finger tracking [Letessier et Bérard, 2004]), reconnaissance de patchs (ARToolkit), ou reconnaissance de la position d'un doigt en 3D (AirMouse [Ortega et Nigay, 2009]), cette dernière basée sur l'utilisation conjointe de deux caméras infrarouges (Figure 7.24).

Finalement, nous avons inclus des composants Dispositif pour exploiter des plate-formes de capteurs physiques. Les plate-formes Interface-Z [Interface-Z, 2010], Phidgets [Greenberg et Fitchett, 2001] et Arduino [Buechley *et al.*, 2008] ont ainsi été intégrées. Ces plate-formes permettent de connecter différents capteurs physiques, tels que des capteurs de torsion, de pression et de lumière. Avec ces capteurs, nous avons conçu différents dispositifs, tels que la Wiisoft, une sorte de Wiimote pliable (capteur de torsion), ou un dispositif Ballon, qui permet d'interagir en exerçant une pression (capteur de pression) (Figure 7.25). L'intérêt de ces plate-formes réside dans la possibilité de prototyper rapidement des dispositifs innovants en ré-utilisant le composant

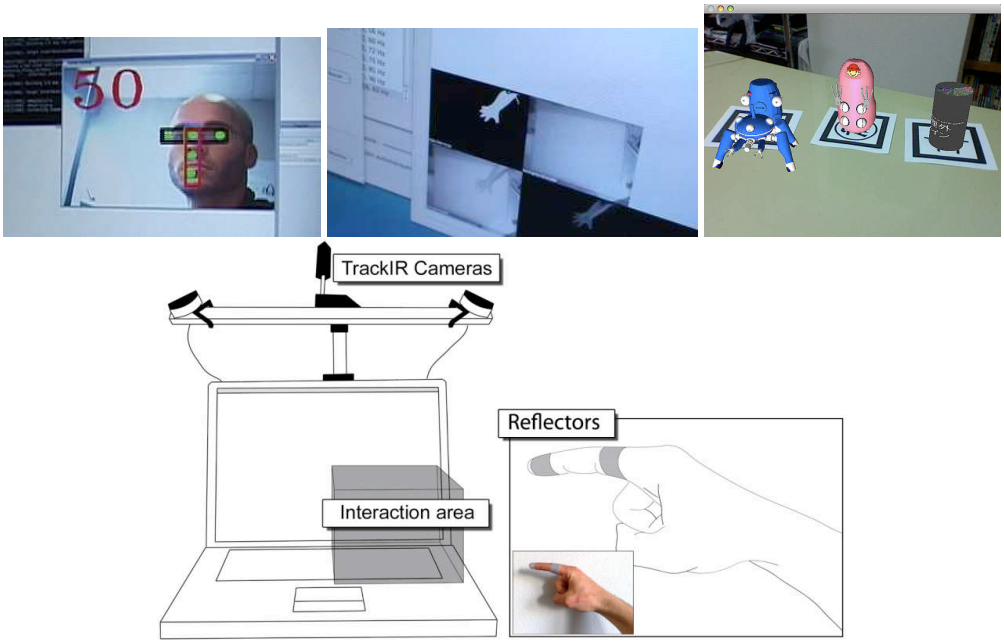


FIGURE 7.24 – Dispositifs de reconnaissance vidéo temps réel : de gauche à droite et de haut en bas, headtracking, fingertracking [Letessier et Bérard, 2004], ARToolkit et AirMouse [Ortega et Nigay, 2009].

logiciel correspondant.

FIGURE 7.25 – Dispositifs construits à partir de capteurs physiques interface-Z [Interface-Z, 2010] : le Wiisoft, à gauche (capteur de torsion) et le ballon, à droite (capteur de pression).

En synthèse, le Tableau 7.1 résume les composants Dispositif développés dans le cadre du projet OpenInterface. Tous ces composants Dispositif ont été développés par les partenaires du projet OpenInterface. Nous avons ensuite réalisé l'intégration en écrivant le MCDL de chaque composant.

7.4.2 Composants de transformation

Plusieurs composants de composition ont été développés et intégrés à la plateforme. Ce développement a suivi plusieurs directions : nous avons développé d'un côté des composants de transformation à comportement indépendant et d'un autre

Type	Nombre
Dispositif de manipulation directe	4
Geste 3D	7
Surface tactile	3
Reconnaissance vocale	2
Reconnaissance vidéo	4
Capteurs	3

TABLE 7.1 – Synthèse des composants Dispositif incorporés dans la plate-forme OpenInterface.

côté des composants de transformation implémentés pour des besoins spécifiques et dépendants d'une application ou d'un dispositif donné.

Les composants de transformation indépendants opèrent des transformations paramétrables de différentes natures, par exemple mathématique ou de filtrage. La Figure 7.26 illustre deux composants indépendants : un composant de filtrage passe-bas et un composant de filtrage de commandes. Le premier réalise un filtrage passe-bas des données en entrée, par exemple pour réduire le bruit dans les données d'un dispositif accéléromètre. Le deuxième composant permet de générer des commandes selon les valeurs en entrée. La fonction de correspondance entre les valeurs en entrée et les commandes en sortie est spécifiée dans un fichier texte externe.

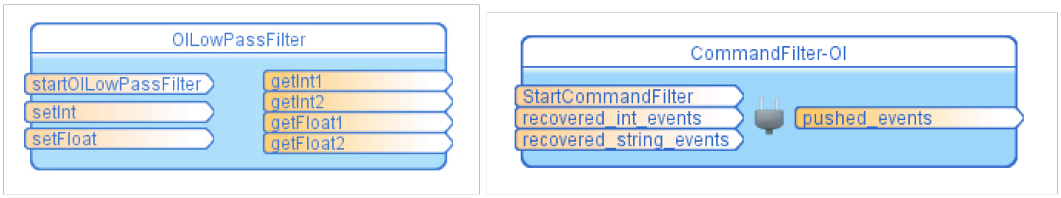


FIGURE 7.26 – Composants de transformation indépendants : filtre passe-bas, filtre de commandes.

Des composants dépendants des dispositifs ont été implémentés surtout pour les dispositifs qui intègrent plusieurs capteurs, comme la Wiimote ou la Shake. Des composants dépendants ont également été implémentés pour traiter les données multi-point venant des dispositifs tactiles, comme le Cubtile. Dans ce cas, le composant de transformation permet de générer des évenements haut niveau à partir des points, comme des actions de zoom (lorsqu'on éloigne ou approche deux doigts sur la surface) ou de rotation (lorsqu'on fait tourner deux doigts sur la surface).

Des composants dépendants d'applications ont également été crées. Par exemple nous avons créé un composant pour adapter l'interaction du Cubtile (composant ILightInteraction sur la Figure 7.27) à la sémantique du composant GoogleEarth (composant ILightInteractionGoogleEarth sur la Figure 7.27) . Ainsi, les deux composants contiennent des ports identiques afin d'être compatibles : un port Translation, un port Angle et un port Scale qui permettent de véhiculer les différentes données d'interaction générées par le Cubtile.

En synthèse, le Tableau 7.2 résume les composants Transformation développés dans le cadre du projet OpenInterface. Tous ces composants Transformation ont été

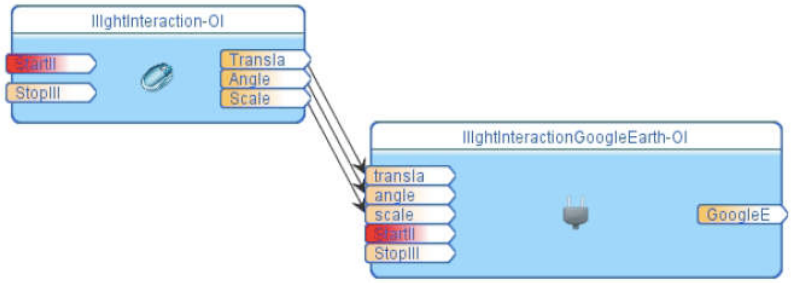


FIGURE 7.27 – Composant de transformation ILightInteractionGoogleEarth (à droite) dépendant de l'application GooglEarth.

développés par les partenaires du projet OpenInterface. Nous avons ensuite réalisé l'intégration en écrivant le MCDL de chaque composant.

Type	Nombre
Indépendants	8
Dépendants des dispositifs	14
Dépendants de l'application	10

TABLE 7.2 – Synthèse des composants Transformation incorporés dans la plate-forme OpenInterface.

7.4.3 Composants de composition

Les composants de composition développés et intégrés sont génériques. Pour implémenter le mécanisme de fusion temporel selon CARE, décrit au chapitre 4 section 4.3, nous avons développé les trois composants de la Figure 7.28 : Equivalence, Redondance et Complémentarité.

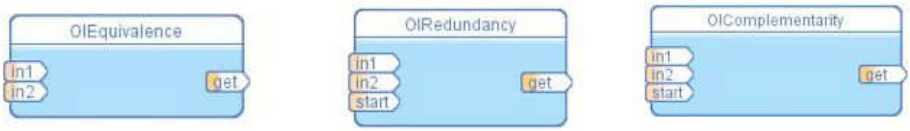


FIGURE 7.28 – Composants de composition intégrés à la plate-forme : redondance, complémentarité et equivalence.

Nous avons conçu ces composants pour réaliser la fusion temporelle de deux modalités en entrée. Les composants possèdent donc deux ports en entrée et un port *start* de démarrage. Le port *start* permet de spécifier le type de relation temporelle à utiliser dans la fusion : anachronisme, séquence, concomitance, coïncidence ou parallèlisme (section 4.3.4 du chapitre 4).

Nous avons implémenté et intégré ces trois composants dans la plate-forme OpenInterface.

7.4.4 Composants tâche

Comme nous l'avons décrit au chapitre 4, nous avons créé trois types de composants tâche.

- Des composants tâches qui communique avec l'application à travers un contrôleur de dialogue. Par exemple, le jeu ArkanOI, développé pendant le projet OpenInterface, est utilisé à travers un composant qui envoit les commandes issues de l'interaction par une connexion UDP, à un contrôleur de dialogue.
- Des composants tâche qui sont eux-mêmes des applications. C'est le cas du diaporama multimodal que nous avons implémenté et intégré dans la plateforme comme un composant. Ainsi, chaque port du composant représente une tâche interactive de l'application (zoom, transition, etc.).
- Des composants qui simulent des entrées d'interaction. Ainsi, nous avons créé un générateur de touches clavier (code de la touche en Ascii), ainsi qu'un émulateur de cliques souris.

En synthèse, le Tableau 7.3 résume les composants Tâche développés dans le cadre du projet OpenInterface. Tous ces composants Tâche ont été développés par les partenaires du projet OpenInterface. Nous avons ensuite réalisé l'intégration en écrivant le MCDL de chaque composant.

Type	Nombre
Contrôleur de dialogue	2
Applications	5
Simulation d'entrées	2

TABLE 7.3 – Synthèse des composants Tâche incorporés dans la plate-forme OpenInterface.

7.4.5 Composants utilitaires

En dehors des composants précédents, définis selon notre modèle conceptuel, nous avons créé des composants utilitaires. Nous distinguons deux types de composants utilitaires.

Composants de communication vers l'extérieur de l'environnement OIDE

Les composants de communication implémentent des protocoles de communication afin de permettre la communication avec des éléments extérieurs à l'assemblage et à l'OIDE. Ces composants ont pour objectif d'augmenter l'inter-opérabilité et la flexibilité de l'approche. Ces composants de communication sont :

- Composant socket : ce composant permet d'ouvrir une socket et d'envoyer ou recevoir des informations.
- Composant OSC : ce composant réalise l'interface avec le prototocole OSC (OpenSoundControl). Ce protocole est très utilisé pour les applications audio et vidéo temps réel.

- Composant Bluetooth : ce composant permet de découvrir et se connecter à un dispositif bluetooth. Ce composant permet en particulier de communiquer avec des applications exécutées sur téléphones mobiles.

- Composant MIDI : ce composant permet d'envoyer des événements avec le protocole de communication MIDI (Musical Instrument Digital Interface).

Ces composants permettent donc de communiquer avec des éléments extérieurs implémentant le protocole correspondant, par exemple d'autres composants extérieurs à l'assemblage. Lorsque ces composants sont utilisés pour recevoir des événements de l'extérieur, ils sont généralement utilisés au niveau Dispositif ou Transformation de notre modèle. Lorsque ces composants sont utilisés pour envoyer des événements vers l'extérieur, ils sont généralement utilisés au niveau Tâche de notre modèle.

Composants de visualisation

Des composants ont été créés afin de visualiser les données du flot de données défini par l'assemblage [Gray *et al.*, 2007]. Outils dédiés à la conception et à la mise au point des composants, ces composants permettent d'augmenter la lisibilité dans l'OIDE. La Figure 7.29 illustre deux composants de visualisation : un composant oscilloscope, qui permet de visualiser des données numériques, et un composant visualisateur de données, qui permet de voir textuellement des données numériques ou textuelles.

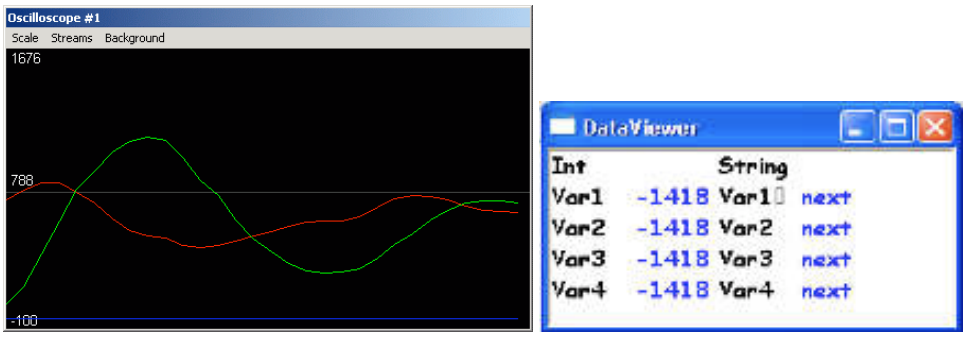


FIGURE 7.29 – Deux composants de visualisation : oscilloscope et visualisateur de données.

Par exemple, le composant oscilloscope permet de mettre au point la reconnaissance de gestes avec le Shake pour naviguer dans Google Earth en modifiant la valeur du seuil de l'accéléromètre dans l'OIDE après avoir effectué des tests en utilisant l'oscilloscope (Figure 7.30).

En synthèse, le Tableau 7.4 résume les composants Utilitaire développés dans le cadre du projet OpenInterface. Nous avons développé deux composants tandis que les autres ont été développés par les partenaires du projet OpenInterface. Nous avons ensuite réalisé l'intégration en écrivant le MCDL de chaque composant.

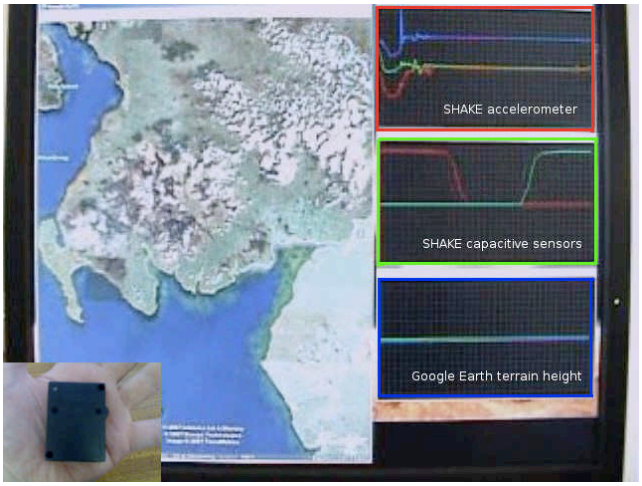


FIGURE 7.30 – Utilisation temps-réel du Shake (en bas à gauche) avec le composant oscilloscope afin de visualiser les données de l'accéleromètre. Illustration extraite de [Gray *et al.*, 2007].

Type	Nombre
Communication	4
Visualisation	6

TABLE 7.4 – Synthèse des composants Utilitaire incorporés dans la plate-forme OpenInterface.

7.5 Composants disponibles pour le développement de prototypes simulés

Notre environnement OIDE vise à intégrer de façon continue le développement de prototypes simulés et interactifs. Cette caractéristique de l'OIDE est originale vis-à-vis de l'état de l'art des outils existants (chapitre 6). Tandis que le développement de prototypes interactifs repose sur les composants décrits à la section précédente, nous définissons et illustrons dans cette section des composants dédiés au développement de prototypes simulés. Nous notons ces composants OpenWizard. Plus généralement OpenWizard est le nom de l'application du OIDE au développement de prototypes simulés ou expériences Magicien d'Oz.

7.5.1 Approche OpenWizard : un continuum du prototype simulé au prototype interactif

Dans le cadre d'une conception itérative centrée sur l'utilisateur, il semble particulièrement intéressant de pouvoir, au sein d'un même outil, développer des prototypes totalement simulé jusqu'à des prototypes interactifs en passant par des prototypes partiellement simulés. Nous illustrons ce continuum au sein de l'OIDE par la métaphore du puzzle (Figure 7.31). Étant donné un prototype multimodal comme un puzzle, si des pièces manquent, le prototype ne peut pas fonctionner car il n'est pas complet. Notre approche consiste alors à compléter ce puzzle avec des composants OpenWizard, qui permettent de simuler les pièces manquantes. Ainsi, le

prototype partiellement interactif, partiellement simulé, sera formé par des composants fonctionnels (section 7.4) et des composants de simulation (les composants OpenWizard).

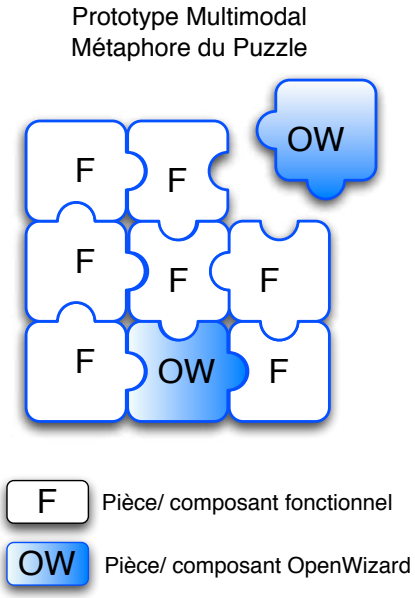


FIGURE 7.31 – Prototype multimodal non fonctionnel comme un puzzle, complété en utilisant des pièces/composants OpenWizard.

7.5.2 Composants OpenWizard

Nous avons développé plusieurs composants OpenWizard (OW) génériques et réutilisables. En terme de dépendance comportementale, tous les composants OW sont indépendants. En ce qui concerne la forme syntaxique des interfaces, nous avons utilisé des syntaxes normalisées lorsque cela était possible. Tous les composants OW développés partagent la même organisation de l'interface graphique qui sera utilisée par le compère pour simuler le composant. L'interface compère des composants OW est constituée de plusieurs zones. La Figure 7.32 illustre la configuration générale d'un composant OW correspondant à un dispositif physique. Ce composant OW permet de simuler des dispositifs à une dimension (bouton, rotation de molette).

Dans la partie inférieure de l'interface, deux consoles affichent les événements entrants et sortants. Les événements en entrée correspondent à des événements reçus par le composant (notons que certains composants n'ont pas d'entrée). Les événements en sortie correspondent aux événements envoyés par le composant donc spécifiés par le compère. En haut à gauche de l'interface, un panel permet de générer les événements OW, dans ce cas précis avec un slider. En haut à droite, un panel présente les éléments de configuration du composant, différents selon le composant : dans l'exemple de la figure, l'utilisateur peut configurer le domaine de valeurs en sortie (la taille du slider) et la résolution du slider. Ces deux panels supérieurs sont différents suivant les composants OW. Notons également que l'interface

ne contient pas de retour vidéo où le compère peut observer les actions de l'utilisateur (sujet de l'expérimentation). Dans notre approche, nous considérons que cet affichage vidéo est présenté sur un écran dédié à côté de l'écran de l'ordinateur, comme cela se fait habituellement dans les studios d'évaluation.

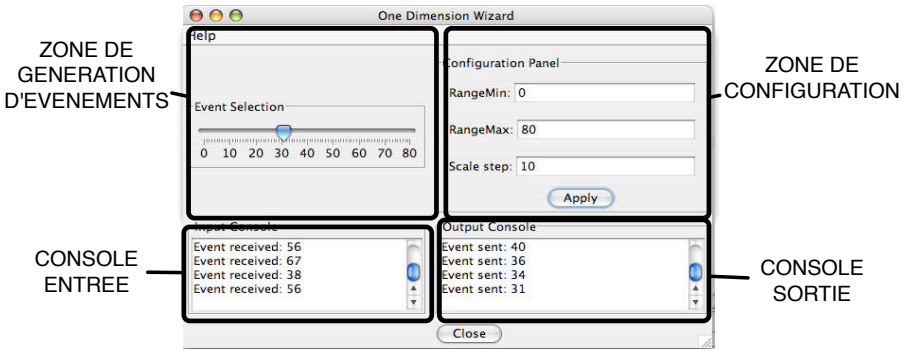


FIGURE 7.32 – Interface graphique générale d'un composant OpenWizard.

Nous avons développé des composants OW Dispositif, de Transformation et de Composition.

Composants OW Dispositif

Afin de simuler des dispositifs en entrée, nous avons développé deux composants OW Dispositif. Pour augmenter la compatibilité syntaxique de ces composants, nous normalisons les interfaces en adoptant la syntaxe inspirée de la taxonomie de Buxton, définie à la section 4.4.4 du chapitre 4. Les deux composants OW Dispositif implémentés sont les suivants :

- OW Dispositif à 1 dimension : permet de simuler un dispositif à une dimension. Nous avons déjà illustré ce composant à la Figure 7.32. Ce composant permet de simuler les trois propriétés définies dans Buxton : Mouvement, Pression et Position. Nous définissons ainsi trois interfaces normalisées en sortie de ce composant, une par propriété.

- OW Dispositif à 2 dimensions : permet de simuler un dispositif à deux dimensions. L'utilisateur interagit avec la souris sur une espace 2D afin de générer les événements (x, y) en sortie. La Figure 7.33 illustre l'interface compère de ce composant. La zone de configuration permet de définir la taille en pixels de l'espace 2D (et donc le domaine des valeurs en sortie). Ce composant ne permet pour l'instant de simuler que des interactions non-continues (clique souris), ce qui correspond à la propriété Position de Buxton. La simulation d'une interaction continue soulève des difficultés, en raison du temps de latence du composant OW (exécuté à distance sur une autre machine), ce qui représente une limitation de notre approche. Ainsi, nous avons défini une seule interface normalisée en sortie, pour la propriété Position.

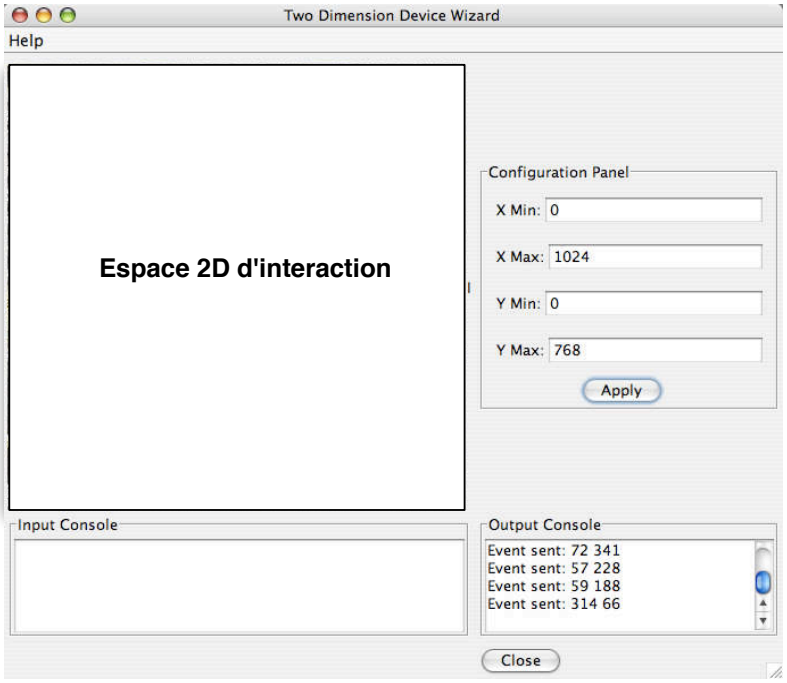


FIGURE 7.33 – Interface compère du composant OW Dispositif à 2 dimensions.

Composants OW Transformation

Les composants OW Transformation permettent de générer des événements à plus haut niveau d'abstraction. Nous avons identifié et implémenté un composant OW Transformation de génération de commandes. Ce composant est indépendant et son interface est ad hoc.

L'interface compère de ce composant (Figure 7.34) est basée sur l'utilisation de boutons pour simuler des commandes : cette interaction est simple et permet au compère de se concentrer sur le sujet de l'expérience. Cette interface est réutilisable et configurable : le compère peut générer ou effacer des boutons de commande en utilisant la zone de configuration (à droite à la Figure 7.34). Ainsi, le compère peut préparer son interface avant l'expérience Magicien d'Oz selon les commandes à simuler.

Composants OW Composition

L'utilisation des composants OW Composition permet au compère de décider en temps réel le type de relation temporelle à utiliser dans la composition. En effet, plusieurs études montrent que des différences significatives apparaissent entre les utilisateurs en ce qui concerne le patron d'intégration multimodal [Oviatt *et al.*, 2005] (c'est-à-dire la façon dont l'utilisateur utilise temporellement les différentes modalités d'interaction). Ces études signalent en particulier que la plupart des utilisateurs possède un patron d'intégration défini, soit simultané soit séquentiel, de façon permanente.

Afin d'adapter la relation temporelle de la composition des modalités, le compère peut utiliser un composant OW Composition. Les composants OW Composi-

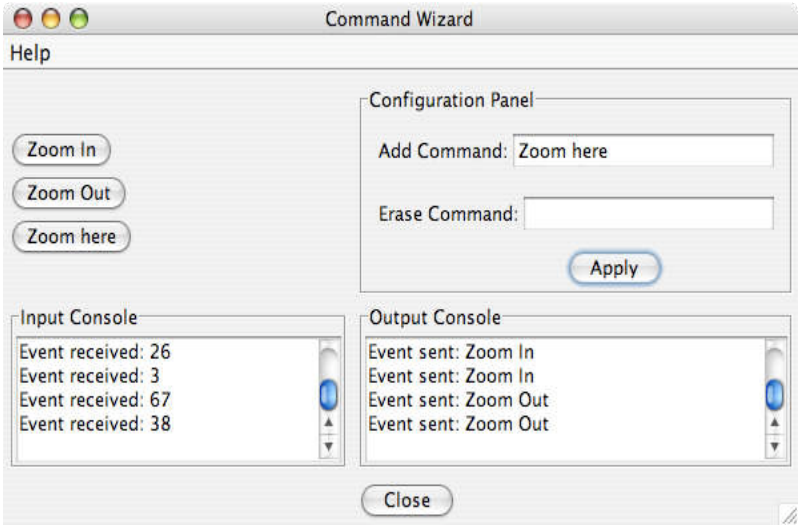


FIGURE 7.34 – Interface compère du composant OW Transformation implémenté.

tion permettent de remplacer une des deux modalités en entrée, en utilisant toujours les données fournies par l'autre modalité. Le composant OW Composition va générer les événements en sortie en fonction des événements en entrée, des événements générés par le compère et de la relation temporelle spécifiée par le compère. Afin de spécifier une relation temporelle, le compère peut sélectionner une des cinq relations de Allen, affichées sur l'interface compère du composant (Figure 7.35).

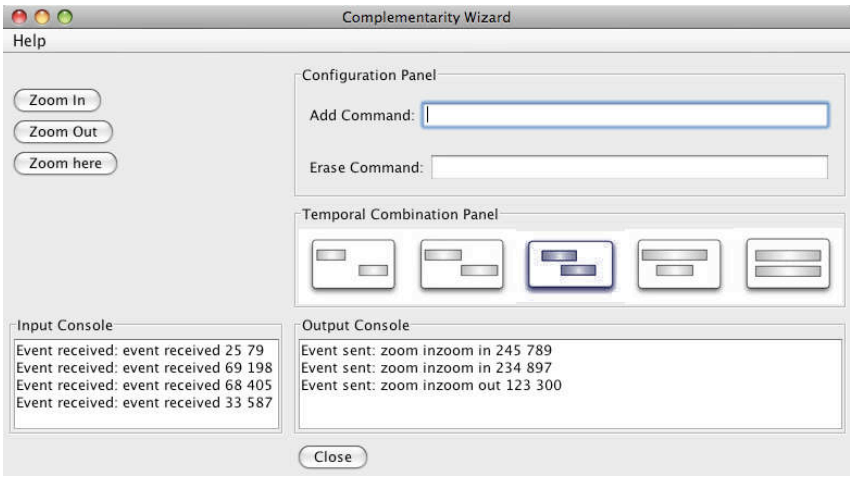


FIGURE 7.35 – Interface compère du composant OW Composition Complémentarité.

Nous avons implémenté trois composants OW Composition correspondant aux trois composants de composition présentés à la section 7.4.3 : équivalence, redondance et complémentarité (Figure 7.35). Ces composants sont indépendants d'un point de vue du comportement et possèdent des interfaces ad hoc.

En synthèse, le Tableau 7.5 résume les composants OW développés. Nous avons développé et intégré tous les composants au sein de la plate-forme OpenInterface.

Type	Nombre
OW Dispositif	2
OW Transformation	1
OW Composition	3

TABLE 7.5 – Synthèse des composants OW incorporés dans la plate-forme OpenInterface.

Charge cognitive du compère

Deux solutions principales ont été adoptées dans le but de réduire la charge cognitive du compère dans le cas où plusieurs composants OW sont intégrés dans un même assemblage :

- Dans un premier cas, tous les composants OW affichés sur une même machine seront intégrés dans une seule interface. Dans les sections suivantes, nous présentons des exemples d'intégration de plusieurs interfaces OW.

- Dans un deuxième cas, nous considérons une configuration multi-compère (chapitre 6 section 6.3.1). Dans une configuration multi-compère, les composants OW sont exécutés de façon distribuée sur différentes machines. Chaque compère contrôle un composant OW sur une machine différente (Figure 7.36). Cette configuration multi-compère permet par exemple d'assigner un compère à chaque composant simulé. Ainsi, la charge cognitive de chaque compère est réduite étant donné qu'il peut effectivement focaliser son attention sur un seul rôle de simulation lié au composant OW correspondant.

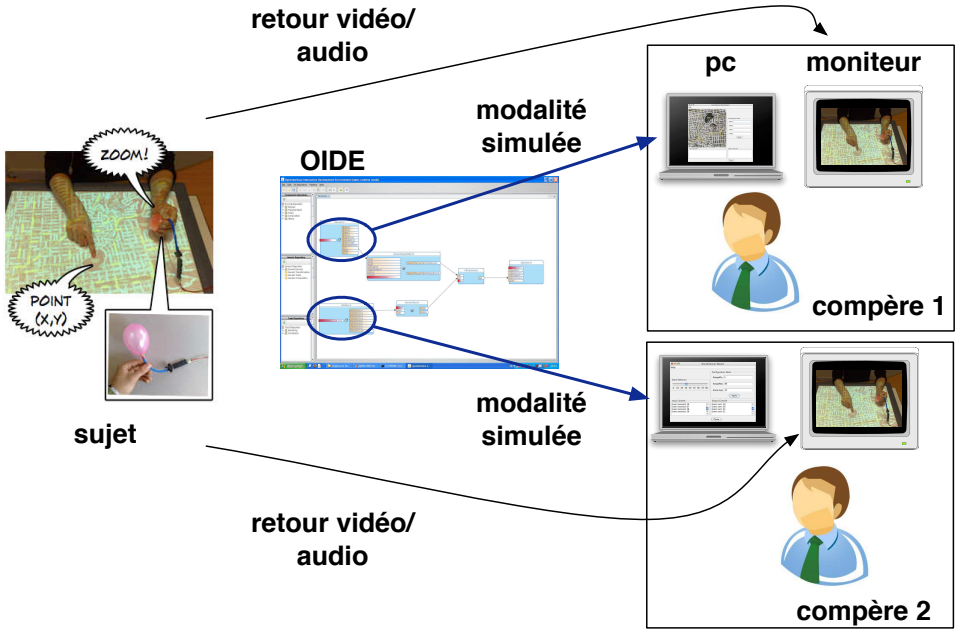


FIGURE 7.36 – Configuration multi-compère.

7.5.3 Degrés de simulation du prototype

Dans cette section, nous présentons un exemple illustratif selon différents degrés de fonctionnalité. Notre exemple illustre aussi des composants OW à différents niveaux dans le flot de donnés (Dispositif, Transformation et Composition).

L'utilisation d'un composant OW dans un assemblage OI permet de simuler une partie de l'interaction non disponible dans la plate-forme OpenInterface. Dans le cadre d'une conception itérative centrée sur l'utilisateur, une démarche serait de commencer par des prototypes complètement simulés afin d'avoir un premier retour rapide sur l'interaction multimodale conçue puis de continuer la conception par des prototypes de moins en moins simulés pour obtenir un prototype interactif fonctionnel. Le degré de simulation du prototype est alors décroissant. Néanmoins cette approche de conception ne traduit pas toujours la réalité de conception et nous l'avons observé dans le cadre du projet OpenInterface. En effet tandis que le prototype est fonctionnel sans simulation, il est envisageable de décider de tester une nouvelle modalité ou un nouveau dispositif. Si ce dernier n'est pas disponible alors il est simulé.

Par exemple, dans une interaction multimodale geste + parole sur une carte, nous avons développé un prototype interactif en exploitant le composant DiamondTouch avec une carte projetée à plat sur la table puis nous avons décidé d'explorer une projection de la carte verticale. Pour cela nous avons utilisé un composant OW pour simuler le geste de l'utilisateur sur le mur où était projetée la carte. Dans cet exemple nous passons d'un prototype interactif fonctionnel à un prototype partiellement simulé.

De notre expérience au sein du projet OpenInterface, nous concluons que la caractéristique primordiale de notre outil est de permettre une intégration sans couture de composants fonctionnels et simulés, sans contraindre la démarche de conception par exemple du prototype simulé au prototype interactif. Ainsi le degré de simulation du prototype peut être faible puis très élevé et vice-versa.

Nous illustrons différents degrés de simulation d'un prototype en considérant un exemple. Nous considérons l'exemple du système d'exploration multimodale de cartes géographiques utilisé au chapitre 4 (que nous détaillons ensuite dans le chapitre 8). Ce système permet la manipulation d'une carte grâce à différentes modalités d'interaction. Les tâches réalisées consistent à zoomer ou à se déplacer dans la carte. Par exemple, le zoom sur un point spécifique de la carte peut se réaliser en combinant la parole et le geste. L'utilisateur peut également interagir de façon bi-manuelle en combinant la pression d'un ballon (capteur Interface-Z [Interface-Z, 2010]) de la main non dominante avec un geste de désignation de la main dominante (Figure 7.37). Une forte pression sur le ballon signifie zoom avant, et une faible pression zoom arrière. La Figure 7.37 montre un exemple d'assemblage de composants correspondant à cette interaction.

Prototype faiblement simulé

Les composants OW peuvent être utilisés pour simuler un seul composant d'un assemblage OI. Ceci permet de définir un prototype à haute fonctionnalité faible-

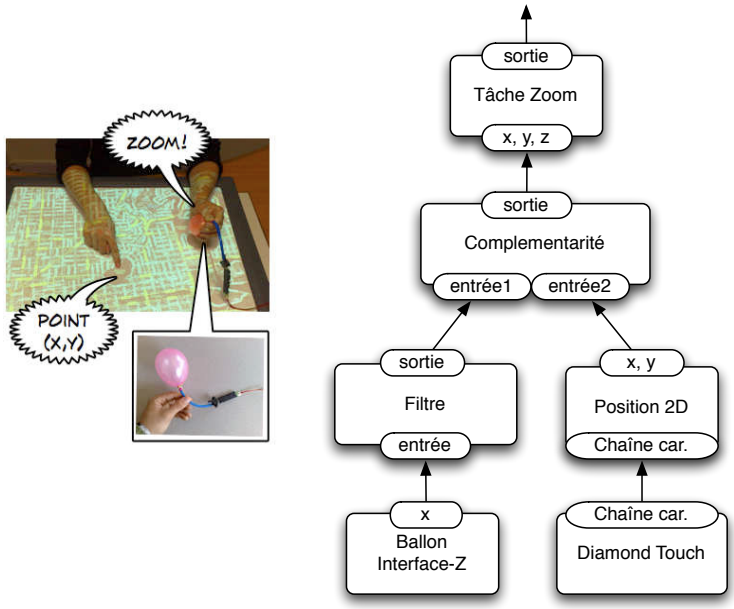


FIGURE 7.37 – Exemple d'interaction multimodale sur une carte géographique et assemblage correspondant.

ment simulé. Le compère simule alors qu'une petite partie de l'interface multimodale. Par exemple nous considérons la simulation d'un dispositif, tous les autres composants sont disponibles dans la plate-forme.

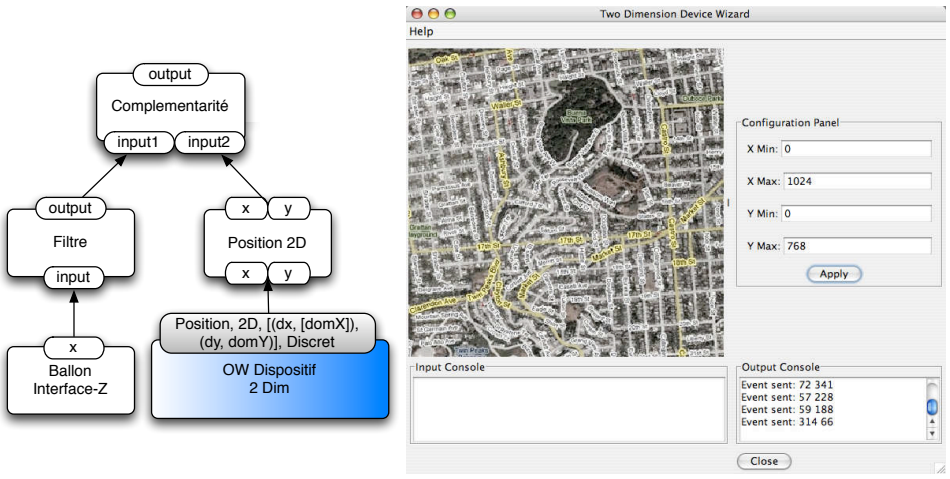


FIGURE 7.38 – Composant OW Dispositif dans un prototype faiblement simulé. À droite, assemblage de composants. À gauche, interface du compère

Dans notre exemple, nous voulons évaluer l'exploration d'une carte multimodale sur un mur au lieu d'une table. Or nous ne disposons pas de solution pour capturer la position du doigt sur un mur. Afin d'évaluer cette interaction, nous utilisons un composant OW Dispositif à deux dimensions afin de remplacer le composant DiamondTouch, comme le montre la Figure 7.38. Ce composant permet au

compère de générer une position (x, y) (interface normalisée selon Buxton) en cliquant sur l'image affichée. Le compère peut également modifier les dimensions de l'image dans la zone de configuration de l'interface.

Le composant OW Dispositif 2D permet donc de simuler n'importe quel dispositif à deux dimensions (doigt sur iPhone, clic souris, position de l'utilisateur). Evidemment, ce composant peut ne pas être complètement adapté à un dispositif particulier, comme un joystick 2D. Dans ce cas, un composant non réutilisable, dépendant du dispositif simulé mais intégrant une interface normalisée, peut être développé.

Prototype moyennement simulé : transformation

Parfois la partie manquante n'est pas un dispositif mais un algorithme : par exemple, l'algorithme qui génère une commande de zoom à partir de la position de deux doigts sur une table tactile. Le degré de simulation est alors plus élevé que dans le cas précédent. Le compère ne génère plus des données brutes mais des données à plus haut niveau d'abstraction. C'est le cas lorsque l'on utilise un composant OW de transformation.

Dans notre exemple de la carte, l'utilisateur peut faire un zoom avant ou arrière sur un point spécifique de la carte en utilisant le ballon. Pour cela, il utilise la pression sur le ballon : une forte pression se traduit par un zoom avant, tandis qu'une faible pression se traduit par un zoom arrière. Néanmoins, il est assez difficile de traiter les événements du ballon en raison de la sensibilité du capteur de pression. Dans ce cas, nous utilisons notre composant OW Transformation de génération de commandes pour simuler les commandes de zoom, comme le montre la Figure 7.39.

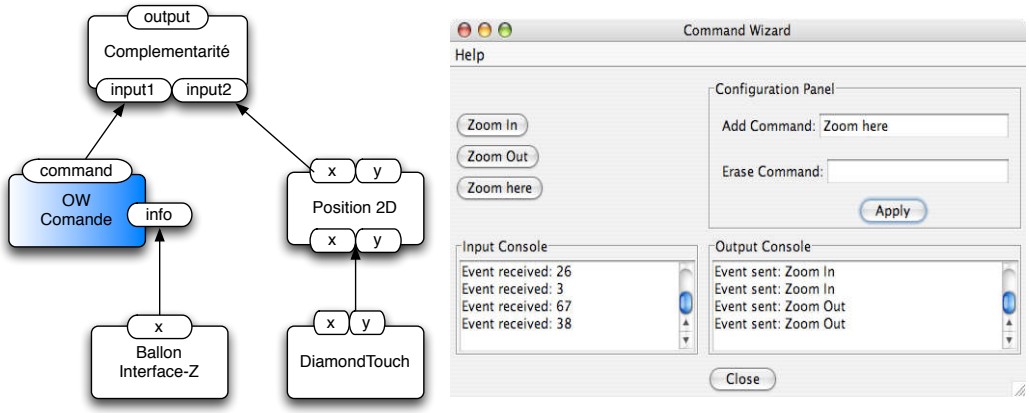


FIGURE 7.39 – Composant OW Transformation (a) assemblage de composants (b) interface du compère.

Nous constatons que dans le cas d'un composant OW transformation, les composants inférieurs dans le flot de données défini par l'assemblage ne sont pas nécessaires pour exécuter le prototype. Par exemple dans notre exemple, le compère peut uniquement se baser sur la vidéo de l'utilisateur en train de manipuler le ballon et le composant Dispositif "Ballon Interface-Z" n'est pas utile. Néanmoins, si le

composant es disponible, les évenements en provenance du dispositif sont affichés dans l'interface du compère et définissent une autre source d'information utile pour décider quelle commande générer. À la Figure 7.39 (b) les informations en provenance du dispositif sont affichés de façon textuelle dans la partie en bas à gauche. Il est néanmoins important d'employer une technique de visualisation des données appropriée aux types de données. Dans notre exemple, les données sont affichées textuellement sur la console, ce qui ne représente pas la meilleure solution. Pour cela, le composant OW Transformation peut utiliser des outils de visualisation intégrés au sein de la plateforme OpenInterface, comme par exemple le composant oscilloscope (section 7.4.5).

Prototype simulé : composition

Les composants OW Composition permettent de remplacer une des deux modalités en entrée, en utilisant toujours les données fournies par l'autre modalité. La Figure 7.40 illustre l'usage d'un composant OW Composition Complémentarité.

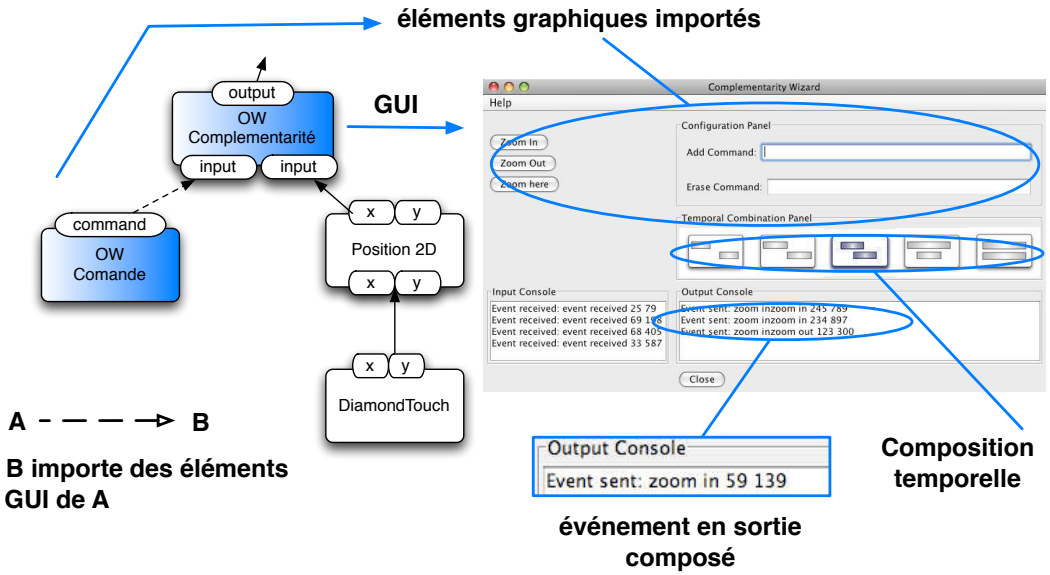


FIGURE 7.40 – Composant OW Composition dans un prototype à basse fonctionnalité.

Les principales caractéristiques des composants OW Composition sont les suivantes :

- Les composants OW Composition peuvent importer des éléments des interfaces des autres composants OW. La Figure 7.40 illustre les éléments graphiques importés par le composant de composition. Les éléments importés correspondent à la zone de génération et à la zone de configuration. De cette façon, le compère contrôle dans son interface le composant en entrée.

- Les composants OW Composition utilisent les événements en entrée pour générer la sortie. Cette sortie équivaut à la composition selon CARE de l'événement reçu en entrée et de l'événement généré par le compère.

- La relation temporelle entre les deux modalités est affichée sur l'interface graphique du composant. Le compère peut sélectionner une composition temporelle sur la liste et définir sa valeur temporelle en faisant double-clique dessus. Cela permet de définir quel événement de la DiamondTouch est utilisé dans la composition lorsque le compère clique sur une commande.

Prototype complètement simulé

Un prototype complètement simulé peut être développé. En adoptant une approche multi-compère, plusieurs composants OW présentés précédemment sont utilisés. Par contre, si nous adoptons une approche mono-compère, l'interface du composant OW doit permettre de simuler complètement l'interaction. Dans l'exemple présent, nous connectons deux composants OW à un composant OW Composition. Comme nous avons vu précédemment, ce composant de composition va intégrer certains éléments des interfaces des deux composants afin de créer une interface composée unique (Figure 7.41).

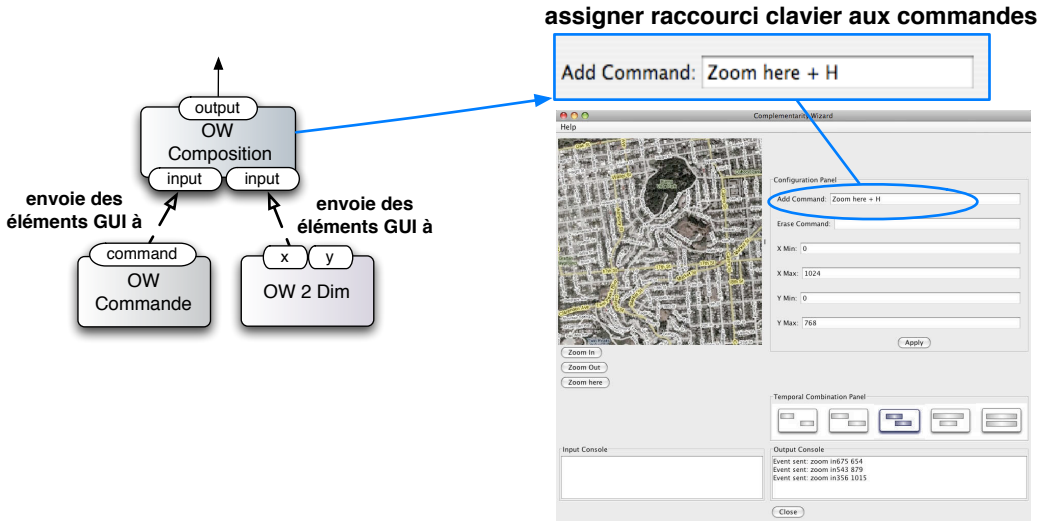


FIGURE 7.41 – Composant OW Composition dans un prototype non fonctionnel.

Cette interface composée est contrôlée par le compère pour simuler l'interaction multimodale. Afin de générer les événements composés, le compère a deux choix. D'une part il peut utiliser un bouton de commande pour activer un mode de commande particulier. Par exemple, le compère peut utiliser le bouton 'zoom in' pour activer le mode zoom avant. A partir de cet instant, chaque clique sur la carte équivaut à un zoom avant sur ce point. D'autre part, le compère peut utiliser des raccourcis clavier et la souris pour générer les commandes multimodales. Sur la Figure 7.41 le compère assigne la touche H à la commande 'zoom here'.

7.6 Conclusion

Dans ce chapitre, nous avons présenté notre environnement OIDE pour le prototypage d'interfaces multimodales. Les caractéristiques de notre outil vis-à-vis de l'état de l'art des outils existants (chapitre 6) sont :

- Un haut niveau de pouvoir descriptif et génératif : pour cela la spécification du prototype repose explicitement sur notre modèle conceptuel du chapitre 4.
- La possibilité de développer des prototypes multimodaux interactifs ou simulés.

L'OIDE fait environ 17000 lignes de code et est disponible sur le site forge du projet OpenInterface (https ://forge.openinterface.org/). Cet environnement OIDE a largement bénéficié du contexte du projet européen OpenInterface. Tout d'abord l'environnement est peuplé de nombreux composants, certains que nous avons développés et d'autres développés par les membres du consortium européen. De plus l'OIDE a été utilisé pour développer plusieurs prototypes, démonstrateurs du projet européen. Ainsi notre environnement a été utilisé par de nombreux partenaires au-delà des collègues de l'équipe. Le chapitre suivant présente l'évaluation et la validation de l'outil à travers les expériences menées et les prototypes réalisés.

Chapitre 8

Évaluation de l'outil de prototypage

« Ne croirait-on pas [...] qu'il n'y a jamais eu qu'un premier animal, prototype de tous les animaux »

Denis Diderot, Écrivain et philosophe français (1713-1784)

Contenu du chapitre

Dans ce chapitre, nous présentons les évaluations de l'OIDE, l'outil de prototypage présenté au chapitre précédent. L'évaluation d'un outil logiciel peut prendre plusieurs formes. Une première approche consiste à mener des évaluations empiriques contrôlées. Une autre forme d'évaluation consiste à montrer la variété des prototypes multimodaux possibles avec l'outil.

La partie dédiée à l'évaluation empirique est divisée en deux sections. À la section 8.1 nous présentons une évaluation contrôlée de l'outil qui focalise sur la définition de prototypes interactifs tandis qu'à la section 8.2 nous détaillons une évaluation empirique de l'outil pour la conception de prototypes simulés.

Dans la deuxième partie du chapitre, dédiée à la validation de l'outil, nous commençons par présenter les prototypes réalisés au sein du projet OpenInterface. Deux domaines d'application ont été considérés dans le projet : les grands espaces d'information et les jeux. Plusieurs versions de chaque prototype ont été construites selon une démarche de conception itérative centrée utilisateur.

Ensuite, nous introduisons des prototypes réalisés par des étudiants. Ces étudiants ont utilisé l'OIDE afin de tester rapidement des nouvelles modalités d'interaction. Finalement, nous présentons des prototypes développés pour différentes démonstrations de notre outil OIDE.

8.1 Évaluation empirique de la conception de prototypes interactifs

Une évaluation empirique de l'OIDE a été réalisée par des chercheurs de l'Université de Glasgow [McGee-Lennon *et al.*, 2009]. Le but de cette évaluation est d'analyser comment et à quel point l'OIDE représente une aide à la conception de prototypes multimodaux dans une démarche itérative de conception centrée utilisateur. Nous résumons les principaux aspects de cette évaluation.

8.1.1 Objectifs

Cette expérience contrôlée a focalisé sur les questions suivantes :

- Comment s'intègre l'OIDE dans une démarche itérative de conception centrée utilisateur ?
- Comment l'OIDE aide les concepteurs à développer des interactions multimodales ?
- Quelles limitations de l'OIDE frustrent ou bloquent les concepteurs ?
- Quels usages non-prévus apparaissent lors du travail avec l'OIDE ?
- Comment améliorer l'OIDE pour l'adapter d'avantage aux habitudes de travail des concepteurs d'interaction ?

8.1.2 Profil des sujets

L'expérience a été conduite sur 16 sujets, répartis en 5 groupes de conception. Les équipes de conception ont été créées aléatoirement. Elles diffèrent en taille et les sujets n'avaient jamais travaillé ensemble auparavant.

Les sujets étaient tous des chercheurs en informatique (étudiants, doctorants et chercheurs post-doctoraux), avec une expertise dans la conception d'interaction.

8.1.3 Protocole

Chaque groupe est placé devant un PC avec l'OIDE et un ensemble de composants déjà installés. Différents dispositifs sont disponibles : un microphone, une manette de Wii et plusieurs capteurs physiques Phidgets (Figure 8.1). Les composants OI correspondant à ces dispositifs sont installés dans l'OIDE. Les sujets disposent également de matériel pour réaliser du prototypage basse-résolution : papier, ciseaux, colle, etc.

Les participants sont introduits à l'usage de l'OIDE pendant 90 minutes avant de réaliser l'exercice de conception Les équipes de conception disposent ensuite de 90 minutes pour réaliser un exercice de conception. Le sujet de l'exercice demande de concevoir une interaction multimodale pour contrôler GoogleEarth dans le contexte d'une exposition sur la déforestation dans un musée.

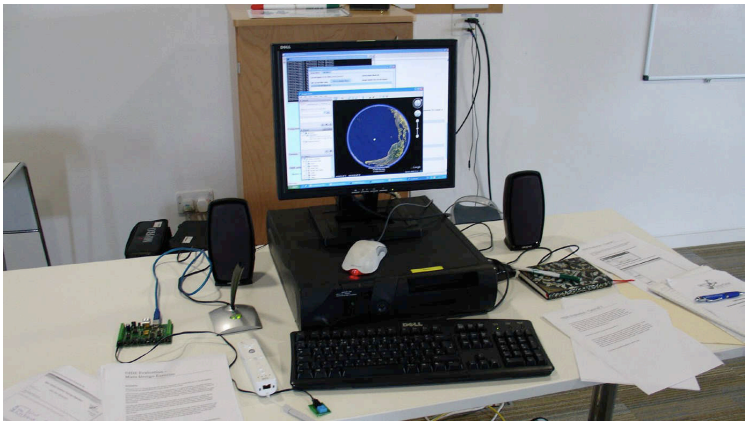


FIGURE 8.1 – Configuration de l'expérience : PC avec OIDE, GoogleEarth et plusieurs dispositifs d'interaction (phidgets, wiimote et microphone).

8.1.4 Résultats

Toutes les équipes ont réussi à compléter l'exercice de conception. L'expérience a montré que l'outil a servi d'instrument de communication et d'exploration de nouvelles solutions.

Trois approches de conception ont émergé. Certaines équipes utilisaient des schémas sur papier pour exprimer leurs idées en même temps qu'elles utilisaient l'OIDE pour explorer l'espace de conception. D'autres équipes plus créatives travaillaient premièrement le prototype papier, sans utiliser l'OIDE. La Figure 8.2 illustre un de ces prototypes papier. Ces équipes concevaient ensuite leur prototype avec l'OIDE. Enfin, des équipes plus techniques regardaient d'abord les composants disponibles et ne concevaient que des interactions qu'ils étaient sûrs de pouvoir réaliser.

FIGURE 8.2 – Évaluation empirique du OIDE : groupe de conception en plein travail avec l'évaluateur (à gauche) et prototype papier réalisé (à droite).

Accompagné ou non de conception de prototypes papier, cette expérience a montré l'intérêt de l'outil pour explorer rapidement l'espace des possibilités.

8.2 Évaluation empirique de la conception de prototypes simulés

Pour compléter cette évaluation empirique faite à Glasgow, nous avons mené une expérience à Grenoble qui focalisait sur une expérience magicien d'oz, et donc un prototype multimodal simulé developpé avec l'OIDE, et sur le travail des compères avec l'interface des composants OW utilisés.

Cette évaluation est donc complémentaire à celle menée à Glasgow : elle ne focalise pas sur la construction de l'assemblage avec l'OIDE, mais sur l'exécution d'un assemblage qui contient des composants OW et sur le travail des compères associés à ces composants.

8.2.1 Objectif de l'évaluation empirique et application simulée

L'objectif de cette évaluation est d'évaluer la pratique de l'approche OpenWizard. Pour cela, nous avons choisi de réaliser une expérience qualitative. Nous rappelons qu'une évaluation qualitative est une évaluation menée sur un nombre limité de sujets. Elle permet de rendre compte de la diversité des usages.

L'application simulée est un diaporama multimodal, projeté sur un mur et contrôlé par le geste et la voix. L'utilisateur peut naviguer dans les diapositives en utilisant le geste ou la voix. L'utilisateur peut également faire un zoom dans une diapositive et naviguer avec un geste bi-manuel (Figure 8.3) ou avec la voix.

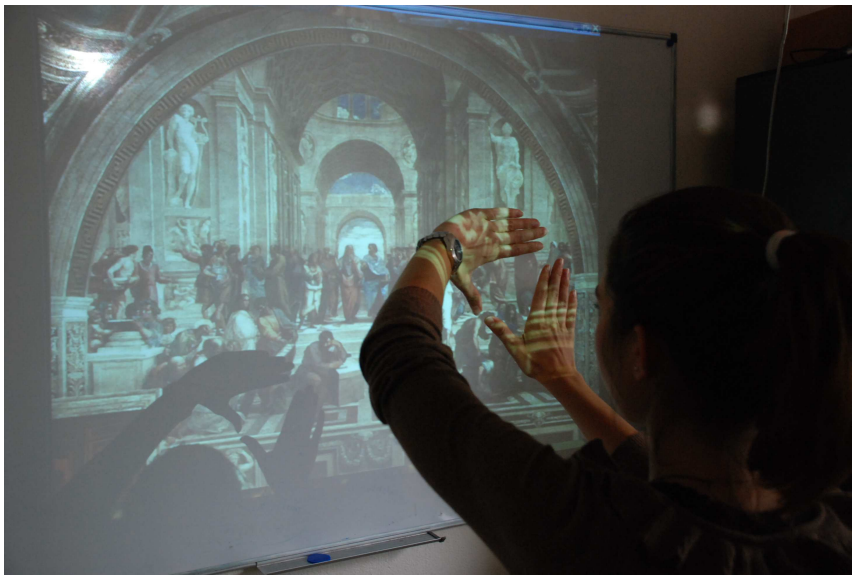


FIGURE 8.3 – Sujet réalisant un geste bi-manuel de zoom sur le diaporama multimodal, modalité simulée par l'un des compères.

8.2.2 Profil des sujets

L'évaluation a impliqué deux groupes de sujets : les utilisateurs et les compères. Le groupe d'utilisateurs est composé de cinq personnes de profils hétérogènes : un étudiant, un responsable de communication d'un lieu culturel, un graphiste, un doctorant et une enseignante en collège (Tableau 8.1). Les sujets n'étaient pas au

courant de la nature simulée de l'interface, ni de l'observation par les compères de leurs actions et paroles.

Nombre	Moyenne d'âge	Sexe	Fonctions
5	23	2 femmes, 3 hommes	1 étudiant, 1 responsable de communication, 1 graphiste, 1 doctorant, 1 enseignante en collège

TABLE 8.1 – Profil des sujets utilisateurs

Le groupe de compères est composé de quatre personnes : un doctorant, un post-doctorant, un ingénieur de recherche et un maître de conférences (Tableau 8.2). Les compères ont déjà participé à l'évaluation empirique de notre approche conceptuelle (section 5.3 du chapitre 5) et connaissent donc le modèle conceptuel.

Nombre	Moyenne d'âge	Sexe	Fonctions
4	23	4 hommes	1 doctorant, 1 post-doctorant, 1 ingénieur de recherche, 1 maître de conférences

TABLE 8.2 – Profil des compères

8.2.3 Protocole

Nous avons utilisé deux méthodes complémentaires des sciences humaines et sociales (SHS) afin de réaliser l'expérience : le **focus group** et le **cultural probe**. Le *focus group* [Krueger et Casey, 2000] est une méthode qualitative qui consiste à discuter avec un groupe de personnes séléctionnées. Le *cultural probe* [Bernhaupt *et al.*, 2008] consiste à évaluer un produit dans son contexte d'utilisation sans la participation d'évaluateurs afin de ne pas influencer l'expérience. Nous recourons aux *work probes* (cultural probes dans un contexte de travail) afin de mettre les compères dans une situation réelle de simulation. Une fois les work probes réalisés, nous analysons leur déroulement en focus group. L'expérience s'est donc déroulée en deux parties : une expérience magicien d'oz en executant le prototype simulé (work probe) et un focus group.

Le work probe a été réalisé par les compères en binômes. Les binômes ont simulé une interaction multimodale en situation réelle. La configuration de cette séance est illustrée à la Figure 8.4. Le sujet se trouvait dans la salle d'expérience et les deux compères se trouvaient dans une autre salle. Afin de pouvoir observer les actions de l'utilisateur, les compères disposaient d'un retour audiovisuel grâce aux caméras et microphones installés dans la salle d'expérience. Chaque compère simulait une modalité différente : le geste et la voix.

Ensuite, un focus group avec tous les compères a permis d'analyser le work probe. Les compères remplissent également une grille points forts/points faibles ainsi que des post-it avec des propositions d'amélioration de l'approche. Les ex-

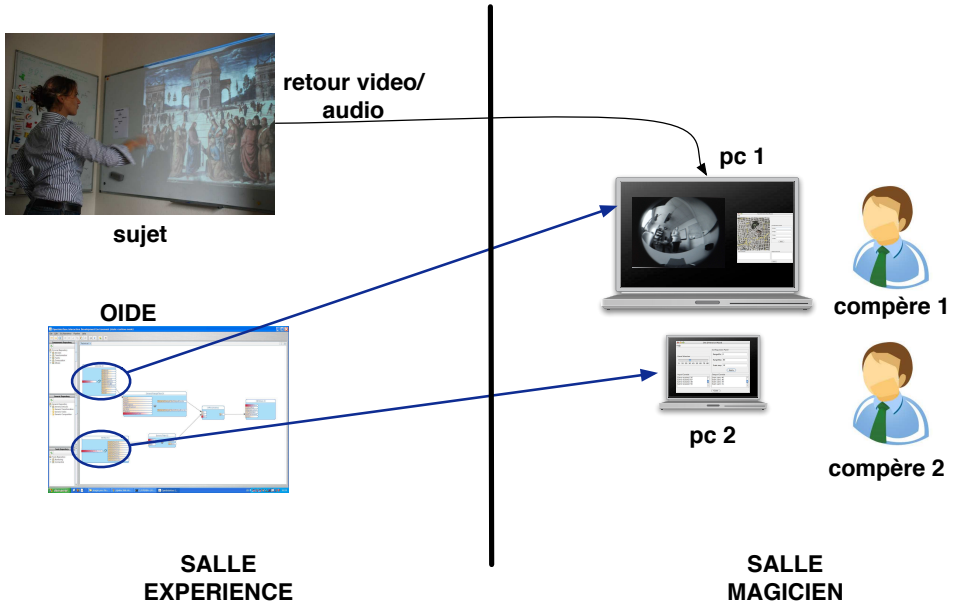


FIGURE 8.4 – Architecture de l'expérience.

périences Magicien d'Oz duraient en moyenne 15 minutes et le focus group mené juste après l'expérience a duré en moyenne 45 minutes.

8.2.4 Résultats de l'expérience de simulation

Les données produites par l'expérience sont les suivantes :

- Enregistrement audio et vidéo des compères pendant la simulation ;
- Grilles points forts-points faibles (compères) ;
- Questionnaire d'utilisabilité (sujets).

Évaluation de la fidélité de la simulation

Hypothèse Notre hypothèse est que la simulation de l'interaction avec notre outil est assez fidèle vis-à-vis de l'utilisateur pour que celui-ci pense interagir avec un système fonctionnel.

Exercice Les sujets sont invités à interagir avec un système interactif sans être avertis du caractère simulé du système. A la fin de l'expérience nous dévoilons la nature simulée de celle-ci et nous demandons aux sujets le degré de crédibilité du système simulé.

Résultats Tous les utilisateurs sauf un affirment avoir cru interagir avec un système fonctionnel même si le type d'interaction, comme le geste, était nouveau pour eux. Seul le sujet doctorant en informatique questionne l'authenticité de l'interaction. Les compères ont par contre pensé que la latence du système, c'est-à-dire le temps entre le lancement d'une action de simulation et la réalisation effective côté

utlisateur, était un peu élevée. Même si cette latence n'empêche pas de réaliser un prototype simulé crédible, c'est un frein à la simulation de certains types d'interaction, comme la manipulation directe qui nécessite un couplage fort entrée/sortie.

Évaluation de l'approche multi-compère

Hypothèse Notre hypothèse est que l'approche multi-compère permet de mieux gérer la simulation de l'interaction multimodale.

Exercice Pour cela, notre expérience comporte deux modalités d'interaction, le geste et la voix. Chaque compère simule une modalité d'interaction différente. Les deux compères sont placés ensemble et partagent le retour audiovisuel (Figure 8.5).

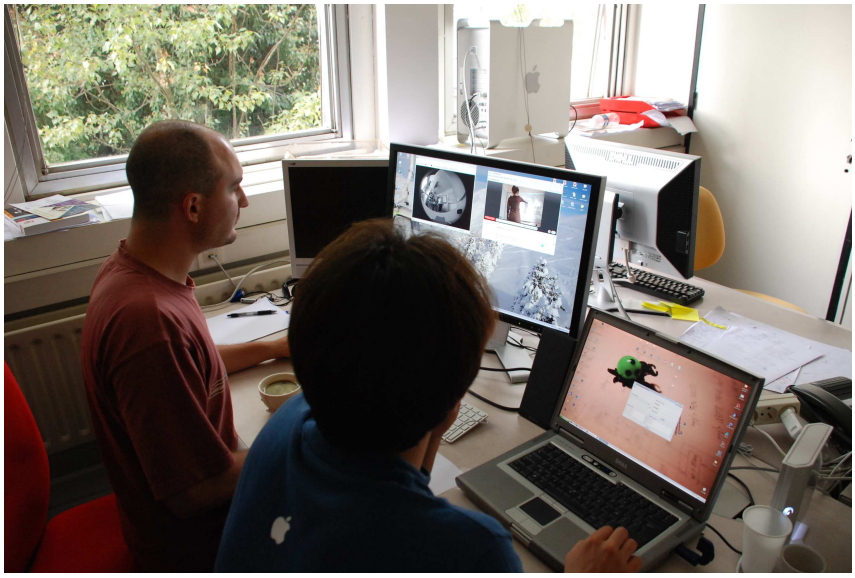


FIGURE 8.5 – Compères simulant une interaction multimodale. L'écran de gauche affiche le retour vidéo du sujet.

Résultats Les compères ont affirmé que l'approche multi-compère était positive pour la simulation de l'interaction multimodale. D'un côté, cela permettait de limiter la charge cognitive de chaque compère. De plus, la présence d'un autre compère pendant l'expérience permettait également de diminuer le stress et de s'entraider pendant la simulation. Certains compères ont souligné que la division des modalités à simuler doit être bien définie afin de ne pas avoir de conflits ou d'erreurs pendant la simulation.

8.3 Utilisation de l'outil : prototypes interactifs réalisés

Une autre forme d'évaluation d'un outil est de montrer la variété des prototypes qui peuvent être développés avec l'outil mais aussi de faire utiliser l'outil par des collègues. Le projet OpenInterface a été pour nous un terrain propice pour que notre outil soit utilisé par de nombreux utilisateurs en dehors de notre équipe. Ce

point est à souligner car il est rare que dans le temps imparti d'une thèse (3 ans) les outils développés soient utilisés à l'extérieur de l'équipe de recherche.

Dans cette section nous présentons d'abord les prototypes multimodaux développés dans le cadre du projet OpenInterface puis deux prototypes élaborés par des étudiants et enfin trois prototypes que nous avons développés pour démontrer notre outil.

8.3.1 Prototypes du projet OpenInterface

Nous introduisons d'abord l'approche de prototypage adoptée dans le projet OpenInterface (A) puis les deux domaines d'application considérés pour les prototypes : les grands espaces d'informations (B) et les jeux (C).

A) Approche de prototypage dans le projet OpenInterface

STU visés

Le projet OpenInterface considère deux domaines d'application différents : les grands espaces d'information et les jeux. Le domaine des grands espaces d'information (GEI) englobe toutes les applications où l'utilisateur doit chercher des informations dans un grand espace informationnel. Dans le projet, nous avons focalisé sur les applications de navigation sur une carte géographique. Le domaine des jeux vise avant tout les jeux multimodaux sur téléphone mobile. Ce domaine d'application est propice à l'expérimentation multimodale, comme nous l'avons montré à la section 2.4 du chapitre 2.

Ainsi, chaque domaine d'application vise un STU (Situation, Tâche, Utilisateur) différent. Les GEI visent une situation (S) de recherche d'information dans un lieu public (gare, office de tourisme) ou sur un dispositif mobile ; une tâche (T) d'interaction multimodale afin d'effectuer la recherche et ; un profil d'utilisateur (U) "grand public". Le domaine du jeu vise une situation (S) de jeu collaboratif sur support mobile, avec une tâche (T) de manipulation multimodale du jeu par les joueurs (U), public généralement initié aux nouvelles techniques d'interaction. Le tableau 8.3 synthétise les deux types de STU.

domaine	Situation (S)	Tâche (T)	Utilisateur (U)
GEI	Besoin d'information dans un lieu public ou sur support mobile	Recherche d'information	Grand public : internautes, touristes ;
Jeux	Jeux collaboratifs sur support mobile	Manipuler de façon multimodale un jeu	Utilisateurs jeux

TABLE 8.3 – STU visés par les deux domaines d'application des prototypes réalisés au sein du projet OpenInterface.

Démarche de conception

La démarche, que nous avons adoptée dans le projet OpenInterface, correspond à une conception itérative centré utilisateur qui s'appuie sur des prototypes, suivie d'une phase de développement afin de créer le système multimodal final (Figure

8.6). Ainsi les deux systèmes multimodaux finaux ont été développés sans l'OIDE sur téléphone mobile. La démarche itérative démarre par une spécification initiale des besoins puis est suivie de plusieurs itérations, chaque itération étant composée d'une phase de conception, prototypage et évaluation. Pour chaque domaine, nous avons effectué quatre itérations. Cette démarche de conception correspond au processus de conception et développement IntuiSign [Schlienger et Chatty, 2007].

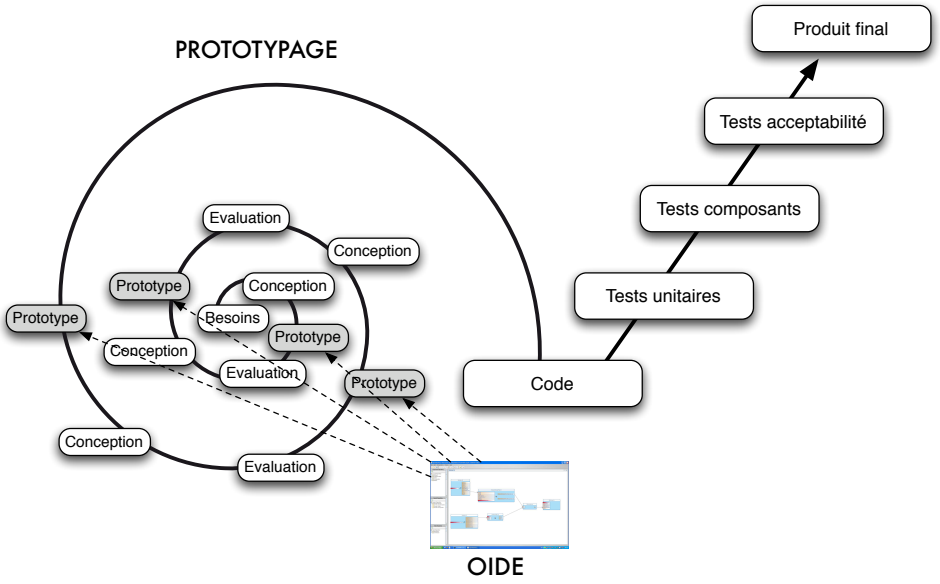


FIGURE 8.6 – Démarche itérative de prototypage adoptée dans le projet OpenInterface.

Comme illustré à la Figure 8.6, l'OIDE est utilisé à chaque itération afin de produire le prototype. Néanmoins, l'OIDE représente également une aide à la conception, comme l'a montré l'évaluation empirique de l'outil (section 8.1 du chapitre 7). Dans les sections suivantes, nous présentons les quatre versions réalisées de chaque prototype, ainsi que leur conception et leur évaluation.

B) Prototypes OpenInterface : Grands espaces d'information

Pour le cas des grands espaces d'information (GEI), nous avons travaillé sur des cartes interactives. Quatre prototypes de navigation multimodale dans une carte ont été développés et évalués.

Prototype 1 : Carte multimodale simple

Pour ce premier prototype nous travaillons sur une carte statique (une grande image fixe). L'objectif est de permettre la navigation multimodale de la carte. Afin de simplifier ce premier prototype, nous ne considérons pas les points d'intérêt (POI[1]).

L'application utilisée est donc un navigateur de cartes. La carte affichée correspond à la ville de San Francisco. Nous définissons trois tâches interactives : zoomer

[1]Points of Interest

sur un point de la carte, naviguer sur la carte et centrer l'image visible sur un point de la carte.

En ce qui concerne l'interaction, nous souhaitons valider notre approche en implémentant une interaction du type *Put-That-There* [Bolt, 1980] (présentée à la section 2.4 du chapitre 2). Ainsi, les deux techniques d'interaction utilisées sont le geste et la voix (Figure 8.7). Pour capter le geste, nous utilisons la Diamond Touch [Dietz et Leigh, 2001] et pour la reconnaissance vocale nous utilisons un composant développé par l'un des partenaires du projet, Multitel.

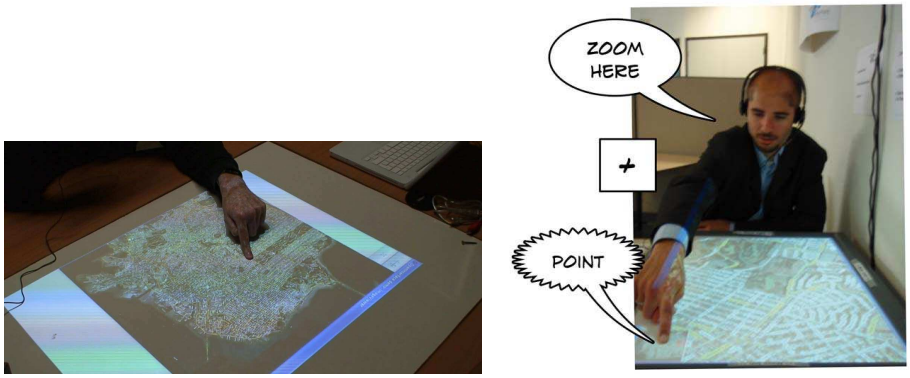


FIGURE 8.7 – Interaction multimodale dans le premier prototype GEI : un navigateur de cartes.

L'assemblage de composants inclut quatre composants : un composant DiamondTouch, un composant Reconnaissance Vocale, un composant de composition et enfin le composant qui représente l'application multimodale (Figure 8.8). Dans cet exemple, aucun composant de transformation n'est utilisé. En effet, la carte accepte des coordonnées dans le repère écran ce qui permet d'utiliser directement le point fourni par la table tactile, et le composant vocal fournit également les commandes attendues par l'application.

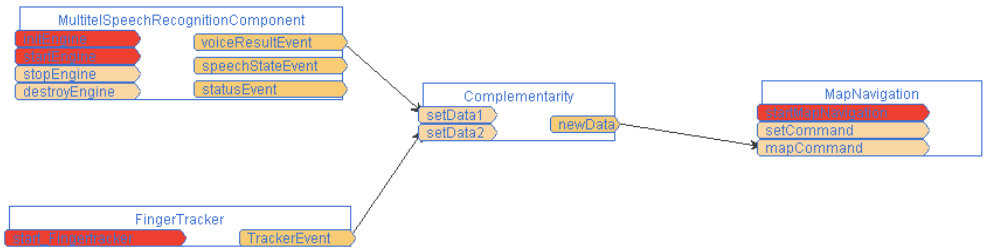


FIGURE 8.8 – Assemblage de composants du premier prototype GEI.

Une évaluation de ce prototype a été menée avec des étudiants de l'Université de Grenoble. Le groupe d'évaluation était composé de 6 sujets (1 homme, 5 femmes), étudiants en informatique, entre 19 et 25 ans. Les tâches demandées aux sujets consistaient à chercher des lieux sur une carte de San Francisco. Les sujets devaient réaliser les commandes vocales en anglais en raison du composant de reconnaissance utilisé.

Cette première évaluation n'a pas fourni de résultats remarquables. En effet la modalité parole a été peu utilisée car les utilisateurs étaient Français et devaient parler en Anglais : le taux de reconnaissance était alors faible. Aussi nous avons observé un phénomène classique en multimodalité de spécialisation d'une modalité pour compenser une autre fonctionnant mal. Le seul sujet qui avait un bon accent en Anglais a utilisé indifféremment les deux modalités.

Prototype 2 : Carte multimodale sur un serveur

La deuxième version du prototype utilise comme application un serveur de cartes s'exécutant à distance (France Télécom) (Figure 8.9), au lieu de l'image statique de la version 1.

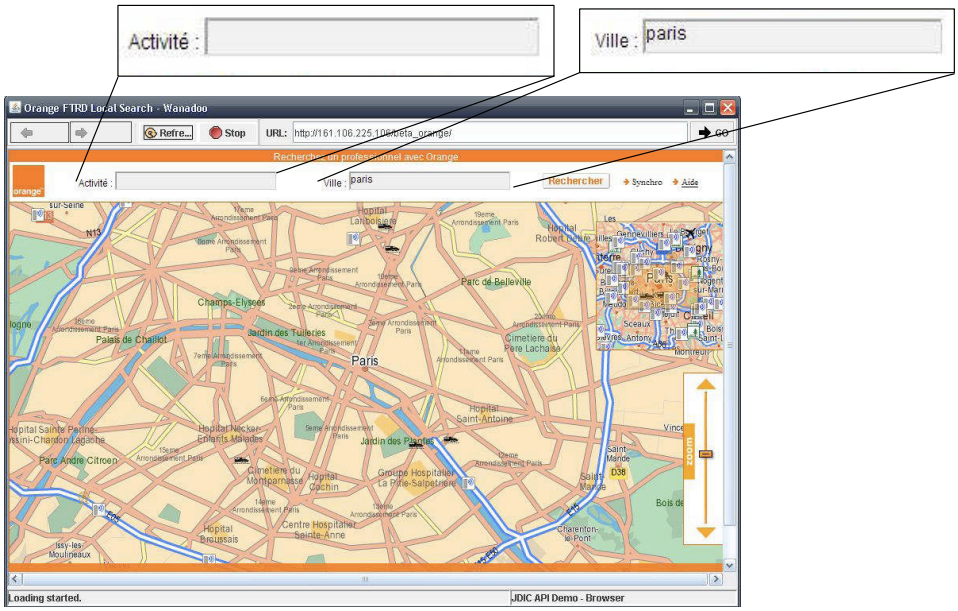


FIGURE 8.9 – Interface graphique de la version 2 de la carte multimodale avec détail des champs de recherche.

Ce prototype focalise non pas sur la complémentarité des modalités mais sur la connexion d'un assemblage à un serveur distant et l'utilisation d'une modalité d'interaction basée sur le geste 3D pour naviguer. Ainsi, dans cette deuxième version,le toucher est remplacé par le geste 3D avec le Shake (dispositif au centre de la Figure 8.10). Ainsi, l'utilisateur peut utiliser le geste pour se déplacer sur la carte (haut/bas/droite/gauche). La reconnaissance vocale en français a été également intégrée suite aux résultats de l'évaluation de la première version du prototype. L'utilisateur peut spécifier avec la voix le contenu des champs de recherche de la carte (Figure 8.9) : Activité et Ville. L'utilisateur a le choix entre trois villes (*Paris, Brest et Monaco*) et entre trois activités différentes (*restaurant, taxi et cinéma*). Les champs sont automatiquement remplis par les valeurs énoncées par l'utilisateur, qui doit ensuite cliquer sur Rechercher pour lancer le moteur de recherche du serveur de cartes.

L'assemblage de composants utilisé est illustré sur la Figure 8.11. Cet assemblage est constitué de trois composants Dispositif : le composant Shake, le com-

FIGURE 8.10 – Modalités d'interaction utilisées dans la version 2 de la carte multimodale.

posant de reconnaissance vocale (*MultitelSpeechRecognitionSystem*) et le composant Clavier. Le Clavier est utilisé de façon redondante au Shake pour se déplacer sur la carte avec les flèches haut/bas/droite/gauche. Le composant Dispositif Shake est connecté à un composant de Transformation dépendant du Shake (*ShakeDirection*), qui transforme les événements de l'accéléromètre du Shake en commandes de direction (haut/bas/droite/gauche). Finalement, le composant Tâche (*Browser*) reçoit de façon équivalente les commandes du Shake et du Clavier pour le déplacement, et celles de la reconnaissance vocale pour les POI. Nous observons que l'équivalence entre le Shake et le Clavier n'est pas exprimée ici par un composant de composition, mais par une connexion redondante au composant Tâche.

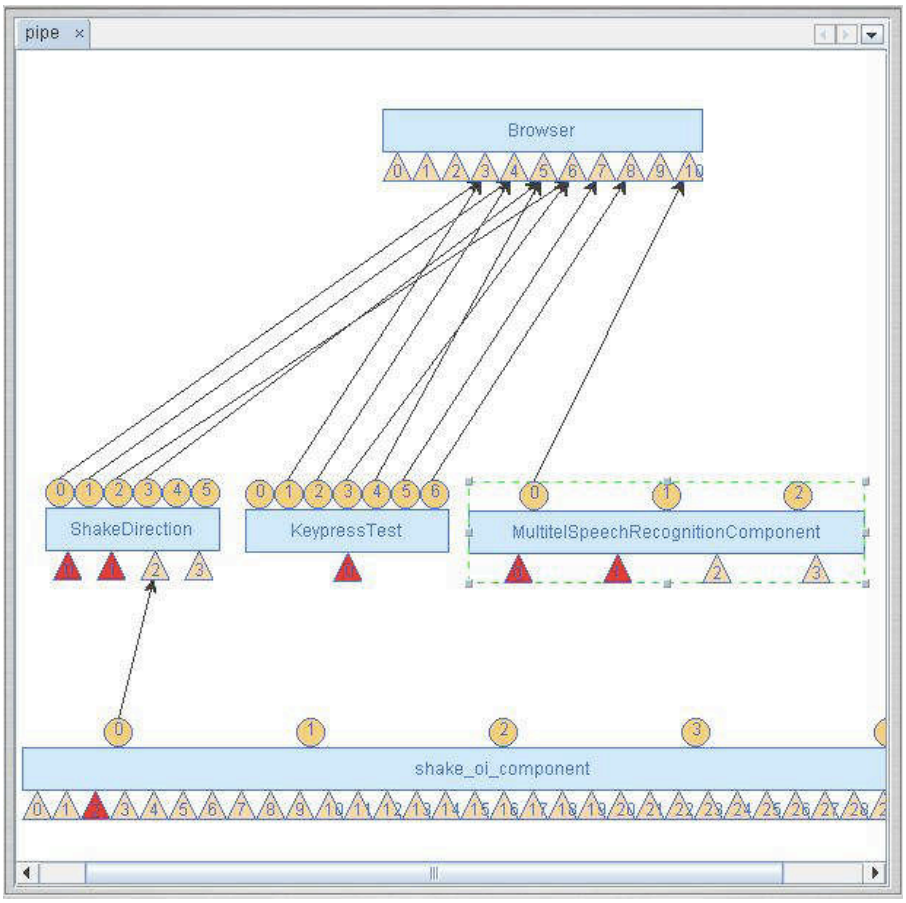


FIGURE 8.11 – Assemblage de composants du deuxième prototype GEI.

Lors de l'évaluation de cette version, les sujets devaient réaliser une série de tâches simples de recherche, comme "dans quelle rue se trouve le restaurant ?". Les résultats de l'évaluation montrent que les sujets ont eu des difficultés à utiliser le Shake pour naviguer en raison de sa haute sensibilité. De plus, malgré ce manque d'utilisabilité, les sujets ont exprimé le souhait de disposer de plus de liberté avec le Shake, par exemple pour zoomer. La reconnaissance vocale a été largement moins utilisée que le geste en raison principalement du manque de familiarité des sujets avec cette modalité d'interaction.

Prototype 3 : Découvrir Paris

Dans la troisième version, nous considérons l'aspect collaboratif pour la tâche de rechercher un lieu sur une carte. L'objectif du prototype est d'évaluer comment l'interaction multimodale influence ce type de tâche et d'améliorer les problèmes d'utilisabilité identifiés dans la version précédente.

L'interaction multimodale repose toujours sur le geste (capté grâce au Shake) avec la voix. L'inclinaison du Shake vers les côtés permet de se déplacer dans la carte et la voix de préciser les lieux et activités qui définissent les POI. L'assemblage de composants de la troisième version du prototype (Figure 8.12) est donc assez semblable à celui de la deuxième version, à exception du composant Clavier, absent de cette version. Il est constitué de deux composants Dispositif, le composant Shake et le composant MultitelSpeechRecognition. Comme pour la version précédente, le composant Shake est connecté à un composant de Transformation afin de générer des commandes de déplacement à partir de l'inclinaison. Ce composant de Transformation a été paramétré afin de rendre plus utilisable le Shake.

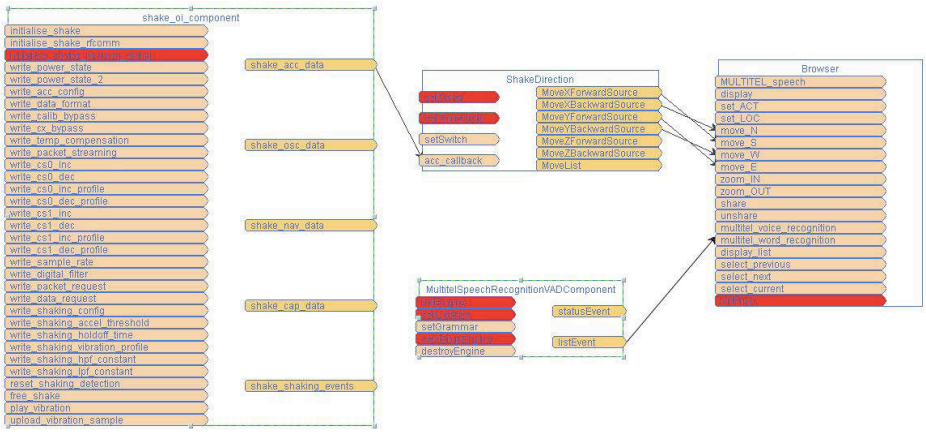


FIGURE 8.12 – Assemblage de composants du troisième prototype GEI.

L'évaluation réalisée avec ce prototype consiste à comparer la performance et l'expérience d'interaction pour des tâches de planification de parcours dans les rues de Paris. Les utilisateurs avaient un temps limité pour visiter un certain nombre d'endroits et ils devaient, en binômes, se mettre d'accord sur le parcours à emprunter et sur les modalités à utiliser. La performance de cette version est comparée à celle précédente.

Le résultat montre que le Shake présente toujours des problèmes d'utilisabilité : l'absence d'une commande de zoom avec ce dispositif oblige les utilisateurs à poser le Shake afin de réaliser le zoom avec les boutons de l'interface graphique de la carte. De plus, la reconnaissance vocale, comme pour la version précédente, est toujours peu exploitée.

Le résultat montre néanmoins que les modalités utilisées dans ce prototype favorisent la collaboration : par exemple, les sujets s'échangent souvent le Shake afin de naviguer dans la carte. L'évaluation démontre aussi qu'il faut avoir des dispositifs d'affichage adaptés aux activités collaboratives : en effet, un écran d'ordinateur n'est pas un bon support de collaboration. Dans le prototype suivant nous étudions d'autres supports d'affichage ainsi que des nouvelles modalités d'interaction.

Prototype 4 : Découvrir Paris II

Dans cette quatrième version, la carte est projetée afin d'etudier l'influence sur la collaboration entre les utilisateurs. Ainsi, la carte est projetée sur un mur et affichée sur un écran plat posé sur une table (à gauche sur la Figure 8.13). Nous utilisons une interaction multimodale basée sur le geste, en ajoutant une nouvelle modalité pour capturer le geste : une caméra infrarouge permet de capter les gestes des doigt sur l'écran plat (à droite sur la Figure 8.13). L'utilisateur peut ainsi scroller la carte en utilisant ses doigts. Le Shake n'est plus utilisée que pour activer la reconnaissance des gestes : l'utilisateur appuie le bouton du Shake pour indiquer au système de démarrer la capture des gestes par caméra.

FIGURE 8.13 – Carte projetée sur le mur et affichée sur un écran plat (à gauche) et détail sur le dispositif de capture de la position du doigt, formé de deux caméras trackIR (à droite) dans la version 4 du prototype multimodal GEI.

L'assemblage de cette version (Figure 8.14) contient deux composants Dispositif : le composant Shake et le composant FingerTrackerIR (la caméra ingrarouge). Un composant ad hoc de Composition (MiddleComponent) fait la fusion entre les données du bouton du Shake et les données de la caméra afin de générer des événements gestuels composés. Ces événements sont ensuite traités par un nouveau composant de Transformation (GestureRecognizer) qui génère des commandes de navigation à partir des événements gestuels composés du bouton du Shake et de la reconnaissance réalisée par la caméra.

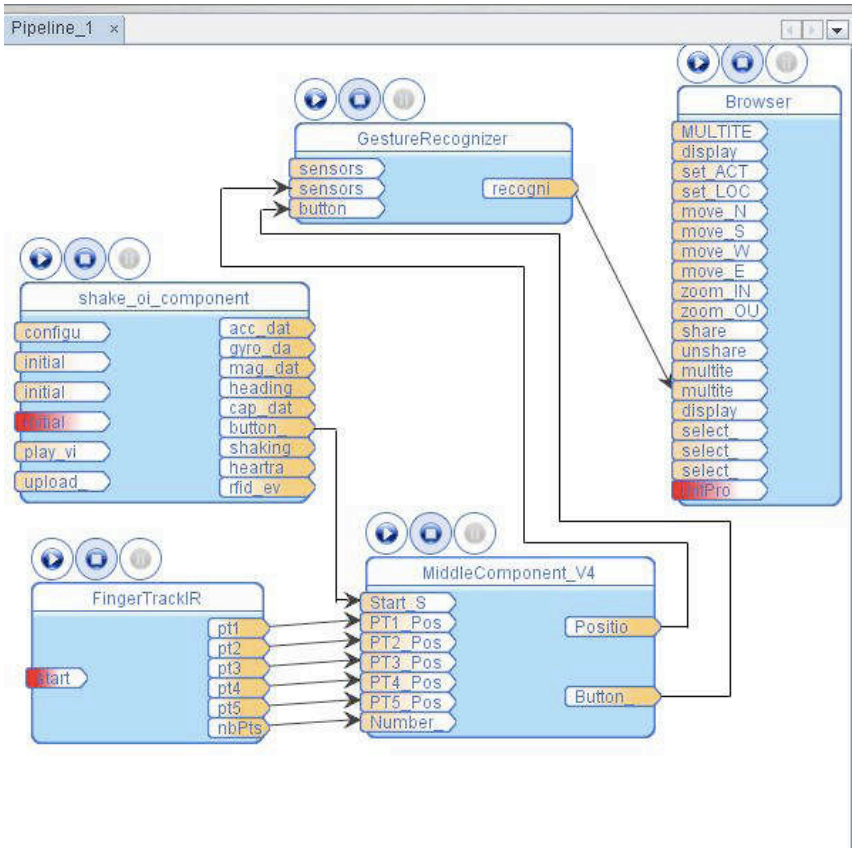


FIGURE 8.14 – Assemblage de composants du quatrième prototype GEI.

L'évaluation de ce prototype reprend les mêmes tâches que celles de l'évaluation précédente : des binômes de sujets doivent concevoir des parcours sur une carte de Paris. Les résultats montrent que l'affichage sur grand écran favorise l'utilisation des modalités gestuelles, par rapport à la version précédente, où la souris et le clavier étaient d'avantage utilisés. Les utilisateurs ont affirmé préférer les modalités innovantes sur les modalités traditionnelles, malgré le manque de robustesse de certaines modalités (en particulier de la reconnaissance de gestes par caméra).

Version finale : Free Trip

Une version finale de l'application GEI a été implémentée entièrement sur un téléphone portable, le Nokia N95, et utilise l'accéleromètre et le GPS internes comme dispositifs d'interaction (Figure 8.15). Cette version finale n'a pas été développée avec l'OIDE et a suivi un cycle de développement classique (partie droite de la Figure 8.6) à partir des spécifications établies après quatre itérations.

C) Prototypes OpenInterface : Jeux

Le deuxième domaine d'application considéré dans le projet OpenInterface est dédié aux jeux, principalement sur dispositif mobile.

FIGURE 8.15 – Version finale de validation du GEI sur téléphone Nokia N95.

Prototype 1 : Jeu 3D de vol

Dans la première version du prototype, nous utilisons comme application un jeu 3D de vol sur PC (Figure 8.16). Dans ce jeu, l'utilisateur manipule un hélicoptère qui avance et doit éviter des obstacles en se déplaçant latéralement ou verticalement. Ce premier prototype, réalisé dans une première itération très rapide, nous a surtout permis de tester l'OIDE à travers un exemple simple, l'utilisation du jeu sur téléphone n'étant pas encore disponible. Aussi aucune évaluation contrôlée de ce premier prototype n'a été menée.

FIGURE 8.16 – Interface graphique du jeu 3D de vol.

Les dispositifs utilisés dans cette première version du jeu sont le Shake, le TrackIR et la plate-forme de capteurs physiques Interface-Z. Le jeu peut être manipulé avec des gestes en utilisant le Shake ou avec Interface-Z à l'aide de capteurs pluggés sur un volant en jouet (Figure 8.17). Le TrackIR est utilisé afin de modifier la vue du joueur (la vue du pilote de l'hélicoptère).

La Figure 8.18 illustre l'assemblage de composants utilisé dans cette première version du jeu. Dans cet assemblage, deux composants Dispositif correspondant à l'Interface-Z et au TrackIR sont connectés au composant du jeu (à droite). Alors

FIGURE 8.17 – Contrôle du jeu de vol avec le geste en utilisant le Shake (à gauche) et en utilisant les capteurs interface-Z pluggés sur un volant de jouet en plastique (à droite).

que le composant TrackIR est directement connecté au port CamControl du jeu qui permet de contrôler la vue du joueur, le composant Interface-Z est connecté à un composant de Transformation, qui envoie les événements modifiés au port HeliControl, afin de contrôler la position de l'hélicoptère.

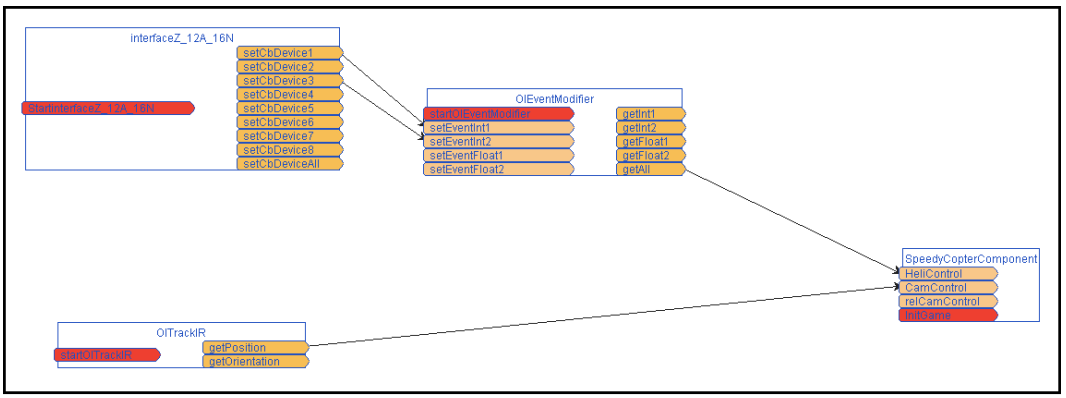


FIGURE 8.18 – Assemblage de composants dans la version 1 du prototype Jeu.

Prototype 2 : Jeu 3D sur téléphone mobile

Dans la deuxième version du prototype, nous utilisons un jeu sur support mobile. Ainsi, l'objectif est de tester de façon prospective des modalités sur l'ordinateur afin de contrôler le jeu sur téléphone mobile. Cela permet en autre d'utiliser des modalités pas encore existantes sur téléphone et d'évaluer leur pertinence dans le cadre des dispositifs mobiles.

Le jeu utilisé est un jeu en 3D où un personnage se déplace sur un plan (au centre sur la Figure 8.19). La tâche interactive dans cette version est le déplacement du personnage sur le plan. Afin de connecter l'OIDE aux tâches interactives sur le téléphone, nous utilisons un composant Bluetooth qui envoie les informations du niveau Tâche vers le téléphone.

Plusieurs modalités sont testées afin de contrôler le personnage : le déplacement de la souris, un composant de suivi de la tête (HeadTracking), avec lequel l'utilisateur peut déplacer le personnage en bougeant la tête face à la webcam de

FIGURE 8.19 – Interface graphique du jeu 3D sur téléphone mobile.

l'ordinateur (à droite sur la Figure 8.20), la voix, le Shake et un composant ARToolkit qui permet de détecter le mouvement du téléphone par rapport à un patch (à gauche sur la Figure 8.20).

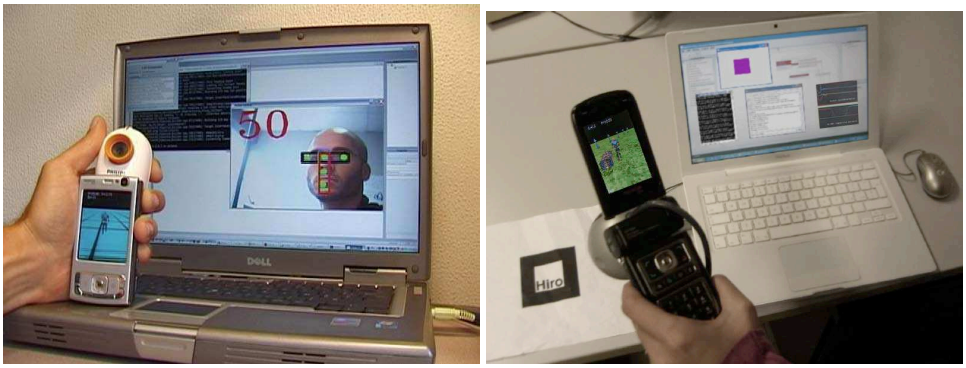


FIGURE 8.20 – Jeu 3D sur téléphone mobile contrôlé avec le déplacement de la tête (à gauche) et avec le déplacement du téléphone en utilisant Artoolkit (à droite).

La Figure 8.21 illustre l'assemblage utilisé dans cette version du prototype, réalisé avec une des premières versions de l'OIDE. Cette assemblage contient cinq composants Dispositif : de gauche à droite, un composant de reconnaissance vocale (MultitelSpeechRecognition), un composant Clavier (Win32Keyboard), le composant ARToolkit, le composant Shake et le composant Souris (Win32Mouse). Tous ces composants, utilisés de façon équivalente pour contrôler le personnage, sont connectés à un seul composant de Transformation, EventAdapter. Le composant EventAdapter a un comportement dépendant des dispositifs en entrée et des interfaces ad hoc équivalentes aux interfaces ad hoc des différents dispositifs. Finalement le composant Tâche envoie les commandes de déplacement du personnage au jeu via Bluetooth.

L'objectif a été de montrer que de nombreuses modalités pourraient être rapidement connectées à des tâches classiques d'un jeu première personne, de déplacement d'un personnage. Nous avons aussi mené une étude sur la connexion bluetooth de l'OIDE vers l'application jeu executée sur le téléphone mobile. Adop-

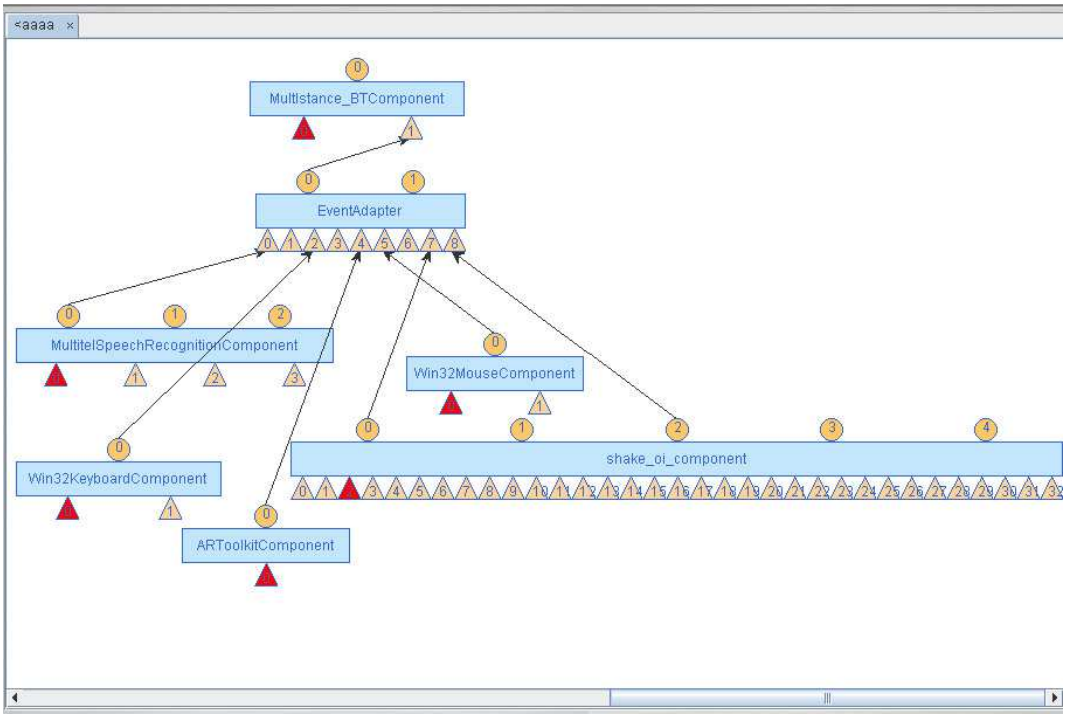


FIGURE 8.21 – Assemblage de composants dans la version 2 du prototype Jeu.

tant un niveau d'abstraction intermédiaire pour les informations échangées selon le protocole défini, la solution conçue consiste à envoyer des tâches génériques du type HAUT, BAS, DROITE, GAUCHE à l'application. Aussi un autre jeu du type première personne peut être immédiatement connecté à l'OIDE.

Prototype 3 : Jeu 3D *Ant's Life*

La troisième version du prototype utilise un jeu sur téléphone mobile, *Ant's Life*, où l'utilisateur prend la place d'une fourmi qui doit combattre d'autres fourmis afin de gagner des graines (Figure 8.22). Les affrontements prennent la forme de combats selon le jeu de pierre-papier-ciseaux.

Afin de sélectionner une des trois options, pierre, papier ou ciseaux, les joueurs disposent de plusieurs modalités d'interaction gestuelles et de la reconnaissance vocale (Figure 8.23). Les modalités gestuelles sont réalisées avec le Shake ou avec un dispositif gant intégré de capteurs, le 5DT Data Glove, capable de capter les gestes de tous les doigts .

La Figure 8.24 montre l'assemblage qui implémente l'interaction avec le Shake et la reconnaissance vocale. Cet assemblage comporte deux composants Dispositif, le composant Shake et le composant de reconnaissance vocale (MultitelSpeech). Les données de l'accéléromètre du Shake sont traités par un premier composant de Transformation (GestureRecognizer), qui génère des commandes selon les gestes effectués avec le Shake. Ensuite, un autre composant de Transformation (Game-v3-Adapter) transforme ces commandes gestuelles génériques en commandes relatives au jeu. Un dernier composant de Transformation (GameComponent) transforme les données du jeu en données compréhensibles par le composant Bluetooth.

FIGURE 8.22 – Screenshots du jeu sur téléphone mobile.

FIGURE 8.23 – Modalités d'interaction utilisées : geste avec un gant, reconnaissance vocale et geste avec le Shake.

Nous observons aussi que cet assemblage contient deux modalités en sortie, matérialisées par le composant AudioComponent (modalité sonore) et par la connexion entre le composant GameComponent et le port *play-vib* du composant Shake (modalité de vibration). Dans l'assemblage utilisé à l'évaluation, le composant Shake sera remplacé par le composant du gant 5DT Data Glove.

L'évaluation de cette version du jeu consistait à faire s'affronter deux sujets. L'évaluation a porté surtout sur l'utilisation de la modalité du gant et son intégration dans le jeu. Ainsi, les résultats de l'évaluation montrent que le jeu ne fournissait pas assez de retour sur l'interaction gestuelle avec le gant. Étant donné que les sujets n'étaient pas habitués à ce genre de dispositifs, cette absence de retour rendait *Ant's Life* moins jouable.

Prototype 4 : Jeu 3D *Ant's Life* II

Dans cette quatrième version du jeu, nous avons rajouté le dispositif GPS, ce qui permet d'utiliser la position de l'utilisateur comme modalité d'interaction. Ainsi, le joueur peut bouger son personnage en se déplaçant lui-même, afin de retrouver des fourmis et les affronter pour gagner des graines (Figure 8.25).

La Figure 8.26 illustre l'assemblage qui implémente l'interaction de cette version du jeu, réutilisant la plupart des composants de la version précédente et ajoutant un composant GPS.

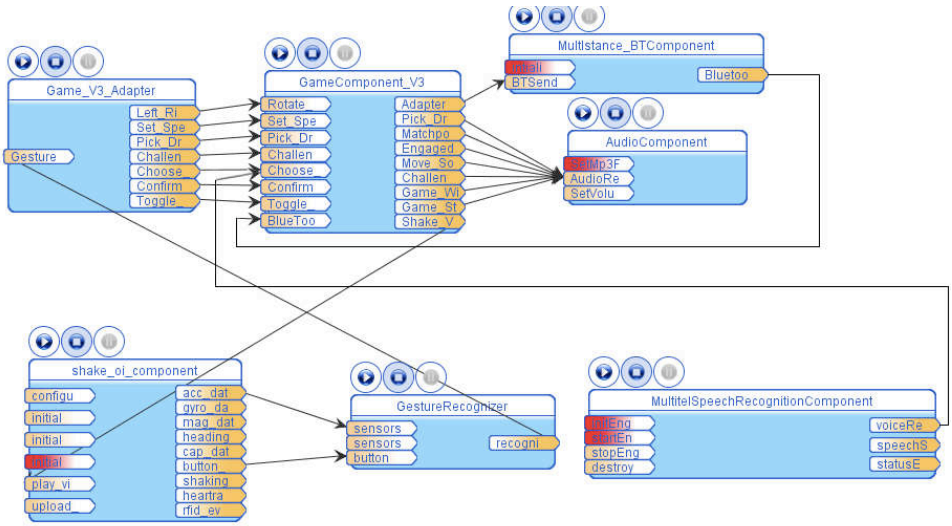


FIGURE 8.24 – Assemblage de composants pour la version 3 du jeu multimodal.

FIGURE 8.25 – Prises d'écran du jeu multimodal *Ant's Life*, où le joueur incarne une fourmi qui doit affronter des fourmis ennemies afin de gagner des graines.

L'évaluation s'est faite en demandant aux utilisateurs de porter un ordinateur dans un sac à dos, sur lequel s'exécutait l'OIDE, afin d'évaluer en particulier l'utilisation du GPS (Figure 8.27). L'évaluation a porté sur un groupe de six sujets, jouant au jeu pendant des séances de 30 minutes. Les résultats de l'évaluation montrent que l'utilisation du GPS était trop sensible parfois, rendant le contrôle du jeu assez difficile. L'utilisation de gestes a été bien perçu par les utilisateurs, qui ont trouvé les gestes assez naturels. Ces gestes ont d'ailleurs été adaptés par les utilisateurs en les rendant plus courts, ce qui augmentait la fiabilité de la détection.

Version finale

La démarche de prototypage effectuée a permis de créer un jeu 3D multimodal, multi-joueur, temps-réel et pervasif sur téléphone mobile. La version finale de validation du jeu utilise un nokia N95 et les dispositifs intégrés dans ce téléphone ainsi que le Shake. L'interaction finale est basée sur la localisation de la position de l'utilisateur avec le GPS du Nokia, sur l'utilisation des boutons avec une main ainsi que sur l'utilisation de gestes 3D avec l'autre main qui tient le Shake. Cette version a été développée sans l'OIDE selon un cycle de développement classique (partie droite

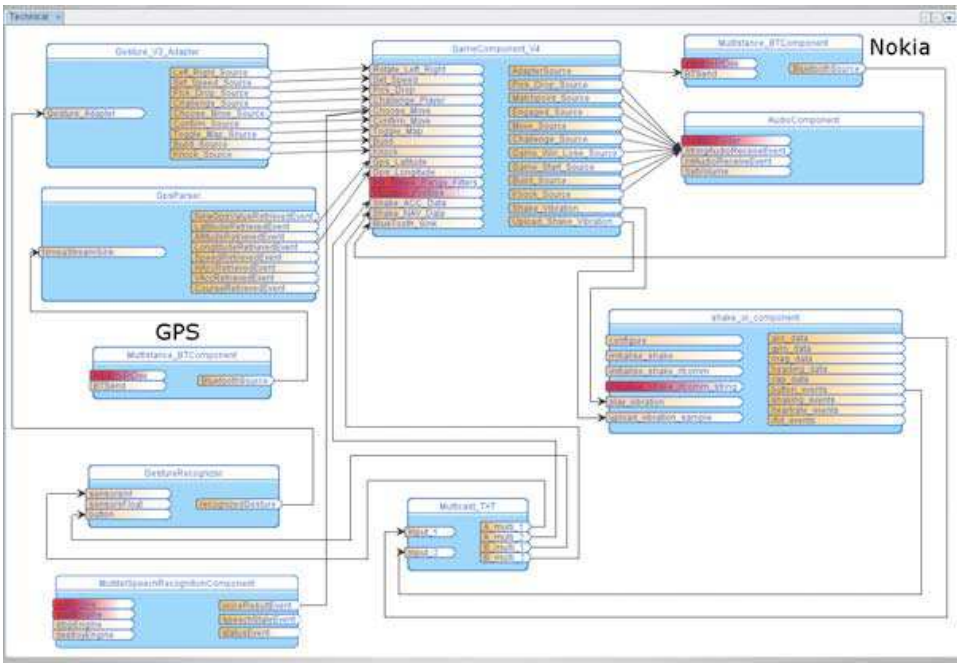


FIGURE 8.26 – Assemblage de composants pour la version 4 du jeu multimodal.

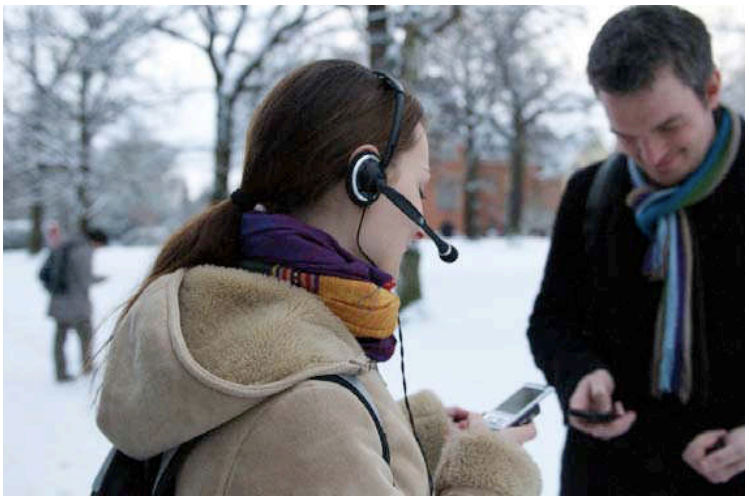


FIGURE 8.27 – Evaluation de la version 4 du jeu Ant's Life.

de la Figure 8.6).

8.3.2 Projets d'études

Plusieurs étudiants de Master ont utilisé l'OIDE afin d'implémenter des nouvelles techniques d'interaction. À la différence de la section précédente, où l'OIDE est utilisé afin de pouvoir réaliser des itérations rapides dans une démarche itérative de conception centrée utilisateur, ici l'OIDE est utilisé en tant que plate-forme de test. En effet, l'utilisation de l'OIDE permet aux concepteurs de se concentrer sur l'implémentation de la technique d'interaction, en autorisant ensuite le test rapide sur des applications déjà existantes ou la combinaison avec d'autres modalités d'interaction. Nous présentons dans cette section deux projets réalisés par des étudiants de Master 2 à l'Université Joseph Fourier (Grenoble).

Navigation d'une carte avec des gestes sur iPhone

Le premier projet, réalisé par Seif Eddine Mouelhi, avait pour objectif d'étudier l'utilisation du iPhone pour la navigation de cartes projetées sur une grande surface. Ainsi, le iPhone n'est étudié que comme dispositif en entrée et non pas en tant que dispositif en sortie (la carte n'étant pas affichée sur l'écran du iPhone). Plusieurs interactions avec le iPhone ont été implémentées, utilisant les différentes capacités d'interaction du dispositif : le geste avec l'accéléromètre et le toucher sur l'écran tactile. Ainsi, l'utilisation du geste avec l'accéléromètre permet de déplacer la carte de Google Earth dans le même sens que l'inclinaison du iPhone (Figure 8.28).

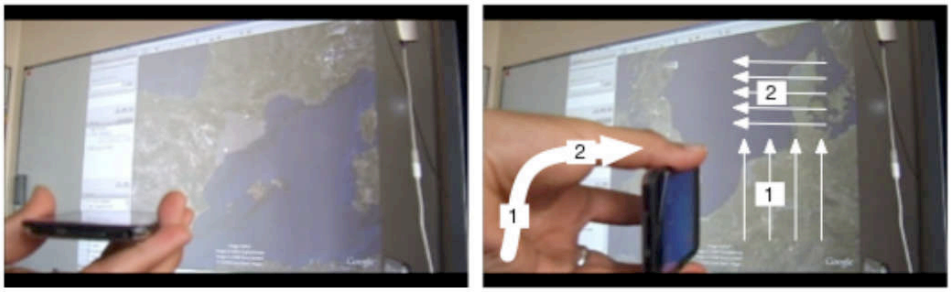


FIGURE 8.28 – Navigation d'une carte en utilisant le geste avec le iPhone.

En ce qui concerne le toucher, trois modalités d'interaction ont été implémentées. La première permet de contrôler la carte avec une télécommande tactile, où l'utilisateur peut choisir le sens de déplacement grâce à quatre boutons Haut/Bas/Gauche/Droite. La deuxième permet de déplacer la carte de façon absolue en utilisant la position de deux doigts sur l'écran de l'iPhone. La troisième permet de déplacer la carte de façon relative, en utilisant la position d'un seul doigt sur l'écran de l'iPhone (Figure 8.29).

Afin de réaliser ces techniques d'interaction, l'étudiant a utilisé un composant OISocket qui communique avec le iPhone et une série de composants de traitement implémentés afin d'analyser et transformer les événements de l'accéléromètre et du toucher sur l'écran. Pour les événements de l'accéléromètre il a également pu

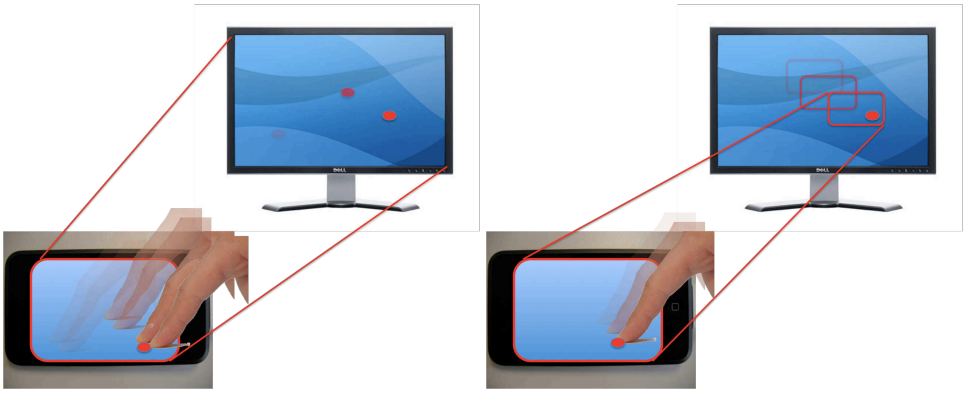


FIGURE 8.29 – Déplacement absolu (à gauche) et relatif (à droite) d'une carte en utilisant le toucher sur l'iPhone.

utiliser les composants de transformation que nous avions déjà utilisés avec le dispositif Shake, en particulier le GenericRangeFilter, capable de générer différentes commandes selon la valeur des trois paramètres en entrée (les x, y et z de l'accéléromètre).

Reconnaissance de gestes

Le deuxième projet, réalisé par Christophe Amman, avait pour objectif d'étudier l'utilisation de gestes pour la navigation de très grandes listes de données. Afin de pouvoir tester l'implication de différents dispositifs dans ces gestes, l'étudiant a implémenté l'algorithme de reconnaissance de gestes comme un composant de transformation au sein de l'OIDE. En entrée, le composant prend des valeurs en 2D pour ensuite générer, en sortie, le geste identifié.

Le composant implémente la reconnaissance en deux phases : une phase d'apprentissage puis une phase de reconnaissance. Dans la phase d'apprentissage, l'utilisateur doit réaliser les gestes afin d'entraîner le système (comme illustré sur la partie supérieure de la Figure 8.30). Le composant Transformation inclut une interface graphique qui permet de spécifier le nom de la commande associée au geste qui vient d'être effectué. Ensuite, dans la phase de reconnaissance, l'utilisateur peut refaire les gestes enregistrés, qui seront reconnus par l'algorithme de reconnaissance (Figure 8.30).

Ce composant de transformation a été utilisé avec différents dispositifs, comme le trackIR afin de reconnaître des gestes réalisés avec la tête (Figure 8.30) ou la wiimote afin de reconnaître des gestes réalisés avec les mains (Figure 8.31). Le composant a également été utilisé avec le jeu multimodal sur téléphone du projet OpenInterface (section 8.3.1 C), afin de contrôler le personnage 3D sur téléphone avec le geste de la Wiimote (Figure 8.31).

Ce projet d'étude a permis de bien montrer l'intérêt de notre modèle conceptuel qui repose sur la distinction entre composant Dispositif et composant Transformation pour définir une modalité. En développant un composant Transformation réutilisable par son interface normalisée, cela a permis ensuite d'exploiter plusieurs dispositifs différents, comme le schématise la Figure 8.32.

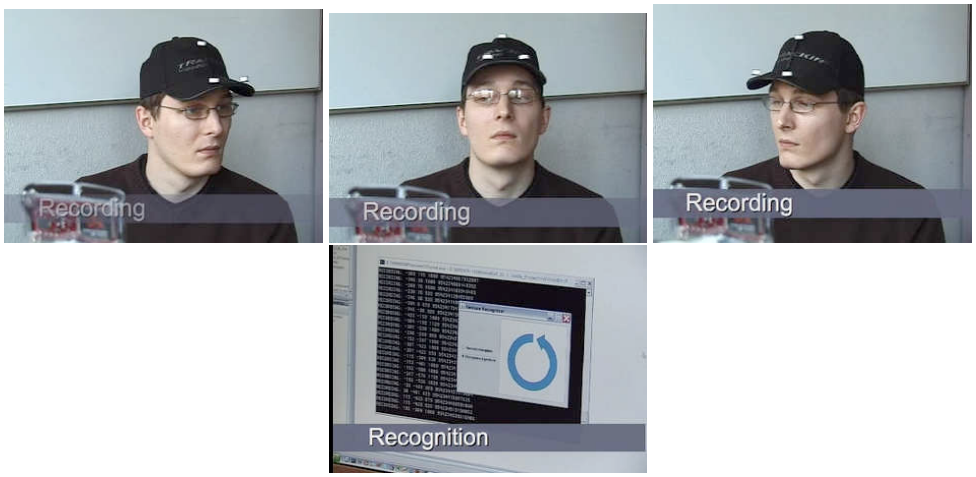


FIGURE 8.30 – Reconnaissance d'un geste de rotation de la tête en utilisant le trackIR et le composant de reconnaissance de gestes.

FIGURE 8.31 – Reconnaissance des gestes avec la wiimote afin d'interagir sur un jeu sur téléphone mobile.

8.3.3 Prototypes de démonstration

D'autres prototypes ont été implémentés au sein du projet OpenInterface de façon indépendante des deux domaines d'application. Ces prototypes ont pour objectif de tester des nouvelles modalités d'interaction ou de réaliser des démonstrations de l'OIDE.

Nous présentons trois prototypes : un prototype de manipulation 3D, un prototype de diaporama multimodal et un prototype de jeu collaboratif.

Manipulation 3D

Le prototype de manipulation 3D est issu du travail effectué avec GoogleEarth dans le domaine des grands espaces d'informations (GEI). La principale différence entre

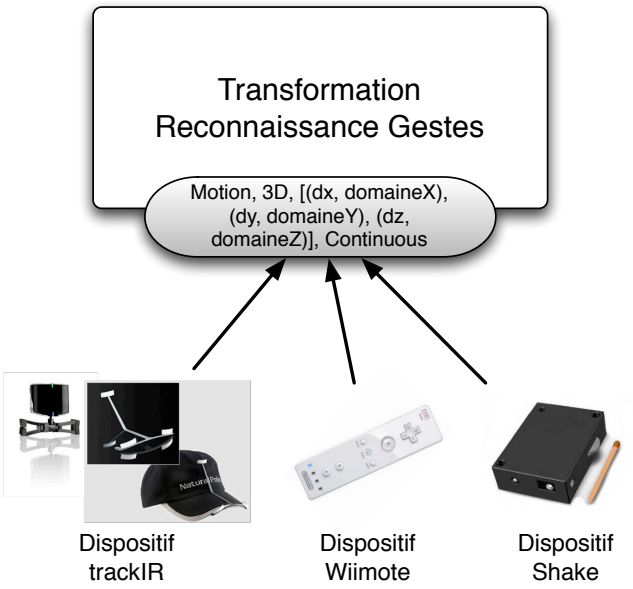


FIGURE 8.32 – Plusieurs modalités exploitant un composant Transformation unique.

GoogleEarth et les applications que nous avons utilisées dans GEI (section 8.3.1 B) est que ces dernières sont en 2D alors que GoogleEarth est en 3D.

Notre objectif est d'étudier l'interaction multimodale 3D, aussi nous avons utilisé le Cubtile, un dispositif tactile en forme de cube qui permet de capturer le toucher sur ses cinq faces (à gauche sur la Figure 8.33). Une interaction multimodale, utilisant le Cubtile combiné avec des gestes (avec le Shake) et des commandes vocales a été implémentée (à droite sur la Figure 8.33).

FIGURE 8.33 – Utilisation du Cubtile seul (à gauche) ou avec d'autres modalités, comme la voix et le Shake (à droite) pour une tâche de manipulation 3D avec l'application GoogleEarth.

L'interaction tactile sur le cube est utilisée afin de déplacer la carte. Il est possible de naviguer directement vers certains villes en utilisant la reconnaissance vocale.

La voix est également utilisé en composition avec le Shake afin de marquer des lieux : avec la voix l'utilisateur indique le nom de la marque (par exemple, "plage") et le mouvement du Shake assigne ce nom à la position actuelle sur GoogleEarth.

La Figure 8.34 montre l'assemblage de composants réalisé afin d'implémenter cette interaction multimodale. Cet assemblage contient trois composants Dispositif : le composant Shake, le composant Cubtile et le composant MultitelSpeechRecognition. Un bouton du composant Shake permet de démarrer la reconnaissance vocale (connexion sur le port StartEngine du composant MultitelSpeechRecognition). Les données du Shake et de la reconnaissance vocale sont ensuite composés à travers un composant de transformation puis envoyés au composant GoogleEarthAPI. Le Cubtile est connecté au composant GoogleEarthAPI à travers un composant de transformation dépendant de l'application, le OILightInteractionGoogleEarth. Ce composant contient plusieurs interfaces à syntaxe ad hoc en sortie qui correspondent à la syntaxe ad hoc des interfaces en entrée du composant GoogleEarthAPI.

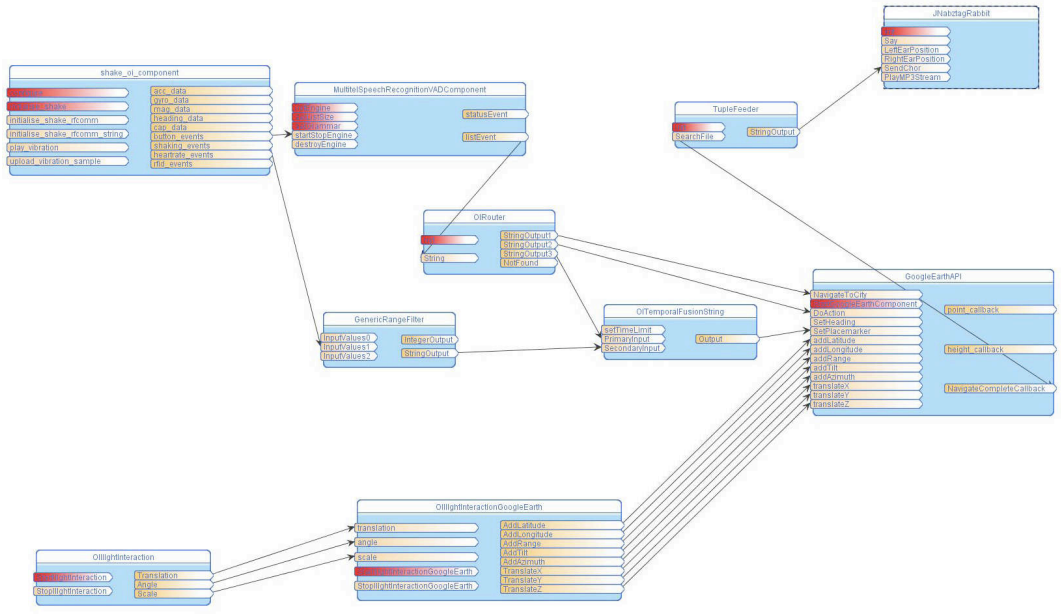


FIGURE 8.34 – Assemblage de composants implémentant une interaction multimodale avec le Cubtile, le Shake et la reconnaissance vocale afin de contrôler GoogleEarth.

Diaporama multimodal

Afin de présenter le projet OpenInterface, il est apparu naturel de réaliser des diaporamas multimodaux, sur lesquels nous pourrions illustrer les concepts de notre approche. Cette application a été utilisée lors des présentations du projet et a permis au présentateur d'utiliser plusieurs modalités innovantes pour avancer dans ses transparents. L'application a été implémentée par Michael Ortega, ingénieur de recherche de l'équipe IIHM.

Même si nous avons utilisé l'OIDE pour contrôler les applications de diaporama traditionnelles (PowerPoint) en utilisant des événements clavier ou souris, cette

nouvelle application a été implémentée afin d'offrir de nouvelles fonctions pendant la présentation : comme le changement de transition et le zoom.

Plusieurs interactions multimodales ont été implémentées avec l'OIDE. Nous avons par exemple utilisé des capteurs Interface-Z afin de créer des interfaces tangibles, comme un cahier qui permet de faire défiler les diapositives en tournant les pages (grâce à un capteur de torsion inséré à l'intérieur du cahier), ou la Wiisoft (Figure 8.35), qui permet de zoomer sur une diapositive en tordant une mousse en forme de Wiimote. La tâche zoom étant localisée sur un point spécifique de la diapositive, nous avons réalisé une composition entre la Wiimote comme dispositif de pointage et la Wiisoft afin de réaliser l'action de zoom (Figure 8.35).

FIGURE 8.35 – Composition de la wiimote et de la wiisoft afin de naviguer dans un diaporama multimodal affiché sur un écran.

La Figure 8.36 illustre l'assemblage qui implémente cette interaction multimodale pour la tâche de zoom. Cet assemblage contient deux composants Dispositif, le composant Wiimote et le composant Interface-Z. Chacun de ces composants est connecté à un composant de Transformation afin de générer des commandes à partir des événements des dispositifs. Ces deux modalités sont ensuite connectées au composant de Composition Complémentarité et finalement au composant qui implémente l'application.

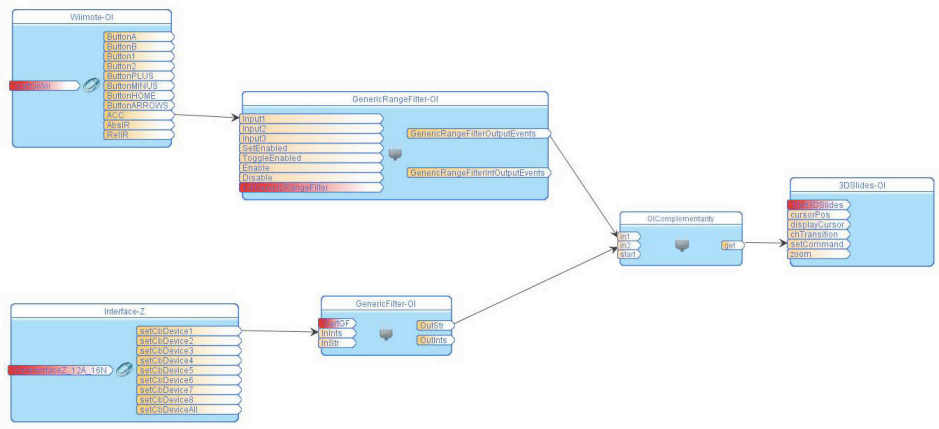


FIGURE 8.36 – Assemblage de composants de la composition complémentaire entre la wii et la wiisoft pour la tâche de zoom de l'application de diaporama multimodal.

Jeu ArkanOI

Un jeu collaboratif a également été implémenté par Michael Ortega. Ce jeu, appellé ArkanOI, est basé sur le jeu classique Arkanoid, où l'utilisateur doit déplacer une plate-forme afin d'éviter que la boule détruise ses briques et doit essayer de détruire les briques de l'adversaire. La principale différence avec le jeu Arkanoid traditionnel réside dans la possibilité d'incliner les plate-formes (les pucks) sur les côtés, ce qui rend ce jeu propice à l'interaction multimodale.

Différentes interactions multimodales ont été implémentées pour manipuler ce jeu. Une configuration multimodale, multijouer et multi-affichage a été réalisée. Sur cette configuration, plusieurs utilisateurs jouent ensemble en manipulant différentes plate-formes (les pucks) avec différentes modalités d'interaction (Figure 8.37). Parmi les modalités d'interaction figurent la DiamondTouch, le iPhone et une table tactile avec retro-affichage (Figure 8.38).

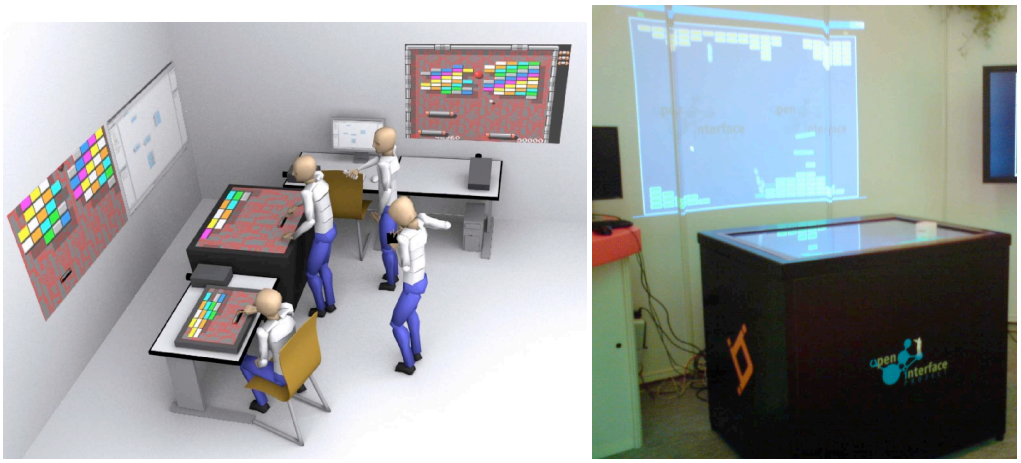


FIGURE 8.37 – Configuration multi-joueur et multi-affichage sur le jeu ArkanOI.

FIGURE 8.38 – Interaction multimodale avec iPhone et Wiimote (à gauche) et interaction de type manipulation directe sur une table tactile (à droite).

La Figure 8.39 illustre l'assemblage implémentant une interaction multimodale avec la table tactile, la wiimote et la wiifit. La table tactile (composant OILight) permet de contrôler autant le mouvement des pucks que leur inclinaison à travers un composant de transformation dépendant du dispositif et de l'application, le com-

posant OILightPuck. La wiimote (composant OIWii) permet de contrôler l'inclinaison à travers le composant de transformation dépendant du dispositif OIWiimoteAdapter. La wiifit (composant WiiBalanceBoard) permet de contrôler le mouvement (le joueur peut déplacer le puck en se penchant latéralement). Ce composant est directement connecté au composant du jeu.

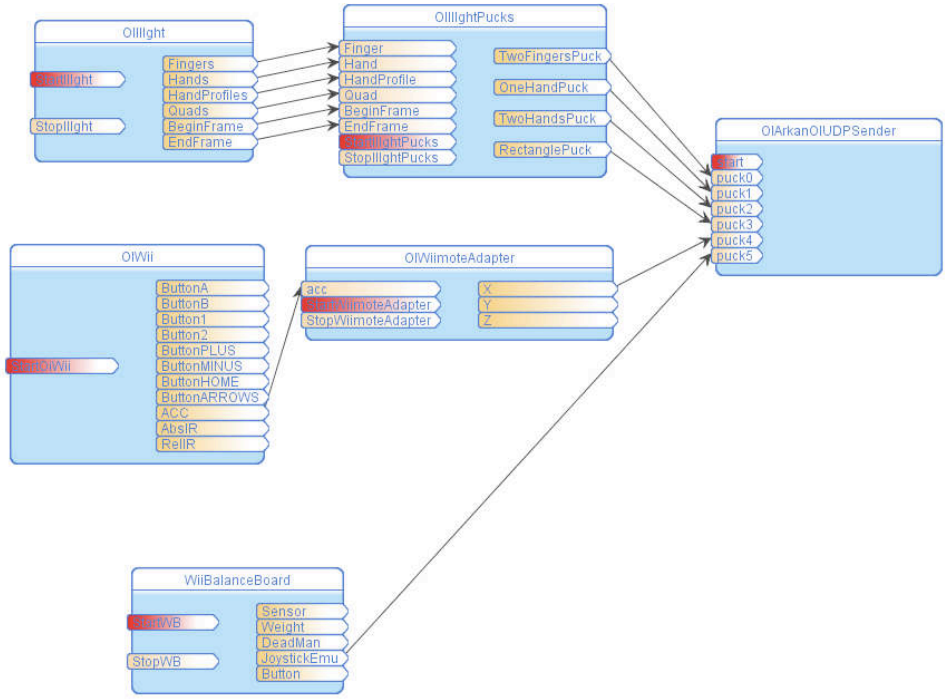


FIGURE 8.39 – Assemblage de composants implémentant la manipulation avec la table tactile, avec la wiimote et avec la wiifit pour contrôler le jeu ArkanOI.

8.4 Conclusion

Dans ce chapitre, nous avons montré les différentes évaluations de l'OIDE, l'outil de prototypage présenté au chapitre 7. Nous avons mené deux types d'évaluation : des évaluations empiriques contrôlées et l'évaluation de l'outil par son utilisation pour le prototypage.

Deux évaluations empiriques ont été menés : nous avons présenté à la section 8.1 une évaluation contrôlée de l'outil qui focalise sur la définition de prototypes interactifs et à la section 8.2 une évaluation empirique de l'outil pour la conception de prototypes simulés. Ces évaluations ont montré que l'outil est un instrument de communication et d'exploration de nouvelles solutions pour la conception de prototypes interactifs. Pour la conception de prototypes simulés, l'outil permet de créer des prototypes simulés crédibles et la configuration multi-compère représente une aide dans la tâche de simuler d'interaction multimodale.

L'utilisation de l'OIDE dans le projet OpenInterface a permis ensuite de montrer l'utilité de l'OIDE pour le prototypage d'interactions multimodales dans une démarche itérative de conception centrée utilisateur. L'OIDE permet de modifier

rapidement et facilement les prototypes entre chaque itération de la conception : changer les modalités utilisées, changer l'application, modifier les composants de transformation afin d'utiliser autrement ou d'améliorer l'interaction d'un certain dispositif.

Nous avons également montré comment l'OIDE peut aider les concepteurs de nouvelles modalités d'interaction à tester rapidement leur techniques avec des applications déjà existantes ou en combinaison avec d'autres modalités d'interaction. Ainsi, nous avons présenté les travaux de deux étudiants qui ont utilisé l'OIDE pour développer des nouvelles modalités pour la navigation dans une carte projetée avec le iPhone et pour tester un algorithme de reconnaissance de gestes avec différents dispositifs en entrée.

Tous les prototypes décrits dans ce chapitre soulignent la variété des modalités d'interaction offerte par l'OIDE. De plus plusieurs de ces prototypes (mobiles ou non) ont été développés avec plusieurs versions de l'OIDE par des utilisateurs extérieurs à l'équipe. En particulier le contexte du projet européen nous a permis une mise au point itérative de l'OIDE grâce aux retours d'expérience des partenaires.

Ces prototypes ont également permis de mettre en évidence certaines limites de l'OIDE. Tout d'abord, l'OIDE ne considère pas le feedback de l'interaction vers l'utilisateur. Nous avons vu que cela peut poser des problèmes lorsque l'utilisateur manipule une modalité innovante qu'il ne connaît pas, comme le gant de données (prototype Jeu v3). Ensuite, nous avons observé des problèmes d'utilisabilité lorsque les assemblages deviennent trop grands : le concepteur se voit alors obligé à agencer les composants verticalement afin de les faire tous rentrer dans la zone d'assemblage (Figures 8.14, 8.24, 8.26 et 8.34).

Chapitre 9

Conclusion et perspectives

Contenu du chapitre

Les travaux présentés dans ce manuscrit contribuent au domaine de l'ingénierie de l'interaction multimodale en entrée et focalisent sur le prototypage d'interfaces multimodales en entrée. Motivé par l'étendue des possibilités en terme de techniques d'interaction en entrée, et par les pratiques souvent ad hoc de conception et de développement d'interfaces multimodales (dépendantes des modalités d'interaction, des plates-formes matérielles et logicielles cibles et des domaines d'application), nos travaux visent à faciliter l'exploration de ce vaste espace de conception par le prototypage d'interaction multimodale.

Pour conclure ce manuscrit, nous rappelons nos contributions qui concernent la conception et le développement de prototypes d'interfaces multimodales. Nos contributions sont conceptuelles et logicielles. Une prise de recul par une analyse de nos résultats nous permet ensuite d'identifier plusieurs perspectives à nos travaux, à moyen et long terme.

9.1 Résumé des contributions

Nos contributions pour le prototypage d'interfaces multimodales sont à la fois conceptuelles, avec la définition d'un modèle conceptuel pour le prototypage, et logicielles, avec un outil logiciel de prototypage basé sur notre modèle conceptuel.

9.1.1 Modèle conceptuel pour le prototypage

Pour le prototypage d'interfaces multimodales, nous avons défini un modèle conceptuel basé sur une approche à composants (chapitre 4). Ce modèle offre un fort pouvoir descriptif et génératif en alliant les atouts de deux modèles existants, Icon et Icare. Notre modèle conceptuel définit un espace de caractérisation des composants logiciels pour le prototypage d'interfaces multimodales. Cet espace est structuré selon deux axes : un axe noté Structure qui définit par une typologie des composants la structure de l'assemblage des composants et un axe noté Réutilisation qui caractérise le degré de réutilisabilité des composants.

L'axe Structure définit quatre niveaux dans le flot de données des dispositifs aux tâches : dispositif, transformation, composition et tâche. Cette structure représente une solution mixte, basée sur les approches Icon et Icare, augmentant le pouvoir de génération de notre modèle grâce à l'utilisation du modèle en cascade de Icon, tout en conservant le pouvoir expressif du modèle Icare. Cette typologie des composants définit le principe de réification, c'est-à-dire la façon dont les concepts sont convertis en composants, au sein de notre modèle.

L'axe Réutilisation est composé de deux dimensions qui permettent de caractériser la réutilisabilité d'un composant. La première dimension décrit la dépendance du comportement du composant. Trois valeurs sont identifiées sur cette dimension : dépendant de l'application, dépendant d'autres composants et indépendant. La deuxième dimension décrit la compatibilité syntaxique des composants, définie par le type de syntaxe des interfaces des composants. Nous identifions deux types de syntaxe pour les interfaces : ad hoc et normalisée. Une interface d'un composant qui a une syntaxe ad hoc est constituée d'un ensemble de paramètres avec des types de données, sans adhérer à aucune convention. Dans ce cas, le composant est faiblement compatible avec les autres composants. Normaliser la syntaxe de l'interface des composants de notre modèle permet de facilement les connecter au sein d'un assemblage. Nous avons ainsi proposé de normaliser l'interface en sortie des composants Dispositif en exploitant les traits caractéristiques de la taxonomie des dispositifs de Buxton [Buxton, 1983] et l'interface en entrée des composants Tâche en nous reposant sur la taxonomie de Foley [Foley *et al.*, 1984].

Nous avons mené deux types d'évaluation complémentaires de notre modèle conceptuel : une évaluation analytique, qui consiste à analyser notre modèle au regard de critères d'évaluation recensés dans la littérature, et une évaluation empirique, qui repose sur des entretiens et des exercices réalisés par un panel de sujets et qui permet d'obtenir un retour expérimental sur l'usage de notre modèle. De l'évaluation analytique, nous retenons les limites de notre modèle quant au passage à l'échelle et à la visibilité des grands assemblages de composants. Cette limite est liée à l'approche à composants que nous avons adoptée. L'évaluation empirique a permis de valider les éléments fondateurs de notre modèle ainsi que d'identifier les limites du modèle quant à la description de la logique temporelle de l'interaction, par exemple lorsqu'on considère une interaction multimodale qui varie selon les états du système.

9.1.2 Outil logiciel pour le prototypage

Basé sur notre modèle conceptuel, nous avons conçu et développé un outil logiciel pour le prototypage, le OpenInterface Interaction Development Environment (OIDE). Les deux caractéristiques importantes de notre outil sont les suivantes :

- L'outil permet le développement de prototypes simulés et de prototypes interactifs : nous avons souligné l'absence d'outil qui permette le développement de ces deux types de prototypes qui s'inscrivent de façon continue dans une démarche de conception itérative centrée utilisateur.

- L'outil offre un fort pouvoir descriptif et génératif en reposant explicitement sur notre modèle conceptuel.

Nous avons peuplé l'environnement de nombreux composants, certains que nous avons développés et d'autres développés par les membres du projet OpenInterface.

Nous avons mené deux types d'évaluation de notre outil OIDE : des évaluations empiriques contrôlées et l'évaluation de l'OIDE par son utilisation pour le prototypage. Deux types d'évaluation empirique ont été menés : une évaluation contrôlée de l'outil lors de la conception de prototypes interactifs et une évaluation empirique de l'outil pour la conception de prototypes simulés. Ces évaluations ont montré que l'outil est un instrument de communication et d'exploration de nouvelles solutions pour la conception de prototypes interactifs. Pour la conception de prototypes simulés, l'outil permet de créer des prototypes simulés crédibles du point de vue du sujet qui participe à l'expérience magicien d'oz. De plus la possibilité de configuration multi-compère offerte par l'outil est une solution pour réduire la charge cognitive du magicien, identifiée élevée dans le cas de multimodalité.

L'utilisation de l'OIDE dans le projet OpenInterface a permis ensuite de montrer l'utilité de l'OIDE pour le prototypage d'interactions multimodales au sein d'une démarche itérative de conception centrée utilisateur. L'OIDE permet de modifier rapidement et facilement les prototypes entre chaque itération de conception : changer les modalités utilisées, changer le type de multimodalité, changer les tâches ou l'application, modifier les composants de transformation afin d'utiliser différemment un dispositif ou d'améliorer l'interaction avec un dispositif donné. Nous avons également montré comment l'OIDE peut aider les concepteurs de nouvelles modalités d'interaction à tester rapidement leur techniques avec des applications déjà existantes ou en combinaison avec d'autres modalités d'interaction. Tous les prototypes décrits dans ce manuscrit soulignent la variété de modalités d'interaction offerte par l'OIDE. Nous soulignons enfin que le projet OpenInterface a permis l'utilisation de notre outil en dehors de notre équipe de recherche par les partenaires européens du projet, une forme de validation de l'outil qui nous semble importante et rarement faite dans le temps imparti d'une thèse.

Nous entrevoyons pour nos travaux de multiples perspectives. Là encore, la dualité, conception et réalisation logicielle apparaît dans nos propositions. Ces dernières s'organisent en deux parties : les extensions ou prolongements à moyen terme qui concernent les limites identifiées de nos résultats et les prolongements à plus long terme. Nous présentons dans la section suivante les extensions à moyen terme, avant d'introduire les perspectives à long terme.

9.2 Limites identifiées et extensions à moyen terme

9.2.1 Description de la logique temporelle de la composition : composants CARE à états

Une des limites identifiées pendant l'évaluation empirique de notre modèle conceptuel concerne la description de la logique temporelle de l'interaction. En effet, le modèle à flot de données permet de définir une tâche de façon continue. Il est néanmoins difficile de décrire la logique d'une interaction multimodale qui varie selon les états du système. Prenons par exemple la carte multimodale présentée au chapitre 8 : la composition des modalités peut varier selon l'état de la carte. Lorsque la carte est au niveau minimum de zoom, l'utilisateur peut utiliser le doigt et la voix pour zoomer sur un point. Ensuite, lorsque la carte est au niveau maximum de zoom, l'utilisateur peut utiliser le doigt ou le geste (avec le Shake) pour se déplacer sur la carte. Typiquement pour les mêmes modalités, l'utilisateur de l'OIDE doit décrire trois assemblages de composants pour les tâches zoom-in, zoom-out et déplacement. De plus il n'est pas possible de décrire la disponibilité d'un assemblage selon des conditions, par défaut tous les assemblages étant exécutés à un instant donné.

Pour ce problème, au niveau conceptuel, une solution consisterait à intégrer des automates à états dans le flot de données (dans le comportement des composants). Cette solution, adoptée récemment par Caroline Appert dans FlowStates [Appert *et al.*, 2009], permet d'augmenter le pouvoir d'expression et la flexibilité de l'approche de conception. Cette solution diminue néanmoins la visibilité de l'assemblage et augmente la difficulté de notre approche (seuil plus haut).

Une extension conceptuelle de notre modèle qui permettrait d'intégrer une machine à états dans un assemblage de façon moins complexe serait de concevoir des composants de composition CARE à états. Ainsi, la logique temporelle serait limitée au niveau composition dans notre flot de données, ce qui diminuerait l'impact sur la visibilité. Ces composants permettraient d'aiguiller la composition des modalités selon des états définis par le concepteur. De plus, selon l'état, le composant pourrait effectuer un type de composition CARE différente.

La Figure 9.1 illustre l'assemblage de l'exemple de la carte multimodale donné précédemment. Dans cet assemblage, le comportement du composant *Composition CARE à états* est défini par une machine à états. Cette machine contient trois états qui correspondent aux trois états du système (la carte) : état vue générale de la carte (niveau de zoom minimum, État 0), état vue partielle de la carte (niveau de zoom intermédiaire, État 1) et état vue détaillée d'une partie de la carte (niveau de zoom maximum, État 2).

Les transitions entre les états sont définies selon les valeurs de la modalité en entrée de reconnaissance vocale ("Zoom in" et "Zoom out") et celle de la variable du système "niveau-zoom". Selon l'état, le composant *Composition CARE à états* va composer de façon différente des modalités et les aiguiller vers des tâches différentes. Ainsi, à l'état 0 et à l'état 1, les modalités de reconnaissance vocale et de toucher sont utilisées de façon Complémentaire pour définir la tâche de Zoom. À l'état 2, les modalités de toucher et gestuelle (avec le Shake) sont utilisées de façon Equivalente pour définir la tâche de Déplacement.

Nous voyons, par cet exemple, le besoin de décrire la logique d'interaction (en-

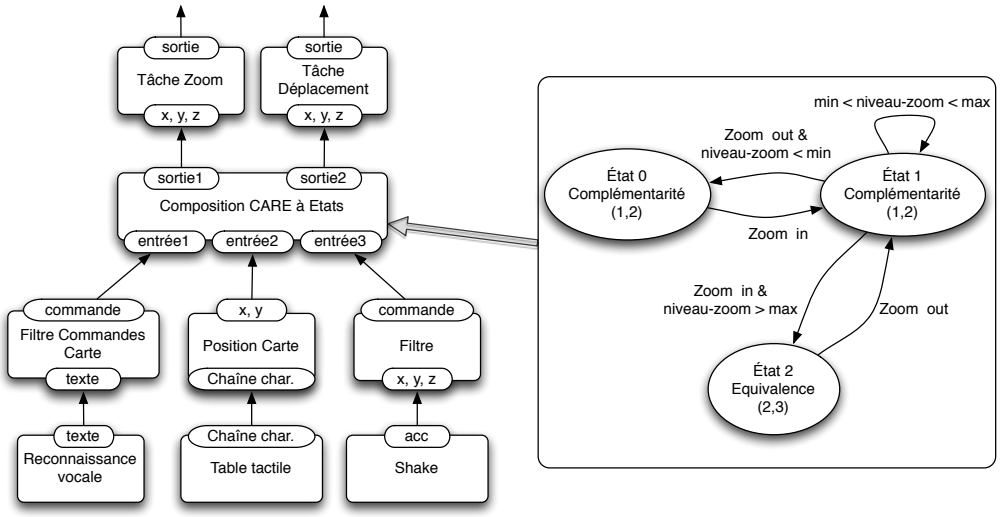


FIGURE 9.1 – Composant de composition CARE à états.

chaînement des tâches) au sein de l'assemblage. Typiquement cela est décrit dans le Contrôleur de Dialogue (selon une architecture Arch), ce dernier serait alors en charge d'activer ou de dé-activer un assemblage de composants selon l'état du dialogue et donc la disponibilité de certaines tâches. Nous voyons dans cette solution de description de la logique d'interaction basée sur des automates au sein d'un assemblage un exemple du phénomène « skinky » lié au modèle Arch où des fonctionnalités glissent d'un composant fonctionnel à un autre.

Au niveau de l'outil OIDE, il serait alors intéressant de l'étendre afin de permettre au concepteur de définir graphiquement la machine à états, puis de l'associer à un composant *CARE à états* au sein d'un assemblage à composants. Ainsi notre outil de prototypage permettrait autant de définir le flot de données spécifiant une tâche que la machine à états spécifiant la logique de l'interaction.

La capture de la trace d'exécution de la machine à états serait aussi un outil intéressant pour les concepteurs afin d'analyser l'usage du système par l'utilisateur final. Cette capture des données, non implémentée dans l'OIDE, est une limite de notre outil que nous analysons dans la section suivante.

9.2.2 Capture et analyse des données de tests

L'absence d'une solution pour la capture et l'analyse des données de tests représente une limite de notre modèle conceptuel et constitue une extension intéressante de notre outil OIDE. Cette capture offrirait un support important dans le cadre d'une conception itérative centrée sur l'utilisateur basée sur le prototypage.

Conceptuellement, il faudrait définir un format pour la capture des données qui soit facilement exploitable pour l'analyse des résultats. Nous avons déjà effectué un travail sur la capture des données d'usage pour la multimodalité en sortie [Serrano *et al.*, 2006], utilisant un format nommé ACICARE basé sur le modèle Icare. De la même façon que ce format était basé sur le modèle Icare, notre format de données pourrait être basé sur notre modèle conceptuel afin de pouvoir exploi-

ter les caractéristiques des composants définies dans notre modèle et en particulier le type de composants au sein de l'assemblage.

De plus, l'utilisation de notre modèle permettrait de définir le format des données capturées, selon le niveau dans le flot de données des dispositifs aux tâches (Figure 9.2). Ainsi, le concepteur peut choisir à quel niveau capturer les données : au niveau dispositif, transformation, composition ou tâche. À chaque niveau, les données peuvent contenir celles des niveaux inférieures afin de ne pas perdre d'informations qui pourraient être utiles à l'analyse.

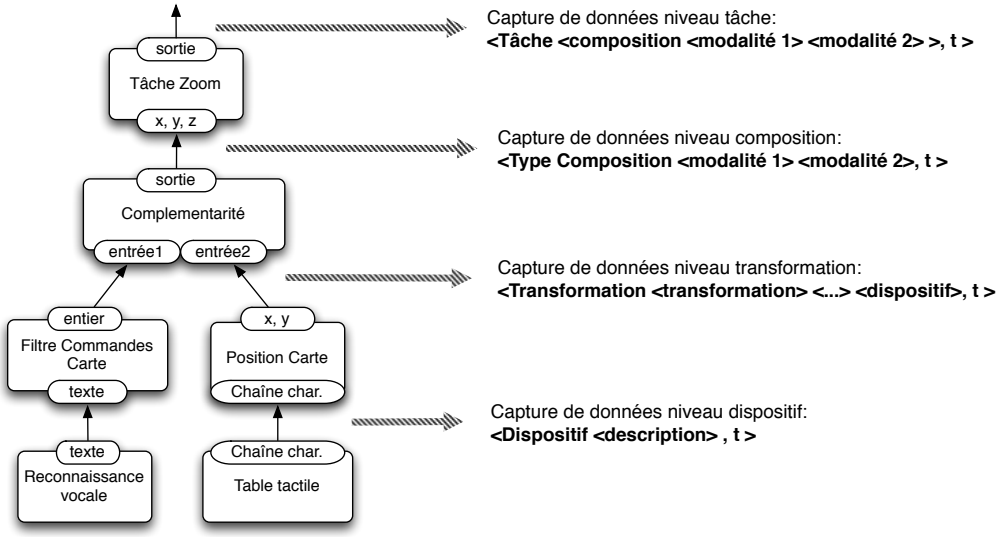


FIGURE 9.2 – Capture de données à différents niveaux de l'assemblage.

Au niveau de l'outil, il faudrait implémenter les mécanismes qui permettent de capturer, enregistrer et analyser les données des tests. Nous identifions deux solutions pour implémenter la capture. La capture peut passer par des solutions intégrées dans la plate-forme sous-jacente, ou par l'utilisation de composants logiciels de traçage que le concepteur pourrait placer dans l'assemblage afin de capturer certains événements. Ensuite, l'enjeu serait d'étendre l'OIDE afin d'intégrer des outils de visualisation des données capturées avec des filtres selon les niveaux identifiés dans le flot de données.

9.2.3 Aide à l'exploration de l'espace des possibilités

L'aspect statique de l'assemblage de composants représente une autre limite de notre contribution. Notre modèle conceptuel ne considère par explicitement l'assemblage dynamique de composants. De plus, malgré la présence d'un mode d'exécution dynamique dans notre outil, en pratique les assemblages sont souvent réalisés de façon statique, en grande partie à cause des limitations du mode dynamique actuel.

Il serait intéressant d'étendre notre outil afin de fournir une aide au concepteur pour remplacer un composant par un autre. Cette aide s'appuierait sur les caractéristiques des composants issues de notre modèle conceptuel : le niveau dans le flot de données, la dépendance comportementale ou la syntaxe de l'interface. Par

exemple, si dans un assemblage le concepteur veut remplacer un composant dispositif (comme le Shake) par un autre, le modèle peut indiquer au concepteur quels composants Dispositif sont équivalents en termes de syntaxe de l'interface. Pour cela le concepteur pourrait par exemple faire un clique droit sur le composant à remplacer afin d'obtenir une liste de composants candidats à ce changement.

9.3 Perspective à long terme : dynamisme du contexte

Une perspective à long terme de notre travail serait de prendre en compte le dynamisme du contexte d'interaction : les dispositifs ou d'autres composants peuvent devenir disponibles ou indisponibles pendant l'exécution. Pour cela, une approche à services (section 3.3.3 chapitre 3) serait plus appropriée que l'approche à composants que nous avons adoptée dans nos travaux.

Ainsi, un premier travail serait d'adapter notre outil afin qu'il s'exécute sur une plate-forme à services, comme OSGi [Tavares et Valente, 2008]. Ensuite, il conviendrait d'étendre notre approche pour permettre la découverte (ou disparition) dynamique de composants d'interaction à l'exécution. Enfin et sans doute le plus difficile, il conviendrait de fournir des mécanismes pour adapter automatiquement ou semi-automatiquement (en remettant l'utilisateur dans la boucle), l'interaction multimodale. Ce mécanisme s'appuyant sur l'assemblage de services vise à l'enrichir ou à combler des trous dans le flot de données dynamiquement.

Voici un ensemble de travaux et de questions à traiter pour la gestion dynamique de l'interaction multimodale qui va au-delà du dynamisme géré actuellement dans notre outil reposant sur une description complète des possibilités, figée à la conception. Ces travaux font l'objet du travail de thèse de Pierre-Alain Avouac, doctorant dans les équipes IIHM et ADELE au Laboratoire d'Informatique de Grenoble. L'approche adoptée repose sur iPOJO, un modèle à composants orientés services basé sur OSGi.

Annexes

Annexe A

Liste complète des publications

Conférences internationales de large diffusion avec comité de lecture sur texte complet

- Serrano, Nigay. Temporal Aspects of CARE-based Multimodal Fusion : From a Fusion Mechanism to Composition Components and WoZ Components (2009). In IMCI '09 : Proceedings of the 11th international conference on Multimodal interfaces, November 2-6, Cambridge, MA, USA. ACM, pp. 177-184.
- Serrano, Juras, Nigay. A Three-dimensional Characterization Space of Software Components for Rapidly Developing Multimodal Interfaces (2008). In IMCI 08 : Proceedings of the 10th international conference on Multimodal interfaces, October 20-22, Chania, Crete, Greece, ACM, pp. 149-156.
- Juras, Nigay, Ortega, Serrano. Multimodal slideshow : demonstration of the openinterface interaction development environment (2008). In ICMI 08 : Proceedings of the 10th international conference on Multimodal interfaces, October 20-22, Chania, Crete, Greece, ACM.
- Serrano, Nigay, Demumieux, Descos, Losquin. Multimodal Interaction on Mobile Phones : Development and Evaluation Using ACICARE (2006). In Proceedings of MobileHCI 2006, The 8th International Conference on Human Computer Interaction with Mobile Devices and Services, Epoo, Finland, 12-15 september 2006, ACM, pp. 129-136.

Revues internationales avec comité de lecture

- Serrano, Nigay. A wizard of oz component-based approach for rapidly prototyping and testing input multimodal interfaces (2010). In Journal on Multimodal User Interface, Springer Publ. 3(3).
- Benoît, Bonnaud, Caplier, Damousis, Jourde, Lawson, Nigay, Serrano, Tzoradas (2007). Multimodal signal processing and interaction for a driving simulator : component-based architecture. In Journal on Multimodal User Interface, Springer Publ. 1(1).

Conférences nationales avec comité de lecture sur texte complet

- Serrano, Nigay (2009). OpenWizard : Une approche pour la création et l'évaluation rapide de prototypes multimodaux. In proc. IHM'09, 21ème Conférence Francophone sur l'Interaction Homme-Machine, ACM, pp. 101-109.

- Serrano, Juras, Ortega, Nigay (2008). OIDE : un outil pour la conception et le développement d'interfaces multimodales. In 4èmes journées Francophones Mobilité et Ubiquité (UbiMob'08), ACM, pp. 91-92.

Chapitres d'ouvrages

- Nigay, Bouchet, Juras, Mansoux, Ortega, Serrano, Lawson (2008). Software Engineering for Multimodal Interactive Systems. In Multimodal user interfaces : from signals to interaction.

Autres conférences et colloques avec actes

- Serrano, Nigay, Lawson, Ramsay, Murray-Smith, Denef. The OpenInterface framework : a tool for multimodal interaction. (2008). In Extended Abstracts of CHI 2008, April 5-10, Florence, Italy, ACM, pp. 3501-3506.
- Gray, Ramsay, Serrano. A Demonstration of the OpenInterface Interaction Development Environment (2007). In Adjunct Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology, UIST 2007, ACM, pp. 39-40.
- Benoît, Bonnaud, Caplier, Damousis, Tzoradas, Jourde, Nigay, Serrano, Lawson. Multimodal Signal Processing and Interaction for a Driving Simulator : Component-based Architecture (2006). In Rapport sur le projet Multimodal Driving Simulator, Workshop de 4 semaines organisé dans le cadre d'eNTERFACE 2006, Réseau Européen d'EXellence SIMILAR, Dubrovnick, Croatie, NoE SIMILAR.

Annexe B

Evaluation empirique de notre modèle conceptuel

Dans cet annexe nous présentons les différents documents utilisés dans l'évaluation empirique de notre modèle, décrite au chapitre 5. Ces documents ont été préparés en collaboration avec Mme Nadine Mandran, ingénieur en méthodologies des sciences sociales de la plateforme Marvelig [Marvelig, 2009]. Nous présentons les documents suivants :

- Fiche d'habitudes
- Fiche de suivi
- Cahier de charges

Fiche Habitudes

Thèse Marcos Serrano, IIHM, Novembre 2009 -

Prénom | **date** | **Année de naissance**

La dernière fois que vous avez développé un logiciel ou un système quelles étaient les grandes étapes par lesquelles vous êtes passé ?

Avez-vous l'habitude d'utiliser des méthodes de conception de systèmes d'information ou de développement logiciel ?

○ 1.Très souvent ○ 2.Souvent ○ 3.Parfois ○ 4.Jamais

Pourquoi en utilisez vous ?

Lequelles ?

Pourquoi, n'en utilisez vous pas ?

Quand vous utilisez une méthode de conception, vous ...

☐ 1.vous l'appliquez à la lettre ☐ 2.vous l'adaptez à vos besoins ☐ 3.vous en utilisez plusieurs en même temps

Vous utilisez une méthode de conception..

☐ 1.Pour toutes les phases ☐ 2.pour les phases importantes ☐ 3.pour les phases de collaboration

Est ce que vous avez l'habitude de présenter, même partiellement; vos méthodes à d'autres personnes ?

○ 1.Très souvent ○ 2.Souvent ○ 3.Parfois ○ 4.Jamais

Est ce que vous avez l'habitude de décrire, même partiellement, vos propres méthodes ?

○ 1.Très souvent ○ 2.Souvent ○ 3.Parfois ○ 4.Jamais

Dans quels contextes vous arrive-t-il de présenter vos méthodes ?

Quels sont les outils et langages que vous utilisez pour décrire vos méthodes ?

Comment mettez vous en place le suivi de votre méthode ?

Ajouter des questions spécifiques aux compétences pour utiliser le modèle de marcos

Fiche de suivi

Thèse de Marcos Serrano, IIHM.

Suivi de la réalisation du projet

Date de travail ☐ **Heure début** ☐

Notez le prénom des personnes

Objectif de la séance

☐ 1.organiser ☐ 2.concevoir ☐ 3.confronter ☐ 4.tester ☐ 5.développer

Précisez ces objectifs ...

Si vous avez eu à travailler ensemble, les objectifs étaient-ils ...

◯ 1.Tout à fait, les mêmes ◯ 2.Plutôt, les mêmes ◯ 3.Plutôt pas, les mêmes ◯ 4.Pas du tout, les mêmes

Pouvez-vous préciser les différences entre vos objectifs ?

Notez les principaux résultats de la séance

Les objectifs fixés ont-ils été atteints ?

◯ 1.Oui, tout à fait ◯ 2.Oui,partiellement ◯ 3.Non

Si non pourquoi ?

Aviez-vous l'ensemble des ressources nécessaires à votre disposition ?

◯ 1.Oui, tout à fait ◯ 2.Oui,partiellement ◯ 3.Non

si non, précisez ce qui vous a manqué ?

Fiche de suivi (suite)

Thèse de Marcos Serrano, IIHM.

Quelles ont été les difficultés que vous avez rencontrées lors de cette séance ?

Avez-vous fait appel à des ressources extérieures et lesquelles ?

Comment la méthode aurait pu vous éviter ces difficultés ?

Autres remarques sur la séance .....

Heure de fin

Expérience Méthode de conception pour interfaces multimodales

Cahier de charges

Description générale:
Le binôme doit réaliser la conception d'une interface multimodale en utilisant le modèle à composants présenté à la première séance.

Application :
Une carte d'une ville projetée sur une table.

Attention : Faire les exercices dans l'ordre sans lire les exercices suivants.

Exercice 1 :

- Pensez des tâches interactives pour cette carte. Précisez les noms et paramètres des méthodes associées aux tâches (une méthode par tâche).

- zoomer
- se déplacer sur la carte } fonctionner
- chercher un lieu précis
- afficher des informations sur un lieu précis
- afficher des "types" bâtiments et services → superposer différents "calques" d'informations
- calcul/visualisation d'itinéraires

~~(voir au dos pour les paramètres)~~

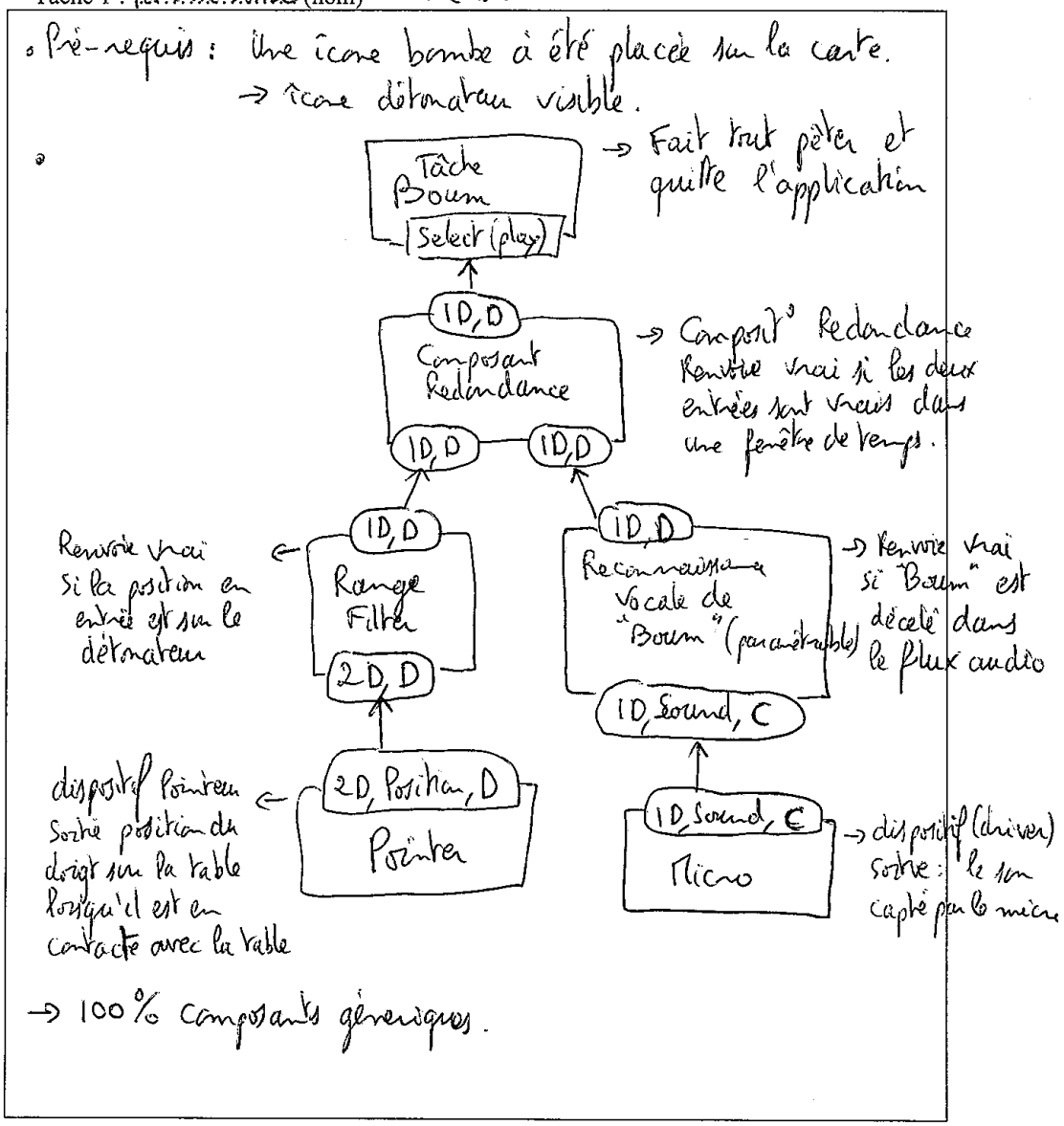
- Pensez aux interactions possibles sans tenir compte des contraintes matérielles ou logicielles. Dites, pour chaque interaction : les sens humains en jeu, les dispositifs et la tâche associée.

- Zoomer : • A 2 mains : 1 doigt pointe le centre du zoom, l'autre doigt de l'autre ~~main~~ modifie le ration en glissant sur une scrollbar imaginaire sur la table
- se déplacer sur la carte : faire glisser la carte en "balayant" la ~~carte~~ avec la main (plusieurs doigts) métaphore de la planisphère.
- chercher un lieu précis : paume de main sur la table pour ouvrir un menu contextuel, choix option "chercher" avec l'autre main ou en l'énonçant puis énoncé de la requête.
- afficher des infos sur un lieu précis : • "qu'est-ce que c'est ?" + doigt pointant (équivalent "Put That There")
  • paume de main → menu contextuel dont on sort en la faisant glisser un outil "question mark" à la MagicLens qu'on peut déplacer sur la carte et qu'on active en cliquant dessus

Exercice 2 :
Prenez 3 tâches parmi celles que vous avez conçues précédemment. Définissez-les avec un assemblage de composants qui implémente l'interaction associée, en suivant les contraintes suivantes

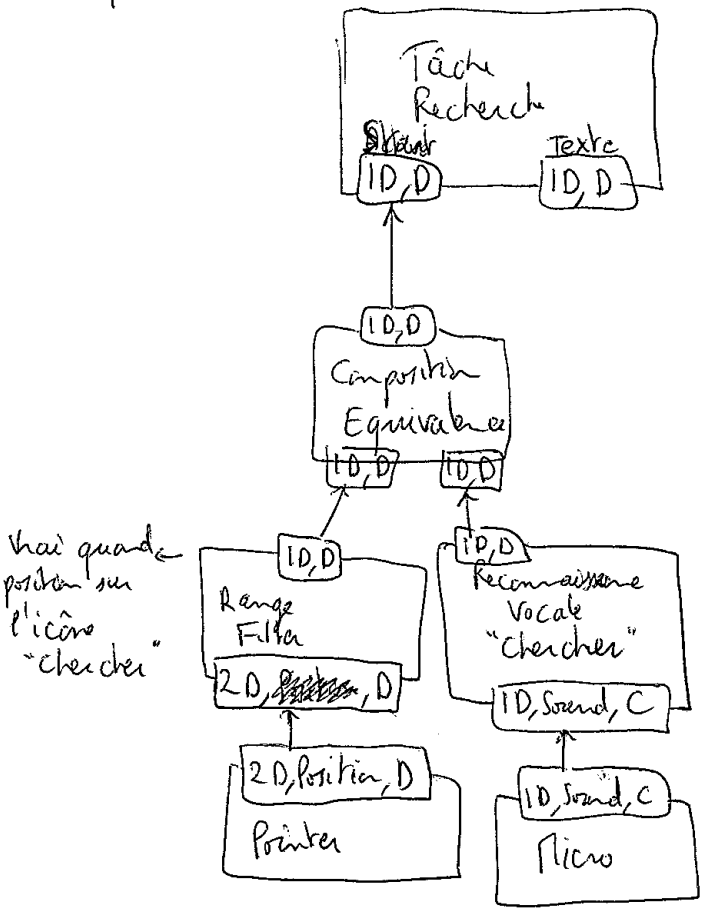
- Utiliser au moins une fois chaque composant de composition : Equivalence, Complémentarité, Redondance
- Pour chaque composant, donner le nom des ports en entrée/sortie et les paramètres associés, décrire brièvement le comportement du composant, déterminer sa généricité

Tâche 1 : ~~Poser une bombe~~ (nom) Faire sauter la bombe.



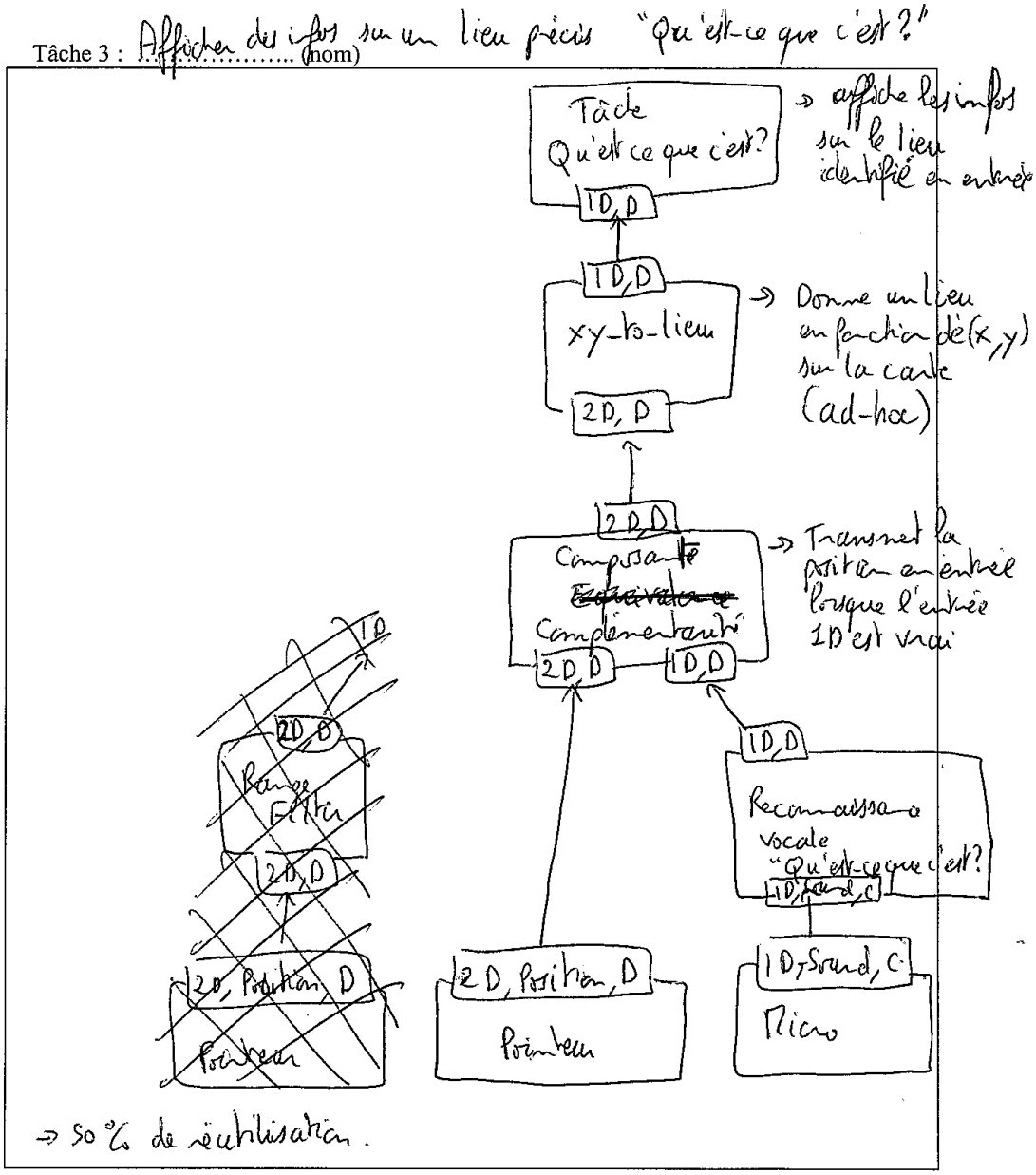
Tâche 2 : Chercher un (nom) lieu précis

Pré-requis : Menu ouvert avec Icône "chercher" visible



→ 100 % générique

→ Réutilisat° : Dispositifs et Transformations


Tâche 3 : Afficher des infos sur un lieu précis "Qu'est-ce que c'est ?"
..... (nom)
Tâche
Qu'est ce que c'est?
1D, D
→ affiche les infos sur le lieu identifié en entrée
1D, D
xy-to-lieu
2D, D
→ Donne un lieu en fonction de (x,y) sur la carte (ad-hoc)
2D, D
Composante
Complémentarité
2D, D
1D, D
→ Transmet la position en entrée lorsque l'entrée 1D est vrai
1D
2D, D
Range Filter
2D, D
2D, Position, D
Pointeur
2D, Position, D
Pointeur
1D, D
Reconnaissance vocale
"Qu'est-ce que c'est?
1D, Sound, C
1D, Sound, C
Micro
→ 50 % de réutilisation.

Exercice 3 :
Voici les tâches de l'application :

- **Centre (x, y) :** Permet de centrer la carte sur le point donné par x et y
- **Zoom(x, y, z) :** Permet de zoomer sur le point donné par x et y, avec une valeur de zoom définie par z. La valeur de zoom est une valeur relative au niveau de zoom courant (+/- 10). L'application gère les limites (si on fait +10 alors qu'on est au niveau maximal de zoom, aucune action n'est effectuée).
- **Translation(dx, dy)** : Déplace la carte d'une valeur relative définie par dx et dy. L'application gère les limites du plan de la carte.

Comme dispositifs, nous avons :

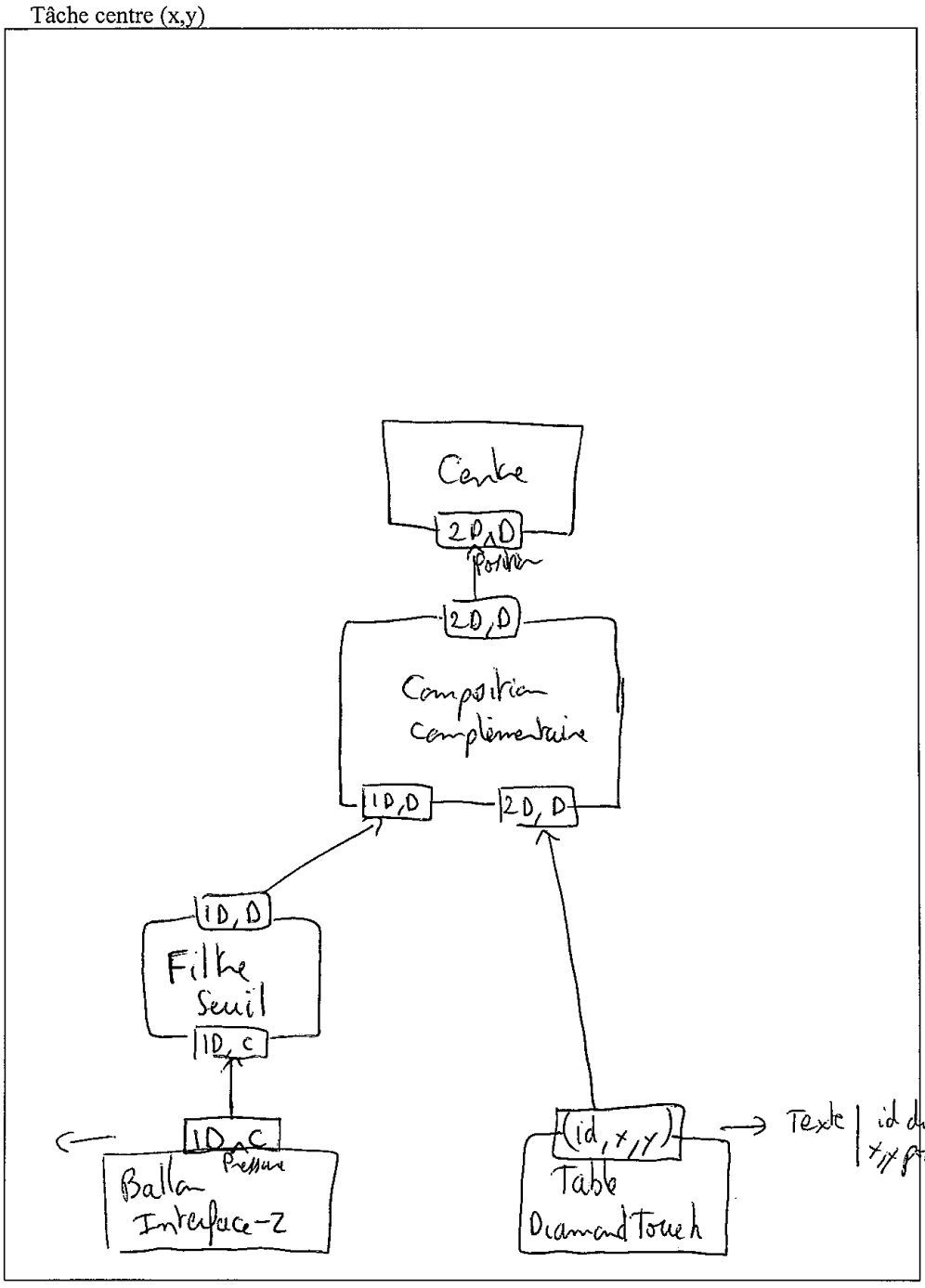
- **Table DiamondTouch :** Table tactile multi-doigt. Sa sortie est une chaîne de caractères avec les champs suivants (id, x, y), où id représente le id du doigt capté, x et y sa position.
- **Reconaissance vocale** : Algorithme de reconnaissance vocale. Reconnaît les mots définis par un dictionnaire.
- **Ballon Interface-Z :** Ballon gonflable connecté à un capteur de pression qui capture la pression exercée dessus.
- **Wiimote :** Manette sans fils. En sortie donne les valeurs de l'accéléromètre (x, y, z), les valeurs des boutons (un entier par bouton, 0 si pas activé, 1 si activé) et la position détectée par la caméra IR (x, y).

Nous aimerions tester les interactions suivantes :

- Composition complémentaire de la pression du ballon et d'un point sur la table pour désigner un nouveau centre de la carte affichée.
- Composition complémentaire de la voix et du geste pour désigner un point à zoomer
- Composition équivalente du geste avec la wiimote et du geste avec la diamond touch pour définir une translation de la carte

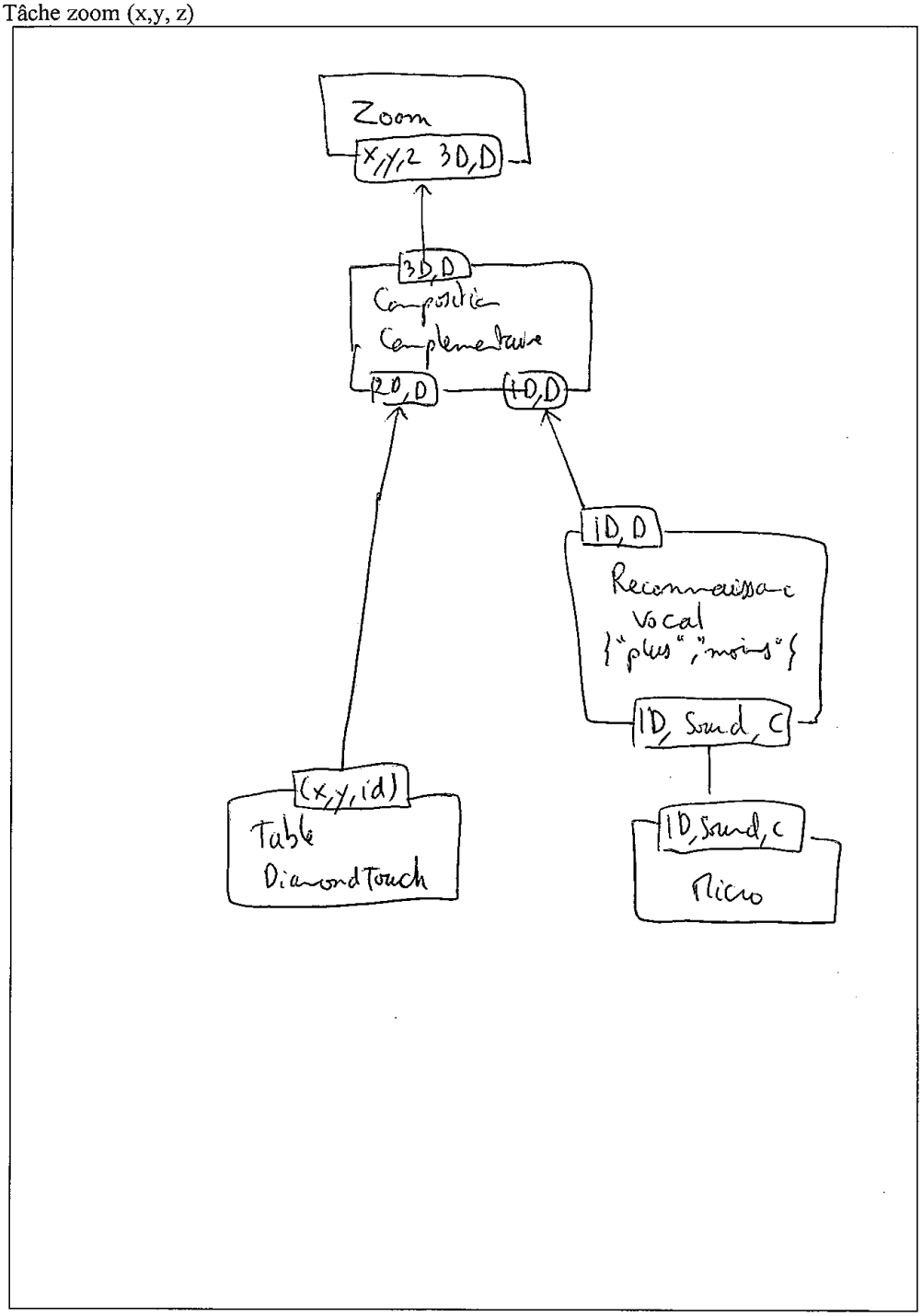
Exercice :

1. Réalisez les assemblages de composants correspondant à ces tâches.
2. Donnez, pour l'un des assemblages, la description complète de tous ses composants (utiliser la description MCDL). Pour chaque composant, décrire également tous ses ports.

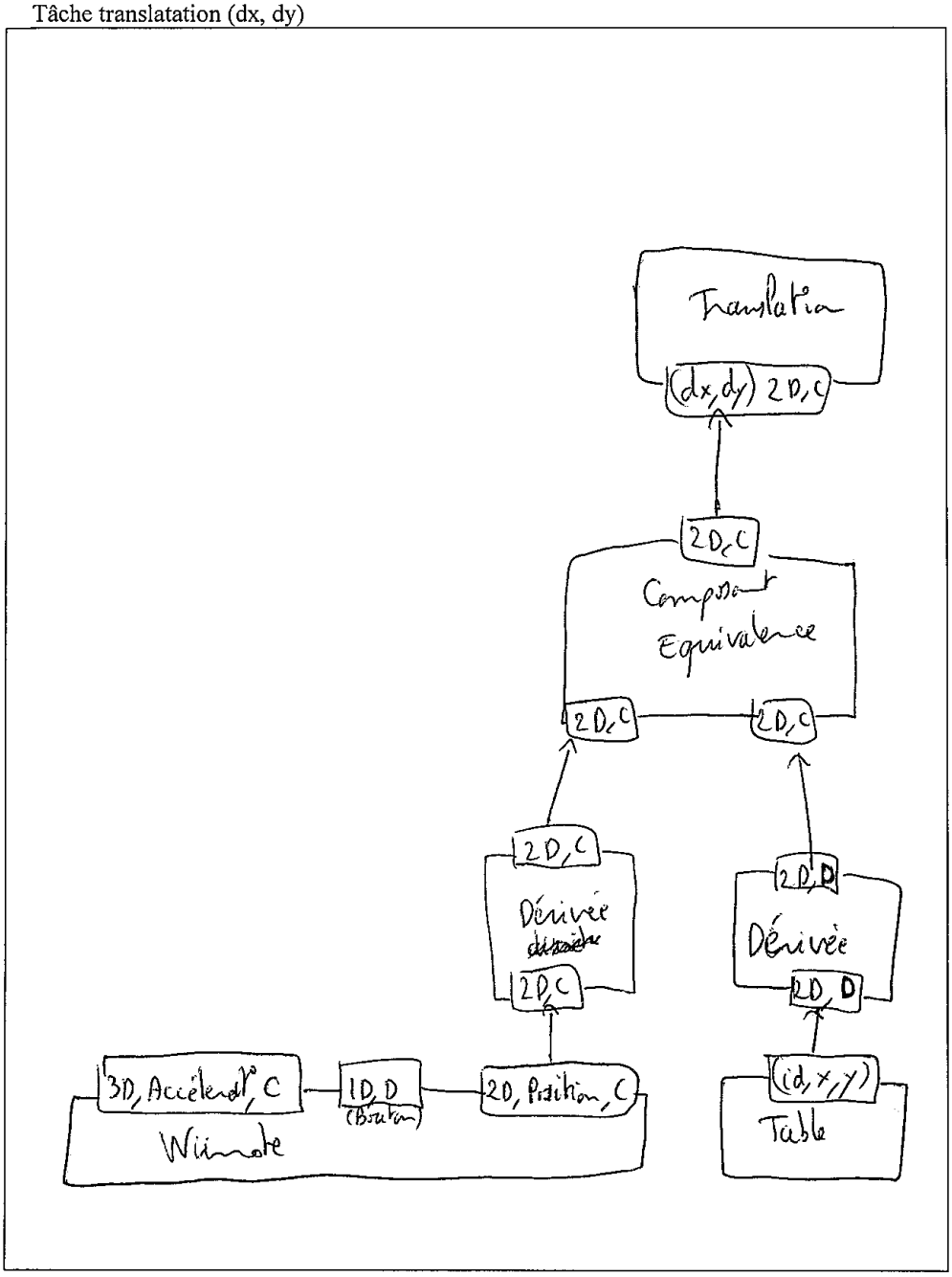


Descript° au dos de la dernière page →

Tâche zoom (x,y, z)



Tâche translatation (dx, dy)



Annexe C

Multimodal Component Description Language (MCDL)

Dans cet annexe, nous illustrons le MCDL à travers la description complète du composant Dispositif Wiimote.

```
<?xml version="1.0"?>
<InteractionComponent xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xmlns:MCDL="MCDL" xs:schemaLocation="MCDL http://www.dcs.gla.ac.uk/~adr/MCDL.xsd">

    <InteractionDescription>

        <Entity_Name>Wiimote</Entity_Name>
        <Type>Device</Type>
        <Dependence>Independent</Dependence>

        <DeviceDescription>

            <Name>Wii Remote</Name>
            <Manufacturer>Nintendo</Manufacturer>
            <Size>148mm (L) x 36.2mm (W) x 30.8mm (D)</Size>
            <Wireless>True</Wireless>
            <Connectivity>Bluetooth</Connectivity>
            <PowerSupply>Two AA batteries</PowerSupply>
            <SensorsDescription>
                <Sensor>
                    <Name>Accelerometer</Name>
                    <NumberofDimensions>3</NumberofDimensions>
                </Sensor>
                <Sensor>
                    <Name>Button</Name>
                    <Number>8</Number>
                </Sensor>
                <Sensor>
                    <Name>IR Sensor</Name>
                </Sensor>
            </SensorsDescription>

        </DeviceDescription>

    </InteractionDescription>


    <ComponentSpecification>
        <TailoredInterface>
            <Name>Accelerometer</Name>
            <parameter><name>x</name><type>int</type></parameter>
            <parameter><name>y</name><type>int</type></parameter>
            <parameter><name>z</name><type>int</type></parameter>
        </TailoredInterface>

        <NormalizedInterface>
            <Name>Button A</Name>
            <normtype>Buxton</normtype>
            <BuxtonInterface>
                <property>Pressure</property>
                <dim>1<dim>
                <parameter>
                    <name>buttonvalue</name>
                    <type>int</type><values>0-1</values>
                </parameter>
                <frequency>continuous</frequency>
            </BuxtonInterface>

        </NormalizedInterface>

        <NormalizedInterface>
            <Name>Button B</Name>
            <normtype>Buxton</normtype>
            <BuxtonInterface>
                <property>Pressure</property>
                <dim>1<dim>
```

```
        <parameter>
            <name>buttonvalue</name>
            <type>int</type><values>0-1</values>
        </parameter>
        <frequency>continuous</frequency>
    </BuxtonInterface>

</NormalizedInterface>


<NormalizedInterface>
    <Name>Button 1</Name>
    <normtype>Buxton</normtype>
    <BuxtonInterface>
        <property>Pressure</property>
        <dim>1<dim>
        <parameter>
            <name>buttonvalue</name>
            <type>int</type><values>0-1</values>
        </parameter>
        <frequency>discrete</frequency>
    </BuxtonInterface>

</NormalizedInterface>

<NormalizedInterface>
    <Name>Button 2</Name>
    <normtype>Buxton</normtype>
    <BuxtonInterface>
        <property>Pressure</property>
        <dim>1<dim>
        <parameter>
            <name>buttonvalue</name>
            <type>int</type><values>0-1</values>
        </parameter>
        <frequency>continuous</frequency>
    </BuxtonInterface>

</NormalizedInterface>

<TailoredInterface>
    <Name>ButtonPlus</Name>
    <parameter><name>buttonvalue</name><type>int</type></parameter>
</TailoredInterface>

<TailoredInterface>
    <Name>ButtonMinus</Name>
    <parameter><name>buttonvalue</name><type>int</type></parameter>
</TailoredInterface>

<TailoredInterface>
    <Name>ButtonArrows</Name>
    <parameter><name>buttonvalue</name><type>int</type></parameter>
</TailoredInterface>

    <TailoredInterface>
    <Name>ButtonHome</Name>
    <parameter><name>buttonvalue</name><type>int</type></parameter>
</TailoredInterface>

<TailoredInterface>
    <Name>IRInput</Name>
    <parameter><name>x</name><type>int</type></parameter>
    <parameter><name>y</name><type>int</type></parameter>
    <parameter><name>z</name><type>int</type></parameter>
</TailoredInterface>
```

```
        </ComponentSpecification>


        <ComponentImplementation>
            <Author>University of Grenoble</Author>
            <Licence>BSD</Licence>
            <Price>free</Price>
            <Type>OI</Type>
            <CIDL_id>openinterface.prototype.component.cplusplus.oiwii</CIDL_id>
        </ComponentImplementation>

    </InteractionComponent>
```

Annexe D

Component Interaction Description Language CIDL

Dans cet annexe, nous illustrons le CIDL, créé par un des partenaires du projet OpenInterface, l'Université Catholique de Louvain (UCL), à travers la description complète du composant Dispositif Wiimote.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Component PUBLIC "-//OpenInterface//DTD Component XML V0.1//EN"
"component.dtd">
<Component id="openinterface.prototype.component.cplusplus.oiwii">
  <Name value="OIWii"/>
  <Language value="c++"/>
  <Container>
    <Name value="OIWii"/>
    <Format value="so"/>
    <Location>OiWii</Location>
  </Container>

  <IO>
    <Facet id="oiwii">
      <Bin>
        <CustomType type="cppclass" name="" def="OIWii.h"/>
      </Bin>

      <Source id="ButtonA">
        <Callback>
          <Interface type="static_function" ref="GetButtonA">
            <Name value="my_cb"/>
            <Argument>
                <Param name="v">
                    <Descr>value of Button A </Descr>
                    <PrimitiveType name="int"/>
                </Param>
            </Argument>
          </Interface>
          <Setter>
            <Interface type="static_function">
              <Name value="GetButtonA"/>
              <Argument>
            <Param name="cback">
              <Descr>Callback for Button A info fonction </Descr>
              <CustomType type="cCallback" name="GetButtonA" def="OIManymouse.h"/>
            </Param>
              </Argument>
            </Interface>
          </Setter>
        </Callback>
      </Source>

      <Source id="ButtonB">
        <Callback>
          <Interface type="static_function" ref="GetButtonB">
            <Name value="my_cb"/>
            <Argument>
                <Param name="v">
                    <Descr>value of Button B </Descr>
                    <PrimitiveType name="int"/>
                </Param>
            </Argument>
          </Interface>
          <Setter>
            <Interface type="static_function">
              <Name value="GetButtonB"/>
              <Argument>
            <Param name="cback">
              <Descr>Callback for Button B info fonction </Descr>
              <CustomType type="cCallback" name="GetButtonB" def="OIManymouse.h"/>
            </Param>
              </Argument>
            </Interface>
          </Setter>
        </Callback>
```

```
</Source>

<Source id="Button1">
  <Callback>
    <Interface type="static_function" ref="GetButton1">
      <Name value="my_cb"/>
      <Argument>
          <Param name="v">
              <Descr>value of Button 1 </Descr>
              <PrimitiveType name="int"/>
          </Param>
      </Argument>
    </Interface>
    <Setter>
      <Interface type="static_function">
        <Name value="GetButton1"/>
        <Argument>
      <Param name="cback">
        <Descr>Callback for Button 1 info fonction </Descr>
        <CustomType type="cCallback" name="GetButton1" def="OIManymouse.h"/>
      </Param>
        </Argument>
      </Interface>
    </Setter>
  </Callback>
</Source>

<Source id="Button2">
  <Callback>
    <Interface type="static_function" ref="GetButton2">
      <Name value="my_cb"/>
      <Argument>
          <Param name="v">
              <Descr>value of Button 2 </Descr>
              <PrimitiveType name="int"/>
          </Param>
      </Argument>
    </Interface>
    <Setter>
      <Interface type="static_function">
        <Name value="GetButton2"/>
        <Argument>
      <Param name="cback">
        <Descr>Callback for Button 2 info fonction </Descr>
        <CustomType type="cCallback" name="GetButton2" def="OIManymouse.h"/>
      </Param>
        </Argument>
      </Interface>
    </Setter>
  </Callback>
</Source>

<Source id="ButtonPLUS">
  <Callback>
    <Interface type="static_function" ref="GetButtonPLUS">
      <Name value="my_cb"/>
      <Argument>
          <Param name="v">
              <Descr>value of Button PLUS </Descr>
              <PrimitiveType name="int"/>
          </Param>
      </Argument>
    </Interface>
    <Setter>
      <Interface type="static_function">
        <Name value="GetButtonPLUS"/>
```

```
        <Argument>
      <Param name="cback">
        <Descr>Callback for Button PLUS info fonction </Descr>
        <CustomType type="cCallback" name="GetButtonPLUS" def="OIManymouse.h"/>
      </Param>
        </Argument>
      </Interface>
    </Setter>
  </Callback>
</Source>

<Source id="ButtonMINUS">
  <Callback>
    <Interface type="static_function" ref="GetButtonMINUS">
      <Name value="my_cb"/>
      <Argument>
          <Param name="v">
              <Descr>value of Button MINUS </Descr>
              <PrimitiveType name="int"/>
          </Param>
      </Argument>
    </Interface>
    <Setter>
      <Interface type="static_function">
        <Name value="GetButtonMINUS"/>
        <Argument>
      <Param name="cback">
        <Descr>Callback for Button MINUS info fonction </Descr>
        <CustomType type="cCallback" name="GetButtonMINUS" def="OIManymouse.h"/>
      </Param>
        </Argument>
      </Interface>
    </Setter>
  </Callback>
</Source>

<Source id="ButtonHOME">
  <Callback>
    <Interface type="static_function" ref="GetButtonHOME">
      <Name value="my_cb"/>
      <Argument>
          <Param name="v">
              <Descr>value of Button HOME </Descr>
              <PrimitiveType name="int"/>
          </Param>
      </Argument>
    </Interface>
    <Setter>
      <Interface type="static_function">
        <Name value="GetButtonHOME"/>
        <Argument>
      <Param name="cback">
        <Descr>Callback for Button HOME info fonction </Descr>
        <CustomType type="cCallback" name="GetButtonHOME" def="OIManymouse.h"/>
      </Param>
        </Argument>
      </Interface>
    </Setter>
  </Callback>
</Source>

<Source id="ButtonARROWS">
  <Callback>
    <Interface type="static_function" ref="GetButtonARROWS">
      <Name value="my_cb"/>
      <Argument>
```

```
            <Param name="v">
                <Descr>value of Button ARROWS </Descr>
                <PrimitiveType name="string"/>
            </Param>
        </Argument>
      </Interface>
      <Setter>
        <Interface type="static_function">
          <Name value="GetButtonARROWS"/>
          <Argument>
        <Param name="cback">
          <Descr>Callback for Button ARROWS info fonction </Descr>
          <CustomType type="cCallback" name="GetButtonARROWS" def="OIManymouse.h"/>
        </Param>
          </Argument>
        </Interface>
      </Setter>
    </Callback>
</Source>

<Source id="ACC">
    <Callback>
      <Interface type="static_function" ref="GetACC">
        <Name value="my_cb"/>
        <Argument>
            <Param name="x">
                <Descr>X value of ACC </Descr>
                <PrimitiveType name="int"/>
            </Param>
        <Param name="y">
                <Descr>Y value of ACC </Descr>
                <PrimitiveType name="int"/>
            </Param>
        <Param name="z">
                <Descr>Z value of ACC </Descr>
                <PrimitiveType name="int"/>
            </Param>
        </Argument>
      </Interface>
      <Setter>
        <Interface type="static_function">
          <Name value="GetACC"/>
          <Argument>
        <Param name="cback">
          <Descr>Callback for ACC info fonction </Descr>
          <CustomType type="cCallback" name="GetACC" def="OIManymouse.h"/>
        </Param>
          </Argument>
        </Interface>
      </Setter>
    </Callback>
</Source>


<Source id="AbsIR">
    <Callback>
      <Interface type="static_function" ref="GetAbsIR">
        <Name value="my_cb"/>
        <Argument>
            <Param name="x">
                <Descr>Absolute X value of IR </Descr>
                <PrimitiveType name="float"/>
            </Param>
        <Param name="y">
                <Descr>Absolute Y value of IR </Descr>
                <PrimitiveType name="float"/>
```

```
                </Param>
              </Argument>
            </Interface>
            <Setter>
              <Interface type="static_function">
                <Name value="GetAbsIR"/>
                <Argument>
              <Param name="cback">
                <Descr>Callback for IR info fonction </Descr>
                <CustomType type="cCallback" name="GetAbsIR" def="OIManymouse.h"/>
              </Param>
                </Argument>
              </Interface>
            </Setter>
          </Callback>
        </Source>

      <Source id="RelIR">
          <Callback>
            <Interface type="static_function" ref="GetRelIR">
              <Name value="my_cb"/>
              <Argument>
                  <Param name="dx">
                      <Descr>Relative X value of IR </Descr>
                      <PrimitiveType name="int"/>
                  </Param>
              <Param name="dy">
                      <Descr>Relative Y value of IR </Descr>
                      <PrimitiveType name="int"/>
                  </Param>
              </Argument>
            </Interface>
            <Setter>
              <Interface type="static_function">
                <Name value="GetRelIR"/>
                <Argument>
              <Param name="cback">
                <Descr>Callback for IR info fonction </Descr>
                <CustomType type="cCallback" name="GetRelIR" def="OIManymouse.h"/>
              </Param>
                </Argument>
              </Interface>
            </Setter>
          </Callback>
        </Source>

          <Sink id="StartOiWii">
              <Interface type="static_function">
                  <Name value="startOiWii"/>
              </Interface>
              </Sink>


      </Facet>
    </IO>
  </Component>
```

Annexe E

Pipeline Description Language PDCL

Dans cet annexe, nous illustrons le PDCL, créé par un des partenaires du projet OpenInterface, l'Université Catholique de Louvain (UCL), à travers la description complète de l'assemblage implémentant le premier prototype du testbed Grands Espaces d'Informations.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Pipeline SYSTEM "c:/OpenInterface_v0.1/OpenInterface/common/pipeline.dtd">
<Pipeline>
    <ComponentList>
        <Component id="c1" descr="../installed_components/ImageViewer-lis.xml"/>
        <Component id="c3" descr="../installed_components/MultitelSpeechRecognition.xml"/
>
        <Component id="c4" descr="../installed_components/FingerTracker-win32.xml"/>
        <Component id="c0" descr="../installed_components/FingerPoint-win32.xml"/>
        <Component id="c5" descr="../installed_components/zoomAdapter-win32.xml"/>
    </ComponentList>

    <FacetList>
        <!--Viewer-->
        <Facet id="my_viewer" name="my_viewer" component="c1">
            <Factory>
                <Constant type="string" value="files/test.txt"/>
            </Factory>
        </Facet>

       <!--FingerTracker-->
        <Facet id="tracker" name="tracker" component="c4"/>

        <!--FingerPoint-->
        <Facet id="point" name="pointer" component="c0"/>

        <!--MultitelSpeech-->
        <Facet id="multitel_engine" name="VAD" component="c3"/>

      <!--ZoomAdapter-->
        <Facet id="zoomer" name="zoomer" component="c5"/>

        <Facet id="FingerTracker_CompleteLanguage" name="FingerTracker_CompleteLanguage"
component="c1"/>

    </FacetList>
    <PinList>

        <!--FingerTracker-->
        <Pin id="FingerTrackerData" facet="tracker" name="TrackerEvent"/>

        <!--FingerPoint-->
        <Pin id="PointData" facet="point" name="PointEvent"/>
        <Pin id="setFingersData" facet="point" name="setFingersData"/>

        <!--FingerTracker Adpater-->
        <Pin id="FingerTracker_EventSink" facet="FingerTracker_CompleteLanguage"
name="My_Finger_Manager"/>
        <Pin id="CompleteListener_Source" facet="FingerTracker_CompleteLanguage"
name="CompleteListener"/>

          <!--ZoomAdapter-->
        <Pin id="zoom_source" facet="zoomer" name="ZoomEvent"/>
        <Pin id="SpeechEventSink" facet="zoomer" name="newSpeechEvent"/>
        <Pin id="PointEventSink" facet="zoomer" name="newPointEvent"/>

        <!--MultitelSpeech-->
        <Pin id="InitSpeechEngine" facet="multitel_engine" name="initEngine"/>
        <Pin id="StartRecognition" facet="multitel_engine" name="startEngine"/>
        <Pin id="VoiceEventSource" facet="multitel_engine" name="VoiceEvent"/>


        <!--Viewer-->
        <Pin id="commandSelectorSink" facet="my_viewer" name="selectCommand"/>
        <Pin id="commandFinderHelper" facet="my_viewer" name="getCommandAt"/>
        <Pin id="screenToImageLocationConverter" facet="my_viewer"
```

```
name="getLocationOnImage"/>
        <Pin id="imageLoader" facet="my_viewer" name="loadImage"/>
        <Pin id="zoomCenterParam_Sink" facet="my_viewer" name="zoomCenterParam"/>

    </PinList>


    <Pipe>


   <Plug source="FingerTrackerData" target="setFingersData">
        <Filter>
                <TargetValue target="state">
                    <SourceValue source="state"/>
                </TargetValue>
                <TargetValue target="x">
                    <SourceValue source="x"/>
                </TargetValue>
                <TargetValue target="y">
                    <SourceValue source="y"/>
                </TargetValue>
                <TargetValue target="oid">
                    <SourceValue source="oid"/>
                </TargetValue>
            </Filter>
        </Plug>

        <Plug source="zoom_source" target="FingerTracker_EventSink">
            <Filter>
                <TargetValue target="type">
                    <SourceValue source="event"/>
                </TargetValue>
                <TargetValue target="x">
                    <SourceValue source="x"/>
                </TargetValue>
                <TargetValue target="y">
                    <SourceValue source="y"/>
                </TargetValue>
            </Filter>
        </Plug>

    <Plug source="PointData" target="PointEventSink">
            <Filter>
                <TargetValue target="xp">
                    <SourceValue source="x"/>
                </TargetValue>
                <TargetValue target="yp">
                    <SourceValue source="y"/>
                </TargetValue>
            </Filter>
        </Plug>

        <Plug source="VoiceEventSource" target="SpeechEventSink">
            <Filter>
                <TargetValue target="event">
                    <SourceValue source="words"/>
                </TargetValue>
            </Filter>
        </Plug>


        <MultiPlug source="CompleteListener_Source">
            <Plug source="setCommand" target="commandSelectorSink">
                <Filter>
                    <TargetValue target="cmd">
```

```
                    <SourceValue source="cmd"/>
                </TargetValue>
                </Filter>
            </Plug>

            <Plug source="zoomCenterParam" target="zoomCenterParam_Sink">
                <Filter>
                <TargetValue target="x">
                    <SourceValue source="centerX"/>
                </TargetValue>
                <TargetValue target="y">
                    <SourceValue source="centerY"/>
                </TargetValue>
                <TargetValue target="scale">
                    <SourceValue source="scale"/>
                </TargetValue>
            </Filter>
            </Plug>
        </MultiPlug>

        <Ignite source="InitSpeechEngine" threaded="no">
            <Constant type="string" value="../../third_part_lib/MT_SDK/
enu16fdd_02040001.lng"/>
            <Constant type="string" value="./enu_navigationCommands.app"/>
        </Ignite>

        <Ignite source="StartRecognition" threaded="yes" />

    </Pipe>
</Pipeline>
```

Bibliographie

[Abate et Jewell, 2001] Frank R. Abate et Elizabeth. Jewell. *The new Oxford American dictionary / edited by Frank Abate, Elizabeth J. Jewell*. Oxford University Press, New York :, 2001.

[Allen, 1983] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11) :832–843, 1983.

[Appert *et al.*, 2009] Caroline Appert, Stéphane Huot, Pierre Dragicevic, et Michel Beaudouin-Lafon. Flowstates : prototypage d'applications interactives avec des flots de données et des machines à états. In *IHM '09 : Proceedings of the 21st International Conference on Association Francophone d'Interaction Homme-Machine*, pages 119–128, New York, NY, USA, 2009. ACM.

[Apple, 2010] Apple, 2010. www.apple.com.

[Bailly *et al.*, 2005] Gilles Bailly, Laurence Nigay, et David Auber. 2M : un espace de conception pour l'interaction bi-manuelle. In *Conférence UBIMOB 2005, Deuxièmes Journées Francophones : Mobilité et Ubiquité 2005*, 2005.

[Ballagas *et al.*, 2005] Rafael Ballagas, Michael Rohs, et Jennifer G. Sheridan. Sweep and point and shoot : phonecam-based interactions for large public displays. In *CHI '05 : CHI '05 extended abstracts on Human factors in computing systems*, pages 1200–1203, New York, NY, USA, 2005. ACM.

[Ballagas *et al.*, 2007] Rafael Ballagas, Faraz Memon, Rene Reiners, et Jan Borchers. istuff mobile : rapidly prototyping new mobile phone interfaces for ubiquitous computing. In *CHI '07 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1107–1116, New York, NY, USA, 2007. ACM.

[Barboni *et al.*, 2007] Eric Barboni, Stéphane Conversy, David Navarre, et Philippe Palanque. Model-based engineering of widgets, user applications and servers compliant with arinc 661 specification. *Interactive Systems. Design, Specification, and Verification*, pages 25–38, 2007.

[Baroth et Hartsough, 1995] Ed Baroth et Chris Hartsough. Visual programming in the real world. *Visual object-oriented programming : concepts and environments*, pages 21–42, 1995.

[Batory et O'Malley, 1992] Don Batory et Sean O'Malley. The design and implementation of hierarchical software systems with reusable components. *ACM Trans. Softw. Eng. Methodol.*, 1(4) :355–398, 1992.

[Baudel, 1995] Thomas Baudel. *Aspects morphologiques de l'interaction homme-machine : étude de modèles d'interaction gestuels*. Thèse de doctorat, Université de Paris Sud, 1995.

[Beaudouin-Lafon et Mackay, 2000] Michel Beaudouin-Lafon et Wendy E. Mackay. Reification, polymorphism and reuse : three principles for designing visual interfaces. In *AVI '00 : Proceedings of the working conference on Advanced visual interfaces*, pages 102–109, New York, NY, USA, 2000. ACM.

[Beaudouin-Lafon, 1997] Michel Beaudouin-Lafon. Interaction instrumentale : de la manipulation directe à la réalité augmentée. In *Actes Neuvièmes journées francophones sur l'Interaction Homme Machine (IHM'97)*, Septembre 1997.

[Beaudouin-Lafon, 2004] Michel Beaudouin-Lafon. Designing interaction, not interfaces. In *AVI '04 : Proceedings of the working conference on Advanced visual interfaces*, pages 15–22, New York, NY, USA, 2004. ACM.

[Bérard, 1999] François Bérard. The perceptual window : Head motion as a new input stream. In *Proc. Interact99, Edinburgh, A. Sasse & ; C. Johnson Eds, IFIP IOS Press Publ*, 1999. 238 pages.

[Bernhaupt *et al.*, 2008] Regina Bernhaupt, Marianna Obrist, Astrid Weiss, Elke Beck, et Manfred Tscheligi. Trends in the living room and beyond : results from ethnographic studies using creative and playful probing. *Comput. Entertain.*, 6(1) :1–23, 2008.

[Berry, 1989] Gérard Berry. Real time programming : special purpose or general purpose languages. Technical report, Centre de Mathématiques Appliquées - CMA - Mines ParisTech - INRIA Sophia Antipolis - INRIA, 1989.

[Beugnard *et al.*, 1999] Antoine Beugnard, Jean-Marc Jézéquel, Noël Plouzeau, et Damien Watkins. Making components contract aware. *Computer*, 32(7) :38–45, 1999.

[Bieber et Carpenter, 2001] Guy Bieber et Jeff Carpenter. Introduction to service-oriented programming (rev 2.1). *OpenWings Whitepaper*, 2001.

[Biermann et Krishnaswamy, 1976] A. W. Biermann et R. Krishnaswamy. Constructing programs from example computations. *IEEE Trans. Softw. Eng.*, 2(3) :141–153, 1976.

[Blanch et Beaudouin-Lafon, 2006] Renaud Blanch et Michel Beaudouin-Lafon. Programming rich interactions using the hierarchical state machine toolkit. In *AVI '06 : Proceedings of the working conference on Advanced visual interfaces*, pages 51–58, New York, NY, USA, 2006. ACM.

[Blanch et Ortega, 2009] Renaud Blanch et Michaël Ortega. Rake cursor : improving pointing performance with concurrent input channels. In *CHI '09 : Proceedings of the 27th international conference on Human factors in computing systems*, pages 1415–1418, New York, NY, USA, 2009. ACM.

[Blanch, 2005] Renaud Blanch. *Architecture logicielle et outils pour les interfaces hommes-machines graphiques avancées*. Thèse de doctorat, Laboratoire de Recherche en Informatique de l'Université Paris-Sud XI, 2005.

[Bolt et Herranz, 1992] Richard A. Bolt et Edward Herranz. Two-handed gesture in multi-modal natural dialog. In *UIST '92 : Proceedings of the 5th annual ACM symposium on User interface software and technology*, pages 7–14, New York, NY, USA, 1992. ACM.

[Bolt, 1980] Richard Bolt. Put-that-there : voice and gesture at the graphics interface. In *Proceedings of ISIGGRAPHÕ80*, pages 262–270, 1980.

[Bolt, 1982] Richard A. Bolt. Eyes at the interface. In *Proceedings of the 1982 conference on Human factors in computing systems*, pages 360–362, New York, NY, USA, 1982. ACM.

[Bolt, 1987] Richard Bolt. The integrated multi-modal interface. *Transactions of the Institute of Electronics, Information and Communication Engineers (Japan)*, J70-D(11) :2017–2025, November 1987.

[Bouchet *et al.*, 2004] Jullien Bouchet, Laurence Nigay, et Thierry Ganille. Icare software components for rapidly developing multimodal interfaces. In *ICMI '04 : Proceedings of the 6th international conference on Multimodal interfaces*, pages 251–258, New York, NY, USA, 2004. ACM.

[Bouchet *et al.*, 2008] Jullien Bouchet, Laya Madani, Laurence Nigay, Catherine Oriat, et Ioannis Parissis. Formal testing of multimodal interactive systems. pages 36–52, 2008.

[Bouchet et Nigay, 2004] Jullien Bouchet et Laurence Nigay. ICARE : Approche à composants pour l'interaction multimodale. In *Actes des Premières Journées Francophones : Mobilité et Ubiquité 2004*, pages 36–43, 2004. Nice Sophia-Antipolis, France.

[Bouchet, 2006] Jullien Bouchet. *Ingénierie de l'interaction multimodale en entrée. Approche à composants ICARE*. Thèse de doctorat, L'UNIVERSITÉ JOSEPH FOURIER - GRENOBLE I, Décembre 2006.

[Bourguet, 2002] Marie-Luce Bourguet. Outil de prototypage pour la conception et l'evaluation d'interfaces utilisateur multimodales. In *IHM '02 : Proceedings of the 14th French-speaking conference on Human-computer interaction (Conférence Francophone sur l'Interaction Homme-Machine)*, pages 239–242, New York, NY, USA, 2002. ACM.

[Bourguet, 2003] Marie-Luce Bourguet. Designing and prototyping multimodal commands. In *Human-Computer Interaction INTERACT'03*, pages 717–720. IFIP, IOS Press, September 2003.

[Buechley *et al.*, 2008] Leah Buechley, Mike Eisenberg, Jaime Catchen, et Ali Crockett. The lilypad arduino : using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *CHI '08 : Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 423–432, New York, NY, USA, 2008. ACM.

[Buxton et Myers, 1986] W. Buxton et B. Myers. A study in two-handed input. In *CHI '86 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 321–326, New York, NY, USA, 1986. ACM.

[Buxton, 1983] Will Buxton. Lexical and pragmatic considerations of input structures. In *ACM SIGGRAPH*, pages 31–37, 1983.

[Buxton, 2007] Bill Buxton. *Sketching User Experiences : Getting the Design Right and the Right Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[Camurri *et al.*, 2004] A. Camurri, P. Coletta, A. Massari, B. Mazzarino, Peri M., Ricchetti M., Ricci A., et Volpe G. Toward real-time multimodal processing : Eyesweb 4.0. In *Proc. AISB 2004 Convention : Motion, Emotion and Cognition*, Leeds, UK, 2004.

[Card *et al.*, 1991] Stuart K. Card, Jock D. Mackinlay, et George G. Robertson. A morphological analysis of the design space of input devices. *ACM Trasactions on Information Systems*, 9(2) :99–122, April 1991.

[Carr, 1991] Robert M. Carr. The point of the pen. *BYTE*, 16(2) :211–ff., 1991.

[CHI, 2009] CHI. Chi '09 : Proceedings of the 27th international conference on human factors in computing systems, 2009. General Chair-Olsen,Jr., Dan R. and General Chair-Arthur, Richard B. and Program Chair-Hinckley, Ken and Program Chair-Morris, Meredith Ringel and Program Chair-Hudson, Scott and Program Chair-Greenberg, Saul.

[Clarisse et Chang, 1986] O. Clarisse et SK. Chang. Vicon ; a visual icon manager. In *Visual Languages*, pages 151–190, 1986.

[CNRTL-CNRS, 2009] CNRTL-CNRS. Centre national de ressources textuelles et lexicales, http ://www.cnrtl.fr/, October 2009.

[Cohen *et al.*, 1997] Philip R. Cohen, Michael Johnston, David McGee, Sharon Oviatt, Jay Pittman, Ira Smith, Liang Chen, et Josh Clow. Quickset : multimodal interaction for distributed applications. In *MULTIMEDIA '97 : Proceedings of the fifth ACM international conference on Multimedia*, pages 31–40, New York, NY, USA, 1997. ACM.

[Cohen *et al.*, 1998] Philip R. Cohen, Adam Cheyer, Michelle Wang, et Soon Cheol Baeg. An open agent architecture. pages 197–204, 1998.

[Cohen et McGee, 2004] Philip R. Cohen et David R. McGee. Tangible multimodal interfaces for safety-critical applications. *Commun. ACM*, 47(1) :41–46, 2004.

[Collet, 1999] Christophe Collet. *Capture et suivi du regard par un système de vision dans le contexte de la communication homme-machine*. Thèse de doctorat, École Normale Supérieure de Cachan, 15 janvier 1999.

[COM, 2009] COM. Component object model. http ://www.microsoft.com/com, 2009.

[Computer, 1987] Apple Computer. *Apple Human Interface Guidelines : The Apple Desktop Interface*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1987.

[Corba, 2002] Corba. Formal/02-06-65 : Corba components, v3.0 full specification. OMG, 2002. - CORBA. www.corba.org.

[Corradini *et al.*, 2002] Andrea Corradini, Richard M. Wesson, et Philip R. Cohen. A map-based system using speech and 3d gestures for pervasive computing. In *ICMI '02 : Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, page 191, Washington, DC, USA, 2002. IEEE Computer Society.

[Coutaz *et al.*, 1996] Joelle Coutaz, Daniel Salber, Eric Carraux, et Nathalie Portolan. Neimo, a multiworkstation usability lab for observing and analyzing multimodal interaction. In *Proceedings of CHI'96*, pages 402–403. ACM Press, 1996.

[Coutaz et Nigay, 1994] Joëlle Coutaz et Laurence Nigay. Les propriétés CARE dans les interfaces multimodales. In *Actes de la conférence IHM'94, Lille*, pages 7–14, 1994.

[Coutaz, 1987] Joëlle Coutaz. PAC, on object oriented model for dialog design. In *Interact'87*, 1987. 6 pages.

[Coutaz, 1993] Joëlle Coutaz. *Architectural Design for User Interfaces ; The Encyclopedia of Software Engineering*, pages 38–49. J. Marciniak Ed., Wiley & Sons Publ, 1993. J. Marciniak Ed., Wiley & Sons Publ.

[Coutrix *et al.*, 2005] Céline Coutrix, Laurence Nigay, et Philippe Renevier. Modèle d'interaction mixte : la réalité mixte à la lumière des modalités d'interaction. In *UbiMob '05 : Proceedings of the 2nd French-speaking conference on Mobility and ubiquity computing*, pages 153–160, New York, NY, USA, 2005. ACM.

[Coyette, 2007] Adrien Coyette. *A Methodological Framework for Multi-Fidelity Sketching for User Interfaces.* Thèse de doctorat, Université Catholique de Louvain, 2007.

[Davis *et al.*, 2007] Richard C. Davis, T. Scott Saponas, Michael Shilman, et James A. Landay. Sketchwizard : Wizard of oz prototyping of pen-based user interfaces. In *Proceedings of UIST'07*, pages 119–128. ACM Press, 2007.

[Davis et Ellis, 1964] M. R. Davis et T. O. Ellis. The rand tablet : a man-machine graphical communication device. In *AFIPS '64 (Fall, part I) : Proceedings of the October 27-29, 1964, fall joint computer conference, part I*, pages 325–331, New York, NY, USA, 1964. ACM.

[Davis, 1994] James Davis. Visual gesture recognition. In *IEE proceedings. Vision, image and signal processing*. Univ. cent. Florida, computer vision lab, 1994.

[Dietz et Leigh, 2001] Paul Dietz et Darren Leigh. Diamondtouch : a multi-user touch technology. In *Proceedings of UIST'01*, pages 219–226. ACM Press, 2001.

[Dow *et al.*, 2005] Steven Dow, Blair MacIntyre, Jaemin Lee, Christopher Oezbek, Jay David Bolter, et Maribeth Gandy. Wizard of oz support throughout an iterative design process. *IEEE Pervasive Computing*, 4(4) :18–26, 2005.

[Dragicevic et Fekete, 2004] Pierre Dragicevic et Jean-Daniel Fekete. Support for input adaptability in the icon toolkit. In *ICMI '04 : Proceedings of the 6th international conference on Multimodal interfaces*, pages 212–219, New York, NY, USA, 2004. ACM.

[Dragicevic, 2004] Pierre Dragicevic. *Un modèle d'interaction en entrée pour des systèmes interactifs multi-dispositifs hautement configurables*. Thèse de doctorat, Ecole Nationale Supérieure des Techniques Industrielles et des Mines de Nantes, mars 2004.

[Dumas *et al.*, 2008a] Bruno Dumas, Denis Lalanne, Dominique Guinard, Reto Koenig, et Rolf Ingold. Strengths and weaknesses of software architectures for the rapid creation of tangible and multimodal interfaces. In *TEI '08 : Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 47–54, New York, NY, USA, 2008. ACM.

[Dumas *et al.*, 2008b] Bruno Dumas, Denis Lalanne, et Rolf Ingold. Démonstration : Hephaistk, une boîte à outils pour le prototypage d'interfaces multimodales. In *IHM '08 : Proceedings of the 20th International Conference of the Association Francophone d'Interaction Homme-Machine*, pages 215–216, New York, NY, USA, 2008. ACM.

[Dumas *et al.*, 2008c] Bruno Dumas, Denis Lalanne, et Rolf Ingold. Prototyping multimodal interfaces with smuiml modeling language. In *Proc. of CHI 2008 Workshop on UIDLs for Next Generation User Interfaces*, pages 63–66, Florence (Italy), April 2008. ACM Press.

[Eckert *et al.*, 1979] R. Eckert, G. Enderle, K. Kansy, et F. Prester. Gks'79 - proposal of a standard for a graphical kernel system. In *Proceedings of Eurographics'79*, 1979.

[EJB, 2009] EJB. Enterprise java beans. http ://java.sun.com/products/ejb/, 2009.

[EMMA, 2010] W3C Language EMMA. http ://www.w3.org/tr/emma/, 2010.

[Escoffier *et al.*, 2007] C. Escoffier, R. S. Hall, et P. Lalanda. ipojo an extensible service-oriented component framework. In *IEEE International Conference on Service Computing (SCC'07)*, 2007.

[Foley *et al.*, 1984] James D. Foley, Victor L. Wallace, et Peggy Chan. The human factors of computer graphics interaction techniques. *IEEE Comput. Graph. Appl.*, 4(11) :13–48, 1984.

[Gauthy-Sinéchal et Vandercammen, 2005] Martine Gauthy-Sinéchal et Marc Vandercammen. *Études de marchés : méthodes et outils*. De Boeck Université, 2 edition, 2005.

[Gautier et Guernic, 1987] Thierry Gautier et Paul Le Guernic. Signal : A declarative language for synchronous programming of real-time systems. In *FPCA*, pages 257–277, 1987.

[Genesereth et Ketchpel, 1994] Michael R. Genesereth et Steven P. Ketchpel. Software agents. *Commun. ACM*, 37(7) :48–ff., 1994.

[Goldberg et Robson, 1983] Adele Goldberg et David Robson. *Smalltalk-80 : the language and its implementation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1983.

[Gray *et al.*, 2007] Phil Gray, Andrew Ramsay, et Marcos Serrano. A demonstration of the openinterface interaction development environment. In *Adjunct Proceedings of UIST'07*, pages 39–40. ACM Press, 2007.

[Green, 1989] T. R. G. Green. Cognitive dimensions of notations. In *Proceedings of the fifth conference of the British Computer Society, Human-Computer Interaction Specialist Group on People and computers V*, pages 443–460, New York, NY, USA, 1989. Cambridge University Press.

[Green, 2000] Thomas R. G. Green. Instructions and descriptions : some cognitive aspects of programming and similar activities. In *AVI '00 : Proceedings of the working conference on Advanced visual interfaces*, pages 21–28, New York, NY, USA, 2000. ACM.

[Greenberg et Buxton, 2008] Saul Greenberg et Bill Buxton. Usability evaluation considered harmful (some of the time). In *CHI '08 : Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 111–120, New York, NY, USA, 2008. ACM.

[Greenberg et Fitchett, 2001] Saul Greenberg et Chester Fitchett. Phidgets : easy development of physical interfaces through physical widgets. In *UIST '01 : Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 209–218, New York, NY, USA, 2001. ACM.

[Halbert, 1993] Daniel C. Halbert. Smallstar : programming by demonstration in the desktop metaphor. *Watch what I do : programming by demonstration*, pages 103–123, 1993.

[Halbwachs *et al.*, 1991] N. Halbwachs, P. Caspi, P. Raymond, et D. Pilaud. The synchronous dataflow programming language lustre. In *Proceedings of the IEEE*, pages 1305–1320, 1991.

[Harada *et al.*, 2006] Susumu Harada, James A. Landay, Jonathan Malkin, Xiao Li, et Jeff A. Bilmes. The vocal joystick : : evaluation of voice-based cursor control techniques. In *Assets '06 : Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility*, pages 197–204, New York, NY, USA, 2006. ACM.

[Hartmann *et al.*, 2006] Björn Hartmann, Scott R. Klemmer, Michael Bernstein, Leith Abdulla, Brandon Burr, Avi Robinson-Mosher, et Jennifer Gee. Reflective physical prototyping through integrated design, test, and analysis. In *UIST '06 : Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 299–308, New York, NY, USA, 2006. ACM.

[Hartmann *et al.*, 2007] Björn Hartmann, Leith Abdulla, Manas Mittal, et Scott R. Klemmer. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *CHI '07 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 145–154, New York, NY, USA, 2007. ACM.

[Heap et Hogg, 1996] T. Heap et D. Hogg. Towards 3d hand tracking using a deformable model. In *FG '96 : Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition (FG '96)*, page 140, Washington, DC, USA, 1996. IEEE Computer Society.

[Hewitt, 1984] W. T. Hewitt. Phigs – programmer's hierarchical interactive graphics system. In *Computer Graphics Forum*, volume 3(4), pages 299–300, 1984.

[Hinckley *et al.*, 1998] Ken Hinckley, Mary Czerwinski, et Mike Sinclair. Interaction and modeling techniques for desktop two-handed input. In *UIST '98 : Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 49–58, New York, NY, USA, 1998. ACM.

[Huang *et al.*, 1993] Xuedong Huang, Fileno Alleva, Mei-Yuh Hwang, et Ronald Rosenfeld. An overview of the sphinx-ii speech recognition system. In *HLT '93 : Proceedings of the workshop on Human Language Technology*, pages 81–86, Morristown, NJ, USA, 1993. Association for Computational Linguistics.

[Igarashi et Hughes, 2001] Takeo Igarashi et John F. Hughes. Voice as sound : using non-verbal voice input for interactive control. In *UIST '01 : Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 155–156, New York, NY, USA, 2001. ACM.

[Interface-Z, 2010] Interface-Z. http ://www.interface-z.com, 2010.

[Ishii *et al.*, 2004] H. Ishii, C. Ratti, B. Piper, Y. Wang, A. Biderman, et E. Ben-Joseph. Bringing clay and sand into digital design — continuous tangible user interfaces. *BT Technology Journal*, 22(4) :287–299, 2004.

[Johnston *et al.*, 2004] Wesley M. Johnston, J. R. Paul Hanna, et Richard J. Millar. Advances in dataflow programming languages. *ACM Comput. Surv.*, 36(1) :1–34, 2004.

[Jones *et al.*, 2007] William Jones, Jared Spool, Jonathan Grudin, Victoria Bellotti, et Mary Czerwinski. "get real !" : what's wrong with hci prototyping and how can we fix it ? In *CHI '07 : CHI '07 extended abstracts on Human factors in computing systems*, pages 1913–1916, New York, NY, USA, 2007. ACM.

[Kaiser *et al.*, 2003] Ed Kaiser, Alex Olwal, David McGee, Hrvoje Benko, Andrea Corradini, Xiaoguang Li, Phil Cohen, et Steven Feiner. Mutual disambiguation of 3d multimodal interaction in augmented and virtual reality. In *Proceedings of the 5th international conference on Multimodal interfaces*, ICMI '03, pages 12–19, New York, NY, USA, 2003. ACM.

[Kelley, 1984] J. F. Kelley. An iterative design methodology for user-friendly natural language office information applications. *ACM Trans. Inf. Syst.*, 2(1) :26–41, 1984.

[Klemmer *et al.*, 2000] Scott R. Klemmer, Anoop K. Sinha, Jack Chen, James A. Landay, Nadeem Aboobaker, et Annie Wang. Suede, a wizard of oz prototyping tool for speech user interfaces. In *Proc. of UIST'00*, page 1Ð10. ACM Press, 2000.

[König *et al.*, 2010] Werner A. König, Roman Rädle, et Harald Reiterer. Interactive design of multimodal user interfaces - reducing technical and visual complexity. In Springer, éditeur, *Journal on Multimodal User Interfaces*, volume 3, pages 197–213, 2010.

[Krasner et Pope, 1988] Glenn E. Krasner et 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, 1988.

[Krueger et Casey, 2000] R.A. Krueger et M.A. Casey. *Focus Groups : A Practical Guide for Applied Research*. Sage Publication Publisher, 2000.

[Lawson *et al.*, 2009] Jean-Yves Lionel Lawson, Ahmad-Amr Al-Akkad, Jean Vanderdonckt, et Benoit Macq. An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components. In *EICS '09 : Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 245–254, New York, NY, USA, 2009. ACM.

[Lécuyer *et al.*, 2008] Anatole Lécuyer, Fabien Lotte, Richard B. Reilly, Robert Leeb, Michitaka Hirose, et Mel Slater. Brain-computer interfaces, virtual reality, and videogames. *Computer*, 41(10) :66–72, 2008.

[Lemur, 2010] Lemur. http ://www.jazzmutant.com, 2010.

[Letessier et Bérard, 2004] Julien Letessier et François Bérard. Visual tracking of bare fingers for interactive surfaces. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 119–122, 2004.

[Li *et al.*, 2004] Yang Li, Jason I. Hong, et James A. Landay. Topiary : a tool for prototyping location-enhanced applications. In *UIST '04 : Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 217–226, New York, NY, USA, 2004. ACM.

[Li *et al.*, 2007] Yang Li, Jason I. Hong, et James A. Landay. Design challenges and principles for wizard of oz testing of location-enhanced applications. In *Proceedings of IEEE Pervasive Computing*, volume 6, pages 70–75, 2007.

[Lieberman et Watson, 2009] Zachary Lieberman et Theodore Watson. Openframeworks, http ://www.openframeworks.cc, October 2009.

[Lieberman, 1982] Henry Lieberman. Constructing graphical user interfaces by example. In *Graphics Interface'82*, pages 295–302, Toronto, Ont, Mar. 17-21 1982.

[Lieberman, 1993] Henry Lieberman. Mondrian : a teachable graphical editor. In *CHI '93 : Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, page 144, New York, NY, USA, 1993. ACM.

[Lim *et al.*, 2008] Youn-Kyung Lim, Erik Stolterman, et Josh Tenenberg. The anatomy of prototypes : Prototypes as filters, prototypes as manifestations of design ideas. In ACM Press, éditeur, *Transactions on Computer-Human Interaction (TOCHI)*, volume 15, 2008.

[Mackay, 1998] Wendy E. Mackay. Augmented reality : linking real and virtual worlds : a new paradigm for interacting with computers. In *AVI '98 : Proceedings of the working conference on Advanced visual interfaces*, pages 13–21, New York, NY, USA, 1998. ACM.

[Malik *et al.*, 2005] Shahzad Malik, Abhishek Ranjan, et Ravin Balakrishnan. Interacting with large displays from a distance with vision-tracked multi-finger gestural input. In *UIST '05 : Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 43–52, New York, NY, USA, 2005. ACM.

[Martin *et al.*, 1999] David L. Martin, Adam J. Cheyer, et Douglas B. Moran. The open agent architecture : A framework for building distributed software systems. *Applied Artificial Intelligence*, 13(1-2) :91–128, January-March 1999. OAA.

[Martin *et al.*, 2005] Nicolas Martin, Samuel Degrande, et Christophe Chaillou. Une utilisation du modèle mvc pour une plate-forme de travail virtuel. In *IHM 2005 : Proceedings of the 17th international conference on Francophone sur l'Interaction Homme-Machine*, pages 131–138, New York, NY, USA, 2005. ACM.

[Martin, 1999] Jean-Claude Martin. Tycoon six primitive types of cooperation for observing, evaluating and specifying cooperations. In *AAAI Technical Report FS-99-03*, 1999.

[Marvelig, 2009] Marvelig. Pépinière d'expérimentation scientifique du laboratoire d'informatique de grenoble. https ://marvelig.liglab.fr, 2009.

[McCurdy *et al.*, 2006] Michael McCurdy, Christopher Connors, Guy Pyrzak, Bob Kanefsky, et Alonso Vera. Breaking the fidelity barrier : an examination of our current characterization of prototypes and an example of a mixed-fidelity success. In *CHI '06 : Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1233–1242, New York, NY, USA, 2006. ACM.

[McGee-Lennon *et al.*, 2009] Marilyn Rose McGee-Lennon, Andrew Ramsay, David McGookin, et Philip Gray. User evaluation of oide : a rapid prototyping platform for multimodal interaction. In *EICS '09 : Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 237–242, New York, NY, USA, 2009. ACM.

[Millán, 2006] José del R. Millán. Brain-computer interaction. In *UIST '06 : Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 277–278, New York, NY, USA, 2006. ACM.

[Myers *et al.*, 1995] Brad A. Myers, Dario A. Giuse, Roger B. Dannenberg, Brad Vander Zanden, David S. Kosbie, Edward Pervin, Andrew Mickish, et Philippe Marchal. Garnet : comprehensive support for graphical, highly interactive user interfaces. *Human-computer interaction : toward the year 2000*, pages 357–372, 1995.

[Myers *et al.*, 2000] Brad Myers, Scott E. Hudson, et Randy Pausch. Past, present, and future of user interface software tools. *ACM Trans. Comput.-Hum. Interact.*, 7(1) :3–28, 2000.

[Myers et Buxton, 1986] Brad A. Myers et William Buxton. Creating highly-interactive and graphical user interfaces by demonstration. In *SIGGRAPH '86 : Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 249–258, New York, NY, USA, 1986. ACM.

[Myers, 1986] B. A. Myers. Visual programming, programming by example, and program visualization : a taxonomy. In *CHI '86 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 59–66, New York, NY, USA, 1986. ACM.

[Myers, 1991] Brad A. Myers. Separating application code from toolkits : eliminating the spaghetti of call-backs. In *UIST '91 : Proceedings of the 4th annual ACM symposium on User interface software and technology*, pages 211–220, New York, NY, USA, 1991. ACM.

[Myers, 1998] Brad A. Myers. A brief history of human computer interaction technology. *ACM interactions*, 5(2) :44–54, March 1998.

[Navarre *et al.*, 2008] David Navarre, Philippe Palanque, et Sandra Basnyat. A formal approach for user interaction reconfiguration of safety critical interactive systems. In *SAFECOMP '08 : Proceedings of the 27th international conference on Computer Safety, Reliability, and Security*, pages 373–386, Berlin, Heidelberg, 2008. Springer-Verlag.

[Newman, 1994] William Newman. A preliminary analysis of the products of hci research, using pro forma abstracts. In *CHI '94 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 278–284, New York, NY, USA, 1994. ACM.

[Nigay et Coutaz, 1991] Laurence Nigay et Joëlle Coutaz. Building user interfaces : Organizing software agents. In *Proc. ESPRIT'91 conference, Bruxelles*, pages 707–719, 1991.

[Nigay et Coutaz, 1996] Laurence Nigay et Joëlle Coutaz. Espaces conceptuels pour l'interaction multimédia et multimodale. *TSI, spécial Multimédia et Collecticiel, AFCET & Hermes Publ.*, 15(9) :1195–1225, 1996.

[Olsen, 2007] Dan R. Olsen, Jr. Evaluating user interface systems research. In *UIST '07 : Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 251–258, New York, NY, USA, 2007. ACM.

[Olwal *et al.*, 2003] Alex Olwal, Hrvoje Benko, et Steven Feiner. Senseshapes : Using statistical geometry for object selection in a multimodal augmented reality system. In *ISMAR '03 : Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, page 300, Washington, DC, USA, 2003. IEEE Computer Society.

[OpenInterface, 2006] OpenInterface, 2006. European project. IST Framework 6 STREP funded by the European Commission (FP6-35182). www.oi-project.org.

[Ortega et Nigay, 2009] Michael Ortega et Laurence Nigay. AirMice : Finger gesture for 2D and 3D interaction. In *Conference Proceedings of INTERACT '09, the*

Twelfth IFIP TC13 International Conference on Human-Computer Interaction, Springer LNCS (Lecture Notes in Computer Science), 2009.

[Oviatt *et al.*, 2000] Sharon Oviatt, Phil Cohen, Lizhong Wu, John Vergo, Lisbeth Duncan, Bernhard Suhm, Josh Bers, Thomas Holzman, Terry Winograd, James Landay, Jim Larson, et David Ferro. Designing the user interface for multimodal speech and pen-based gesture applications : state-of-the-art systems and future research directions. *Hum.-Comput. Interact.*, 15(4) :263–322, 2000.

[Oviatt *et al.*, 2005] Sharon Oviatt, Rebecca Lunsford, et Rachel Coulston. Individual differences in multimodal integration patterns : What are they and why do they exist ? In *Proceedings of CHI'05*, pages 241–249. ACM Press, 2005.

[Oviatt, 1999] Sharon Oviatt. Ten myths of multimodal interaction. In *Communications of the ACM*, volume 42, Issu 11, pages 74–81. ACM Press, 1999.

[Palanque *et al.*, 1993] Philippe Palanque, Remi Bastide, et C.Sibertin-Blanc. Formal specification, design and validation of user-driven interfaces using a petri net and objects based model. In *Research Symposium INTERCHI'93*, Amsterdam, 1993.

[Palanque *et al.*, 2007] Philippe Palanque, Sandra Basnyat, Regina Bernhaupt, Ronald Boring, Chris Johnson, et Peter Johnson. Beyond usability for safety critical systems : how to be sure (safe, usable, reliable, and evolvable) ? In *CHI '07 : CHI '07 extended abstracts on Human factors in computing systems*, pages 2133–2136, New York, NY, USA, 2007. ACM.

[Petri, 1962] C.A. Petri. Kommunikation mit automaten. In *Bonn : Institute f/Jr Instrumentelle Mathematik, Schriften des IIM Nr. 3*, 1962.

[Pfaff et Hagen, 1985] Pfaff et Ten Hagen. Uims. In *Seeheim workshop on User Interface Management Systems*. Springer-Verlag, 1985.

[Puckette, 1988] Miller Puckette. The patcher. In *Proceedings, ICMC. San Francisco : International Computer Music Association*, pages 420–429, 1988.

[Puckette, 1996] Miller Puckette. Pure data. In *Proceedings, International Computer Music Conference. San Francisco : International Computer Music Association*, pages 269–272, 1996.

[Puk et McConnell, 1986] Richard F Puk et John I McConnell. Gks-3d : a three-dimensional extension to the graphical kernel system. *IEEE Comput. Graph. Appl.*, 6(9) :42–49, 1986.

[Quartz Composer, 2010] Apple Quartz Composer. http ://developer.apple.com/, 2010.

[Reas et Fry, 2003] Casey Reas et Benjamin Fry. Processing : a learning environment for creating interactive web graphics. *ACM SIGGRAPH 2003 Web Graphics*, page 1, 2003.

[Rousseau, 2006] Cyril Rousseau. *Présentation multimodale et contextuelle de l'information*. Thèse de doctorat, Université Paris-Sud XI, Orsay, France, 13 Décembre 2006.

[Saarinen *et al.*, 2005] Rami Saarinen, Janne Järvi, Roope Raisamo, et Jouni Salo. Agent-based architecture for implementing multimodal learning environments for visually impaired children. In *ICMI '05 : Proceedings of the 7th international conference on Multimodal interfaces*, pages 309–316, New York, NY, USA, 2005. ACM.

[Salber et Coutaz, 1993] Daniel Salber et Joëlle Coutaz. A wizard of oz platform for the study of multimodal systems. In *CHI '93 : INTERACT '93 and CHI '93 conference companion on Human factors in computing systems*, pages 95–96, New York, NY, USA, 1993. ACM.

[Salber, 1992] Daniel Salber. Le projet NEIMO : une plate-forme magicien d'oz multimodale. In *IHM'92, Paris, Telecom Paris Publ*, pages 66–71, 1992.

[Sanou *et al.*, 2006] Loe Sanou, Patrick Girard, et Laurent Guittet. La programmation sur exemple : principes, utilisation et utilité pour les applications interactives. In *Ergo'IA 2006, ESTIA, Bidart-Biarritz*, pages 201–208, 2006.

[Schlienger et Chatty, 2007] Schlienger et Stephane Chatty. An iterative and participatory hci design process in the industry context : Bringing together utility, usability and innovation ... within budget. In *Interfaces Magazine, HCI Publications*, volume 72, 2007.

[Schmucker, 1996] Kurt J. Schmucker. Rapid prototyping using visual programming tools. In *CHI '96 : Conference companion on Human factors in computing systems*, pages 359–360, New York, NY, USA, 1996. ACM.

[Schyn *et al.*, 2003] Amélie Schyn, David Navarre, Philippe Palanque, et Luciana Porcher Nedel. Formal description of a multimodal interaction technique in an immersive virtual reality application. In *IHM 2003 : Proceedings of the 15th French-speaking conference on human-computer interaction on 15eme Conference Francophone sur l'Interaction Homme-Machine*, pages 150–157, New York, NY, USA, 2003. ACM.

[Schyn, 2005] Amélie Schyn. *Une approche fondée sur les modèles pour l'ingénierie des systèmes interactifs multimodaux*. Thèse de doctorat, Université Toulouse III, 2005.

[Serrano *et al.*, 2006] Marcos Serrano, Laurence Nigay, Rachel Demumieux, Jérôme Descos, et Patrick Losquin. Multimodal interaction on mobile phones : development and evaluation using acicare. In *MobileHCI '06 : Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 129–136, New York, NY, USA, 2006. ACM.

[Serrano et Nigay, 2009] Marcos Serrano et Laurence Nigay. Temporal aspects of care-based multimodal fusion : from a fusion mechanism to composition components and woz components. In *ICMI-MLMI '09 : Proceedings of the 2009 international conference on Multimodal interfaces*, pages 177–184, New York, NY, USA, 2009. ACM.

[Shaw *et al.*, 1975] David E. Shaw, William R. Swartout, et C. Cordell Green. Inferring lisp programs from examples. In *IJCAI'75 : Proceedings of the 4th international joint conference on Artificial intelligence*, pages 260–267, San Francisco, CA, USA, 1975. Morgan Kaufmann Publishers Inc.

[Shneiderman, 1987] B. Shneiderman. Direct manipulation : A step beyond programming languages. *Human-computer interaction : a multidisciplinary approach*, pages 461–467, 1987.

[Shneiderman, 1997] Ben Shneiderman. Direct manipulation for comprehensible, predictable and controllable user interfaces. In *IUI '97 : Proceedings of the 2nd international conference on Intelligent user interfaces*, pages 33–39, New York, NY, USA, 1997. ACM.

[Singh *et al.*, 1990] Gurminder Singh, Chun Hong Kok, et Teng Ye Ngan. Druid : a system for demonstrational rapid user interface development. In *UIST '90 : Proceedings of the 3rd annual ACM SIGGRAPH symposium on User interface software and technology*, pages 167–177, New York, NY, USA, 1990. ACM.

[Sinha et Landay, 2003] Anoop K. Sinha et James A. Landay. Capturing user tests in a multimodal, multidevice informal prototyping tool. In *ICMI '03 : Proceedings of the 5th international conference on Multimodal interfaces*, pages 117–124, New York, NY, USA, 2003. ACM.

[Smith et Graham, 2006] J. David Smith et T. C. Nicholas Graham. Use of eye movements for video game control. In *ACE '06 : Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, page 20, New York, NY, USA, 2006. ACM.

[Smith, 1975] David Canfield Smith. *Pygmalion : a creative programming environment*. Thèse de doctorat, Stanford University, Stanford, CA, USA, 1975.

[Szyperski, 2002] Clemens Szyperski. *Component Software : Beyond Object-Oriented Programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[Szyperski, 2003] Clemens Szyperski. Component technology : what, where, and how ? In *ICSE '03 : Proceedings of the 25th International Conference on Software Engineering*, pages 684–693, Washington, DC, USA, 2003. IEEE Computer Society.

[Tavares et Valente, 2008] Andre L.C. Tavares et Marco Tulio Valente. A gentle introduction to osgi. *SIGSOFT Softw. Eng. Notes*, 33(5) :1–5, 2008.

[Teitelman, 1963] Warren Teitelman. Argus : Real-time handwritten character-recognition system. Technical report, Cambridge, MA, USA, 1963.

[Tucker, 2000] Matt Tucker. Objective viewpoint : make your gui swing. *Crossroads*, 6(4) :3–5, 2000.

[UIMS, 1992] UIMS. A metamodel for the runtime architecture of an interactive system : the uims tool developers workshop. *SIGCHI Bull.*, 24(1) :32–37, 1992.

[UIST, 2009] UIST. Uist '09 : Proceedings of the 22nd annual acm symposium on user interface software and technology, 2009. General Chair-Wilson, Andrew and Program Chair-Guimbretière, François.

[Van Dam, 1988] Andries Van Dam. Phigs+ functional description revision. *SIGGRAPH Comput. Graph.*, 22(3) :125–220, 1988.

[Vernier, 2001] Frédéric Vernier. *La multimodalité en sortie et son application à la visualisation de grandes quantités d'information*. Thèse de doctorat, Université Joseph Fourier - Grenoble 1, 21 Février 2001.

[Vilimek et Zander, 2009] Roman Vilimek et Thorsten O. Zander. Bc(eye) : Combining eye-gaze input with brain-computer interaction. In *UAHCI '09 : Proceedings of the 5th International on ConferenceUniversal Access in Human-Computer Interaction. Part II*, pages 593–602, Berlin, Heidelberg, 2009. Springer-Verlag.

[vvvv, 2009] vvvv. vvvv : a multipurpose toolkit. http ://vvvv.org/, 2009.

[Williamson *et al.*, 2007] John Williamson, Roderick Murray-Smith, et Stephen Hughes. Shoogle : excitatory multimodal interaction on mobile devices. In *CHI '07 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 121–124, New York, NY, USA, 2007. ACM.

[Wu et Balakrishnan, 2003] Mike Wu et Ravin Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *UIST '03 : Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 193–202, New York, NY, USA, 2003. ACM.

Interaction multimodale en entrée : Conception et Prototypage

Les interfaces multimodales se caractérisent par la multiplicité des modalités d'interaction et leurs combinaisons. Ces modalités d'interaction atomiques ou combinées sont offertes à l'utilisateur pour interagir avec un système (modalités en entrée) ou exploités par le système pour fournir des informations (modalités en sortie). L'interaction multimodale en entrée ne se limite plus aujourd'hui à la combinaison de la parole et du geste : de nombreuses modalités d'interaction, comme les interactions multi-doigts et bi-manuelles, le suivi du regard ou les interfaces tangibles, ouvrent un vaste espace de possibilités pour la multimodalité. Face à ce constat, il est donc important de pouvoir explorer efficacement l'espace de possibilités lors de la conception d'interfaces multimodales : nos travaux visent à définir des solutions conceptuelles et logicielles pour l'exploration des possibilités en termes d'interactions multimodales en entrée.

Nous présentons ainsi une approche à composants pour la conception et le prototypage fonctionnel et/ou non fonctionnel d'interfaces multimodales en entrée. Nous adoptons une approche par flots de données, reposant sur un assemblage de composants qui décrit l'interaction multimodale en entrée pour une tâche utilisateur donnée. Nos contributions sont conceptuelles et logicielles. Notre contribution conceptuelle consiste en un espace de caractérisation de composants logiciels pour la multimodalité en entrée. Cet espace permet de caractériser les composants logiciels au sein de l'assemblage de composants qui décrit une interaction multimodale. Il offre au concepteur un cadre conceptuel afin de lui permettre de décrire précisément des alternatives de conception. Notre contribution logicielle consiste en un outil de prototypage basé sur un modèle à composants qui implémente notre espace de caractérisation (notre contribution conceptuelle). L'outil permet au concepteur de créer et d'exécuter un assemblage de composants définissant une interaction multimodale. Nous avons développé de nombreux prototypes multimodaux sur PC et sur téléphones mobiles à l'aide de cet outil, qu'ils soient fonctionnels ou en partie simulés par un compère humain selon une approche Magicien d'Oz. Ces prototypes mettent en œuvre une grande variété de modalités d'interaction comme des gestes 2D et 3D, le pointage 3D, la voix, les mouvements de la tête, les mouvements du corps, la géolocalisation ou encore des interactions tangibles.

Input Multimodal Interaction : Design and Prototyping

In our work, we focus on the design of interactive systems. We are specifically interested in multimodal interactive systems that include several interaction modalities used by the user (input modalities) or by the system (output modalities) in an independent or combined way. Recent years have seen the advent of a variety of interaction modalities including multi-touch, two-handed and tangible interfaces: Multimodal interaction is no longer reduced to the combination of speech and gesture and we are facing a wide range of possibilities for multimodal interaction. In this context, our doctoral research is dedicated to conceptual and software tools for efficiently exploring the huge set of possibilities during the design phase of input multimodal interfaces.

We present a component-based approach for designing and prototyping functional and/or simulated input multimodal interfaces. Our approach relies on the data-flow, from input devices to the user-tasks, depicted by an assembly of components. Our results are both conceptual and practical. Our conceptual model consists of a characterization space for describing components. This space helps interaction designers to define component assemblies and to consequently consider different design alternatives for input multimodality. Our software tool is a component-based prototyping tool that implements the characterization space (i.e., our conceptual model). We have implemented several multimodal prototypes using this tool, both functional and simulated by a human using a Wizard Of Oz approach. The developed multimodal prototypes involve a large variety of interaction modalities including 2D/3D gesture, 3D pointing technique, speech commands, head and body movements, geolocalization techniques as well as tangible techniques.

discipline : **informatique**, spécialité : **interaction homme-machine**
mots clés : **Interaction multimodale, prototypage, magicien d'oz, approche à composants.**

thèse préparée au **Laboratoire d'Informatique de Grenoble**.