Les interactions multimodales dans le contexte de l'internet des objets

Fabrice Poirier

COLLEGES BRETAGNE LOIRE
SCIENCES POUR L'INGENIEUR ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS-DE-LA-LOIRE - IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 648
Sciences pour l'ingénieur et le numérique
Spécialité : *Informatique*

Par
Fabrice POIRIER

Les interactions multimodales dans le contexte de l'internet des objets

Thèse présentée et soutenue à Orange Innovation - Cesson Sévigné, le 12/01/2023
Unité de recherche : Lab-STICC UMR 6285 CNRS
Thèse N° : 2023IMTA0343

Rapporteurs avant soutenance :

Frédéric MÉRIENNE Professeur, ENSAM, Chalon-sur-Saône
Marcos SERRANO Maître de Conférences HdR, Université de Toulouse 3 Paul Sabatier

Composition du Jury :

Président :	Frédéric MÉRIENNE	Professeur, ENSAM, Chalon-sur-Saône
Examinateurs :	Gaëlle CALVARY	Professeure, Ensimag - Grenoble INP - UGA
	Marcos SERRANO	Maître de Conférences HdR, Université de Toulouse 3 Paul Sabatier
Dir. de thèse :	Thierry DUVAL	Professeur, IMT Atlantique
Co-encadrants :	Anthony FOULONNEAU	Chercheur, Orange
	Jérémy LACOCHE	Chercheur, Orange

Remerciements

Tout d'abord, je tiens à remercier mon directeur de thèse et mes encadrants pour leur support indéfectible tout au long de ce parcours de 3 ans. Un grand merci ...

À Thierry pour avoir suivi de près et avec bienveillance mon travail de thèse, et cela malgré la distance.

À Anthony pour m'avoir fait découvrir le domaine de la multimodalité et m'avoir guidé dans mes moments de doutes.

À Jérémy pour m'avoir aidé à réaliser mes objectifs, que cela soit pour du développement, des expérimentations, ou pour un déménagement.

Je tiens aussi à remercier mon jury de thèse. Merci à Frédéric et Marcos pour avoir accepté d'être mes rapporteurs et de faire partie de mon jury de soutenance. Vos retours et questions m'ont permis de prendre plus de recul sur mes travaux. Merci à Gaëlle pour l'intérêt et les remarques plus que pertinentes que tu as apportées pendant les entretiens et à la soutenance. Ton énergie et ta force de volonté malgré tes maux me laisse moi aussi sans voix.

Finalement, je tiens à remercier tous ceux qui m'ont accompagnés et soutenus pendant ces 3 années, dans les moments difficiles comme dans les meilleurs moments. En particulier, je pense à tous ceux qui m'accompagnait dans mes pauses plus ou moins longues pour de grandes discutions sur des sujets très diverses. Je pense aussi à ces compagnons d'aventure qui m'ont permis de me changer les idées, que cela soit pendant des soirées ludiques, calmes ou effervescentes. Enfin, je pense à ceux que je connais depuis longtemps et qui ont toujours cru en moi, qu'ils soient dans le Sud-Ouest, la Provence ou même en Île-de-France.

Table of Contents

Introduction

Contexte

Le principe d'interaction multimodale, présenté la première fois par R.Bolt[27], consiste à utiliser plusieurs modalités (parole, gestes, ...) pour rendre les interactions Humain-Machine plus proches des interactions que l'on retrouve entre humains, et donc plus intuitives.

Les premiers systèmes étaient réalisés dans des environnements contrôlés (i.e. la topologie de l'espace interactif, l'équipement et sa disposition dans l'espace sont connus), mais l'arrivée des objets connectés ces dernières années offre l'opportunité de concevoir des systèmes interactifs capables de s'adapter plus facilement à différents environnements (domiciles, bureaux, usines, villes). Ces systèmes, que nous appelons MIBS (Multimodal IoT-Based System), promettent l'arrivée dans notre quotidien d'interactions plus accessibles et naturelles. En effet, les MIBS ont potentiellement à leur disposition une grande variété de modalités disséminées dans les environnements où les utilisateurs se situent. Il est donc possible de créer des applications multimodales dont les interfaces utilisateurs évoluent dans l'espace et le temps pour être utilisables en toutes circonstances, sans être trop envahissantes. Les services de contrôle de maisons intelligentes (ex. contrôle de l'éclairage, de la température, de la qualité de l'air) [135] et de suivi de l'état de santé de personnes âgées [151] sont des exemples de ce que les MIBS peuvent offrir.

Toutefois, créer des MIBS clé en main est un travail complexe.

Prenons l'exemple d'un éditeur de logiciels qui souhaite commercialiser des MIBS à destination des entreprises du tertiaire. En particulier, il veut commercialiser une application de réservation de salles de réunion et une application de guidage d'utilisateurs dans les bâtiments. Les bâtiments, ainsi que les objets connectés qui y sont installés, sont différents d'un client à un autre, et le comportement des applications doit être adapté en conséquence. Il serait très coûteux de proposer des solutions *ad-hoc* (i.e. concevoir les applications pour un environnement spécifique) pour tous ses clients. L'éditeur doit donc concevoir ces applications pour que n'importe quel client puisse les installer dans ses locaux.

Or chaque client a ses propres exigences sur le fonctionnement du système interactif en fonction de l'environnement d'exécution. Par exemple, un des clients peut vouloir interagir avec l'application de réservation au travers des microphones et de caméras déjà installés dans les salles de réunion. Un autre client pourrait préférer utiliser des interfaces tactiles placées dans un couloir. De la même façon, l'utilisation des objets connectés pour indiquer

le chemin à suivre dans l'application de guidage dépend de la topologie des bâtiments et du placement des objets connectés dans ceux-ci.

Ainsi, ces applications doivent être capables d'utiliser les objets connectés et les techniques d'interaction spécifiques à chaque bâtiment des clients, et cette dépendance à l'environnement d'exécution complique la création de MIBS par l'éditeur de logiciels. En particulier, il existe un nombre incalculable d'objets connectés sur le marché [1], et chaque objet a ses propres caractéristiques (ex. champ de vision d'une caméra) qui doivent être prises en compte quand on souhaite l'intégrer dans un système interactif. Concevoir et développer de zéro des alternatives pour chaque combinaison d'objets est impossible, et proposer une solution entièrement générique l'est tout autant.

En résumé, les MIBS fournissent un espace d'interaction lié au monde physique, et ils sont donc très sensibles au contexte d'usage. Or, les environnements possibles de fonctionnement d'un MIBS sont très variés et difficiles (voire impossibles) à anticiper : on y retrouve un ensemble changeant de capteurs et d'actionneurs fixes ou bien mobiles, des utilisateurs aux profils variés (ex. parents et enfants à domicile, simples utilisateurs et administrateurs dans les bureaux, et même visiteurs) pouvant interagir avec le système interactif seuls ou en groupes.

Présentation de cas d'usage des applications de guidage et de réservation de salle de réunion

Pour illustrer cette dépendance des MIBS à l'environnement, nous présentons des cas d'usage pour les applications de guidage et de réservation. Pour chacune de ces applications, nous donnons un exemple de configuration qu'une entreprise du tertiaire pourrait souhaiter installer dans un de ses locaux. Nous illustrons ensuite le fonctionnement de ces configurations au travers d'exemples de scénarios d'usage.

Le procédé de guidage présenté ici a fait l'objet d'un dépôt de brevet réalisé au cours de la thèse.

1. `https://techjury.net/blog/how-many-iot-devices-are-there/`

Cas d'usage de l'application de réservation

Besoin du client

L'entreprise veut installer un service de réservation de salles de réunion. Pour cela, elle souhaite utiliser les objets connectés déjà installés dans les salles de réunion : des enceintes connectés et des interfaces murales connectées. Ces interfaces murales sont constituées de caméras, de microphones et d'écrans de télévision. L'entreprise souhaite pouvoir réserver les salles de réunion en utilisant tous les objets connectés. En effet, elle veut que l'application fournisse des interfaces visuelles ou vocales, selon les préférences des utilisateurs (i.e. salariés). De plus, l'application doit pouvoir se synchroniser avec le service de messagerie qui est déjà utilisé par les salariés depuis leur smartphones.

Exemple de scénario d'usage

Le scénario met en scène Dominique, un salarié de l'entreprise, qui souhaite réserver la salle de réunion juxtaposée à son bureau. La salle de réunion est représentée dans la figure 1.

Dominique entre dans la salle de réunion et s'arrête devant l'interface murale. L'écran s'allume sur un message de bienvenue, et le système l'interpelle vocalement :

- "Bonjour Dominique. Que souhaitez-vous faire ?"

- "je voudrais réserver cette salle début octobre. Indique-moi les jours où elle est libre toute la journée".

- "Voici les créneaux disponibles". Une liste de créneaux s'affiche sur l'écran d'affichage.

Dominique fait défiler la liste de mouvement de balayage de la main, puis pointe du doigt un créneau libre en date du 5 Octobre et dit :

- "Celui-ci."

- "Très bien. Veuillez compléter l'invitation sur votre appareil personnel."

Dominique ouvre depuis son smartphone l'invitation générée pour y ajouter l'objet de la réunion ainsi que les adresses mails des autres participants.

Après avoir envoyé l'invitation, Dominique quitte le champ de la caméra. En réponse, l'écran s'éteint, et le système énonce :

- "Rendez-vous ici le 5 Octobre. Bonne journée !"

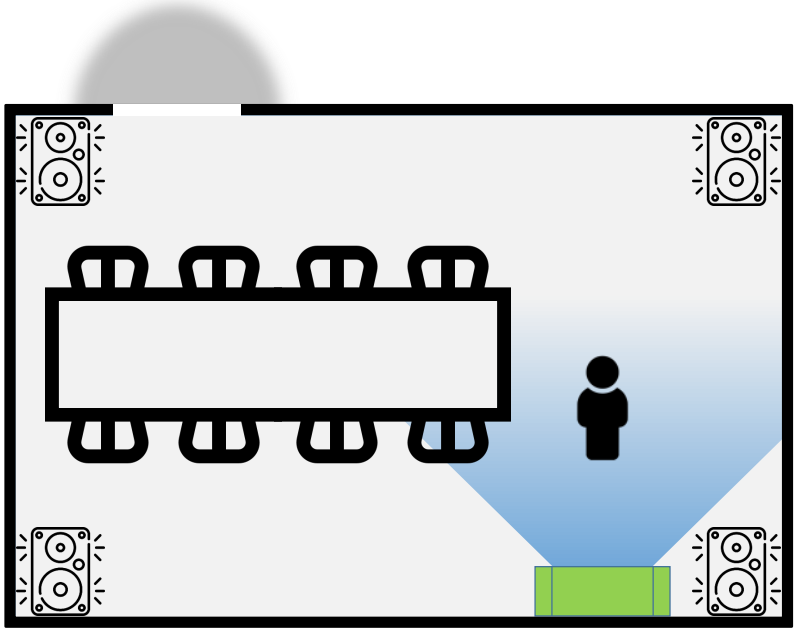


FIGURE 1 – Plan en vue du dessus de la salle de réunion au début de l'interaction entre Dominique et le système interactif. L'interface murale est représentée par le rectangle vert. Le champ de vision de la caméra est représenté par la zone bleue.

Cas d'usage de l'application de guidage

Besoin du client

L'entreprise souhaite déployer l'application de guidage dans un de ses locaux (voir figure 2) pour permettre à un salarié de guider des nouveaux arrivants jusqu'à lui. Ces arrivants commencent leur parcours depuis un portique situé à l'entrée. Après s'être identifiés à l'aide d'une application sur leurs smartphones, les arrivants sont guidés jusqu'à la position du salarié ou de son bureau. L'entreprise veut utiliser les divers équipements installés sur ce site pour diriger les arrivants. En effet, elle veut d'abord utiliser l'enceinte connectée située à l'entrée pour demander aux arrivants de suivre le chemin indiqué par les objets connectés. Ensuite, elle veut donner la direction dans les couloirs en utilisant les ampoules connectées et les écrans connectés placés dans les couloirs. Ces écrans connectés sont constitués d'afficheurs graphiques et d'enceintes, ce qui permet de transmettre le message de guidage visuellement ou vocalement.

Dans l'éventualité où un salarié n'est pas à son bureau, les arrivants doivent être guidés

jusqu'à la position du salarié. Pour obtenir cette position, l'entreprise veut utiliser la position de la montre connectée ou du smartphone de l'employé (avec son consentement). De plus, l'entreprise souhaite que le salarié soit informé quand un nouvel arrivant qui souhaite le rencontrer passe le portique.

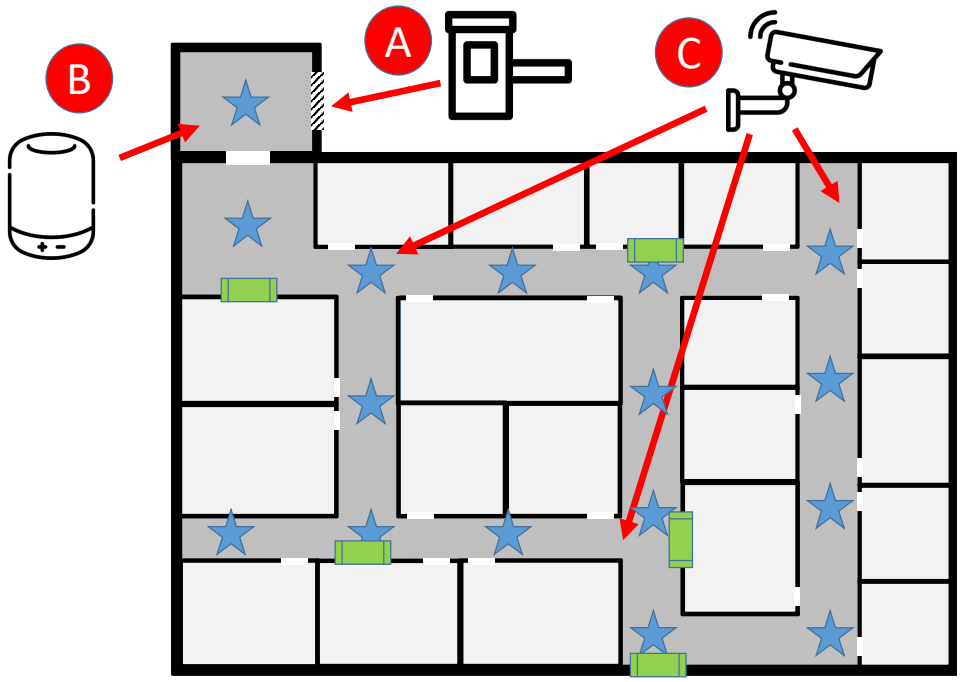


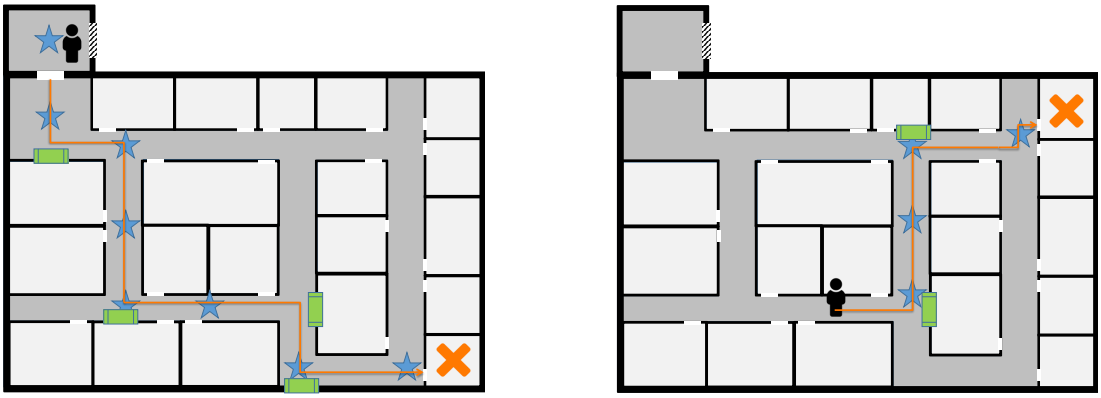
FIGURE 2 – Plan en vue du dessus d'un étage de bâtiment composé de plusieurs pièces et couloirs. L'environnement est équipé de plusieurs objets connectés. (A) Un portique de sécurité et (B) une enceinte connectée sont placé à l'entrée. (C) Trois caméras connectées sont installées. Les ampoules connectées et les écrans connectés sont représentés respectivement par les étoiles bleues et les rectangles verts.

Exemple de scénario d'usage

Ce scénario met en scène Dominique, un salarié travaillant dans les locaux où l'application de guidage a été installée, et Camille qui vient lui rendre visite. Ce scénario est illustré figures 3a et 3b

Camille arrive à l'entrée des locaux de l'entreprise dans laquelle Dominique travaille.

Devant le portique à l'entrée du bâtiment, Camille scanne un QR code avec son smartphone. Cela installe une application qui lui demande un mail comme identifiant, et qui lui demande d'accepter de partager sa position dans le bâtiment.



(a) (b)

FIGURE 3 – Illustration du scénario de guidage. La figure de gauche représente le chemin initial que Camille est amené à emprunter, ainsi que les objets qui sont utilisés par l'application pour le guider. La figure de droite représente le nouveau chemin et les objets utilisés pour le guidage, après que Dominique ait changé la destination. La position de Camille est représentée par le personnage noir, la destination est représentée par la croix orange, et le chemin à suivre est représenté par la flèche orange.

Après s'être identifié, Dominique reçoit une notification signifiant que Camille est arrivé, et qu'il est automatiquement guidé jusqu'au point de rendez-vous.

Le portique s'ouvre pour laisser passer Camille. De plus, un message vocal émis par l'enceinte connectée demande à Camille de suivre les lumières en bleu dans les couloirs, et de se référer aux indications vocales et textuelles transmises aux croisements.

Ainsi, les ampoules connectées menant du point de départ à la destination passent successivement d'une lumière blanche à bleu au passage de Camille. De plus, les écrans (et enceintes) placés aux croisements énoncent vocalement le chemin à suivre si personne d'autre n'est à proximité, et dans le cas contraire, le même message est affiché sur l'écran.

En cours de chemin, Dominique décide de changer depuis sa montre connectée la destination à sa position actuelle.

Suite à cette modification, le parcours est adapté, ce qui permet à Camille de rejoindre Dominique sans problème.

Objectif

L'objectif de cette thèse est de faciliter la création de MIBS, et ainsi participer à l'arrivée de ces systèmes dans notre quotidien. Or, comme évoqué précédemment, l'utilisation d'objets connectés dans les MIBS exacerbe le lien entre les applications et les environnements d'exécution. Pour éviter de recourir à des solutions *ad hoc*, les MIBS doivent donc être conçus et développés pour être les plus génériques possibles, et doivent pouvoir s'adapter aux spécificités des environnements. Pour faciliter la création de MIBS, il faut donc définir ce qui permet de réaliser cette adaptation. Cela soulève la problématique suivante : *Quels méthodes et outils peuvent être utilisés pour faciliter le processus de création des MIBS, de leurs conceptions par les éditeurs à leurs exécutions dans chaque environnement ?*

Méthodologie

Notre démarche est organisée en deux parties.

Pour répondre à cette problématique, nous fournissons une méthodologie pour créer des MIBS qui permet par la suite d'adapter ces systèmes à leurs environnements. Nous proposons notamment une analyse du cycle de vie des MIBS, ainsi qu'une méthode d'aide à la configuration et à l'évaluation des MIBS qui s'inscrit dans leur processus de création.

Dans la première partie, nous montrons en quoi la création des MIBS peut être simplifiée par un framework adapté à ces systèmes. Dans le chapitre 1, nous synthétisons les exigences auxquelles un MIBS doit répondre. À partir de ces exigences, nous étudions dans le chapitre 2 les standards architecturaux et frameworks utilisés pour concevoir des MIBS. Dans le chapitre 3, nous décrivons un framework pour MIBS à l'état de l'art. Notre framework et ceux présentés dans le chapitre 2 répondent à la plupart des exigences exprimées dans le chapitre 1, mais ils ne permettent pas d'assurer l'adaptation des applications aux environnements d'exécution. Une intervention humaine est nécessaire.

Dans la seconde partie, nous cherchons à faciliter cette intervention humaine cruciale au processus d'adaptation. Pour cela, nous étudions les méthodes et outils qui fournissent des éléments de réponse à notre problématique, puis nous proposons et illustrons une méthodologie pour configurer et évaluer les MIBS. Nous proposons dans le chapitre 4 un cycle de vie des MIBS. Cela nous permet d'identifier les étapes de la création des MIBS qui ne sont pas assez outillées (i.e. qui manquent de méthodes et outils logiciels) dans la

littérature pour répondre à notre problématique de recherche. Pour pallier le manque de méthode permettant de valider l'utilisabilité des MIBS dans les environnements d'exécution, nous proposons dans le chapitre 5 une méthode de configuration et d'évaluation de MIBS. Dans le chapitre 6, nous présentons l'expérimentation réalisée pour valider cette méthode. Nous montrons ensuite comment cette méthode peut être utilisée pour déployer les applications de réservation et de guidage. .

Nous concluons par un rappel des contributions réalisées, et nous présentons des perspectives d'évolution des ces travaux.

Première partie

Framework pour faciliter la création de systèmes multimodaux en environnements connectés

Dans cette partie, nous étudions les systèmes multimodaux qui utilisent les objets connectés comme interfaces d'interaction, que nous appellerons MIBS (**M**ultimodal **I**oT-**B**ased **S**ystem) par la suite, et nous étudions en particulier les frameworks qui facilitent leur création. Pour cela, nous définissons dans le chapitre 1 les exigences auxquelles doivent répondre les MIBS, ce qui nous permet de définir précisément ces systèmes. Dans le chapitre 2, nous analysons les architectures et frameworks proposant de créer ce genre de systèmes. A partir de cette analyse, nous synthétisons les limites des approches existantes.

Dans le chapitre 3, nous décrivons un framework pour les MIBS qui synthétise les standards architecturaux utilisés pour répondre aux exigences que nous avons identifiées chapitre 1. Nous illustrons ce framework en l'appliquant aux deux scénarios présentés en introduction.

Nous concluons cette partie par un rappel des limites des frameworks existants, ce qui nous amène à la question de recherche suivante : *quels méthodes et outils peuvent être utilisés pour faciliter le processus de création des MIBS, de leurs conceptions par les éditeurs à leurs exécutions dans chaque environnement ?*

Chapitre 1

Définitions des MIBS et exigences

Les MIBS sont des systèmes multimodaux qui utilisent des objets connectés comme interfaces d'interaction. Les éditeurs de logiciels qui doivent créer et mettre en place des MIBS sont donc confrontés à des problématiques de plusieurs domaines de recherche. En effet, ces systèmes sont l'aboutissement de travaux sur les interactions multimodales, mais aussi de la volonté d'exploiter le domaine plus récent de l'Internet des Objets pour offrir un plus grand panel d'interfaces. Ces systèmes rentrent aussi dans le domaine de l'informatique ubiquitaire comme nous le verrons par la suite.

L'objectif de ce chapitre est d'identifier les exigences auxquelles les MIBS doivent répondre. Pour cela, nous commençons par aborder les concepts utilisés dans le domaine des interactions multimodales. Nous expliquons ensuite en quoi les MIBS sont des systèmes ubiquitaires. Puis nous étudions les problématiques soulevées dans la conceptualisation et la création des MIBS. Enfin, nous donnons une définition complète des MIBS à partir des exigences identifiées. Ces exigences sont synthétisées dans la Table 1.1.

1.1 Interactions multimodales

Le principe des interactions multimodales est de permettre à un utilisateur de communiquer naturellement avec un système interactif en faisant appel à plusieurs modalités d'interactions [19]. L'exemple le plus connu est l'expérience de Bolt [27] qui consiste à utiliser la voix et des gestes pour manipuler une interface graphique. Cette approche conceptuelle des interactions Humain-Machine [117, 120] s'est construite sur un vocabulaire commun dans la littérature. Pour autant, les principes (ex. définition d'une modalité) et la description de ce que l'on peut attendre de ces interactions ne font pas encore consensus. Par exemple, le simple fait d'utiliser un clavier et une souris suffit dans certaines approches (ex. [23]) pour fournir une interface multimodale. Pour d'autres définitions (ex. [109]), une interface ne peut être multimodale que si différent sens (physiologiques) sont sollicités [162].

Dans cette section, l'objectif est donc de fournir la terminologie qui sera utilisée dans la suite de la thèse, ainsi que d'établir les exigences auxquelles les systèmes multimodaux doivent répondre.

1.1.1 Concepts fondamentaux

Modalité

Au cœur des interactions multimodales se trouve la notion de modalité d'interaction. Il en existe plusieurs définitions qui représentent chacune les interactions selon l'un des deux points de vue possibles : le point de vue de l'utilisateur et le point de vue du système interactif [162].

Par exemple, Bernsen [23] propose une définition qui représente l'interaction du point de vue de l'utilisateur. Il définit une modalité comme une représentation de l'information au travers d'un médium physique. Ainsi, un texte affiché sur un écran peut être décrit comme une modalité définie par le couple *lumière-affichage graphique de langage*. Ici, la lumière est le médium entre l'écran et les yeux de l'utilisateur, et l'affichage graphique de langage est l'information transmise.

Dans les approches qui s'adressent principalement aux développeurs, une modalité est souvent définie comme un couple formé par un dispositif physique et un langage d'interaction [117, 14]. Ici, un dispositif physique est un objet capable de capter ou d'agir sur des propriétés physiques. Pour l'exemple du texte sur un écran, le couple <écran, langage naturel> serait une description valable d'une modalité.

La cohabitation de ces points de vue s'explique par les différents besoins exprimés lors de la conception de systèmes interactifs. En effet, les approches centrées utilisateur donnent une meilleure représentation de l'expérience qu'aurait un utilisateur, et permettent donc un meilleur usage des modalités selon le profil des utilisateurs. Quant aux approches techno-centrées, elles permettent d'identifier les points communs et différences entre les capacités d'interaction des dispositifs physiques, et donc de simplifier le travail de développement. Néanmoins, on peut observer qu'un dispositif physique peut fournir des interfaces utilisant plusieurs médiums (ex. interface tactile et visuelle d'une montre connectée), et qu'un médium peut être utilisé par plusieurs dispositifs physiques pour fournir une interface unique (ex. un mur d'écran TV).

Ainsi, ces deux points de vue facilitent la manipulation des modalités à différents moments de la création de systèmes multimodaux. Il est donc nécessaire d'avoir une re-

présentation des modalités qui inclue ces deux facettes. Pour cela, nous utiliserons la définition de la modalité proposée par Bernsen [23], et nous introduisons le terme de *capacité interactive* pour décrire les modalités que chaque dispositif physique peut fournir, et sous quelles conditions. La notion de capacité interactive permet de faire un "catalogue" d'objets répertoriant pour chaque objet les modalités possibles, ce qui permet d'anticiper l'usage que l'on peut faire de ces objets. La taxonomie de Augstein et Neumayr [8] (voir Figure 1.1) convient donc parfaitement à cette représentation des modalités. Dans cette taxonomie, les modalités d'entrée (i.e. de l'utilisateur vers le système) et de sortie (i.e. du système vers l'utilisateur) que les dispositifs physiques peuvent fournir sont regroupées selon la capacité sensori-motrice stimulée de l'utilisateur. Ainsi, cette taxonomie est utilisable à la fois pour les approches techno- et humano- centrées, et s'adapte facilement à un large panel d'applications multimodales. Par exemple, une revue exploratoire des technologies multimodales utilisées en réalité étendue [132] s'appuie sur une version simplifiée de la taxonomie pour catégoriser les dispositifs physiques existants. Nous utiliserons donc cette taxonomie par la suite.

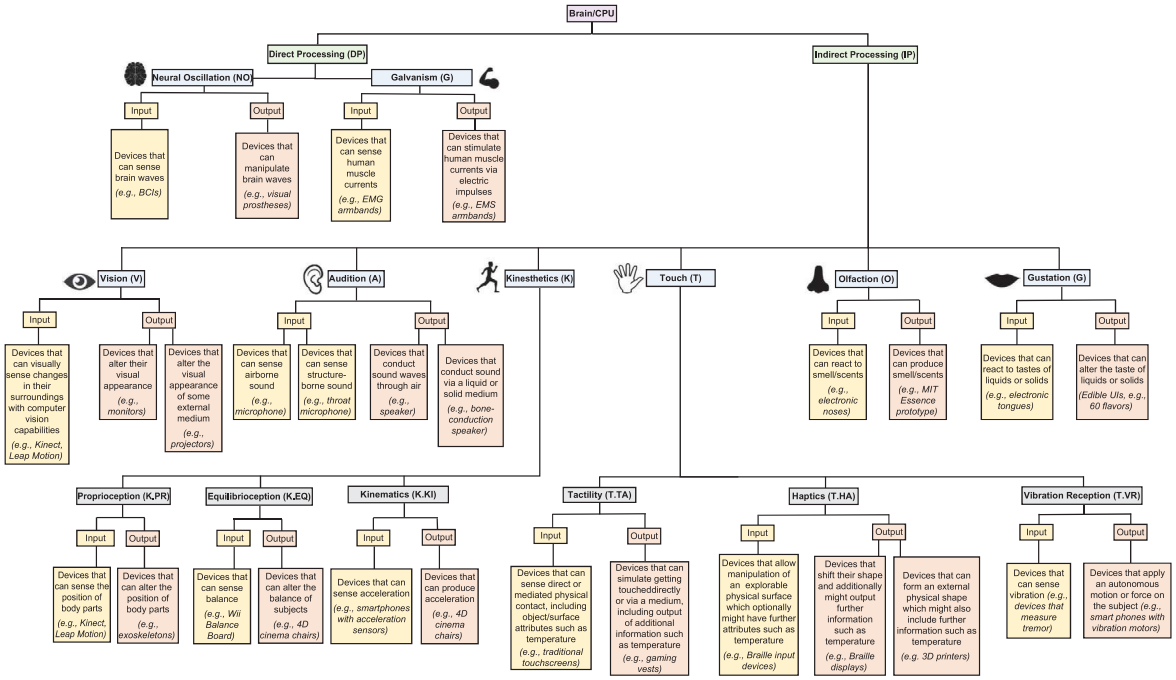


FIGURE 1.1 – Taxonomie des modalités et dispositifs physiques selon Augstein et Neumayr [8].

Interaction multimodale

Une interaction multimodale est définie par l'usage de plusieurs modalités (entrée et sortie confondus) pour réaliser une interaction [117]. Pour faciliter la compréhension des interactions multimodales, deux modèles sont souvent utilisés pour décrire l'usage de plusieurs modalités : les modèles CASE et CARE.

Nigay et Coutaz [116] proposent avec les propriétés CASE (concurrent, alternatif, synergique, exclusif) une classification utile aux concepteurs de systèmes multimodaux. Ici, les interactions sont différenciées selon l'usage des modalités : elles peuvent être utilisées parallèlement ou séquentiellement, et elles peuvent être utilisées de façon indépendante ou de façon combinée. Le modèle CARE [54] (complémentaire, équivalent, assignation, redondance) a été proposé par la suite pour décrire l'état d'un système interactif selon les utilisations possibles des modalités à disposition. Ainsi, ce modèle a pour vocation d'aider à évaluer les systèmes multimodaux dans les phases de test.

Ces deux modèles ont donc des buts différents, et peuvent être utilisés conjointement, à l'instar de l'espace conceptuel dans [117]. Par exemple, les modalités détectant les commandes vocales et gestuelles de l'utilisateur dans l'expérience de Bolt [27] sont utilisées de manière complémentaire (CARE), et elles peuvent être perçues par le système comme alternées ou synergiques (CASE).

Système interactif multimodal

Nigay et Coutaz [116] donnent la définition suivante pour un système interactif multimodal (appelé par la suite système multimodal) : "a multimodal system supports communication with the user through different modalities such as voice, gesture, and typing" (*un système multimodal supporte la communication avec l'utilisateur en utilisant différentes modalités telles que la voix, les gestes et la dactylographie*). Similairement, Bernsen [23] définit un système multimodal ainsi : "a system which uses at least two different modalities for input and/or output" (*un système qui utilise au moins deux modalités différentes en entrée et/ou sortie*). Ces définitions présentent un système multimodal comme un système utilisant différentes modalités. Toutefois, Bernsen considère que deux modalités sont différentes si les structures des données sont différentes, même si le médium est le même. Ainsi, une interface graphique (GUI : Graphical User Interface) affichant une image et du texte serait multimodale selon Bernsen. Or, cela s'oppose au principe que les interactions multimodales font appel à différents sens de l'utilisateur. Nous considérons donc un sys-

tème comme multimodal s'il supporte l'usage de plusieurs modalités utilisant différents médiums physiques.

1.1.2 Exigences propres aux systèmes multimodaux

Bien qu'il existe plusieurs définitions valables des termes couramment employés dans le domaine de la multimodalité, deux exigences élémentaires sont communément acceptées [117, 120] :

— *Support d'interactions multi-sensori-motrices :* un système multimodal doit pouvoir fournir une interface qui requière au moins deux dispositifs (soit d'entrée, soit de sortie) qui sollicite les multiples capacités sensori-motrices naturelles de l'homme ;

— *Abstraction des dispositifs physiques :* "un système multimodal a la faculté d'abstraire donc de comprendre ce qui est reçu ou émis via ses dispositifs d'entrée/sortie"[117].

Cette abstraction passe par une représentation des dispositifs physiques selon les modalités qu'ils peuvent fournir. Ainsi, les applications multimodales peuvent être créées indépendamment des dispositifs physiques, et les objets servant d'interfaces pour ces applications peuvent être plus facilement interchangés.

Bien que ces deux exigences soient simples à comprendre, nous verrons dans le chapitre 2 que de nombreux travaux ont été réalisés au cours de ces 30 dernières années pour y répondre.

Interactions multimodales

Les systèmes multimodaux sont des systèmes interactifs qui supporte l'usage de plusieurs modalités utilisant différents médiums physiques pour réaliser des interactions multimodales. Ces systèmes doivent satisfaire deux exigences : 1) ils doivent supporter les interactions multi-sensori-motrices et 2) les dispositifs physiques doivent être représentés par les modalités qu'ils peuvent fournir. Cette abstraction des dispositifs physiques facilite l'adaptation des systèmes multimodaux aux environnements d'exécution.

1.2 Internet des Objets et informatique ambiante

Nous englobons sous le terme "objets connecté" des capteurs et actionneurs ayant des capacités très variables. Par exemple, un capteur de présence fournit une information très faible sur l'environnement, tandis qu'une enceinte connectée a une capacité forte à comprendre ce que est dit à proximité. Les MIBS utilisent ces objets connectés répartis dans nos environnements comme interfaces d'interaction. Ces systèmes doivent donc être conçus en suivant des exigences non-fonctionnelles liées au domaine de l'Internet des Objets (IoT).

De plus, l'utilisation des objets connectés comme interfaces d'interaction n'est pas un sujet abordé dans le domaine de l'IoT. Il s'agit néanmoins d'un thème étudié dans l'informatique ambiante, où la notion d'expérience utilisateur et sa dépendance à l'environnement et au système interactif ont une grande importance. Ainsi, nous considérons que les MIBS sont des systèmes ubiquitaires. Pour assurer l'aspect ubiquitaire, les MIBS doivent aussi valider les exigences de l'informatique ambiante.

Dans la suite de cette section, nous présentons les paradigmes de l'IoT et de l'Informatique ambiante, ainsi que les exigences que les systèmes IoT et les systèmes ambiants doivent satisfaire. Cela nous permet d'étudier ce qu'implique l'usage de l'Internet des Objets dans les MIBS.

1.2.1 Internet des Objets

L'initiative IEEE IoT [47] donne une définition de l'Internet des Objets (IoT) qui s'applique pour les réseaux simples, constitués seulement d'un petit nombre d'objets (ex. maison intelligente) :

"An IoT is a network that connects uniquely identifiable “Things” to the Internet. The “Things” have sensing/actuation and potential programmability capabilities. Through the exploitation of unique identification and sensing, information about the “Thing” can be collected and the state of the ‘Thing’ can be changed from anywhere, anytime, by anything"

(*un IoT est un réseau qui connecte à l'internet des "objets" identifiables de manière unique. Les "objets" ont des capacités de détection/actionnement et de programmation potentielle. Grâce à l'exploitation de l'identification unique et de la détection, il est possible de collecter des informations sur un "objet" et de modifier l'état de cet "objet" de n'importe où, n'importe quand et par n'importe quel moyen*) ;

Toutefois, les réseaux IoT ne sont pas tous à petite échelle. Comme présenté dans [47], plusieurs réseaux peuvent cohabiter dans des environnements où un grand nombre d'objets sont gérés/possédés par plusieurs entités. Par exemple, il est possible que les MIBS doivent accéder à des objets connectés gérés par des plateformes IoT (ex. l'éclairage dans le scénario de guidage). Cette différence d'échelle entre les environnements met en avant la première caractéristique que les MIBS héritent de l'IoT : l'*extensibilité* ("scalability"). En effet, les réseaux IoT permettent à une entité externe (ex. un service fourni par un MIBS) l'utilisation de n'importe quel nombre d'objets. Cela veut dire que la taille d'un réseau n'est pas fixe, mais peut s'étendre selon les besoins. Similairement, les MIBS peuvent faire appel aux objets personnels des utilisateurs comme les smartphones ou les smartwatchs.

Cette définition des IoT a été comparée aux définitions utilisées en pratique [134], ce qui a permis de dégager les 4 exigences majoritairement reconnues auxquelles doivent répondre les systèmes IoT :

— *Connexion à Internet :* les objets peuvent échanger des données en utilisant internet. On notera que l'usage d'Internet amène à considérer des exigences non-fonctionnelles supplémentaires dans la création de MIBS, comme la fiabilité et la sécurité du réseau. Toutefois, nous ne considérerons pas ces exigences par la suite car elles dépassent le cadre de cette thèse. On notera aussi que les objets n'ont pas tous la même technologie ou le même protocole de communication. Ils peuvent communiquer par un réseau sans fil (ex. Bluetooth, Wifi) ou par câble, et utiliser des protocoles tels que MQTT [79] ou HTTP [1]. Pour éviter de limiter les MIBS à un ensemble restreint d'objets connectés, il faut assurer au maximum la compatibilité entre les MIBS et ces technologies de communication.

— *Capacités de captation ou d'action :* les objets sont soit capables de capter des informations venant de l'environnement, soit capables d'agir sur cet environnement. Cette capacité des objets à capter ou agir est similaire à la capacité de fournir des interfaces multi-sensori-motrices dans le domaine de la multimodalité. En effet, pour interagir avec les utilisateurs, les systèmes multimodaux captent des informations sur l'environnement où se trouvent les utilisateurs et agissent sur ce même environnement. Les plateformes IoT sont aussi capables de capter et d'agir sur des éléments de l'environnement indépendant de l'interaction avec les utilisateurs. Cependant, ces informations ne sont pas nécessaires pour les MIBS. Les deux exigences seront donc représentées par la suite par une seule exigence (celle de la

1. `https://www.w3.org/Protocols/`

multimodalité).

— *Objets identifiables :* les objets dans un système IoT peuvent être identifiés de façon unique. L'intérêt d'une convention de nommage est de faciliter la communication. Cela permet aussi d'intégrer des mécanismes de sécurité en fonction du nom [3].

— *Services à disposition :* les objets offrent des services, qu'il y ait ou non une intervention humaine directe. Ces services peuvent être utilisés par des utilisateurs ou d'autres services. Pour permettre cela, les objets doivent exposer sur le réseau une API sur laquelle d'autres services peuvent dynamiquement se connecter [3]. Par exemple, Philips propose une API RESTful (REpresentational State Transfer) pour contrôler ses ampoules connectées [2].

1.2.2 Informatique ambiante

La dissémination d'objets connectés dans les lieux privés et publics permet à des services IoT de capter et agir en toute circonstance, suivant ainsi la vision du *"anytime, anywhere, anymedia, anything"* [7]. Cette vision se retrouve dans le paradigme de l'*informatique ambiante*, aussi appelée *informatique ubiquitaire*. Le principe d'informatique ambiante a été pour la première fois défini par Weiser [163] comme la disparition, du point de vue de l'utilisateur, des ordinateurs dans l'environnement. Coutaz et Nigay [52] ajoutent que l'informatique ambiante se définit par des "capacités de mobilité et d'intégration des systèmes dans le milieu physique, et ceci de manière spontanée, à de multiples échelles, de la planète au micro-, voire nano-objet". Cette intégration des systèmes dans l'environnement physique requiert une bonne modélisation de ce dernier. Ainsi, le contexte d'usage a une grande importance, et tout particulièrement le contexte associé à l'environnement physique, à l'usage des objets connectés, mais aussi à la localisation/mobilité des objets et utilisateurs. On notera que l'informatique ambiante est à différencier de l'intelligence ambiante. Cette dernière ajoute une notion d'intelligence artificielle pour prendre des décisions et anticiper l'intention de l'utilisateur [99]. L'intelligence ambiante est donc un cas particulier d'informatique ambiante, bien que la distinction ne soit pas toujours faite dans la littérature.

Pour assurer l'adaptation des systèmes ubiquitaires au contexte d'usage, Da Costa et al. [50] synthétisent les 10 problématiques associés à l'informatique ambiante : invisibilité, interaction utilisateur transparente, gestion du contexte, perception du contexte,

2. `https://developers.meethue.com/develop/get-started-2/`

mobilité, inter-opération spontanée, vie privée/confiance, tolérance à l'erreur, extensibilité, hétérogénéité. Les challenges de tolérance à l'erreur, extensibilité, hétérogénéité, vie privée et confiance sont partagés avec le domaine de l'IoT présenté précédemment. Les autres exigences viennent donc s'ajouter aux caractéristiques des MIBS :

— *Perception du contexte :* le système est capable de percevoir ou inférer l'état de l'utilisateur et de l'environnement. Cela considère aussi la détection et le suivi des dispositifs physiques dans l'environnement [33]. Nous verrons par la suite la notion de contexte d'usage plus en détails.
— *Gestion du contexte :* le système peut changer de comportement en fonction du contexte perçu. L'objectif de cette adaptation est d'améliorer la satisfaction des utilisateurs [12].
— *Inter-opération spontanée :* les objets connectés, logiciels et services peuvent apparaître et disparaître fréquemment dans les systèmes ubiquitaires. Il faut donc assurer la communication entre ces éléments hétérogènes malgré ces changements. Il s'agit donc d'une exigence importante pour permettre aux systèmes ubiquitaires de s'adapter dynamiquement aux changements d'état des objets connectés.
— *Mobilité :* l'informatique ambiante suppose que les utilisateurs puissent accéder de partout et à tout moment aux données, et cela même en déplacement. Ainsi, l'usage d'objets transportables tels que les smartphones et montres connectées doit être possible. De plus, Da Costa et al. [50] identifient deux types de mobilité dans les systèmes ubiquitaires : la mobilité logique (i.e. distribution et migration des processus et des données) et la mobilité physique.
— *Interaction utilisateur transparente :* une interaction est transparente quand l'interface utilisateur fournie par le système interactif semble naturelle à l'usage. Cette transparence peut être réalisée par le biais d'interfaces qui sont intuitives à utiliser, comme peuvent l'être les interfaces tangibles. En effet, bien qu'un système ubiquitaire ait pour objectif d'aider l'utilisateur, une UI non-intuitive (ex. du texte sur un écran hors du champ de vision de l'utilisateur) peut nuire à un usage naturel du système dans l'environnement physique.
— *Invisibilité :* le système informatique se confond à l'environnement, de sorte que l'on ne perçoit plus sa présence (en dehors de l'usage que l'on peut en faire). Par exemple, un assistant de navigation (ex. GPS) peut guider vocalement un conducteur, mais peut se faire oublier quand aucune instruction n'est nécessaire.

Les notions de transparence et d'invisibilité sont proches car elles représentent toutes

les deux le lien entre le monde numérique et le monde physique. Toutefois, leur objectifs sont différents : l'invisibilité consiste à faire disparaître le monde numérique, tandis que la transparence consiste à rendre floue la limite entre les deux mondes. Par exemple, un bouton de démarrage caché derrière un obstacle n'est pas une interface transparente car ce n'est ni naturel, ni intuitif. En revanche, le système est invisible tant que l'utilisateur n'a pas besoin d'utiliser ce bouton. En comparaison, les tables tactiles, comme la plupart des interfaces tactiles, offrent des interfaces très transparentes car l'action physique et le retour visuel sont alignés (i.e. on touche directement l'élément graphique avec lequel on souhaite interagir). Néanmoins, ces tables prennent souvent de la place et attirent l'attention. elle peuvent donc empêcher un système d'être invisible.

Internet des Objets et informatique ambiante

Les systèmes IoT doivent satisfaire 4 exigences : 1) être extensibles ; 2) fournir une API pour chaque objet ; 3) rendre disponible cette API via Internet ; 4) pouvoir identifier tous les objets de façon unique. L'usage d'objets connectés comme interfaces d'interaction font des MIBS des systèmes ubiquitaires, ils doivent alors satisfaire 6 autres exigences : 1) pouvoir percevoir le contexte ; 2) mais aussi le gérer ; 3) être capable d'inter-opération spontanée ; 4) supporter la mobilité logique et physique ; 5) supporter les interactions utilisateur transparentes ; 6) pouvoir se confondre avec l'environnement pour devenir invisibles. L'inter-opération spontanée est un aspect essentiel de l'adaptation dynamique des systèmes ubiquitaires aux changements dans un environnement.

1.3 Les MIBS

En plus des exigences précédentes, les MIBS doivent satisfaire des exigences qui leurs sont propres. Selon Neßelrath [114], ces spécificités proviennent de l'utilisation de systèmes cyber-physiques (CPS : *Cyber-Physical System*) pour réaliser des interactions multimodales. Chaque CPS est composé d'un ensemble d'objets, ainsi que des logiciels et services les exploitant pour fournir une API commune. Dans son étude des MIBS, Neßelrath [114] identifie 3 exigences à satisfaire : le *choix libre de modalités et combinaison de modalités*,

la *gestion de la multimodalité massive* et l'*abstraction des modalités.*

Choix libre de modalités et combinaison de modalités : pour assurer une interaction n'importe où et n'importe quand, l'interface utilisateur multimodale doit pouvoir évoluer. Ainsi, un MIBS doit être capable d'interagir avec l'utilisateur en s'adaptant aux modalités disponibles et aux techniques d'interaction que ces modalités permettent de réaliser.

Gestion de la multimodalité massive : les objets considérés dans l'IoT ont des capacités de captation/action et de calculs très variés [29]. Par exemple, un capteur de mouvement bon marché (ex. capteur de présence de Wiz [3]) fournit une information très simple par rapport aux capacités des enceintes intelligentes telles que les enceintes Echo équipées de l'assistant Alexa [4]. De plus, Neßelrath fait la distinction entre les objets demandant une action consciente de l'utilisateur (ex. un bouton d'une télécommande ou un écran TV) et ceux qui n'ont pas vocation à être interactifs (ex. un thermomètre ou un ventilateur). Comme les objets installés sont différents d'un environnement à un autre, les MIBS doivent donc supporter cette "multimodalité massive" (i.e. grande hétérogénéité des objets d'entrée et de sortie), d'autant plus qu'il ne semble pas y avoir une convergence des approches proposées par les fournisseurs d'objets connectés et services IoT [3].

Abstraction des modalités : les informations données par les modalités peuvent être à des niveaux d'interprétation très variés. Si l'on souhaite créer un service qui détecte si une personne dort, on peut entre autres détecter ses mouvements par vision ou mesurer son rythme cardiaque. Concevoir toutes les interprétations possibles serait complexe et nuirait à la réutilisabilité de ces composants d'interprétation. Il faut donc pouvoir concevoir le noyau fonctionnel d'une application indépendamment des modalités. L'abstraction des dispositifs physiques (exigence de la multimodalité) et l'abstraction des modalités pourraient être considérées comme identiques car les deux participent au découplage entre les dispositifs physique et le noyau applicatif, et facilitent donc l'adaptation des MIBS aux environnements d'exécution. Cependant, elles représentent deux niveaux de séparations différentes : entre les objets et le reste du système, et entre le noyau fonctionnel d'une application et le reste du système. Nous gardons donc la distinction entre ces deux exigences par la suite.

La reconfiguration dynamique de l'ensemble des objets disponibles et l'adaptation aux défaillances et indisponibilités des objets sont aussi mis en avant par Neßelrath comme des caractéristiques que les MIBS doivent avoir. Toutefois, ces caractéristiques font partie

3. `https://www.wizconnected.com/fr-be/p/accessoire-detecteur-de-mouvement/8718699788209`

4. `https://www.amazon.fr/b?ie=UTF8&node=15428386031`

de la problématique d'inter-opération spontanée propre à l'informatique ambiante.

Les MIBS

Les MIBS doivent satisfaire 3 exigences qui leur sont spécifiques : 1) permettre de choisir librement les modalités et leurs combinaisons ; 2) supporter la multimodalité massive (i.e. grande hétérogénéité des objets d'entrée et de sortie) ; 3) permettre d'abstraire les modalités pour le noyau fonctionnel. Satisfaire ces 3 exigences contribue au processus d'adaptation des MIBS aux environnements. En effet, le support de la multimodalité massive et l'abstraction des modalités permettent de gérer la diversité des environnements plus facilement, tandis que le choix libre de modalités et combinaison de modalités facilite l'adaptation des interfaces aux modalités disponible.

1.4 MIBS et contexte

Le contexte, ou contexte d'usage, dans un système informatique correspond généralement à un ensemble de données partagé, structuré et évolutif qui est modélisé dans un but spécifique (*evolving, structured, and shared information spaces, and that such spaces are designed to serve a particular purpose*) [53]. Le contexte a une grande importance dans les MIBS : il permet au système d'adapter l'interaction (i.e. le noyau fonctionnel ainsi que le choix des modalités et des techniques d'interaction) à l'environnement d'exécution. De plus, cette adaptation doit être dynamique, car les environnements ubiquitaires changent fréquemment.

Prenons l'exemple d'un utilisateur qui se fait guider dans un bâtiment, de sa position actuelle à la position d'un autre utilisateur (qui peut lui-même se déplacer). Dans un premier temps, l'utilisateur doit être repéré par différents capteurs placés à différents endroits de son itinéraire. Pour cela, le parcours de l'utilisateur doit être analysé dynamiquement, et le système doit sélectionner les capteurs qui peuvent fournir l'information voulue. Il faut ensuite informer cet utilisateur du parcours à suivre. Pour cela, le système doit trouver les objets connectés disponibles et à proximité qui peuvent transmettre ce genre d'informations. Dans ce scénario, le système a besoin de connaître plusieurs informations contextuelles telles que : la structure du bâtiment, la position des capteurs, ou

encore les positions et identités des deux utilisateurs. Ces informations sont ensuite utilisées pour choisir les dispositifs physiques capables de capter la position des utilisateurs et d'informer du chemin à suivre. Ces informations peuvent aussi être utilisées pour adapter le message de guidage (ex. informer l'utilisateur s'il ne prend pas le bon chemin). On remarque à travers cet exemple l'importance de la perception qu'ont les MIBS du monde physique, aussi appelé le "monde observable" [12].

Pour assurer une bonne gestion de ces problématiques, la première étape est d'étudier la notion de contexte et de ce qu'elle recouvre. Pour cela, plusieurs travaux proposent de catégoriser le contexte selon les éléments qui définissent l'état du "monde observable".

Coutaz et Nigay [52] séparent le contexte en trois éléments : l'utilisateur, la plateforme et l'environnement physique et social. Ici, la plateforme est décrite par les ressources de calcul, de communication et d'interaction disponibles en temps réel, et notamment les modalités.

Similairement, plusieurs travaux comme SIAM-DP [114] et l'étude sur les interactions situationnelles dans [26] considèrent que le contexte se sépare entre les informations représentant les utilisateurs, les objets et l'environnement. De plus, les objets sont classifiés dans [114] selon leur position, leur distance d'interaction et le nombre d'utilisateurs potentiels avec lesquels les objets peuvent interagir. Ici, les capacités de calculs et de communication des plateformes ne sont pas considérées.

Hönold et al. [78] séparent les informations de contexte en 8 catégories : application, tâche, environnement, utilisateur, dialogue, information, présentation et composants. Cette classification leur permet de définir les informations utilisables selon le type d'adaptation envisagé. Par exemple, ils considèrent que l'adaptation du comportement d'un dispositif physique en entrée ne dépend que des informations sur l'environnement et l'utilisateur.

Contrairement aux approches précédentes, Knappmayer et al. [87] proposent une classification du contexte centrée sur la notion de *situation*. Ainsi, une situation se décompose en 9 aspects : spatial, temporel, objets, communication, environnement, utilisateur, activité, mental et interaction. Pour reprendre l'exemple de guidage, une situation possible serait : l'utilisateur guidé se situe seul dans un couloir au début de l'interaction, et il écoute attentivement une indication vocale transmise par une enceinte connectée au WiFi. La notion d'environnement est donc ici séparée des informations spatiales, et l'état mental des utilisateurs est séparé de son profil utilisateur (i.e. préférences et habitudes). De plus, la notion de temporalité est considérée comme une information contextuelle.

Enfin, Augusto [12] propose une approche complémentaire aux précédentes en définissant une représentation du contexte dans les environnements intelligents selon 4 aspects : le *nom* du contexte, le *bénéficiaire* (ex. l'utilisateur final), la *condition d'activation* et l'*effet* du contexte. L'intérêt de cette représentation est de définir explicitement l'usage du contexte, ce qui facilite le passage de sa conception à son implémentation en pratique.

En résumé, la perception du contexte est aspect important pour permettre l'adaptation dynamique des MIBS aux environnements d'exécution. De plus, le contexte dans les MIBS peut être défini de différentes façons. Cette diversité de paradigme s'explique par la dépendance entre la représentation du contexte et son usage [52], et plus généralement par l'absence de représentation standardisée du contexte [12]. Toutefois, les approches présentées proposent généralement dans leurs représentation du contexte 4 types d'entités : **l'utilisateur**, **l'environnement**, **les dispositifs physiques** et **le système**. Les autres aspects plus spécifiques à chaque approche peuvent pour la plupart être inclus dans les 4 entités (ex. le mental dans [87]) ou sont au croisement entre deux entités (ex. la plateforme dans [52]). Pour assurer la compatibilité de notre approche du contexte aux méthodes existantes (et possiblement à de futurs standards), nous utilisons par la suite ces 4 entités pour définir le contexte. Nous verrons par la suite que ces entités suffisent pour décrire les informations utiles au processus d'adaptation des MIBS.

1.5 Synthèse

Nous avons vu dans ce chapitre que la conception de MIBS requiert de répondre à des exigences provenant des domaines de la multimodalité, de l'internet des objets, de l'informatique ambiante ainsi que des critères qui leur sont propres (voir tableau 1.1). Ainsi, nous donnons la définition suivante pour les MIBS :

Un MIBS est un système qui utilise les API proposées par des objets connectés à Internet et identifiables pour fournir des interfaces multi-sensori-motrices entre des utilisateurs et des applications interactives. Les interactions proposées peuvent être mobiles, invisibles ou transparentes, et utilisent des informations contextuelles collectées et traitées dynamiquement. Les applications interactives peuvent utiliser de manière interchangeable les objets qui proposent des capacités interactives similaires. Pour cela, ces objets sont représentés selon les modalités qu'ils offrent, et les services sont définis indépendamment de la notion de modalité.

On observe que la capacité d'adaptation des MIBS aux environnements dépend des

réponses apportées à 6 des exigences identifiées. Tout d'abords, les MIBS doivent supporter la multimodalité massive (C13) et doivent permettre l'abstraction des dispositifs physiques (C2) et l'abstraction des modalités (C15). En effet, répondre à ces 3 exigences permet de faciliter le découplage entre les divers objets connectés et les noyaux fonctionnels des applications multimodales, et donc de concevoir et développer les MIBS pour être les plus génériques possibles. Ensuite, les MIBS doivent être capables d'interagir avec les utilisateurs en utilisant les capacités interactives de tous les objets disponibles (C14), et cela même si l'état et la disponibilité de ces objets évoluent dans le temps (C9). Enfin, la perception du contexte (C7) est nécessaire pour permettre cette adaptation dynamique des MIBS aux changements dans un environnement.

Nous analyserons dans le chapitre suivant comment les réponses aux exigences présentées dans ce chapitre sont traitées dans la littérature.

TABLE 1.1 – Cette table synthétise les exigences que les MIBS doivent satisfaire.

Domaine	Identifiant	Description
Multimodalité	C1	Support d'interactions multi-sensori-motrices
	C2	Abstraction des dispositifs physiques
Internet des Objets	C3	Extensibilité
	C4	Connexion à internet
	C5	Objets identifiables
	C6	Services à disposition
Informatique ambiante	C7	Perception du contexte
	C8	Gestion du contexte
	C9	Inter-opération spontanée
	C10	Mobilité
	C11	Interaction utilisateur transparente
	C12	Invisibilité
MIBS	C13	Choix libre de modalités et combinaison de modalités
	C14	Gestion de la multimodalité massive
	C15	Abstraction des modalités

Chapitre 2

Méthodes de création des MIBS dans la littérature

Plusieurs approches, ou frameworks, ont été proposées pour fournir des MIBS qui répondent aux exigences présentées dans le chapitre précédent. Pour répondre à ces exigences, ces frameworks s'inspirent des standards architecturaux qui peuvent provenir des domaines de la multimodalité ou de l'informatique ambiante. C'est pour cela que l'étude de ces standards et des frameworks est une étape importante pour identifier ce qui permet de faciliter la création des MIBS.

Dans ce chapitre, nous abordons d'abord un état de l'art sur les différentes architectures logicielles des MIBS. Nous étudions ensuite les frameworks permettant de faciliter la création des MIBS selon ces architectures. Nous analysons enfin des limites des approches existantes, et des problématiques restantes.

2.1 Architectures des MIBS

Comme vu précédemment, les MIBS héritent des caractéristiques des systèmes multimodaux et des systèmes ubiquitaires. Pour répondre aux exigences provenant de ces domaines, il est courant que les architectures des MIBS soient inspirées des architectures de ces deux familles de systèmes. Pour identifier les points communs, nous commençons par une revue des caractéristiques des architectures des systèmes multimodaux et des systèmes ubiquitaires, et plus particulièrement les caractéristiques qui permettent de répondre aux exigences présentées précédemment. Ensuite, nous étudions les principes architecturaux des MIBS.

2.1.1 Standards architecturaux de la multimodalité

La plupart des systèmes multimodaux proposent les mêmes fonctionnalités pour gérer l'utilisation de plusieurs dispositifs physiques offrant différentes modalités pour communiquer avec l'utilisateur. Comme illustré en figure 2.1, ces fonctionnalités sont intégrées dans une architecture de référence communément utilisée pour concevoir ce genre de système [138, 56]. Dans cette architecture, les modalités communiquent avec le noyau fonctionnel, aussi appelé "comité d'intégration" [61]. Ce comité est lui-même composé de 4 composants :

— un module de fusion ;
— un moteur de dialogue ;
— un module de fission ;
— un gestionnaire de contexte.

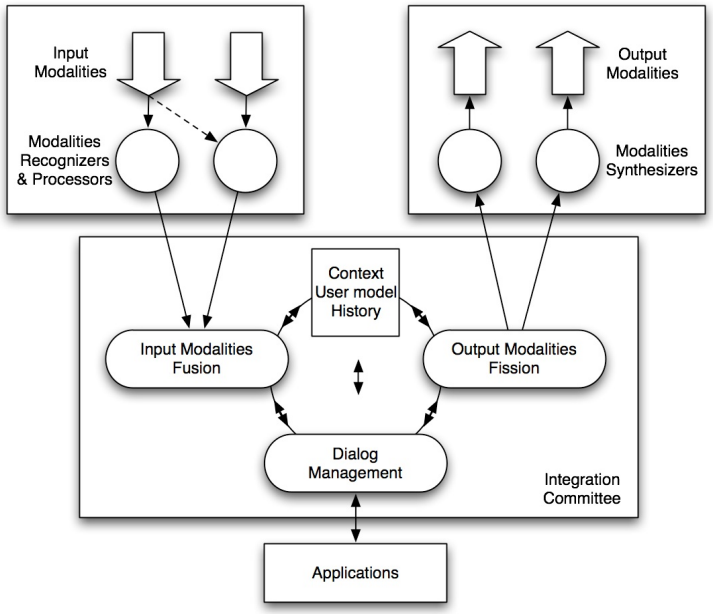


FIGURE 2.1 – Architecture logicielle d'un système multimodal [61]

Le *module de fusion* a pour rôle d'interpréter les données reçues par les modalités d'entrée, ce qui inclut la fusion de ces informations. Ces informations sont transmises au *moteur de dialogue* qui contrôle le déroulement du dialogue entre l'utilisateur et le système interactif. La réponse généré par le moteur de dialogue est restituée par le *module de fission* qui choisit la bonne modalité, le bon dispositif de sortie et le bon format pour cette réponse. Enfin, le *gestionnaire de contexte* assure l'adaptation du système interactif en gérant

l'évolution de la base de connaissances formant le contexte.

Cette architecture permet de satisfaire les exigences associées au domaine de la multimodalité (exigences C1 et C2 dans le tableau 1.1). En effet, les processus d'interprétation et de présentation permettent d'abstraire les dispositifs physiques (C2). De plus, les capacités de fusion en entrée et de fission en sortie facilitent l'utilisation de plusieurs modalités différentes dans une même interaction (C1).

Par la suite, nous utilisons le terme de *module d'interprétation* pour mettre en avant la différence entre les processus d'interprétation par fusion et les autres processus d'interprétation (qui n'utilisent pas des informations provenant de différentes sources). De même, nous utilisons le terme de *module de présentation* pour décrire le composant qui gère le processus de restitution.

2.1.2 Standards architecturaux de l'informatique ambiante

Nous avons vu que la conception de plateformes IoT et de systèmes ubiquitaires impose des exigences qui viennent s'ajouter à celles relatives à la multimodalité. Ainsi, plusieurs plateformes, middleware, et frameworks ont été proposés pour supporter la création de tels systèmes. En particulier, les revues de littérature [144] et [3] permettent d'identifier les standards architecturaux qui donnent des éléments de réponse aux exigences relatives à ces systèmes.

Dans leur étude des technologies utilisés dans le domaine de l'IoT, Sethi et Sarangi [144] expliquent qu'il n'y a pas de consensus sur l'architecture des plateformes IoT. Toutefois, ils identifient deux composants majeurs de ces systèmes : une *couche de perception* et une *couche de communication*. La *couche de perception* assure la collecte de données au travers de différents capteurs, et donc la perception du contexte (C7). Ensuite la *couche de communication* est gérée par un middleware. Ce middleware assure l'échange sur le réseau de données captés par des objets identifiables (C5), quel que soit le protocole de communication (C4). De plus, l'extensibilité (C3) des plateformes IoT est assurée par la distribution logicielle des éléments de l'architecture. En effet, la *couche de communication* permet de réaliser une partie du traitement des données au niveau de la source (i.e. hardware du dispositif physique), et donc de minimiser les besoins en bande passante entre les objets et le serveur hébergeant la plateforme. Cette distributivité permet aussi de supporter la mobilité logique (C10). Pour assurer le suivi du déplacement des objets et utilisateurs, et donc la mobilité physique (C10), la *couche de perception* doit aussi intégrer des mécanismes de suivi.

Dans leur revue de littérature de plateformes IoT et de systèmes ubiquitaires, Alberti et al. [3] examinent 20 propositions pour en déterminer les points communs et les limites. Voici plusieurs points communs observés entre les architectures existantes :

— le protocole de communication favori pour assurer la connexion à internet (C4) est TCP/IP ;
— les modèles de communications majoritaires sont les solutions requête-réponse (principalement pour la simplicité des architectures RESTful [15]) et publish-subscribe (pour sa nature asynchrone et sa méthode de couplage indirecte entre émetteur et récepteur, ce qui facilite l'inter-opération spontanée (C9)) ;
— les objets sont souvent représentés par des interfaces logicielles et des jumeaux numériques qui permettent entre autres d'assurer le nommage des objets (C5) et faciliter l'inter-opération entre les objets (C9), ce qui est un premier pas pour gérer la multimodalité massive (C14).

Ici, un jumeau numérique est un clone virtuel qui décrit entièrement un produit réel ou potentiel [72].

Alberti et al. affirment aussi que les architectures sont généralement constituées de 6 couches : objets (i.e. hardwares et logiciels embarqués), connectivité (ex. accès au réseau (C4)), adaptation (ex. gestion des modèles de données), virtualisation, service (API) et application (à destination des utilisateurs). On notera que la virtualisation consiste ici à représenter en une version virtuelle des objets et autres matériels physiques. Cela permet ainsi d'abstraire des objets par des services (C6).

Enfin, les notions de transparence (C11) et d'invisibilité (C12) ne sont pas des propriétés du système, la validation de ces exigences dépend plutôt de la connaissance qu'a l'utilisateur final de ce système, et de ce qu'il considère comme transparent [142]. Ainsi, seul un choix averti des objets et de leur positionnement dans chaque environnement permet d'installer des MIBS selon ces notions, et elles ne peuvent être validées que par l'usage du système dans l'environnement final [41]. Il ne s'agit donc pas d'exigences auxquelles on peut répondre par des choix d'architecture, donc nous n'évaluerons pas par la suite les approches existantes selon ces critères.

2.1.3 Standards architecturaux des MIBS

Plusieurs architectures pour les MIBS se sont construites sur les principes architecturaux des systèmes multimodaux et ubiquitaires. Prenons par exemple l'architecture du MIBS présentée dans [21] et [77]. Cette architecture permet de réaliser des interactions

multimodales implicites et explicites. De plus, elle intègre une méthode de fission probabiliste qui inclut un mécanisme de raisonnement comme pour un système ubiquitaire.

Toutefois, les MIBS doivent répondre à des exigences qui leurs sont propres. Nous détaillons par la suite les standards architecturaux qui répondent à ces exigences. En particulier, nous présentons deux approches souvent employées pour créer des MIBS : l'architecture proposée par le World Wide Web Consortium (W3C), et le paradigme des architectures orientées composants. Enfin, nous revenons sur les standards architecturaux qui permettent de gérer le contexte.

MIBS et défis architecturaux

Les MIBS doivent faciliter l'utilisation de dispositifs ayant des capacités interactives très variées (C14). Cela ajoute des contraintes supplémentaires sur la conception des composants de modalités. Almeida et al. [4] mettent en avant l'importance de la modularité de l'architecture pour plus facilement séparer de la partie générique les spécificités des dispositifs.

De plus, la gestion des modalités dans les systèmes multimodaux est souvent *ad hoc* [137], ce qui est un frein à la réutilisabilité et l'interchangeabilité des composants de modalités (C13). Il faut donc favoriser les couplages faibles entre composants quand cela est possible. Le besoin de compartimentation des différents éléments des MIBS (i.e. séparer le plus possible les fonctionnalités de ces systèmes dans des composants logiciels indépendants) est une contrainte forte. Nous verrons dans la section 2.1.3 que cela se traduit généralement par des architectures orientées composants.

Nous avons vu dans le chapitre 1.3 que chaque étape d'un dialogue entre un utilisateur et un MIBS peut être réalisé au travers des différentes modalités offertes par les objets connectés. Or, adapter le dialogue aux spécificités de chaque modalité n'est pas possible du fait de la grande diversité des capacités interactives des objets connectés. Il est donc nécessaire d'abstraire les modalités dans la gestion du dialogue (C15). Certaines approches [114, 4] définissent une représentation des actes de communication en des termes indépendants des technologies servant d'interface, communément appelés des actes de dialogue [1]. Un acte de dialogue inclut un ensemble d'attributs, ou annotations, décrivant l'information transmise [36]. Par exemple, la description de la question "Quelle heure est-il ?" peut renseigner l'émetteur, le destinataire, sa fonction de question et le fait que la phrase fait référence au temps. L'utilisation d'un acte de dialogue requiert

1. `https://www.iso.org/fr/standard/76443.html`

des algorithmes de reconnaissance pour chacun de ces attributs. Cela signifie que pour ces approches par acte de dialogue, le module d'interprétation (et fusion) des MIBS doit intégrer ces algorithmes.

Pour proposer des MIBS qui respectent les exigences présentées précédemment, des approches comme [4] se basent sur les travaux du W3C. Nous présentons ces travaux dans la section suivante.

MIBS et W3C

Le groupe de travail sur l'interaction multimodale du W3C [2] est le principal contributeur aux normes multimodales de l'IoT [57]. Par exemple, le groupe a proposé le langage XML EMMA ("**E**xtensible **M**ulti**M**odal **A**nnotation") pour décrire les informations interprétées lors d'une fusion multimodale. C'est pour cela que la plupart des architectures pour les MIBS s'inspirent de l'approche proposée par le W3C [3].

Cette architecture abstraite appelée MMI (voir figure 2.2) réduit les systèmes multimodaux à leurs composants principaux, à savoir un gestionnaire d'interaction (similaire au "comité d'intégration") et un ensemble de composants de modalité qui communiquent par événements asynchrones. Le choix du protocole est libre, mais le protocole choisi doit satisfaire 2 contraintes :

— la communication doit être fiable, et doit pouvoir générer des messages d'erreur en cas de problème ;
— les événements envoyés à un composant par une même source doivent arriver au destinataire dans l'ordre d'émission.

Le gestionnaire d'interaction peut être lié à un composant de données relatives à l'application dans l'éventualité où le gestionnaire d'interaction ne gère pas lui-même ces données. Les gestionnaire d'interaction, les composants de modalité et le composant de données ont tous un identifiant unique, ou URI (Uniform Resource Identifier), ce qui permet d'identifier chaque émetteur et récepteur de message (C5).

Ainsi, le W3C offre avec ses travaux un protocole standardisé de communication entre les composants des MIBS. L'approche du W3C met aussi en avant l'importance du paradigme de la programmation orientée composant que nous présentons par la suite.

2. `https://www.w3.org/2002/mmi/`
3. Copyright © 2012 W3C® (MIT, ERCIM, Keio). This software or document includes material copied from or derived from Multimodal Architecture and Interfaces `https://www.w3.org/TR/mmi-arch/`

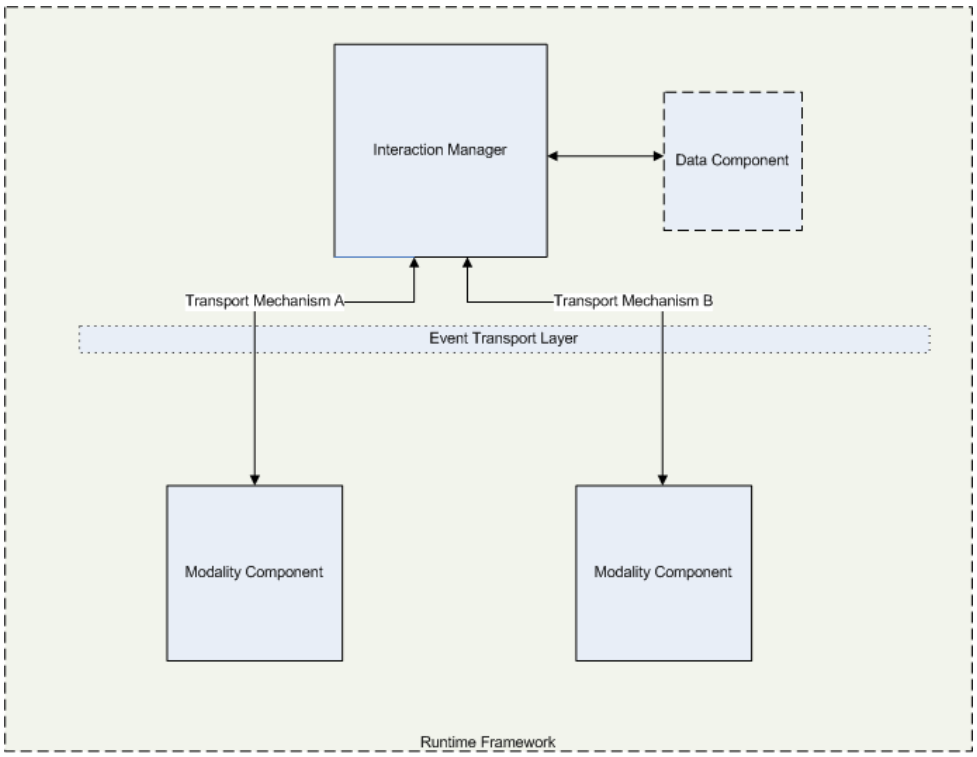


FIGURE 2.2 – Diagramme de l'architecture MMI à l'exécution (proposée par le W3C). Le gestionnaire d'interaction communique avec les composants de modalités (d'entrée et de sortie) pour coordonner le dialogue avec les utilisateurs. Le gestionnaire d'interaction échange aussi avec un composant de données gérant une base de connaissances à propos de l'application exécutée. Les composants de modalités peuvent aussi contenir leurs propres gestionnaires d'interaction et composants de données pour gérer l'interaction à l'échelle de la modalité.

MIBS et architectures orientées composants

L'approche orientée composants est le paradigme de développement le plus utilisé dans les MIBS (voir chapitre 2.2). Les architectures par composants séparent les fonctionnalités en différents blocs qui fonctionnent de manière indépendante, et dont les interfaces (i.e. les entrées et les sorties) sont strictement définies. La popularité de cette approche vient de plusieurs facteurs. Les composants représentent facilement les systèmes distribués [70], ils améliorent la capacité d'adaptation de ces systèmes [91], et ils peuvent être développés indépendamment pour encourager la coopération entre entreprises [57], réduisant ainsi le coût de production [4] et facilitant l'interopérabilité dans l'IoT [45] (C9). Les composants peuvent faire partie du cœur fonctionnel du système interactif ou être spécifiques à une application. Le cœur fonctionnel est exécuté à chaque session, tandis

que l'instanciation des composants applicatifs dépend de l'application exécutée, et peut être dynamique. Les composants en charge des modalités, de la fusion et de la fission sont souvent des composants applicatifs (i.e. conçus pour un usage spécifique), bien qu'il existe quelques exceptions. Par exemple, les composants de fusion et de fission sont uniques dans SMARTKOM [76], mais plusieurs de ces composants peuvent être crées et utilisés dans des prototypes de systèmes multimodaux avec la plateforme OpenInterface [93].

Toutefois, les composants sont souvent fortement couplés entre eux [4], ce qui peut nuire à leur réutilisabilité, et donc à l'adaptabilité des MIBS. Chang et al. [45] suggèrent l'utilisation de plateformes orientées containers logiciels (ex. OSGi [4] ou Docker [5]). Ces containers permettent d'exécuter les composants en isolation, et n'exposent que les fonctionnalités désirées des processus contenus. Cela facilite la cohabitation des processus, et en particulier les mises à jour d'un composant n'impactent pas le fonctionnement des autres composants.

Une autre approche consiste à représenter les composants comme des services [131]. Contrairement à l'approche orientée composants, les processus ne peuvent échanger des données que par messages standardisés sur le réseau. L'échange ne dépend donc pas de l'infrastructure (ex. si les processus sont sur différent hôtes ou sur le même).

MIBS et gestion du contexte

Nous avons vu dans le chapitre 1.4 que la représentation du contexte dépend de la finalité. Le contexte peut par exemple être utilisé pour adapter automatiquement les UI, on parle alors de plasticité des interfaces [52]. Dans cet exemple, les informations contextuelles comme les capacités interactives des objets (ex. la dimension des écrans TV) ou la position relative entre l'utilisateur et les objets connectés peuvent être utilisées pendant cette phase d'adaptation. Le contexte peut aussi être utilisé pour adapter le dialogue. Par exemple l'échange entre le système et l'utilisateur peut être abrégé si le système a accès à l'emploi du temps de l'utilisateur dans l'application de réservation de salle de réunion.

La gestion du contexte ne s'arrête toutefois pas à l'utilisation de ce contexte, mais nécessite de capter et interpréter les données provenant de plusieurs sources (C7 et C8), puis d'enregistrer et partager les informations traitées avec les composants qui en ont besoin. Dans leur étude, Knappmeyer et al. [87] identifient certaines caractéristiques de

4. `https://www.osgi.org/`
5. `https://www.docker.com/`

la gestion du contexte dans les systèmes sensibles au contexte ("context-aware") tels que les MIBS :

— ces systèmes ont généralement un composant (i.e. un serveur central) dédié à la gestion du contexte ;
— le composant de contexte fournit une interface permettant de questionner la base de connaissances, comme des requêtes SQL ;
— l'acquisition et l'interprétation des données peuvent être internes ou externes à ce composant.

Toutefois, les MIBS utilisent déjà des objets connectés pour collecter et interpréter des informations concernant l'utilisateur. Il est donc judicieux que le gestionnaire de contexte puisse acquérir ces informations externes.

La gestion du contexte requiert aussi de choisir une représentation du contexte. Différentes représentations du contexte sont possibles lors de la conception de systèmes ubiquitaires [87]. Par exemple, les modèles clé-valeur représentent le contexte comme des données dans un dictionnaire, et les modèles logiques représentent le contexte comme un ensemble de faits, expressions et règles.

Parmi les modèles existants, le contexte dans les MIBS est plus souvent représentés par des ontologies [131, 114], c'est-à-dire par des relations entre un ensemble d'entités et de propriétés décrivant un domaine particulier de connaissances [6]. L'avantage de ce modèle est de proposer, en théorie, la meilleure modélisation du contexte, notamment car chaque domaine de connaissances peut avoir sa propre ontologie. Cela facilite l'extension de la base de connaissances et évite toutes les ambiguïtés qu'il pourrait y avoir sur un terme utilisé différemment suivant la communauté concernée (ex.entre le domaine de la multimodalité et de l'informatique ubiquitaire).

6. `https://www.w3.org/TR/mediaont-10/`

Standards architecturaux des MIBS

L'étude des standards architecturaux nous a permis d'identifier les solutions apportées pour chacune des exigences. Les standards architecturaux utilisés dans les domaines de la multimodalité et de l'informatique ambiante permettent de répondre à la plupart des exigences auxquelles les MIBS doivent répondre. De plus, plusieurs travaux comme ceux du W3C donnent des réponses pour les exigences propres aux MIBS. Les exigences de transparence (C11) et d'invisibilité (C12) ne peuvent pas être satisfaites en se basant uniquement l'architecture des MIBS, donc nous n'utiliserons pas ces exigences par la suite.

2.2 Frameworks de création de MIBS

Il existe pour chacun des standards architecturaux présentés précédemment des méthodes et outils facilitant leur mise en oeuvre. Toutefois, il est fréquent que les différentes approches ne soient pas directement compatibles, et qu'un travail d'adaptation coûteux entre les différents éléments conçus soit nécessaire [56, 9]. Par exemple, l'interprétation des données provenant d'un objet connecté peut être effectuée par un composant d'interprétation, par un composant de modalité gérant cet objet, ou encore peut être répartie entre les deux. Des composants supplémentaires peuvent être nécessaires pour faire le lien entre des processus d'interprétation développés selon ces différentes approches.

C'est pour cela que nous cherchons un framework capable de satisfaire toutes les exigences présentées précédemment. En particulier, ce framework doit permettre de facilement adapter les MIBS aux spécificités des environnements.

Dans la suite de cette section, nous présentons 6 frameworks proposant des solutions pour créer des MIBS, de la conception au déploiement. Pour chaque framework, nous étudions les points principaux de leurs architectures, nous illustrons les approches sur le cas d'usage sur l'application de guidage présentée en introduction, et nous synthétisons les réponses apportées aux exigences (voir tableau 1.1). '✓' représente si le framework permet de répondre à une exigence, '(✓)' si la solution est partielle et '-' si aucune solution n'est apporté.

2.2.1 AIM

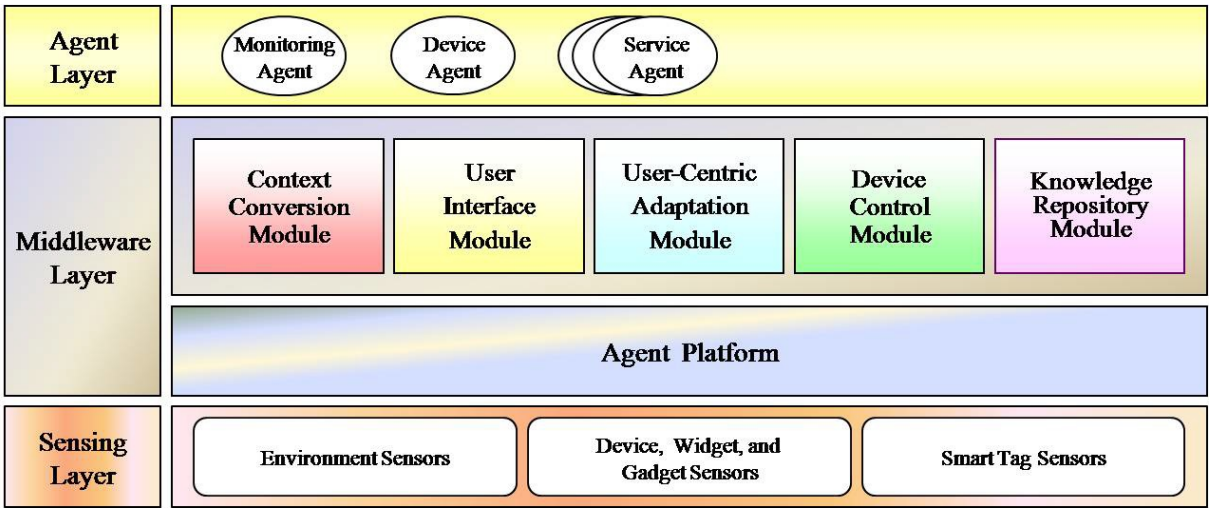


FIGURE 2.3 – Les couches du modèle AIM

Jang et al. [81] proposent le middleware AIM (Agent-based Intelligent Middleware) pour supporter les services centrés utilisateur sensible au contexte. Ce middleware orienté agent est inspiré des approches utilisées en informatique ubiquitaire. Il intègre 5 agents (communication par Internet (C4, C10)) qui s'interposent entre l'environnement et les services (voir figure 2.3) : le *module de conversion de contexte*, le *module d'adaptation*, le *module de contrôle des objets*, le *module de gestion de connaissances* et le *module d'interface utilisateur*. Les objets connectés sont eux-même gérés par des agents qui communiquent avec le *module de conversion de contexte* et le *module de gestion de connaissances*.

Le *module de conversion de contexte* est chargé d'enrichir les données captées provenant des différents objets connectés avec des informations supplémentaires (C6, C7). Les données contextualisées sont enregistrées sous format XML dans une structure appelée "contexte d'intégration" (Integration Context).

Le *module d'adaptation* détecte l'intention de l'utilisateur. On notera que les contextes d'intégration sont utilisés directement pour faire la prédiction, et qu'il n'y a pas d'information sur la méthode utilisée pour inférer l'intention.

Le *module de contrôle des objets* identifie les objets disponibles et leurs caractéristiques. Pour caractériser les objets, ces derniers sont décrits par des ontologies (C14). Néanmoins, il n'y a pas plus de détail sur les ontologies utilisées.

Le *module de gestion de connaissances* gère l'interprétation et le stockage sous format XML des informations telles que les profils utilisateur et les caractéristiques sur l'équipe-

ment (C8). Les autres composants peuvent ensuite questionner cette base pour obtenir les informations de contexte nécessaires (C3). En particulier, les caractéristiques des objets sont utilisées pour construire l'interface utilisateur et contrôler les actionneurs (C1). Il n'y a toutefois pas de détails fournis quant à la distinction faite entre les différents types d'objets (ex. les modalités que les objets peuvent fournir ne sont pas explicitées).

Le *module d'interface utilisateur* adapte l'interface pour qu'elle soit plus naturelle. Pour fournir ces interfaces naturelles, ce module prédit, à partir des événements agrégés dans la base de données, les tâches que l'utilisateur réalise. Puis, à partir de ces prédictions et des informations contenues dans la base de contexte, l'interface utilisateur est générée (C13). L'agrégation des informations des données captées et des données interprétées dans une base de contexte permet d'éviter tout problème potentiel dû à la connexion ou déconnexion de capteurs (C9, C13).

Toutefois, les tâches et services sont fortement couplés aux objets utilisés (C15), ce qui peut nuire à leur réutilisabilité. De plus, peu d'information est donnée sur le fonctionnement exact des agents. L'efficacité de ces composants est donc difficile à estimer et l'utilisation du modèle d'architecture proposé peut se résumer au suivi de recommandations visant à faire respecter ce modèle.

Avec ce framework, pour mettre en place un MIBS qui permette de réaliser le cas d'usage de guidage, chaque tâche doit être conçue et développée en fonction des objets installés. Par exemple, chaque ampoule peut être associée à une tâche de guidage consistant à s'allumer. Au début de l'interaction, le *module d'adaptation* estime que l'utilisateur veut être guidé. Quand l'utilisateur passe à proximité d'une ampoule, le *module d'interface utilisateur* infère que l'utilisateur veut réaliser la tâche de guidage associée à cette ampoule, et donc demande à l'agent qui gère cette ampoule de s'allumer. Exprimer cette dépendance des tâches aux objets peut être relativement long et fastidieux selon le nombre d'objets installés et de tâches que l'on souhaite réaliser.

Les réponses aux exigences sont synthétisées dans le tableau 2.1.

TABLE 2.1 – AIM [81] : réponses apportées aux exigences

Exigences (identifiants)	Validation	Description
Support d'interactions multi-sensori-motrices (C1)	✓	Gestion de la reconnaissance des informations provenant de différents capteurs
Abstraction des dispositifs physiques (C2)	-	Aucun détail sur la représentation des données
Extensibilité (C3)	✓	Uniformisation et centralisation de tous les événements dans une base de données unique
Connexion à internet (C4)	✓	Communication entre agents par Internet
Objets identifiables (C5)	-	Aucun détail sur la représentation des objets
Services à disposition (C6)	(✓)	Communication des données par événements. Contextualisation des données. Connexion manuelle des objets au middleware
Perception du contexte (C7)	✓	Collecte et contextualisation des données
Gestion du contexte (C8)	✓	Interprétation et stockage des données contextualisées dans une base de connaissances
Inter-opération spontanée (C9)	✓	Capture du contexte indépendant de son usage. Processus de présentation sensible au contexte
Mobilité (C10)	(✓)	Utilisation d'agents distribuables. Aucun détail sur la méthode de suivi des entités mobiles
Choix libre de modalités et combinaison de modalités (C13)	✓	Capture du contexte indépendant de son usage. Processus de présentation sensible au contexte
Gestion de la multimodalité massive (C14)	(✓)	Utilisation d'ontologies d'objets dans les processus d'interprétation et présentation. Pas de détails sur les ontologies
Abstraction des modalités (C15)	-	Description des tâches en fonction des objets

AIM (Agent-based Intelligent Middleware)

Avec AIM, la centralisation et la contextualisation des données captées avant leur utilisation permet à la fois de percevoir les informations de contexte et d'inférer l'intention des utilisateurs selon les capteurs disponibles, et les interfaces sont générées et adaptées à partir de cette base commune de connaissances et des capacités interactives des objets. Toutefois, aucune solution n'est apportée pour facilement concevoir les services interactifs séparément des objets utilisés, ce qui complique le travail d'adaptation aux environnements.

2.2.2 DAME

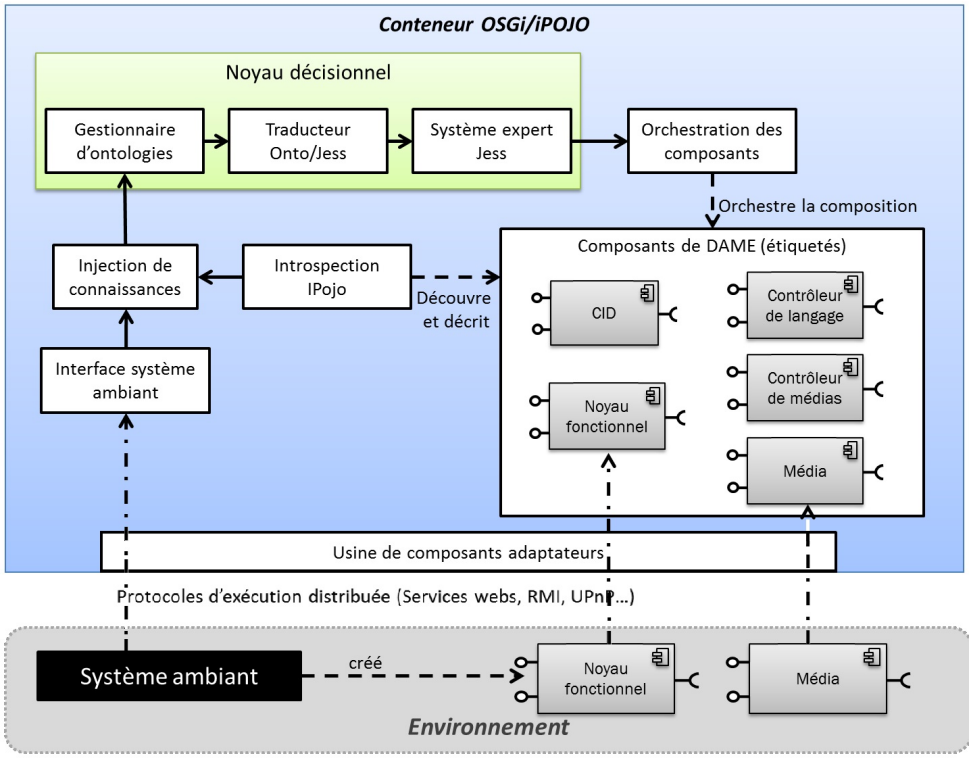


FIGURE 2.4 – Architecture implémentant l'approche DAME

DAME (Distributed Ambient Multimodal Environments) est un environnement encadrant la conception et le développement de systèmes multimodaux par composants selon le paradigme de l'intelligence ambiante. La démarche de création consiste à concevoir et développer des composants, et à définir les règles permettant au système créé de s'adapter à son environnement. Les composants (voir figure 2.4) sont répartis en 5 catégories :

- les composants associés à des média (i.e. dispositifs physiques et services externes) spécifiques (C1) servent d'API pour contrôler ces médias (i.e. ils sont étiquetés (C5), et exposent les capacités interactives des objets (C6)).
- les « contrôleurs de média » (CM) gèrent l'abstraction des ressources d'interaction d'entrée et de sortie (i.e. les médias) en tant que modalité (C2).
- les composants de contrôle de langages gèrent l'interprétation et la restitution.
- les composants d'interface dédiée (CID) définissent le déroulé du dialogue, et s'appuient sur des langages d'interaction indépendants des modalités utilisées (C15).
- les composants du noyau fonctionnel (NF) correspondent aux fonctionnalités du

domaine applicatif de la tâche en cours.

Pour adapter au mieux le système interactif à l'utilisateur, la plateforme génère dynamiquement des chaînes d'interaction, c'est-à-dire des chaînes de composants allant des médias aux composants gérant le dialogue, et sélectionne la meilleure de ces chaînes. Les différentes chaînes candidates sont classées selon des règles arbitraires liées au contexte. Ces critères peuvent être intégrés par les acteurs en charge du déploiement et de la maintenance du système. La méthode de composition dynamique proposée permet de reconfigurer l'interface utilisateur (C13), ce qui facilite l'interchangeabilité des composants dans les limites des modèles de composant à disposition (C9).

Prenons par exemple le scénario de guidage. Un MIBS développé avec DAME va devoir décider quels média parmi les ampoules, les afficheurs et les enceintes connectés ceux qui permettrons de transmettre la direction à suivre. Tout d'abord, il va générer toutes les combinaisons possibles de chaînes d'interaction (i.e. les candidats), en partant des composants du NF qui satisfont les besoins de l'utilisateur pour arriver aux média. On aura donc un candidat pour tous les sous-ensembles de média, ce qui fait 16! possibilités si l'on ne considère que les 16 ampoules connectées. Ensuite, chaque candidat est évalué par un ensemble de métriques et critères pour sélectionner le meilleur. Les candidats qui utilisent des enceintes peuvent être interdits si d'autres personnes se trouvent à proximité, et le candidat qui utilise les objets les plus proches du chemin à suivre peut être sélectionné. D'autres règles pourraient être proposées, mais il y aura toujours un écart entre des règles génériques et les spécificités de chaque environnement, ce qui peut entraîner des situations imprévues. par exemple, le changement de direction dans le scénario de guidage arrive quand l'utilisateur vient juste de dépasser une ampoule. Les candidats qui utilisent cette ampoule seront mieux notés selon la métrique de proximité.

On remarque donc que la génération de tous les candidats peut prendre du temps selon la capacité de calcul à disposition, et les règles permettant d'évaluer ces candidats ne permettent pas d'assurer l'utilisabilité du système déployé.

De plus, les métriques utilisées s'appuient sur des informations contextuelles. Or, la collecte d'information de contexte (C7) n'est pas directement gérée par l'approche DAME mais les informations collectées viennent mettre à jour une base de connaissances par ontologies (C8, C14). Ces ontologies sont par la suite utilisées pour adapter l'interface utilisateur.

Les règles d'adaptation influencent la stratégie de composition automatique de DAME. L'aspect ergonomique du système est considéré dans la conception et la pondération de

ces règles. Le framework fournit 3 outils :

— *Describe* facilite et sécurise l'édition des ontologies formant la base de connaissances du système.
— *Behave* permet l'édition des règles décrivant les situations que les designers UX et ergonomes souhaitent favoriser ou éviter.
— *Simulate* permet à l'ergonome de simuler le comportement du système selon plusieurs configurations.

La plateforme utilise le conteneur OSGi/iPOJO pour faciliter l'implémentation des composants par les développeurs. Ainsi, les composants peuvent être distribués sur le réseau (C3, C4, C10), bien que cela demande d'implémenter des composants adaptateurs tiers.

Les réponses aux exigences sont synthétisées dans le tableau 2.2.

Table 2.2 – DAME [131] : réponses apportées aux exigences

Exigences (identifiants)	Validation	Description
Support d'interactions multi-sensori-motrices (C1)	✓	Utilisation de composants servant d'interfaces entre les objets et le système
Abstraction des dispositifs physiques (C2)	✓	Utilisation de composants de médiation
Extensibilité (C3)	✓	Support de la distribution de composants logiciels (middleware)
Connexion à internet (C4)	✓	Interfaces des composants accessibles depuis le réseau
Objets identifiables (C5)	✓	Gestion d'un registre de composants. Convention de nommage
Services à disposition (C6)	✓	Utilisation de composants servant d'interfaces entre les objets et le système
Perception du contexte (C7)	-	Collecte des informations de contexte externe
Gestion du contexte (C8)	✓	Base de connaissances par ontologies
Inter-opération spontanée (C9)	✓	Composition dynamique de chaînes d'interaction
Mobilité (C10)	(✓)	Architecture distribuée (mobilité des composants). Pas de support spécifique aux déplacements physiques
Choix libre de modalités et combinaison de modalités (C13)	✓	Méthode de génération de chaînes d'interaction
Gestion de la multimodalité massive (C14)	✓	Utilisation d'ontologie et interprétation/présentation par composition
Abstraction des modalités (C15)	✓	Langage d'interaction indépendant des modalités

DAME (Distributed Ambient Multimodal Environments)

La méthode de génération de chaînes d'interaction utilisée dans DAME permet d'assurer l'adaptation dynamique des MIBS aux changements dans les environnements. De plus, ces chaînes sont composées de différents types de composants qui assurent le découplage entre les objets connectés, les techniques d'interaction et le noyau fonctionnel de chaque application. Cependant, aucune solution n'est proposée pour supporter la perception du contexte.

2.2.3 SIAM-DP

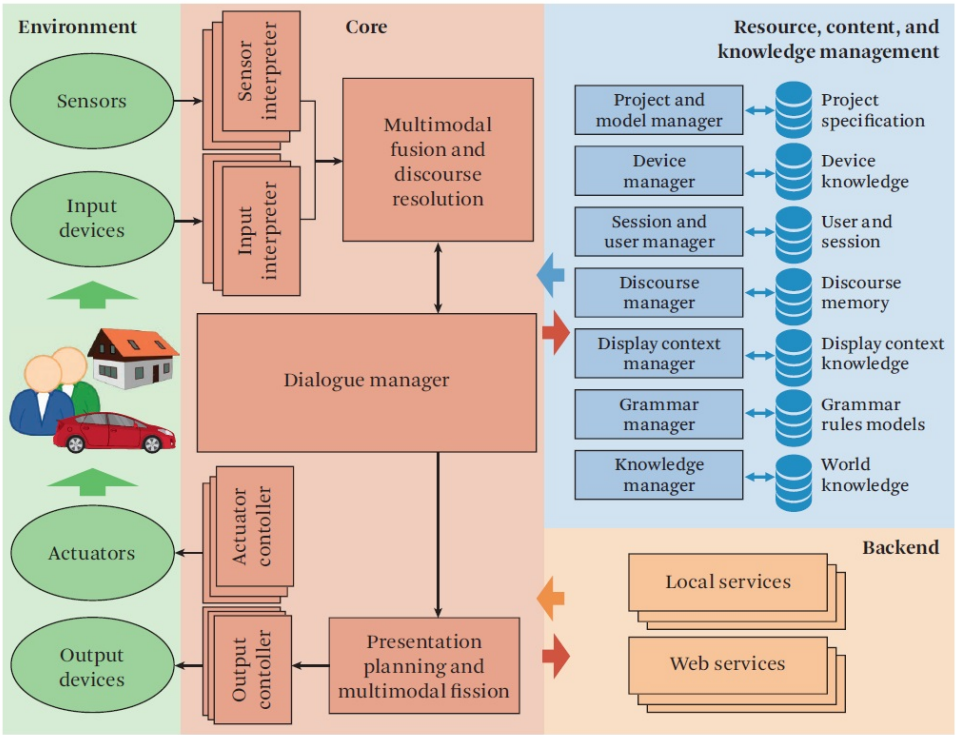


FIGURE 2.5 – Architecture de SIAM-DP

La plateforme SIAM-DP (SItuation Adaptive Multimodal Dialogue Platform) facilite la conception rapide de systèmes de dialogues multimodaux en utilisant des modèles sémantiques. Son architecture modulaire (voir figure 2.5) est composée de 3 composants cœur : un *composant de fusion*, un *gestionnaire de dialogue à machine à états* et un

composant de restitution. Comme pour DAME, les composants peuvent être exécutés par différents ordinateurs (C3, C10) sur le réseau (C4).

Chaque capteur a un interpréteur associé qui fournit au *composant de fusion* l'intention de l'utilisateur selon un modèle sémantique. Le *composant de fusion* utilise les informations sémantiques générées ainsi que les informations de contexte pour contextualiser les intentions des utilisateurs et résoudre les problèmes d'ambiguïté entre plusieurs interprétations. Ainsi, toutes les données captées sont traitées au niveau sémantique avant d'être comparées (C7, C8).

Les états du *gestionnaire de dialogue* sont décrits sous un format XML, et le framework propose un outil graphique d'édition de dialogue qui peut aussi simuler le dialogue pour analyser le coût en calcul (les auteurs prévoient à l'avenir de permettre la visualisation de plus d'informations, comme les étapes du dialogue produisant trop de charges cognitives). Un langage indépendant des modalités est utilisé pour concevoir le dialogue (C15). Le modèle de dialogue par machine à états permet de facilement décrire des dialogues simples, mais est moins adapté pour représenter des dialogues plus complexes. Ainsi, les états peuvent faire appel à des programmes exécutables pour complexifier la structure du dialogue.

Le *composant de restitution* réalise une optimisation de l'interface finale (i.e. comment les informations sont concrètement transmises par les différentes ressources d'interaction de sortie) par comparaison d'interfaces finales candidates selon des variables de contexte (C9, C13). Ces interfaces candidates sont générées à partir de composants de rendu spécifiques à des ressources d'interaction de sortie.

L'architecture intègre les objets de tous types : capteurs, actionneurs et autres dispositifs d'entrée et de sortie (C1). Tous les objets en entrée passent par des interpréteurs, et les objets en sortie ont leurs propres composants de présentation (C2). Comme chaque composant d'interprétation ou de présentation est spécifique à un objet, réutiliser les processus d'interprétation intégrés dans ces composants pour d'autres objets requiert un travail d'adaptation supplémentaire. Par exemple, deux caméras RGB différentes peuvent avoir des caractéristiques différentes, mais les deux peuvent fournir le même type d'information sémantique. Avec SIAM-DP, il faut développer deux composants d'interprétation, un pour chaque caméra.

Toutes les entrées et sorties des objets sont représentées par des messages intégrant des méta-données telles que les identifiants (C5), les caractéristiques des objets et les modalités associées. Cette contextualisation des données permet au système d'obtenir une

description des capacités interactives des objets disponibles, et donc d'identifier dynamiquement les services qu'ils fournissent (C6). De plus, pour gérer la diversité de capacités interactives des objets, l'auteur propose une classification hiérarchique des objets (C14). Les objets dans chaque classe sont aussi décrits par des attributs communs (ex. la position, la distance d'interaction).

Avec ce framework, dans le cas d'usage de l'application de guidage chaque objet connecté doit être associé à son propre interpréteur qui infère l'acte de communication. Par exemple, chaque caméra a un interpréteur qui donne l'information sémantique sur la position estimé de l'utilisateur. Si une des caméras a des caractéristiques différentes (ex. caméra de profondeur et caméra RGB), alors il faudra développer un nouvel interpréteur pour obtenir la même information sémantique. L'autre option est d'uniformiser les informations syntaxiques fournies par les composant gérant les objets connecté, mais cela demande de pré-interpréter les informations provenant des capteurs, ce qui nuit à la réutilisabilité de ces composants dans d'autres services. La granularité de la chaîne d'interprétation que permet SIAM-DP n'est donc pas optimale.

Les réponses aux exigences sont synthétisées dans le tableau 2.3.

Table 2.3 – SIAM-DP [114] : réponses apportées aux exigences

Exigences (identifiants)	Validation	Description
Support d'interactions multi-sensori-motrices (C1)	✓	Support de plusieurs modalités en entrée et en sortie
Abstraction des dispositifs physiques (C2)	✓	Représentation des dispositifs selon un modèle générique
Extensibilité (C3)	✓	Support de la distribution de composants logiciels
Connexion à internet (C4)	✓	Interfaces des composants accessibles depuis le réseau
Objets identifiables (C5)	✓	Utilisation d'identifiants uniques
Services à disposition (C6)	✓	Utilisation de composants servant d'interfaces entre les objets et le système
Perception du contexte (C7)	✓	Base de connaissances alimentée par les données captées
Gestion du contexte (C8)	✓	Gestionnaire de contexte
Inter-opération spontanée (C9)	✓	Adaptation dynamique des interfaces en fonction des objets à disposition
Mobilité (C10)	(✓)	Architecture distribuée (mobilité des composants). Pas de support spécifique aux déplacements physiques
Choix libre de modalités et combinaison de modalités (C13)	✓	Adaptation dynamique des interfaces en fonction des objets à disposition
Gestion de la multimodalité massive (C14)	(✓)	Classification hiérarchique des objets. Composants d'interprétation et de présentation spécifiques à chaque objet
Abstraction des modalités (C15)	✓	Langage d'interaction indépendant des modalités

SIAM-DP (SItuation Adaptive Multimodal Dialogue Platform)

Les composants d'interprétation et de restitution assurent le découplage total entre les dispositifs physiques et le noyau applicatif. Ce découplage, ainsi que le processus d'adaptation automatique des interfaces permet de s'adapter plus facilement aux dispositifs physiques disponibles dans l'environnement. Pour réaliser cette adaptation automatique, les données captées et leurs interprétations sont toutes enregistrées dans une base commune de connaissances. La classification des objets permet de supporter en partie la multimodalité massive. Cependant, la dépendance forte entre les dispositifs et les composants d'interprétation/présentation augmente le coût de développement de ces composants, ce qui est une limite face à la diversité des objets connectés.

2.2.4 MIBO

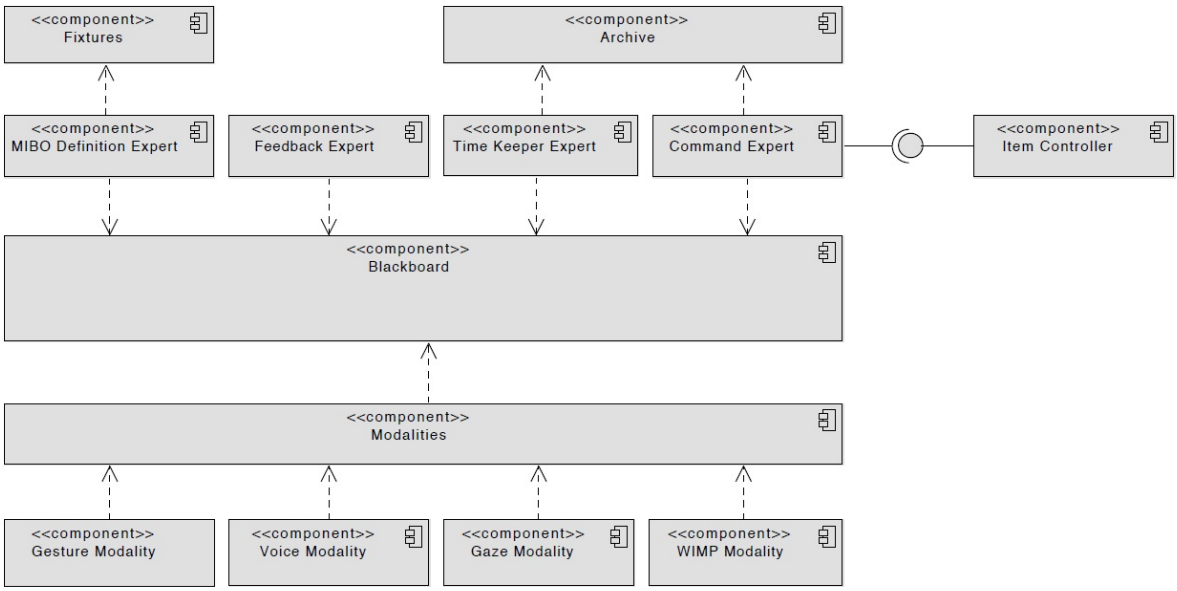


FIGURE 2.6 – diagramme UML des composants de MIBO

MIBO [125] est un framework qui a pour vocation de concevoir des systèmes multimodaux pour contrôler les objets connectés capables d'agir sur l'environnement (i.e. lumière, stores, ...). Il suit une architecture multi-agent avec un agent pour chaque modalité (C1, C2). Les données fournies par ces agents sont ensuite agrégées (C3, C7) dans un tableau (blackboard) (voir figure 2.6). Ce tableau est une base de données centralisées qui peut être mise à jour dynamiquement. De plus, un composant supprime à intervalles réguliers les données invalides (ex. les commandes utilisateurs trop anciennes). Tous les autres composants prennent en entrée les données de ce tableau et écrivent en sortie dans le même tableau. Par exemple, les données dans le tableau sont utilisées par un moteur de règles pour fournir des informations de plus haut niveau (ex. fusion de données). Ces nouvelles données sont stockées dans ce tableau, ce qui permet d'enrichir la base de connaissances de manière itérative (C8, C14).

Pour assurer la communication avec l'ensemble des objets, le framework s'appuie sur OpenHAB [7]. OpenHAB est un logiciel qui permet d'interagir avec les services offerts par des objets connectés (C6) tout en cachant les protocoles de communication sous-jacents (C4).

Il n'y a pas de description de dialogue : chaque interaction est une règle qui agit sur l'environnement dès que certaines conditions sont atteintes (i.e. les événements associés sont déclenchés, et les sélecteurs fournissent les informations désirées). "*Allumer la lumière* (action) quand un utilisateur *pointe du doigt* (condition 1) *une ampoule* (condition 2)" est un exemple de règle possible avec cette approche. Ce modèle fonctionne relativement bien sur des scénarios de contrôle simples, mais demande plus d'effort des développeurs pour réaliser un échange long (les dépendances entre les étapes ne sont pas natives comme dans une machine à états) Ainsi, l'abstraction des modalités dans la modélisation du dialogue ne dépend ici que du contenu du tableau (C15).

L'abstraction des dispositifs physiques en sortie (C2) n'est pas réalisée : les commandes sont directement destinées à des objets spécifiques. En revanche, un composant de feedback permet d'observer à l'exécution l'état du système en s'abonnant à des événements prédéfinis. La gestion des sorties est toutefois limitée à des feedbacks sur les informations captées. Bien que ces retours puissent être représentés par plusieurs modalités, cela demande un travail d'implémentation supplémentaire.

À partir des données dans le tableau, un composant (« MIBO definition Expert ») analyse les connaissances pour réaliser des commandes selon des modèles d'interaction.

7. `https://www.openhab.org/`

Ces modèles sont des règles composées de 5 éléments :

- les objets *"à portée"* de la commande, ce qui correspond aux objets que l'on peut (i.e. à portée) et que l'on veut contrôler avec la commande ;
- le *point d'entrée* est le déclencheur de la règle (i.e. la méthode d'engagement du dialogue) ;
- la *sélection* définit la ou les méthodes de récupération (ex, fusion de modalités) des informations nécessaires à l'interaction ;
- le *contrôle* définit la technique d'interaction utilisée pour réaliser la commande ;
- la *sortie* est le déclencheur qui finit l'interaction (i.e. la méthode de désengagement du dialogue).

On notera qu'il n'est pas nécessaire de spécifier lequel des capteurs est la source des informations nécessaires au dialogue. En effet, les informations sont récupérées auprès du tableau, ce qui évite aux applications de dépendre de la disponibilité des capteurs (C9).

Prenons l'exemple de l'application de guidage. La *"portée"* correspond aux différents actionneurs (ex. ampoule, écrans) qui permettent de guider l'utilisateur au travers du bâtiment. Le passage du portique par l'utilisateur remplit le rôle de *point d'entrée*. Puis, les actionneurs à contrôler sont *sélectionnés* en fonction de la position de l'utilisateur dans le bâtiment. Ensuite, la direction que prend l'utilisateur permet de définir comment *contrôler* les objets (ex. allumer les ampoules qui sont devant l'utilisateur et éteindre les autres). Enfin, l'arrivée de l'utilisateur à destination permet de *sortir* de l'interaction.

Au déploiement, une interface permet de créer et configurer les modèles d'interaction pour choisir parmi les modalités et techniques d'interaction la façon la plus satisfaisante de contrôler des actionneurs (C13). Il est possible de faire cohabiter plusieurs instances de ce type de service. De plus, cette interface alerte l'utilisateur en cas de conflit entre instances. Les réponses aux exigences sont synthétisées dans le tableau 2.4.

TABLE 2.4 – MIBO [125] : réponses apportées aux exigences

Exigences (identifiants)	Validation	Description
Support d'interactions multi-sensori-motrices (C1)	(✓)	Support de plusieurs modalités en entrée. Interaction en sortie limité au contrôle d'objets
Abstraction des dispositifs physiques (C2)	(✓)	Représentation des objets par leurs modalités limitée à l'entrée
Extensibilité (C3)	✓	Mise en commun des données dans un tableau unique
Connexion à internet (C4)	✓	Interfaces des composants accessibles depuis le réseau
Objets identifiables (C5)	-	Aucun détail sur la représentation des objets
Services à disposition (C6)	✓	Communication des données par événements
Perception du contexte (C7)	✓	Collecte des données dans un tableau unique
Gestion du contexte (C8)	✓	Interprétation et stockage des données utilisées par les applications
Inter-opération spontanée (C9)	(✓)	Utilisation des données indépendamment de leurs sources (entrée seulement)
Mobilité (C10)	-	Aucun support
Choix libre de modalités et combinaison de modalités (C13)	(✓)	Interface de configuration (limitée au choix des applications et des techniques d'interaction) au déploiement
Gestion de la multimodalité massive (C14)	(✓)	Processus d'interprétation centralisé. Aucun support en sortie
Abstraction des modalités (C15)	✓	Langage d'interaction indépendant des modalités

MIBO

Les modèles d'interaction dans MIBO sont conçus indépendamment des capteurs utilisés, mais l'abstraction des dispositifs physiques en sortie n'est pas réalisée, ce qui limite le découplage entre les actionneurs et les noyaux fonctionnels des applications. L'utilisation d'un tableau pour centraliser et interpréter les données avant leur utilisation par les noyaux fonctionnels des applications a deux avantages : cela permet de percevoir le contexte et d'assurer l'inter-opérabilité entre les capteurs (mais pas entre les actionneurs).

2.2.5 PHASER

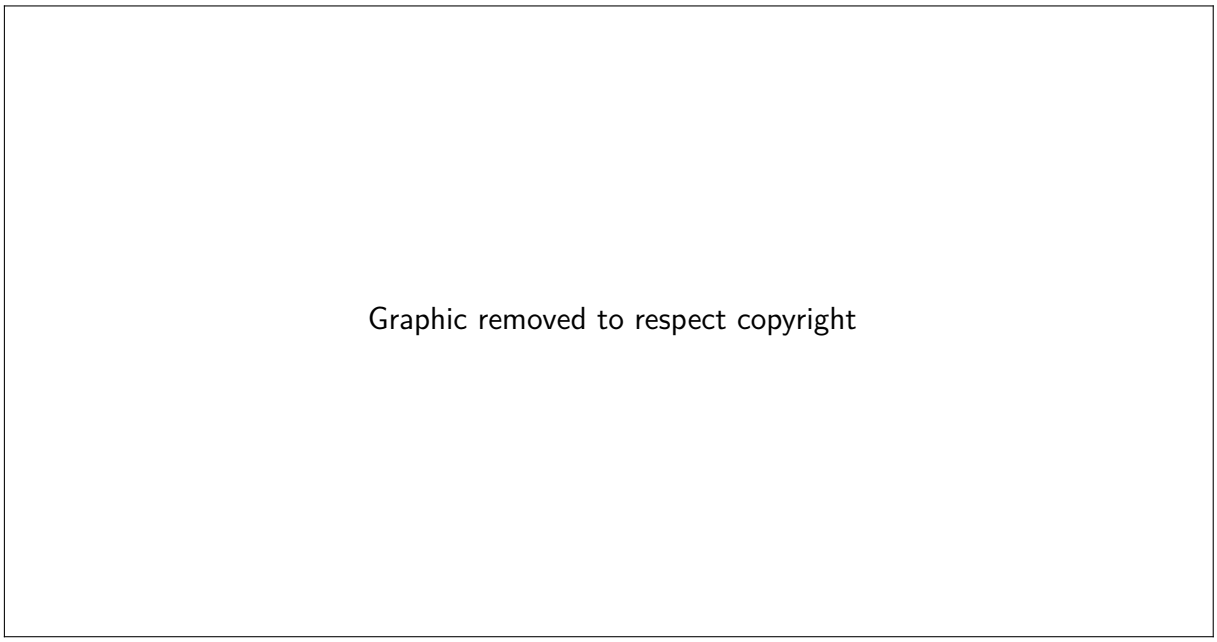


FIGURE 2.7 – Exemple de réseau de nœuds PHASER

PHASER (Pervasive Human-centred Architecture for Smart Environmental Responsiveness) [58] est un framework qui consiste à mettre en œuvre un réseau de systèmes interactifs multimodaux, appelés nœuds, distribués sur un ou plusieurs dispositifs physiques (voir figure 2.7). Chaque nœud intègre un cœur fonctionnel d'un système multimodal, ainsi que des composants d'interprétation et présentation spécifiques aux dispositifs physiques associés (C1). Le cœur fonctionnel est composé de 6 composants :

- un composant pour recevoir les données des autres nœuds ;
- un module de synchronisation (temporelle) des données d'entrées ;
- un module de fusion (i.e. module d'interprétation) ;
- un gestionnaire du dialogue ;
- un module de synchronisation (temporelle) des données en sortie ;
- un composant pour envoyer des données aux autres nœuds.

Le fonctionnement de ces composants n'est pas davantage expliqué : les points majeurs de cette approche sont la modélisation des échanges entre les nœuds, et la configuration de ces nœuds.

Chaque nœud est capable d'interagir avec l'utilisateur selon les modalités des objets associés (C2). Ainsi, il est possible de définir séparément les comportements que doivent

avoir ces objets dans leur environnement direct (i.e. il n'est pas nécessaire d'avoir une vue globale du système et de l'environnement) (C10). Cela signifie aussi que le choix des modalités (C13) est contraint par les capacités interactives du nœud, et non de l'ensemble du réseau.

Pour configurer les nœuds, un outil permet de dessiner en vue du dessus le plan d'une d'un espace, de placer des icônes représentant les objets associés, et de paramétrer le nœud. Ces paramètres sont rangés en deux catégories :

— les paramètres qui correspondent au comportement isolé d'un nœud. ils définissent la capacité du nœud à recevoir et envoyer des données aux autres nœuds ;
— les paramètres qui correspondent au comportement d'un nœud dans un groupement de nœuds. Ces paramètres définissent comment gérer les groupes et communiquer à l'intérieur de ce groupe.

Ainsi, un nœud peut communiquer par Websockets avec les nœuds proches (C4) si les paramètres le permettent (C6). La communication peut être nécessaire pour plusieurs raisons. Tout d'abord, cela permet à un nœud d'utiliser les capacités interactives des autres nœuds quand ses capacités interactives ou les informations à sa disposition sont insuffisantes pour continuer le dialogue. Cela peut aussi être utile pour partager une information au sein d'un groupe (C7).

Prenons l'exemple de l'application de guidage dans le bâtiment d'une entreprise. Imaginons que chaque espace (ex. couloir, salles) a son propre nœud, et qu'un utilisateur initie le service à partir du nœud qui est exécuté sur sa montre connectée. Le nœud de la montre n'a pas directement accès aux capacités interactives lui permettant de diriger l'utilisateur (ex. l'écran est trop petit pour afficher un plan). Il envoie donc une requête spécifiant le besoin d'afficher les informations de navigation au noeud de l'espace actuellement occupé par l'utilisateur. Ce nœud peut donc ensuite utiliser les modalités de sortie disponible (ex. un écran TV) pour guider l'utilisateur jusqu'à l'espace suivant.

L'absence de composant central (hormis la base de données répertoriant les nœuds (C5)), et l'utilisation d'un mécanisme de communication de proche en proche rend cette approche théoriquement capable de réaliser des inter-opérations spontanés (C9) et de fonctionner dans des infrastructures à différentes échelles (C3).

Les réponses aux exigences sont synthétisées dans le tableau 2.5.

TABLE 2.5 – PHASER [58] : réponses apportées aux exigences

Exigences (identifiants)	Validation	Description
Support d'interactions multi-sensori-motrices (C1)	✓	Support des modalités associées à chaque nœud
Abstraction des dispositifs physiques (C2)	✓	Représentation des dispositifs par leur modalités
Extensibilité (C3)	✓	Architecture entièrement distribuée. Mécanisme de communication de proche en proche
Connexion à internet (C4)	✓	Interfaces des nœuds accessibles depuis le réseau
Objets identifiables (C5)	✓	Utilisation d'identifiants uniques
Services à disposition (C6)	✓	Interface unique pour tous les nœuds
Perception du contexte (C7)	✓	Recherche d'informations par communication entre nœuds
Gestion du contexte (C8)	-	Aucun support
Inter-opération spontanée (C9)	✓	Adaptation des interfaces par communication entre nœuds
Mobilité (C10)	✓	Lien direct entre le logiciel (i.e. nœud), l'objet et l'environnement
Choix libre de modalités et combinaison de modalités (C13)	(✓)	Choix des modalités en fonction des capacités interactives des nœuds
Gestion de la multimodalité massive (C14)	-	Aucun support
Abstraction des modalités (C15)	-	Aucun support

PHASER

Dans PHASER, un système interactif, ou nœud, est associé à un ou plusieurs objets localisés dans un espace limité. Ces nœuds sont capables de chercher les informations de contexte et les capacités interactives qui leur font défaut auprès des autres nœuds. Cette coordination des nœuds permet d'adapter dynamiquement les interfaces, bien que le processus d'adaptation favorise les modalités locales plutôt que de considérer l'interface dans sa globalité. Le découplage entre les divers objets connectés et les noyaux fonctionnels des applications multimodales n'est pas un point abordé dans cette approche. La réutilisabilité des composants n'est donc pas considérée.

2.2.6 AM4I

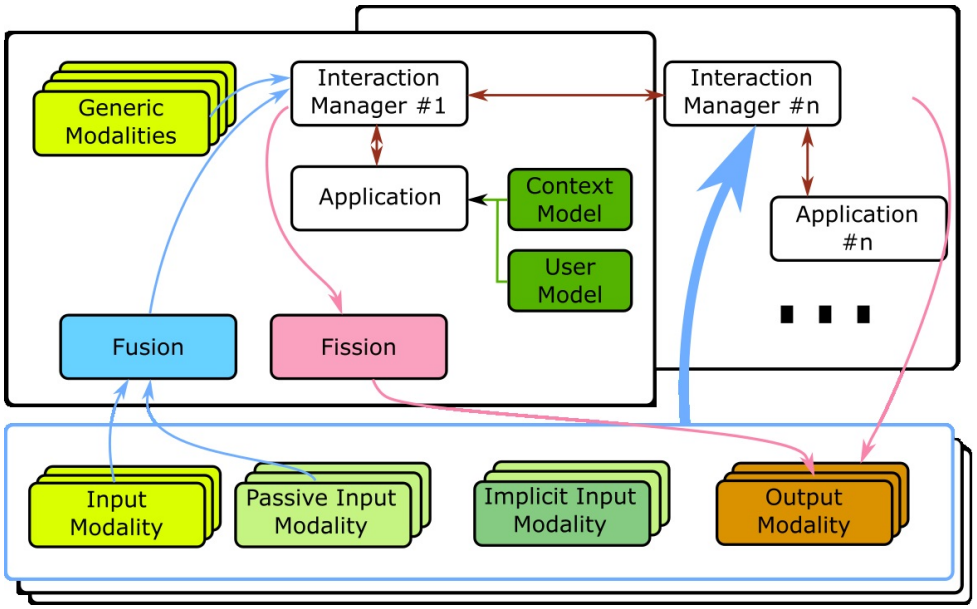


FIGURE 2.8 – Architecture du framework AM4I

AM4I (Adaptive Multiplatform Multidevice Multilingual Multimodal Interaction) [4, 135] est un framework proposant de concevoir des MIBS en se basant sur une architecture modulaire et distribuée (C3, C10). Cette approche s'est construite sur l'architecture par composants proposée par le W3C, et intègre plusieurs de leurs standards.

Ainsi, des composants de modalités identifiés par des URI (C5) fournissent des API (C6) permettant d'interagir avec les dispositifs physiques selon leurs capacités interactives (C1, C2). Les données provenant des modalités d'entrée sont fusionnées et le résultat de la fusion est envoyé à un gestionnaire d'interaction. Ce gestionnaire d'interaction communique ensuite avec des applications pour générer des messages abstraits à transmettre à l'utilisateur. Un composant de fission concrétise chaque message abstrait en plusieurs éléments d'interface et envoie ces éléments aux modalités de sorties disponibles (C13). Plusieurs gestionnaires d'interaction peuvent être exécutés en même temps pour gérer des ensembles d'objets différents, ou pour gérer différentes applications.

Les composants communiquent suivant le protocole HTTP (C4) et par événements tels que spécifiés par le W3C [8].

Les modalités sont réparties en 5 catégories de composants :

8. `https://www.w3.org/TR/mmi-arch/`

— les modalités d'entrée ;
— les modalités de sortie ;
— les modalités implicites/passives qui ont pour rôle de collecter des informations sur le contexte (C7) ;
— les modalités mixtes qui gèrent des modalités d'entrée et de sortie fortement couplées (ex. la voix est considéré comme d'entrée et de sortie) ;
— les modalités génériques.

Ces dernières sont des composants qui gèrent en interne une partie du processus de fusion ou de fission des données, et proposent donc des interfaces directement exploitables par le gestionnaire d'interaction. Les modalités fournissent des événements au niveau sémantique (ici des actes de dialogue), donc les applications gérées par les gestionnaires d'interaction sont créées indépendamment des modalités utilisées (C15).

L'intégration de modalités implicites dans les MIBS a son importance. Cacher les objets dans l'environnement contribue à l'effacement du monde numérique. Mettre à part les modalités implicites et passives permet d'identifier les objets qui les fournissent, et donc de mieux les placer au déploiement. Il n'y a toutefois pas de support supplémentaire pour correctement placer les objets.

Les capacités de mobilité des composants de l'architecture du W3C ont été préservées (C10). AM4I intègre un composant de détection d'utilisateurs, ce qui peut suffire pour les scénarios présentés, mais reste insuffisant pour des scénarios de mobilités plus complexes (ex. le suivi du parcours d'un utilisateur dans une pièce).

Le dialogue est représenté par des machines à états créées en langage SCXML [9] du W3C (machine à état).

Toutefois, la gestion du contexte est externalisée (C8), sauf pour le stockage des données.

Le niveau de granularité trop bas (i.e. fusion au niveau sémantique) est une limite de cette approche. En effet, pour obtenir la position de l'utilisateur dans le cas d'usage de guidage, chaque composant de modalité doit fournir la position de l'utilisateur sous un même format. Pour avoir cette même information sémantique à partir de capteurs ayant des capacités interactives différentes, plusieurs composants doivent être développés. Or, ils ne proposent pas de support pour réutiliser des processus d'interprétation communs.

Les réponses aux exigences sont synthétisées dans le tableau 2.6.

9. `https://www.w3.org/TR/scxml/`

TABLE 2.6 – AM4I [4] : réponses apportées aux exigences

Exigences (identifiants)	Validation	Description
Support d'interactions multi-sensori-motrices (C1)	✓	Représentation des capacités interactives des objets en fonction des modalités possibles
Abstraction des dispositifs physiques (C2)	✓	
Extensibilité (C3)	✓	Architecture distribuée avec centralisation des données captées
Connexion à internet (C4)	✓	Interfaces des composants accessibles depuis le réseau
Objets identifiables (C5)	✓	Identifiant unique des composants de modalité (W3C)
Services à disposition (C6)	✓	API des objets utilisables par plusieurs applications
Perception du contexte (C7)	✓	Collecte d'informations de contexte par des modalités passives
Gestion du contexte (C8)	-	Utilisation d'une base de connaissances externe
Inter-opération spontanée (C9)	✓	Adaptation dynamique des interfaces en fonction des objets à disposition
Mobilité (C10)	(✓)	Architecture distribuée (mobilité des composants). Composant de détection d'utilisateurs
Choix libre de modalités et combinaison de modalités (C13)	✓	Adaptation dynamique des interfaces en fonction des objets à disposition
Gestion de la multimodalité massive (C14)	-	Aucun support
Abstraction des modalités (C15)	✓	Représentation des informations sémantiques en actes de dialogue

AM4I

Pour faciliter la réutilisabilité de ses composants, AM4I supporte l'abstraction des dispositifs physiques avec des composants de modalités, ainsi que la représentation des données captées et interprétées par des informations sémantiques en actes de dialogue. Toutefois, les processus d'interprétation et de présentation sont fortement couplés aux composants de modalités. AM4I intègre des composants de modalités implicites/passives pour percevoir le contexte. Les interfaces sont dynamiquement adaptées aux modalités disponibles et au contexte.

2.3 Synthèse

Dans ce chapitre, nous avons vu les standards d'architectures utilisés pour concevoir des MIBS. Ces standards proviennent pour certains de la multimodalité, et pour d'autres

de l'informatique ubiquitaire.

Les frameworks existants qui proposent des solutions pour concevoir des MIBS de la conception au déploiement utilisent une partie des standards présentés. Toutefois, aucune des approches étudiées ne permet de répondre à toutes les exigences exprimées dans le chapitre 1 (comme illustré dans le tableau 2.7). En particulier, aucun des frameworks ne répond entièrement aux exigences liées à l'adaptation des MIBS aux environnements. En effet, AIM, MIBO et PHASER ne supportent pas assez le découplage entre les dispositifs physiques et le noyau applicatif. SIAM-DP et AM4I réalisent bien ce découplage, mais les processus d'interprétation et de présentation sont trop rigides pour facilement supporter la multimodalité massive. La forte granularité de ces processus dans DAME donne bien la flexibilité suffisante pour supporter l'hétérogénéité des objets connectés, mais la méthode d'adaptation automatique des interfaces ne permet pas d'assurer l'utilisabilité du système.

De plus, il n'est pas possible de d'intégrer ces frameworks aux solutions internes à Orange car leurs implémentations ne sont pas disponibles.

C'est pour cette raison que nous proposons dans le chapitre suivant un framework au niveau de l'état de l'art dans l'objectif de répondre à toutes les exigences du chapitre 1.

TABLE 2.7 – Synthèse des réponses apportées aux exigences par les frameworks existants

Exigences (identifiants)	AIM [81]	DAME [131]	SIAM-DP [114]	MIBO [125]	PHASER [58]	AM4I [4]
Support d'interactions multi-sensori-motrices (C1)	✓	✓	✓	(✓)	✓	✓
Abstraction des dispositifs physiques (C2)	-	✓	✓	(✓)	✓	✓
Extensibilité (C3)	✓	✓	✓	✓	✓	✓
Connexion à internet (C4)	✓	✓	✓	✓	✓	✓
Objets identifiables (C5)	-	✓	✓	-	✓	✓
Services à disposition (C6)	(✓)	✓	✓	✓	✓	✓
Perception du contexte (C7)	✓	-	✓	✓	✓	✓
Gestion du contexte (C8)	✓	✓	✓	✓	-	-
Inter-opération spontanée (C9)	✓	✓	✓	(✓)	✓	✓
Mobilité (C10)	(✓)	(✓)	(✓)	-	✓	(✓)
Choix libre de modalités et combinaison de modalités (C13)	✓	✓	✓	(✓)	(✓)	✓
Gestion de la multimodalité massive (C14)	(✓)	✓	(✓)	(✓)	-	-
Abstraction des modalités (C15)	-	✓	✓	✓	-	✓

Chapitre 3

Proposition d'un framework pour la réalisation et le déploiement des MIBS

Le chapitre précédent nous a permis d'identifier les solutions apportées aux exigences présentées dans le chapitre 1. Toutefois, une des attentes de cette thèse CIFRE était de concevoir et développer un framework permettant de créer des MIBS, et qui s'intègre aux solutions logicielles fournies par Orange, ce qui n'est pas possible avec les frameworks existants (i.e. solutions propriétaires, incompatibilités logicielles). C'est pour cela que nous présentons dans ce chapitre un framework qui s'inspire des différentes solutions apportées par la littérature pour répondre à ces exigences.

Nous commençons par une présentation générale du framework, puis nous présentons ensuite les différents composants du framework qui permettent de réaliser un MIBS fonctionnel et complet. Enfin, nous illustrons notre approche au travers des applications de réservation et de guidage présentées en introduction.

3.1 Vue générale du framework

Nos avons vu que pour faciliter la création de MIBS, les architectures existantes séparent les fonctionnalités de ces systèmes en plusieurs composants. C'est pour cela que nous proposons de concevoir les MIBS en respectant l'architecture orientée composants illustrée en figure 3.1.

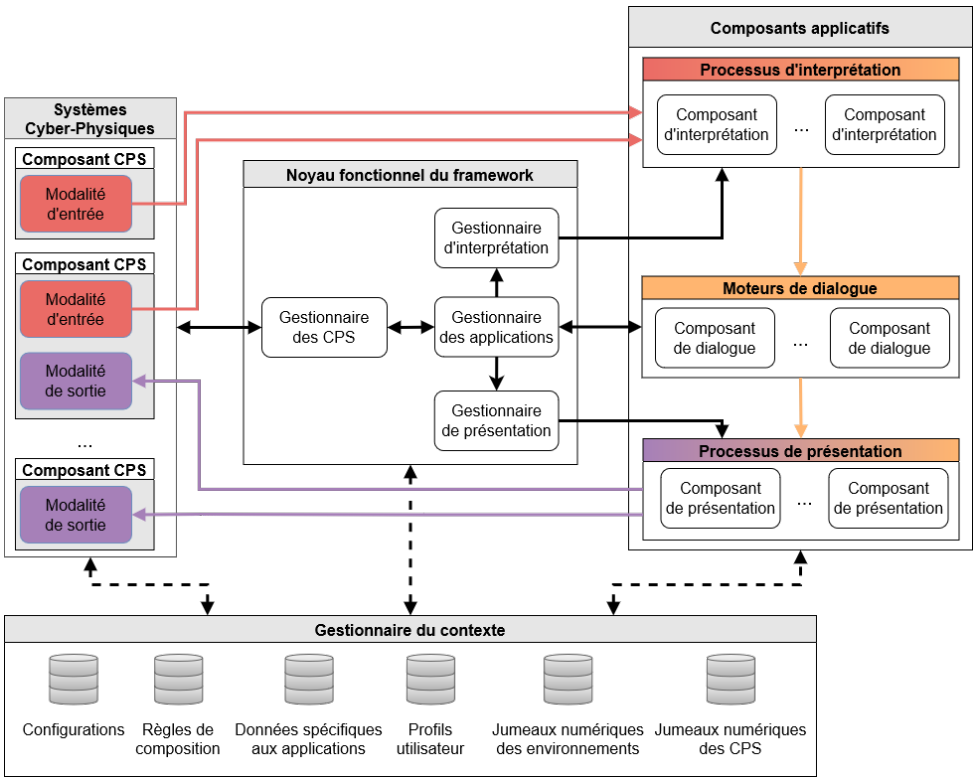


FIGURE 3.1 – Architecture globale de notre framework. Les flèches de couleurs illustrent le flux de données dans une chaîne d'interaction type. Les flèches en pointillés représentent la capacité de tous les composants à accéder aux données sauvegardées par le gestionnaire de contexte. Les cylindres correspondent aux différentes données dans la base de connaissances utilisée par le système et les flèches correspondent aux différents échanges d'informations entre les composants. La base de connaissances est accessible depuis un gestionnaire de contexte externe à l'application, et les composants de dialogue peuvent faire appel à des logiciels tiers.

Le framework comprend 4 catégories de composants :

— les composants CPS, qui servent d'interfaces entre les applications interactives et les systèmes cyber-physiques, ou CPS (voir chapitre 1.3) ;

— les composants formant le noyau du framework, qui gèrent l'instantiation, la modification et l'arrêt des applications en fonction du contexte ;

— le gestionnaire du contexte, qui donne accès à la base de connaissances, et permet l'évolution de cette base en fonction des données recueillies par les CPS et l'état des applications en cours d'exécution ;

— les composants applicatifs, qui sont des composants d'interprétation, de dialogue et de présentation qui forment les applications.

Comme proposé dans [87] nous utilisons une base de connaissances gérée par un gestionnaire de contexte centralisé pour toutes les applications de l'environnement. Ainsi, tous les CPS et applications peuvent participer à l'élaboration et l'enrichissement de la base de connaissances, et donc à la perception du contexte (C7). Le processus d'inférence sur le contexte est réalisé par les composants d'interprétation des chaînes d'interaction, ce qui se rapproche du processus itératif d'interprétation dans MIBO [125]. La séparation entre les applications et le gestionnaire de contexte permet aussi d'intégrer des systèmes préexistants de représentation et d'inférence sur le contexte tels que présentés dans [87] (C8). Les informations enregistrées peuvent être utilisées pour assurer la correspondance entre l'environnement physique et l'environnement numérique. En effet, les composants CPS peuvent utiliser les informations de la base de connaissances (ex : la position de l'objet, l'horodatage des données, l'utilisateur courant du MIBS) pour contextualiser les données captées.

Nous présentons par la suite les différents composants dans leurs catégories respectives.

Vue générale du framework

Le noyau de notre framework suit l'état des CPS et supervise les applications formées à partir de composants d'interprétation, de dialogue et de présentation. Les composants CPS et les composants applicatifs peuvent enrichir une base commune de connaissances, ce qui permet la perception du contexte (C7). L'interprétation des données est réalisée par les composants d'interprétation faisant partie des chaînes d'interaction des applications (C8).

3.2 Noyau du framework

Les systèmes ubiquitaires tels que les MIBS doivent être capables d'assurer l'interopérabilité spontanée (C9) entre les objets connectés. Parmi les frameworks existants, le processus de gestion des MIBS dans DAME [131] fournit le plus grand degré de liberté sur l'adaptation du MIBS au contexte. En effet, le framework surveille l'état des CPS, des applications, des utilisateurs et des informations de contexte. En fonction de ces informations, il contrôle la formation, l'adaptation et l'évolution des applications interactives, ce qui comprend la sélection dynamique des CPS et des chaînes d'interaction.

C'est pour cela que de la même façon le noyau de notre framework fonctionne grâce à la collaboration de ses 4 modules :

- le gestionnaire des CPS, qui découvre et enregistre les CPS disponibles et les modalités qu'ils fournissent ;
- le gestionnaire d'interprétation, qui gère l'instanciation des composants d'interprétation et il s'assure du bon fonctionnement de ces composants ;
- le gestionnaire de présentation, qui gère l'instanciation des composants de présentation et il s'assure du bon fonctionnement de ces composants ;
- le gestionnaire des applications, qui gère l'instanciation des applications en fonction des modalités et des composants d'interprétation et présentation à disposition, et qui ensuite donne aux gestionnaires d'interprétation et de présentation la chaîne de composants à instancier.

Ces composants sont présentés plus en détail dans les sections suivantes.

Nous présentons aussi le protocole de communication et la convention de nommage utilisés pour assurer les échanges entre tous les composants.

3.2.1 Gestionnaire des CPS

Ce composant communique avec les composants CPS pour surveiller la disponibilité, le type et l'état des CPS dans l'environnement. Ces informations sont ensuite utilisées par le gestionnaire des applications pour coupler les CPS aux applications. Nous nous sommes inspirés de la stratégie utilisée dans [91] pour déclencher l'association entre les objets et les applications. Ainsi, le gestionnaire communique avec les CPS à 3 occasions :

- au lancement et à l'arrêt d'une ou plusieurs modalités fournies par un composant CPS. Cela permet de signifier au MIBS l'évolution de la disponibilité du CPS associé ;
- au lancement du noyau du framework ;
- quand une application rencontre un problème concernant la technique d'interaction utilisée. En effet, il peut arriver qu'un composant CPS rencontre une erreur critique (ex. l'objet associé n'est plus connecté au réseau) qui l'empêche de signifier son arrêt au gestionnaire des CPS. Dans ces cas, les composants d'une chaîne d'interaction utilisant ce composant CPS peuvent demander une vérification de l'état des composants de la chaîne (dont le composant CPS). Les mécanismes de gestion de la qualité de service du protocole DDS peuvent être utilisé pour permettre aux composants d'interprétation et de présentation pour détecter ce type de problème.

Pour illustrer le procédé, nous présentons un diagramme de séquence en figure 3.2. Il représente les échanges entre un composant CPS fournissant des données en entrée, un composant d'interprétation utilisant ces données, le gestionnaire des CPS et le gestionnaire des applications à la suite d'un arrêt brusque du composant CPS. Le composant d'interprétation ne reçoit plus de données de la part du composant CPS, et envoie donc une alerte au gestionnaire des applications. Ce dernier contacte les autres composants du noyau du framework pour vérifier l'état des composants de la chaîne d'interaction de l'application en cours d'exécution. En particulier, le gestionnaire des applications demande au gestionnaire des CPS l'état du composant CPS. Le gestionnaire des CPS essaye de joindre le composant CPS, mais n'y arrive pas. Il suppose donc que le CPS n'est plus accessible et en informe le gestionnaire des applications pour que ce dernier adapte l'application pour ne plus utiliser ce CPS. Le gestionnaire des applications procède ensuite à la formation d'une nouvelle chaîne d'interaction qui n'utilise pas le CPS indisponible.

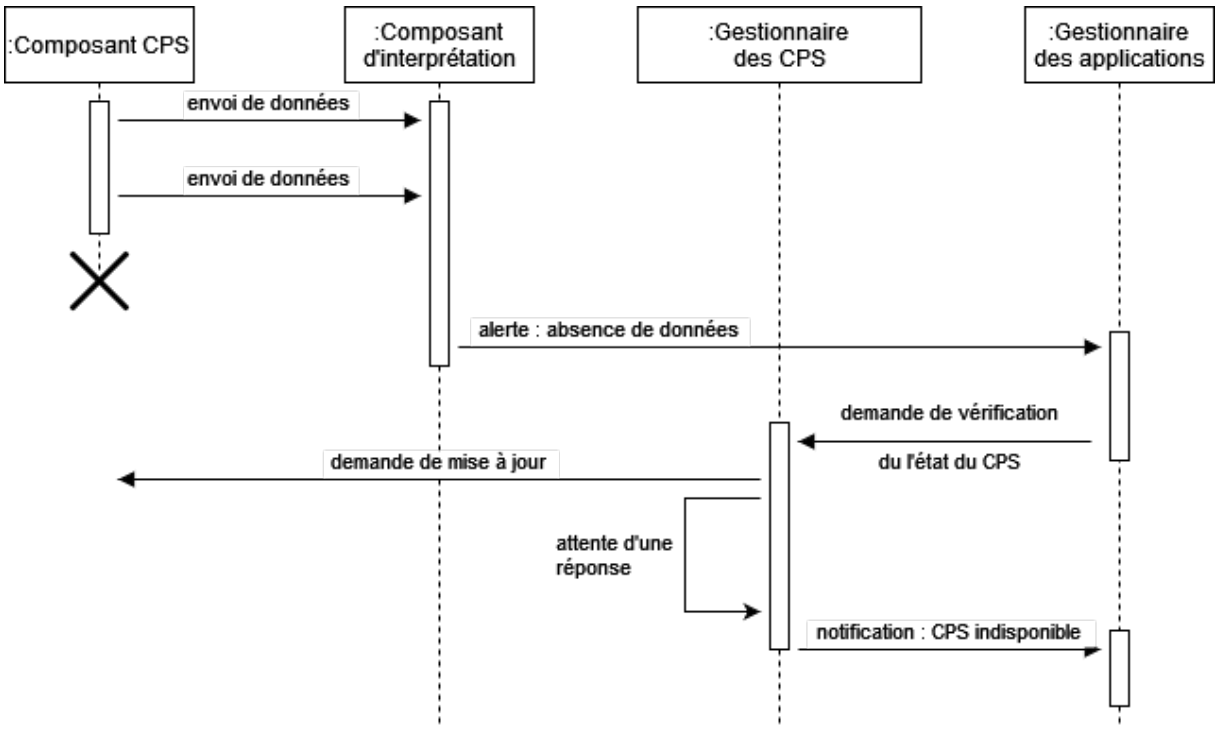


FIGURE 3.2 – Diagramme de séquence représentant le mécanisme pour mettre à jour la base de connaissances concernant les objets disponibles.

3.2.2 Gestionnaire des applications

Un utilisateur de MIBS peut souhaiter utiliser au cours d'une session plusieurs applications. Par exemple, Dans les scénarios présentés en introduction, le salarié peut recevoir la notification de l'arrivée d'un visiteur sur sa montre connectée pendant qu'il réserve une salle de réunion. Dans ce cas, les deux applications sont utilisées en même temps.

Cela veut donc dire que ces applications doivent être gérées dynamiquement pour assurer une bonne cohabitation. C'est le gestionnaire des applications qui a le rôle central d'assurer cette cohésion. Pour cela, nous nous sommes inspiré des algorithmes d'adaptation des interfaces utilisateur proposées par Lindt [96], et du processus d'adaptation de systèmes ubiquitaires proposé dans DAME [131]. En effet, le gestionnaire des applications gère les composants de dialogue en suivant un processus en 4 étapes :

— détection de l'évolution des besoins des composants de dialogue en cours d'exécution, ou de changement dans les modalités disponibles ;
— génération d'alternatives de chaînes d'interaction pour toutes les applications (i.e. composants de dialogue) installée ;
— sélection des chaînes d'interaction pour toutes les applications ;
— envoi des nouvelles chaînes d'interaction en entrée et en sortie respectivement au gestionnaire d'interprétation et au gestionnaire de présentation.

Un graphe orienté avec pour noeuds les composants (CPS, interprétation, présentation) et pour arêtes les interfaces compatibles est parcouru pour générer les alternatives de chaînes d'interaction. Comme pour SiAM-dp [114], le choix des chaînes d'interaction parmi ces alternatives peut être fait automatiquement selon un ensemble de critères, ou à partir de configurations définies au préalable. Si ce processus n'a pas pu aboutir pour une application, le composant de dialogue correspondant reçoit un message d'erreur spécifiant l'origine du problème. Cette gestion des erreurs permet de concevoir des composants de dialogue qui adaptent le fonctionnement de l'application en cas d'imprévu, voire qui proposent une liste de besoins alternatifs.

Deux stratégies de sélection sont intégrées dans le gestionnaire des applications car elles répondent à des besoins différents.

La méthode automatique supporte en théorie l'adaptation spontanée du système à l'environnement (C9), et une grande liberté dans le choix des modalités (C13). Des informations sur les applications, l'équipement, l'environnement et les profils utilisateur (ex. préférences, handicaps) peuvent être utilisées pour évaluer chacune des alternatives. Toutefois, les critères sont définis à partir d'une modélisation simplifiée du contexte qui

ne peut pas prendre en compte les spécificités de chaque environnement d'exécution. Cet écart entre le "monde observable" et la réalité peut pousser le gestionnaire des applications à choisir une alternative sous-optimale. Il serait pertinent dans l'application de guidage de sélectionner en priorité les modalités les plus proches spatialement de l'utilisateur dans les espaces fréquentés, mais cela ne garantit pas en pratique que l'utilisateur peut correctement interagir avec les éléments d'interface (ex. affichage des directions sur un écran derrière l'utilisateur).

Au contraire, la méthode par configuration permet d'assurer que les interfaces des applications correspondent aux besoins des utilisateurs, bien qu'une intervention humaine soit requise avant l'exécution pour définir les chaînes d'interaction à utiliser pour chaque ensemble de CPS possible et pour chaque application.

Dans notre framework, la méthode automatique est utilisée dans les situations qui n'ont pas été prévues par les configurations existantes, ce qui assure l'adaptation du système en toute situation en favorisant les attentes spécifiques à chaque environnement. Nous proposons par défaut un algorithme simple qui consiste à parcourir une liste de toutes les alternatives (de toutes les applications) jusqu'à trouver la première alternative (pour chaque application) qui convient.

Le diagramme de séquence en figure 3.3 illustre le fonctionnement du gestionnaire des applications quand un composant de dialogue lui transmet une nouvelle liste de besoin en actes de dialogue. Par exemple, les besoins initiaux du composant de dialogue dans le scénario de réservation peuvent être exprimés par le code en figure 3.4.

Nous utilisons le vocabulaire proposé par Paul et al. [122] pour annoter les besoins. Ici, l'application ne demande en entrée qu'une seule action obligatoire de l'utilisateur : une salutation. Aucune modalité d'entrée n'est préférée, et il suffit juste que l'identité de l'utilisateur soit détectée. Des exemples de besoins du composant de dialogue pour les autres phases de dialogue de l'application de réservation sont présentés en Annexe A.

Après avoir reçu la liste des besoins, le gestionnaire des applications récupère auprès du gestionnaire des CPS la liste des modalités mises à disposition par les CPS, et récupère la liste des composants d'interprétation et de présentation qui peuvent être instanciés. Des chaînes d'interaction sont ensuite générées, puis transmises aux gestionnaires d'interprétation et de présentation.

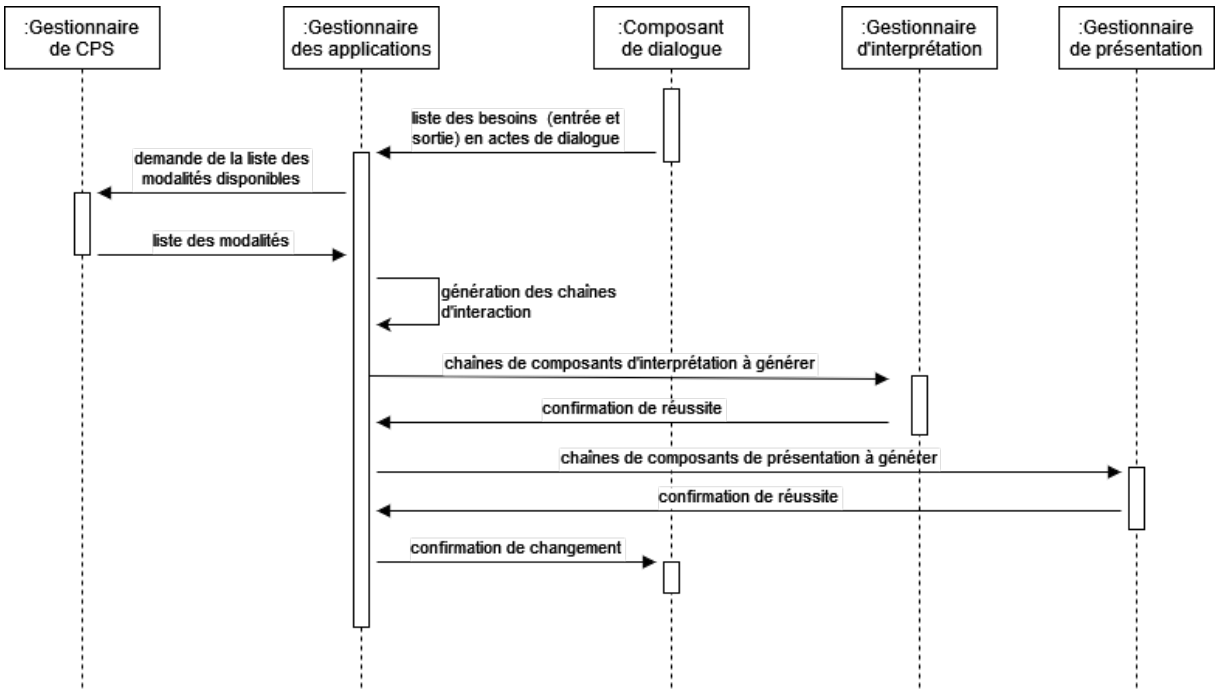


FIGURE 3.3 – Diagramme de séquence représentant le mécanisme pour mettre à jour la chaîne d'interaction d'une application suite à l'évolution des besoins d'un composant de dialogue.

```
{
"application_reservation" :
 {
  "user-hi":
   {
    "optionnel" : Faux,
    "source" : "entrée",
    "préférence" : {},
    "requis" : {"id_utilisateur"},
    "canal" : 0
   }
 }
}
```

FIGURE 3.4 – Besoins initiaux du composant de dialogue de l'application de réservation d'une salle de réunion. L'attente de l'engagement du système par l'utilisateur est représentée par l'attente de l'acte de dialogue "user-hi" en entrée.

3.2.3 Gestionnaires d'interprétation et de présentation

Les gestionnaires d'interprétation et de présentation ont des fonctionnalités similaires. En effet, leur rôles consistent à instancier et paramétrer les composants nécessaires à la formation de la chaîne d'interaction générée par le gestionnaire des applications. Ils gèrent aussi l'arrêt des composants qui ne sont pas dans le message envoyé par le gestionnaire des applications, et qui ne sont donc plus nécessaires. En particulier le gestionnaire d'interprétation gère les composants qui permettent de passer des modalités d'entrée aux composants de dialogue, tandis que le gestionnaire de présentation gère les composants qui permettent de passer du message généré par un composant de dialogue à une interface utilisateur au travers des modalités de sortie. Par exemple, pour répondre aux besoins initiaux de l'application de réservation (voir Figure 3.4), deux composants d'interprétation (i.e. détection de visage et reconnaissance de visage) peuvent être instanciés par le gestionnaire d'interprétation pour former la chaîne d'interaction permettant d'identifier un utilisateur à partir du retour vidéo d'une caméra. Cette chaîne est illustrée dans la figure 3.5. Des exemples de chaînes d'interaction pour les autres phases de l'application de réservation sont présentés en Annexe A.

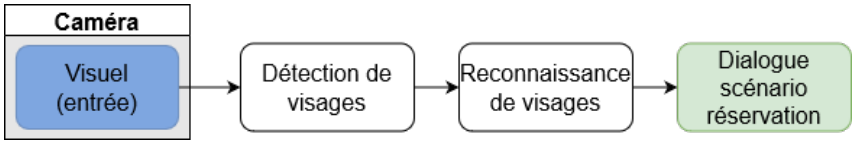


FIGURE 3.5 – Chaîne d'interaction pour la phase initiale dans le scénario d'usage de l'application de réservation.

3.2.4 Réseau

Pour faciliter la distribution des composants (C10) sur un réseau (C4), les composants communiquent entre eux selon le standard industriel de communication Data Distribution Service (DDS) de l'organisme Object Management Group [1] qui utilise un modèle publish-subscribe (C9, voir section 2.1.2) permettant de gérer la qualité de service (par exemple, choisir la latence acceptable entre l'émission et la réception d'un message).

De plus, les noms des canaux de communication entre les composants applicatifs sont construits à partir de l'information qu'ils permettent de transmettre, ainsi que de l'entité de rattachement. En effet, les canaux de communication des composants CPS contiennent

1. `https://www.omg.org/omg-dds-portal/index.htm`

le nom du CPS, le type d'information utilisé (modalité) et le sens de circulation de l'information (entrée ou sortie). De façon similaire, les composants d'interprétation et de présentation utilisent des canaux dont le nom contient le type d'information transmise, le nom de l'application de rattachement et le sens de circulation de l'information. Cette convention de nommage permet d'assurer une meilleure organisation des composants en cours d'exécution (C5).

Noyau du framework

Le noyau du framework supervise le fonctionnement des MIBS. Le suivi de l'état des composants CPS permet d'assurer l'interopérabilité spontanée (C9). Une méthode par configuration de chaînes d'interaction permet d'adapter les applications à chaque environnement d'exécution. Cela demande toutefois une intervention humaine dans ces environnements. Pour couvrir les situations qui n'ont pas été prévues par les configurations, une méthode automatique de génération de chaînes d'interaction prend le relais. Ainsi, les chaînes d'interaction peuvent être adaptées dynamiquement en fonction des modalités disponibles (C13). Le standard de communication utilisé permet de distribuer les composants sur un réseau (C4), et ainsi assurer la mobilité logicielle (C10). Une convention de nommage est utilisée pour assurer que chaque composant est adressable de manière unique (C5).

3.3 Composants applicatifs

Le noyau fonctionnel des applications multimodales tel que décrit dans le chapitre 2.1.1 se retrouve dans les composants applicatifs, à l'exception du gestionnaire du contexte. Ce gestionnaire est séparé des autres composants applicatifs pour faciliter la centralisation des informations contextuelles. Ainsi, les composants CPS, applicatifs et le noyau du framework peuvent tous communiquer avec le gestionnaire du contexte pour accéder à une base commune de connaissances.

Nous avons vu dans le chapitre 2.1.3 que l'encapsulation de mécanismes d'interprétation et de présentation au sein de composants permet une grande modularité et donc

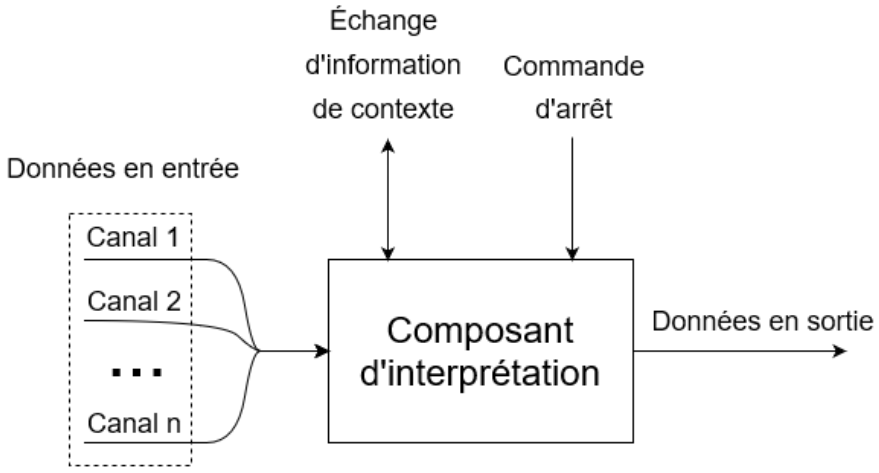


FIGURE 3.6 – Illustration des entrées et sorties d'un composant d'interprétation.

une interchangeabilité des nombreuses technologies utilisées dans les MIBS (C13). Cela permet aussi d'adapter ces processus à la multimodalité massive (C14), et en particulier aux différentes capacités de calcul et API des objets connectés. Par exemple, cela permet d'instancier les composants qui nécessitent de réaliser des calculs lourds sur des cartes graphiques (ex. détection de gestes dans les flux vidéo provenant de plusieurs caméras) sur plusieurs serveurs de calculs. Nous utilisons donc pour tous les composants applicatifs des interfaces utilisant le même formalisme, et qui peuvent communiquer sur le réseau.

3.3.1 Composants d'interprétation et de présentation

Comme représenté en figure 3.6, les composants d'interprétation s'abonnent à plusieurs canaux de communication (*subscriber* dans le modèle Publish/Subscribe), et publient le résultat de leurs interprétations dans un seul canal. Ces composants peuvent communiquer avec le gestionnaire du contexte pour contextualiser les événements reçus.

Les composants de présentation fournissent des interfaces similaires. Toutefois, ils ne proposent qu'un seul canal en entrée, mais ils peuvent publier dans plusieurs canaux.

3.3.2 Composants de dialogue

Il existe plusieurs modèles d'architecture pour réaliser un moteur de dialogue [43, 64]. Par exemple, le dialogue peut être représenté comme une suite d'états et transitions dans un modèle par automate fini, ou comme un ensemble de tâches orchestrées à partir de

l'objectif de l'utilisateur que le système à identifié dans un modèle par plans.

Comme dans la plateforme OpenInterface [92], nous ne spécifions pas de modèle explicite de composants de dialogue pour accorder plus de flexibilité aux développeurs des composants de dialogue. En pratique, ces composants fournissent les mêmes interfaces que les composants d'interprétation et de présentation, à savoir un canal de communication en entrée et en sortie, ainsi qu'un accès à la base de connaissances. Ils intègrent en plus une interface avec le gestionnaire des applications pour fournir une liste de besoins en entrée et sortie, et recevoir la confirmation que les chaînes d'interaction ont été adaptées en fonction des besoins. Ces besoins incluent les informations sémantiques qu'un composant de dialogue requiert en entrée et fournit en sortie. Ces informations sont annotées selon la taxonomie d'actes de dialogue proposée par Paul et al. [122]. Par exemple, l'intention de l'application de guidage à indiquer à l'utilisateur le chemin à suivre peut être représentée par l'acte de dialogue "inform" avec pour information sémantique la liste des salles et couloirs à traverser. C'est l'utilisation d'actes de dialogue pour représenter l'intention d'interaction de l'utilisateur ou de l'application qui permet d'assurer l'abstraction des modalités (C15). Nous proposons toutefois des exemples d'implémentation de composants de dialogue, et des librairies tierces [2]. Des applications externes peuvent aussi facilement s'interfacer avec ce modèle simple de composant.

Composants applicatifs

Le noyau du framework instancie des composants applicatifs et les assemble en chaîne d'interaction, formant ainsi les applications. La séparation des processus d'interprétation et de présentation en plusieurs composants assure une plus grande modularité, et permet de plus facilement gérer l'hétérogénéité des CPS (C14). De plus, l'assemblage dynamique de ces composants permet de facilement utiliser les CPS disponibles comme interfaces d'interaction (C13).

2. Par exemple python-statemachine [3] pour un modèle par machine à état, ou SPADE [4] pour un modèle par agents.

3.4 Composant CPS

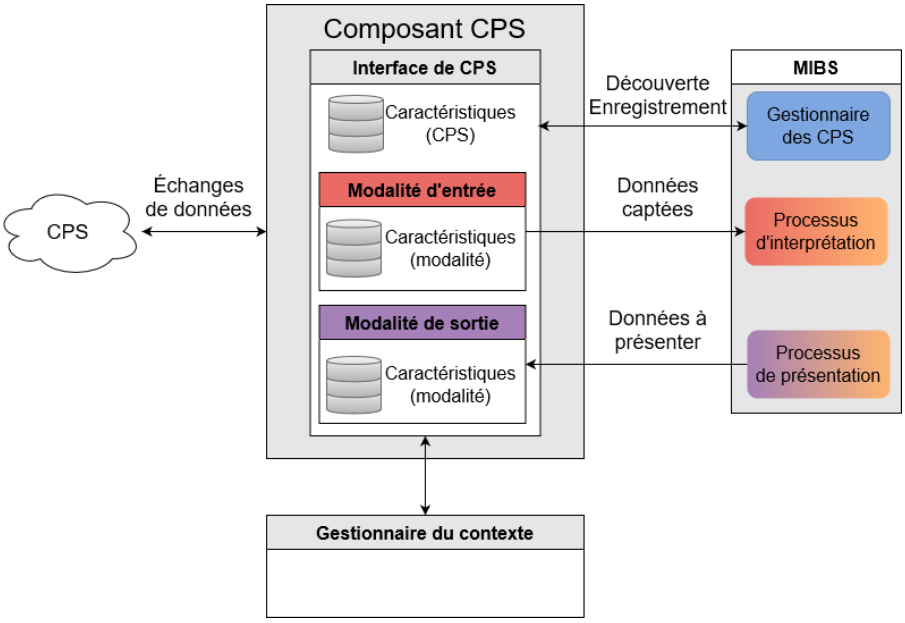


FIGURE 3.7 – Illustration des entrées et sorties d'un composant CPS. Ici, le composant fournit une modalité d'entrée et une modalité de sortie qui s'interfacent avec un MIBS.

Les CPS peuvent être utilisés par plusieurs systèmes et leur installation et retrait ne dépend pas toujours d'un système en particulier. C'est pour cela que les composants CPS sont gérés séparément des applications, à l'instar des composants de modalité dans AM4I [4]. En effet, chaque composant CPS est exécuté et arrêté de manière autonome, et peut être utilisé par plusieurs applications. Ainsi, les applications ont potentiellement accès à tous l'équipement disponible sur le réseau, ce qui contribue à l'extensibilité de notre approche (C3).

Chaque CPS, tout comme les modalités qu'il peut fournir, a des caractéristiques qui lui sont spécifiques. Par exemple, les caméras ont des angles de vue, et les microphones ont des niveaux de sensibilité. De plus, le protocole d'association peut dépendre de la technologie utilisée (ex. bluetooth, WiFi). C'est pour cela que nous proposons une classe passerelle, appelée *interface de CPS*, qui sert d'interface entre les MIBS et les implémentations spécifiques aux CPS (C6). Comme illustré Figure 3.7, un composant CPS intègre une *interface de CPS*. Cette interface contient des informations sur les caractéristiques du CPS et son état. De plus, chaque modalité est représentée par une classe qui intègre les fonctionnalités permettant de communiquer avec les MIBS. Les modalités gèrent aussi les

informations qui leurs sont spécifiques. En particulier, les modalités sont représentées selon le formalisme de la taxonomie de Augstein et Neumayr [8], ce qui permet l'abstraction des dispositifs physiques (C2).

Au cours d'une session d'interaction, les composants CPS alimentent la base de connaissances d'informations contextuelles (ex. position de l'objet), et envoient aux applications les données captées (i.e. modalités d'entrée) dont elles ont besoin. Ces données sont ensuite interprétées pour obtenir les informations sémantiques demandées par les composants de dialogue des applications. Pour communiquer avec l'utilisateur, les composants de dialogue génèrent des messages abstraits qui sont réifiés par une chaîne de composants de présentation, puis transmis aux utilisateur au travers des modalités de sortie fournies par les composants CPS.

Les composants CPS ont aussi accès aux informations de contexte, et transmettent au gestionnaire du contexte les informations décrivant le CPS lorsque celui-ci change d'état. Par exemple, les composants CPS informent le gestionnaire du contexte de la disponibilité du CPS. Les messages échangés entre les applications et les CPS peuvent contenir en plus certaines des méta-données concernant l'objet et sa situation dans l'environnement, comme proposé dans SIAM-DP [114]. Bien que nous proposions un ensemble de messages types que peuvent échanger des composants CPS, nous convenons que la taxonomie utilisée peut être restrictive lors de l'implémentation, surtout pour des objets proposant des interfaces particulières (ex. les *Brain Computer Interface*, ou BCI). C'est pour cela qu'il est possible d'ajouter de nouveaux formats de messages. Comme dans AM4I [4], nous utilisons le formalisme EMMA du W3C pour éviter la création de formats spécifiques à un usage précis et favoriser l'inter-opérabilité des modalités.

La prise en compte des spécificités de chaque CPS permet de supporter les interactions multi-sensori-motrices (C1).

Composants CPS

Les composants CPS fournissent des API (C6) permettant à toutes les applications sur le réseau (C3) d'accéder aux modalités fournies par les dispositifs physiques (C2). Chaque CPS est représenté par son état, ses caractéristiques et les caractéristiques des modalités disponibles. Cette représentation uniformisée facilite leurs usages dans des interactions multi-sensori-motrices (C1).

3.5 Utilisation du framework pour réaliser les applications de réservation et de guidage

Nous avons intégré plusieurs standards architecturaux pour répondre aux exigences présentées dans le chapitre 1. Dans cette section, nous illustrons avec les cas d'usage présentés dans l'introduction que notre framework permet bien de réaliser des interactions multimodales dans des environnements ubiquitaires.

3.5.1 Cas d'usage de l'application de réservation

Description du dialogue

Pour cette application, le dialogue peut être modélisé en 5 phases successives :

— le système attend qu'un utilisateur engage le dialogue ;
— le système salue l'utilisateur et détecte l'intention qu'a l'utilisateur de réserver une salle de réunion ;
— le système informe l'utilisateur que des créneaux sont disponibles, puis fournit la liste de créneaux disponibles, une méthode de parcours de cette liste et une méthode de sélection d'un des créneaux ;
— le système informe l'utilisateur de continuer la réservation depuis l'appareil personnel de l'utilisateur ;
— le système attend que l'utilisateur arrête le dialogue.

Pour réaliser ces tâches, l'entreprise qui déploie l'application de réservation dans la salle de réunion a à sa disposition plusieurs dispositifs physiques : un écran d'affichage, un microphone, une caméra, des enceintes et son smartphone. Chacun de ces objets peut être utilisé indépendamment. Ils ont donc tous leur propre composant CPS. Les modalités que ces objets peuvent fournir sont synthétisées dans le tableau 3.1. On notera que le smartphone n'est utilisé ici que pour accéder à une application tierce (i.e. boîte mail). Ainsi, il est entièrement géré par cette application tierce et il ne sera pas utilisé comme fournisseur de modalités.

OBJETS	Modalités en entrée		Modalités en sortie	
	audio	visuelle	audio	visuelle
Écran				✓
Microphone	✓			
Caméra		✓		
Enceintes			✓	

TABLE 3.1 – Modalités par objet

Description des chaînes d'interaction

Les dispositifs physiques et les techniques d'interaction sont différents à chaque phase du dialogue dans le scénario. Les chaînes d'interaction instanciées pour chaque phase seront donc différentes.

Prenons l'exemple de la phase de sélection du créneau horaire (phase 3). Comme illustré Figure 3.8, le composant de dialogue a besoin de 4 actes de dialogue pour réaliser cette phase :

— l'application doit informer (`"inform"`) l'utilisateur que des créneaux sont disponibles ;
— l'application doit fournir (`"offer"`) la liste des créneaux disponibles ;
— l'application doit permettre à l'utilisateur de parcourir (`"reqalts"`) la liste ;
— l'application doit permettre à l'utilisateur de sélectionner (`"user-confirm"`) un élément de la liste.

Ici, la liste est parcourue en passant d'un élément à l'autre (`"cible" : ["précédent", "suivant"]`), et une cible est requise pour l'acte de dialogue de sélection.

```
{
"application_reservation" :
 {
  "reqalts":
   {
    "optionnel" : Faux,
    "source" : "entrée",
    "préférence" : {},
    "requis" : {"cible" : ["précédent", "suivant"]},
    "canal" : 0
   },
  "user-confirm":
   {
    "optionnel" : Faux,
    "source" : "entrée",
    "préférence" : {},
    "requis" : {"cible"},
    "canal" : 1
   },
  "inform":
   {
    "optionnel" : Faux,
    "source" : "sortie",
    "préférence" : {"modalité" : "audio"},
    "requis" : {"type" : "créneaux", "disponible" : Vrai},
    "canal" : 0
   },
  "offer":
   {
    "optionnel" : Faux,
    "source" : "sortie",
    "préférence" : {"modalité" : "visuelle"},
    "requis" : {"type" : "créneau"},
    "canal" : 1
   }
 }
}
```

FIGURE 3.8 – Besoins du composant de dialogue pour la phase 3, exprimés au format Json.

Pour répondre à ces 4 besoins de l'application de réservation dans le scénario d'usage, le noyau du framework instancie la chaîne d'interaction présentée Figure 3.9.

L'information (`"inform"`) sur les créneaux disponibles est émise par les enceintes après un processus de restitution par synthèse vocale. Pour cela, le message est transformé en une chaîne de caractères ("Voici les créneaux disponibles"), puis en fichier audio à joué sur les enceintes.

La liste de créneaux est affichée sur l'écran de télévision sous un format Web. Pour faciliter le parcours de cette liste, une variable $p_{creneau}$ définit quel créneau à mettre en avant. Un premier composant de présentation transforme la liste en tableau en colorant le créneau pointé par $p_{creneau}$. Ce tableau est ensuite transmis à un autre composant qui l'intègre dans un fichier HTML compatible à la modalité visuelle du composant CPS de l'écran.

Pour parcourir cette liste, les mouvement de balayage de la main de l'utilisateur sont captés par la caméra. un premier composant d'interprétation détecte si l'utilisateur réalise un mouvement de balayage vers la gauche ou vers la droite. Ces mouvements, représentés par les valeurs 0 et 1, sont ensuite analysés (ex. à partir de la fréquence d'arrivée de nouvelles valeurs) pour inférer si l'utilisateur demande une alternative (`reqalts`), et si cela correspond à la précédente (valeur 0) ou à la suivante (valeur 1). Le composant de dialogue renvoie ensuite la liste de créneaux à l'écran d'affichage en indiquant avec $p_{creneau}$ le créneau à mettre en avant.

La sélection d'un créneau se fait à partir d'une commande vocale et d'un geste de pointage. D'abord, le geste de pointage est détecter puis la cible du geste est identifiée. Ensuite, la phrase "Celui-ci" énoncé par l'utilisateur est transformé en chaîne de caractères, puis interprété comme une intention de sélectionner (`"user-confirm"`). Enfin, la cible du geste est associé à la commande de sélection par un composant de fusion.

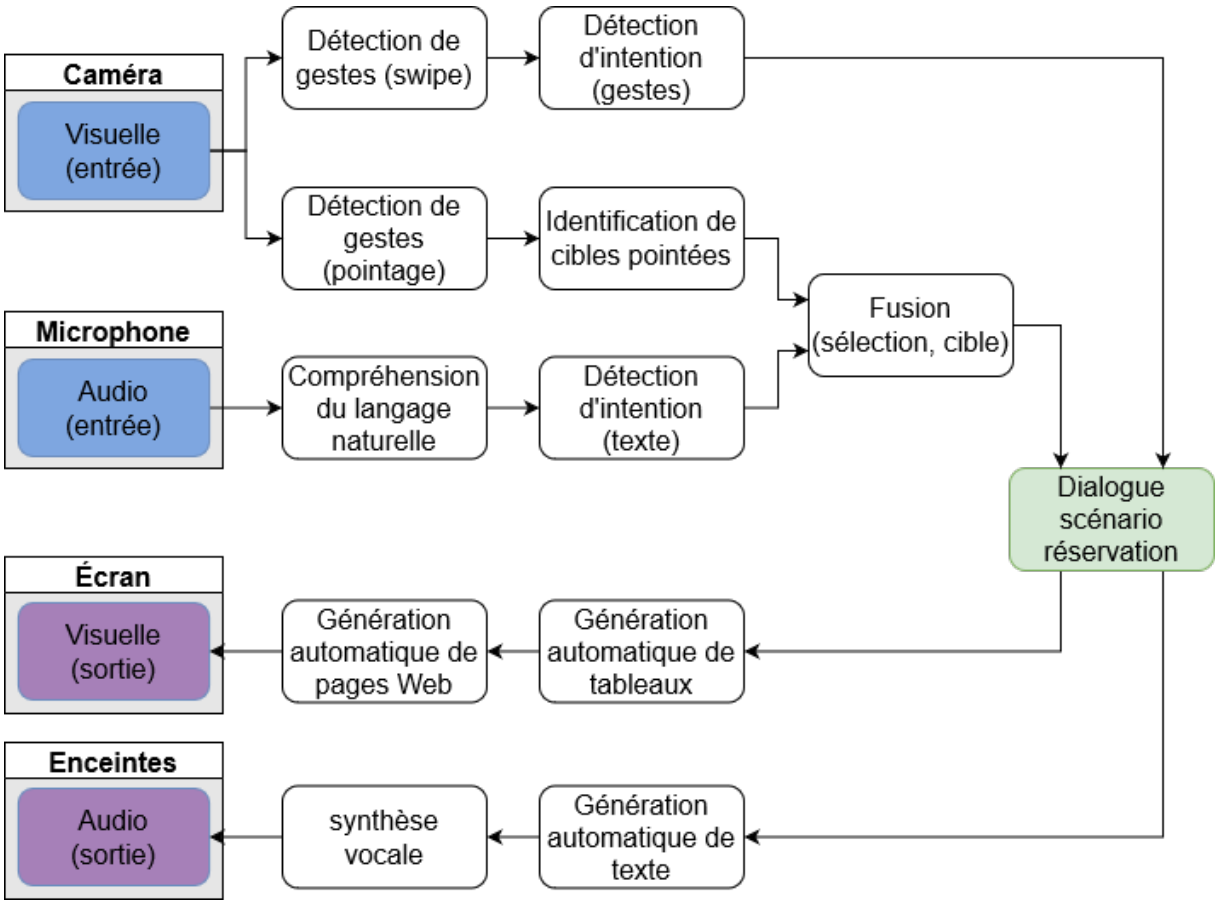


FIGURE 3.9 – Exemple de chaîne d'interaction pour la phase 3.

En résumé, les composants de cette chaîne d'interaction permettent de réaliser la phase 3 de l'application de réservation dans le scénario d'usage. Comme dans DAME [131], la forte granularité des chaînes d'interaction permet de facilement adapter les techniques d'interaction. De plus, la séparation des besoins du composant de dialogue en 4 permet de séparer la chaîne d'interaction en 4 branches indépendantes. Une illustration des actes de dialogue et des chaînes pour les autres phases du scénario d'usage se trouvent en annexe A.

Description des informations contextuelles

Pour pouvoir réaliser le scénario d'usage, la base de connaissances doit contenir plusieurs informations sur l'environnement que les dispositifs physiques ne peuvent pas capter. Tout d'abord, une base de profils utilisateur doit être accessible. Elle doit contenir une représentation des visages pour la phase 1, et les dispositifs personnels (ici, le smart-

phone) doivent être associés à leurs propriétaires. Ensuite, il faut connaître la disposition des objets dans la pièce. Cela permet d'identifier ce que pointe l'utilisateur dans la phase 3, et cela peut permettre de plus facilement détecter le départ de l'utilisateur dans la phase 5.

3.5.2 Cas d'usage de l'application de guidage

Description du dialogue

Sans le changement de destination, le dialogue de l'application de guidage peut être modélisé en deux phases :

— Le système accueille l'utilisateur ;
— Le système guide l'utilisateur à travers le bâtiment.

La phase d'installation de l'application sur le smartphone ne fait pas partie du dialogue en lui-même, mais correspond à une phase d'association d'un nouvel appareil au MIBS. On notera aussi que la destination dans la phase 2 est mise à jour pendant le trajet. Néanmoins, le composant de dialogue est implémenté selon une approche par plan où le plan est décomposé en instructions (ex. tourner à droite, monter au premier étage). Un changement de destination changera le plan, mais n'aura pas d'impact direct sur les besoins du composant de dialogue (i.e. les objets associés à l'application changent toujours en fonction de la position actuelle de l'utilisateur et de la destination).

Pour réaliser le cas d'usage, l'entreprise utilise le portique et l'enceinte connectée placés à l'entrée pour accueillir l'utilisateur, et les ampoules, les écrans et les enceintes installés dans les couloirs pour guider l'utilisateur. Ici, une plateforme logicielle gère les équipements statiques de l'environnement (i.e. les écrans muraux et les ampoules connectées). Nous représentons l'éclairage par un seul composant CPS où chaque modalité représente la capacité de changer les couleurs et la luminosité d'une des ampoules, et donc d'interagir visuellement. De même, un seul composant représente l'ensemble des écrans muraux. Les modalités que ces objets peuvent fournir sont synthétisées dans le tableau 3.2.

OBJETS	Modalités en entrée		Modalités en sortie	
	tactile	visuelle	tactile	visuelle
Écran				✓
Éclairage				✓
Portique			✓	
Montre connectée	✓	✓		✓

TABLE 3.2 – Modalités par objet

Description des chaînes d'interaction

Les besoins du composant de dialogue et la chaîne d'interaction dans la phase d'accueil sont similaires à ce que l'on a présenté dans la phase d'engagement de l'application de réservation. Toutefois, la phase 2 est plus complexe. En effet, les ampoules, les écrans et les enceintes (i.e. les actionneurs) sont utilisées pour guider l'utilisateur, et deux utilisateurs interagissent avec le système. La chaîne d'interaction utilisée durant cette phase est illustrée en Figure 3.10. Par rapport à ce que l'on peut trouver dans la littérature, le composant de *fission* propose un usage très spécifique des lumières dans un environnement connecté. En effet, le composant de *fission* contrôle progressivement les actionneurs pour guider l'utilisateur selon le trajet le plus court $path_{p_i p_f}$ entre la position de l'utilisateur p_i et une position finale p_f (voir Figure 3a). Ce trajet $path_{p_i p_f}$ est défini par une liste de position à parcourir, chacun étant associé à un intersection dans l'environnement physique. À partir de la position de l'utilisateur estimée à partir des données captées par les caméras, le composant de *fission* contrôle les actionneurs en 3 étapes :

— il définit le vecteur $p_i\vec{p}_j$ entre la position de l'utilisateur p_i et le prochain point du trajet p_j ;
— il récupère la position de tous les actionneurs s_k se trouvant dans les pièces traversées par $p_i\vec{p}_j$;
— il calcule la distance (d_{s_k,s'_k}) de chaque actionneur au vecteur, ainsi que la distance orthogonale (d_{p_i,s'_k}) à $p_i\vec{p}_j$ (i.e. Soit $p_i\vec{s}_k$ le vecteur entre p_i et un actionneur s_k, θ l'angle entre $p_i\vec{p}_j$ et $p_i\vec{s}_k$, alors d_{p_i,s'_k} est égale à $p_i\vec{s}_k * cos(\theta)$) ;
— l'actionneur éligible le plus proche de p_i est utilisé.

L'éligibilité d'un actionneur dépend de 2 critères : θ est dans l'intervalle $[-60°, 60°]$ (i.e. dans le champs de vue d'un personne en p_i qui regarde vers p_f [104]) et d_{p_i,s'_k} est inférieure à la norme de $p_i\vec{p}_f$ (i.e. entre la position initiale et finale).

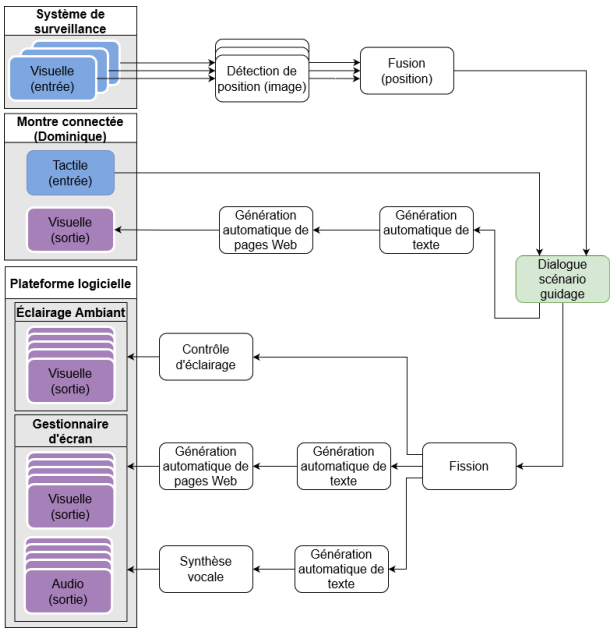


FIGURE 3.10 – Chaîne d'interaction pour l'application de guidage dans la phase 2.

Ces 3 étapes sont répétées à chaque changement de p_i et p_j jusqu'à qu'il n'y ait plus d'actionneurs éligibles (ex. la position initiale s'est suffisamment rapprochée de la position finale pour qu'il n'y ait plus besoin de montrer le chemin). Les actionneurs qui ont été utilisés sont remis dans leur état initial (ex. les ampoules sont éteintes) par la suite.

Si la destination est modifié depuis la montre connectée, le composant de dialogue recalcule le trajet à effectuer, puis informe le composant de fission de la nouvelle liste de position à parcourir.

De plus, dans l'éventualité où un objet connecté devient indisponible (ex. une ampoule grille), le composant CPS de l'éclairage ambiant informe le gestionnaire de contexte du changement d'état. Cette information est ensuite récupérée par le composant de *fission* quand il identifie les actionneurs se trouvant sur le trajet de l'utilisateur.

Description des informations contextuelles

Les composants de dialogue, de *présentation* et de *contrôle d'éclairage* ont besoin de plusieurs informations contextuelles :

— pour construire le parcours de guidage, le composant de dialogue a besoin d'une représentation de l'environnement (ex. un graphe reliant les différentes pièces, couloirs et escaliers). Ce composant requiert aussi la position finale, que cela soit un

élément de la représentation de l'environnement (ex. une salle de réunion) ou la position de la montre connectée de Dominique ;

— la position des caméras dans la représentation de l'environnement est aussi nécessaire pour repérer la position de Camille ;

— le composant de *présentation* nécessite la position des écrans muraux. Cela lui permet d'afficher les instructions du composant de dialogue sur l'écran le plus proche de Camille, et dans le cas où aucun écran n'est assez proche, utiliser l'éclairage pour diriger l'utilisateur ;

— pour réaliser les étapes présentées précédemment, le composant de *fission* a besoin de la position des actionneurs.

Utilisation du framework

L'illustration de notre framework au travers de deux cas d'usage permet de montrer qu'il répond bien aux exigences. La représentation des besoins des applications en actes de dialogue permet de concevoir les applications indépendamment des dispositifs physique et des techniques d'interaction. La méthode de génération dynamique de chaînes d'interaction permet de supporter les processus d'interprétation et de présentation pour une grande variété de dispositifs physique, tout en permettant de réutiliser des composants d'une application à une autre. L'adaptation automatique des chaînes d'interaction permet d'adapter les MIBS aux spécificités des environnements, bien qu'une intervention humaine soit nécessaire pour assurer que le système soit utilisable après adaptation.

3.6 Synthèse

L'objectif de ce chapitre était de définir un framework pour MIBS qui répond aux exigences présentées dans le chapitre 1. Pour cela, nous avons présenté dans la section précédente un framework qui intègre plusieurs standards architecturaux utilisés pour créer des MIBS. Les exigences ainsi que les réponses apportées sont synthétisées dans la table 3.3.

TABLE 3.3 – Cette table synthétise les réponses apportées (i.e. choix architecturaux et méthode de conception) aux exigences présentées dans le chapitre 1.

Exigences (identifiants)	Réponses
Support d'interactions multi-sensori-motrices (C1)	Support de plusieurs modalités en entrée et en sortie
Abstraction des dispositifs physiques (C2)	Représentation des dispositifs par leurs modalités [8]
Extensibilité (C3)	Architecture distribuée, fonctionnement autonome des composants CPS
Connexion à internet (C4)	Interfaces des composants accessibles depuis le réseau
Objets identifiables (C5)	Convention de nommage des composants et de leurs interfaces
Services à disposition (C6)	API des objets utilisables par plusieurs applications
Perception du contexte (C7)	Enrichissement de la base de connaissances par les CPS et les applications
Gestion du contexte (C8)	Utilisation d'un gestionnaire du contexte externe
Inter-opération spontanée (C9)	Modèle Publish-Subscribe (robustesse à l'erreur), composition dynamique de chaînes d'interaction (adaptation) en comportement par défaut
Mobilité (C10)	Architecture distribuée (mobilité des composants), modélisation de l'environnement
Choix libre de modalités et combinaison de modalités (C13)	Architecture par composants et méthodes de génération de chaînes d'interaction
Gestion de la multimodalité massive (C14)	Interprétation/présentation par composition
Abstraction des modalités (C15)	Utilisation d'actes de dialogue et d'informations sémantiques

En particulier, l'abstraction de dispositifs physiques par les composants CPS et l'abstraction des modalités par les composants de dialogue facilitent le découplage entre les objets connectés et les applications. De plus, la méthode de génération automatique de chaînes d'interaction permet de s'adapter dynamiquement aux changements dans l'environnement à partir de données captées par les composants CPS et les composants d'interprétation. Toutefois, il est difficile d'évaluer l'utilisabilité des interfaces générées automatiquement. Par exemple, la caméra dans le scénario de réservation peut être ébloui par

le soleil à certaines périodes de la journée. Or, cette situation est impossible à modéliser par des métriques génériques. De même, une interaction vocale peut ne pas être adaptée à certains environnements comme une cafétéria très bruyante ou à l'inverse un open space dans lequel le silence doit régner. Modéliser l'ensemble de ces paramètres environnementaux est fastidieux, voire impossible dans certains cas. Cependant, ils sont nécessaires à tout algorithme d'adaptation des modalités.

Le framework supporte l'utilisation de configuration statique des applications en fonction des objets connectés disponibles. Dans ce cas, il permet à la personne en charge du déploiement (qui connaît donc l'environnement dans lequel le MIBS sera exécuté), de tenir compte des différents paramètres influençant l'utilisabilité, et de choisir ainsi les chaînes d'interaction les plus adéquates. Pour assurer l'utilisabilité du système, une intervention humaine est alors requise.

Dans la partie suivante, nous réalisons une étude de ces méthodes et outils pour identifier en quoi consiste cette intervention humaine, et ainsi proposer une méthodologie pour faciliter l'adaptation des MIBS aux environnements d'exécution.

Deuxième partie

Méthodes et outils pour faciliter le processus de création des MIBS

Nous avons vu dans la partie précédente que l'adaptation des MIBS aux environnements requiert une intervention humaine que les frameworks existants n'abordent pas. Dans cette partie, nous étudions le processus de création des MIBS pour identifier en quoi consisterait cette intervention humaine, et proposer ainsi une solution.

Dans le chapitre 4, nous proposons une analyse du cycle de vie de ces systèmes, de leurs conceptions par les éditeurs à leurs exécutions dans les environnements. Cette analyse nous donne une vue globale sur le processus de création de ces systèmes, et nous permet ainsi d'identifier le degré de maturité des méthodes pour accomplir toutes les étapes du processus. Dans un second temps, nous identifions les étapes insuffisamment outillées pour répondre à notre problématique de recherche.

À partir de cette analyse, nous présentons dans le chapitre 5 notre solution pour adapter facilement les MIBS aux spécificités des environnements dans lesquels ils sont déployés. En effet, nous proposons une méthodologie de configuration et d'évaluation des MIBS en utilisant la réalité virtuelle (MCEV : MIBS Configuration and Evaluation in Virtual reality). Cette méthode s'intègre au framework décrit dans le chapitre 3, et elle s'applique pendant les étapes d'intégration et de déploiement.

Nous présentons dans le chapitre 6 l'expérimentation réalisée pour évaluer cette méthode, puis nous étudions des scénarios présentant l'application de la méthode MCEV aux cas d'usage de réservation et de guidage.

Chapitre 4

Proposition d'un cadre conceptuel pour la création de MIBS

Notre objectif dans ce chapitre est d'étudier comment la création des MIBS peut être facilitée. En particulier, nous nous intéressons aux approches facilitant l'adaptation des MIBS à leurs environnements d'exécution. La complexité de création de MIBS peut être allégée par des méthodes et outils logiciels. Cependant, les processus de création que ces outils supportent, peuvent être différents. Les outils ne sont donc pas tous interopérables. De plus, ces outils ne prennent en charge que des tâches spécifiques dans leurs processus respectifs. Enfin, chaque outil demande un niveau d'expertise spécifique à un domaine d'activité. Par exemple, les outils pour développeurs peuvent être difficiles à utiliser par un designer.

Pour étudier les outils existants, nous présentons d'abord les tâches liées au cycle de vie des MIBS, ainsi que les rôles qui leurs sont associés. À partir de ces études préliminaires, nous proposons une revue de la littérature des outils existants qui correspondent à ce cycle de vie et à ces tâches. Nous discutons ensuite des limites du soutien apporté par les outils dans le processus de création. Pour finir, nous rappelons les principales conclusions de ce chapitre. Nous illustrons les différentes tâches et analysons les outils existants en nous projetant sur leur utilisation pour créer l'application de réservation de salles de réunion présentée précédemment.

Publication

Ce chapitre a fait l'objet d'une publication dans une conférence internationale : Fabrice Poirier, Anthony Foulonneau, Jérémy Lacoche et Thierry Duval, « Interactive Multimodal System Characterization in the Internet of Things Context », in : *HUCAPP 2022*, Conférence Virtuelle en ligne, France, fév. 2022, URL : `https://hal.archives-ouvertes.fr/hal-03451478` (visité le 22/03/2022)

4.1 Études des méthodologies existantes pour guider le processus de création des MIBS

Le processus de création des MIBS est complexe et requiert l'intervention de plusieurs acteurs. Il est donc nécessaire de construire ces systèmes selon un processus, ou cycle de vie, explicitement défini. En effet, cela permet de construire une réflexion sur un cadre commun, d'assurer une répartition consensuelle des tâches et d'uniformiser l'intégration des spécificités des MIBS. Définir un cadre de travail holistique facilite en outre la convergence des pratiques et l'adoption de standards qui peuvent être intégrés dans des méthodes et outils logiciels. Il s'agit donc d'une étape essentielle pour répondre à notre problématique de recherche. Nous commençons par présenter les méthodes utilisées pour créer des systèmes multimodaux et des systèmes ubiquitaires. Nous étudions ensuite les approches utilisées pour créer des MIBS.

4.1.1 Conception et développement de systèmes multimodaux

D'après [120] et [131], les systèmes multimodaux sont généralement créés selon un processus itératif en 3 étapes qui suit les principes de la conception centrée utilisateur : analyse des besoins, prototypage (i.e. conception et développement) et évaluation. Ainsi, chaque répétition du cycle permet de créer un prototype qui répond mieux aux attentes des utilisateurs. Le produit final peut donc être obtenu après plusieurs cycles.

De plus, la modularité de tels systèmes permet de séparer la conception des éléments du système, ce qui facilite la répartition du travail.

En particulier, Silva et al. [147] mettent l'accent sur l'importance de la séparation entre modalités et applications lors du développement. Chaque composant gérant une modalité peut donc être développé indépendamment de toute application. Ainsi, ces composants génériques peuvent être utilisés dans plusieurs applications, ce qui permet ainsi de réduire les coûts de développement. MIODMIT [55] est un exemple d'architecture basée sur l'utilisation de composants génériques. MIODMIT propose un processus d'adaptation en 3 étapes. Premièrement, il faut définir les objets, modalités et techniques d'interaction considérés pour l'application. Deuxièmement, les composants existants nécessaires pour l'application sont sélectionnés, tandis que les composants qui manquent sont modélisés. Enfin, troisièmement, ces composants manquants sont développés et intégrés dans un prototype à évaluer. L'avantage de ces approches est de séparer l'interaction des cœurs applicatifs, ce qui favorise la réutilisabilité des composants existants. Néanmoins, le rôle des parties prenantes n'est pas clairement défini, ce qui peut freiner leur coopération.

4.1.2 Conception et développement de systèmes ambiants

En informatique ambiante, la méthodologie U-C IEDP ("User-Centred Intelligent Environments Development Process") a été proposée pour pallier au manque de méthodologie [10]. Cette approche (voir figure 4.1) est constituée de plusieurs étapes réparties dans 3 cycles, et chaque cycle fait intervenir toutes les parties prenantes. La première boucle consiste à établir et évaluer les exigences à partir de retours utilisateur. Le système est conçu (en collaboration avec les différents acteurs) et développé dans la deuxième boucle. Enfin, les objets et le système sont installés dans l'environnement cible pour être évalués par les parties prenantes. Ainsi, cette méthodologie prend en compte le besoin de communication entre les différents acteurs dans la conception d'environnements intelligents. De plus, cette méthode de co-création avec les utilisateurs finaux permet de les impliquer, ce qui permet de vérifier que le système répond aux besoins, et d'assurer une bonne acceptation du système. Toutefois, la méthodologie U-C IEDP ne prend pas en compte la réutilisabilité des composants pour d'autres projets. De plus, cette méthode de création en plusieurs cycles centrés sur les parties prenantes requiert l'accès à l'environnement final et aux retours des utilisateurs finaux pour que le système soit correctement adapté à l'environnement d'exécution. Or dans notre cas, l'environnement et les utilisateurs finaux ne sont que rarement connus.

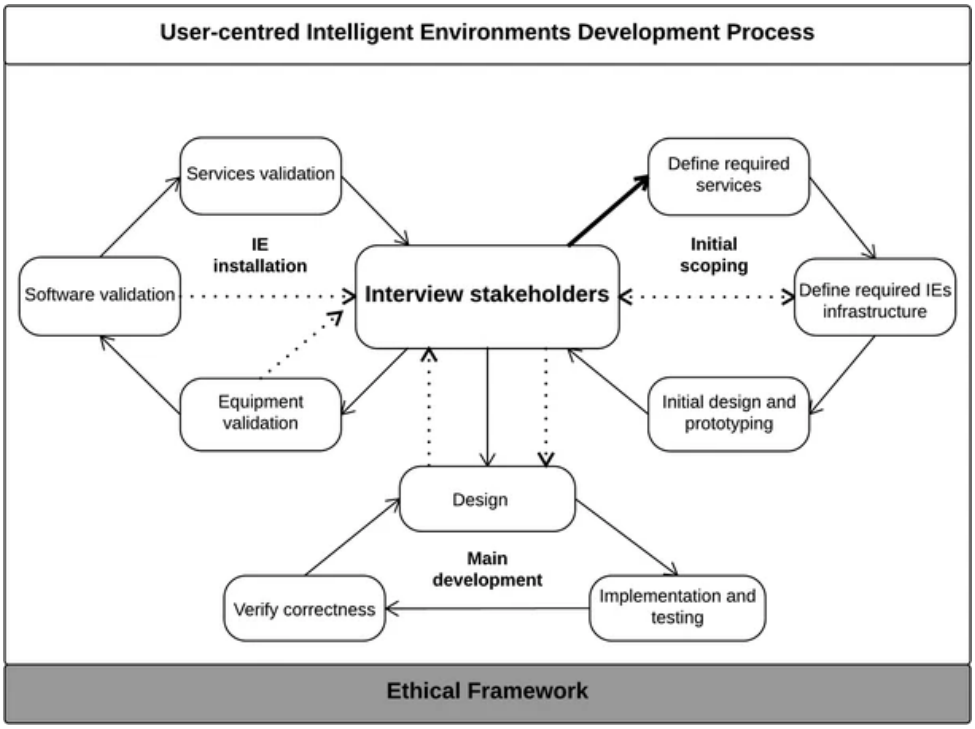


FIGURE 4.1 – Processus de développement d'environnements intelligent proposé par Augusto et al. [10].

4.1.3 Conception et développement des MIBS

Peu d'informations sont données sur les méthodologies utilisées pour créer des MIBS, et ces informations se limitent souvent à l'énonciation des tâches à réaliser. L'approche utilisée dans DAME (voir chapitre 2.2.2) est l'une des rares méthodologies à associer chaque tâche à un rôle, et à montrer comment le travail de chaque acteur agit sur le système global. En particulier, plusieurs rôles spécifiques aux environnements ambiants sont présentés : les designers, administrateurs et ergonomes d'ambiant. Les *designers d'ambiant* créent les MIBS en assemblant des composants créés par différentes équipes de designers et développeurs. Ensuite, les *administrateurs d'ambiant* configurent les MIBS à leurs environnements. Enfin, les *ergonomes d'ambiant* assurent l'utilisabilité du système. Il y a donc une séparation entre les acteurs de "l'ambiant" et les acteurs qui créent les composants individuellement.

Similairement, Barricelli et al. [16] proposent la méthodologie SSW (Software Shaping Workshop). Cette méthodologie suit le paradigme de l'atelier, où chaque acteur a ses propres outils et sa propre perception du projet. Comme illustré en figure 4.2, ces outils

sont créés à un niveau supérieur (niveau "meta design") et utilisés par les différents acteurs (niveau design). Ces développeurs et designers viennent ensuite créer les applications pour les utilisateurs finaux (niveau "use").

L'avantage principal de ces deux méthodologies est la facilitation de la collaboration entre les différents acteurs en cachant les considérations des autres domaines d'expertise. Toutefois, la méthodologie de création de MIBS et les rôles associés dans DAME sont trop spécifiques à leur approche pour fournir une méthode standard en lien avec les travaux sur la multimodalité. La méthodologie SSW ne présente quant à elle ni les composants à créer ni l'interaction concrète entre les acteurs.

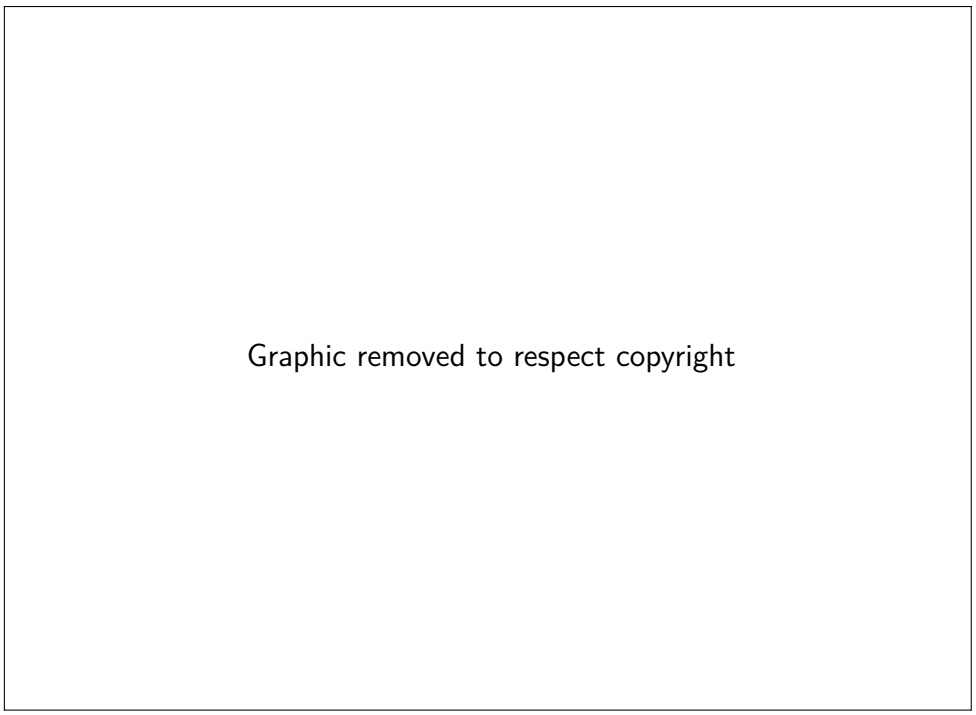


FIGURE 4.2 – Méthodologie SSW utilisée par Barricelli et al. [16] pour concevoir des systèmes multimodaux pervasifs.

4.1.4 Résumé

Les approches spécifiques à la multimodalité abordent cette difficulté en facilitant la réutilisation des composants logiciels d'une application à une autre. Pour autant, la place des différents acteurs intervenant dans la création de MIBS n'est pas clairement définie, ce qui peut compliquer leur collaboration.

La méthodologie U-C IEDP supporte la participation des parties prenantes dans la

création de systèmes ubiquitaires, mais l'environnement d'exécution est considéré comme connu dès les premières étapes.

Enfin, les méthodes utilisées pour les MIBS n'offrent pas un cycle de vie suffisamment concret ou générique.

Dans la section suivante, nous proposons un cycle de vie pour les MIBS en lien avec les standards architecturaux présentés dans le chapitre 2.

4.2 Proposition : Cycle de vie des MIBS

Les méthodologies présentées précédemment n'intègrent pas de solution pour facilement adapter et évaluer les MIBS en fonction des environnements d'exécution. Pourtant, les capacités d'adaptation d'un MIBS à l'exécution dépendent des choix réalisés tout au long du cycle de vie des MIBS, de la modélisation des applications à l'installation du MIBS dans l'environnement.

C'est pour cela que nous décrivons les tâches à réaliser dans le cycle de vie des MIBS que nous séparons en 5 étapes (synthétisé en figure 4.3) :

— l'étape du *design* pour décrire tous les modèles (dialogue, interfaces utilisateur, contexte) nécessaires au système à partir des spécifications faites ;
— l'étape de *développement* pour créer les composants logiciels nécessaires à la mise en œuvre des modèles conçus ;
— l'étape d'*intégration* pour assembler les composants en un système entièrement fonctionnel ;
— l'étape de *déploiement* pour installer le système dans l'environnement désiré ;
— l'étape d'*exécution* (ou "d'opération et maintenance") pour observer le système en cours d'exécution.

Contrairement aux approches précédentes, les étapes d'intégration et de déploiement sont considérées à part car elles regroupent des tâches qui dépassent le cadre du développement, et sont antérieures à l'exécution. En particulier, le passage de MIBS génériques à des systèmes adaptés à leurs environnements se fait à ces étapes.

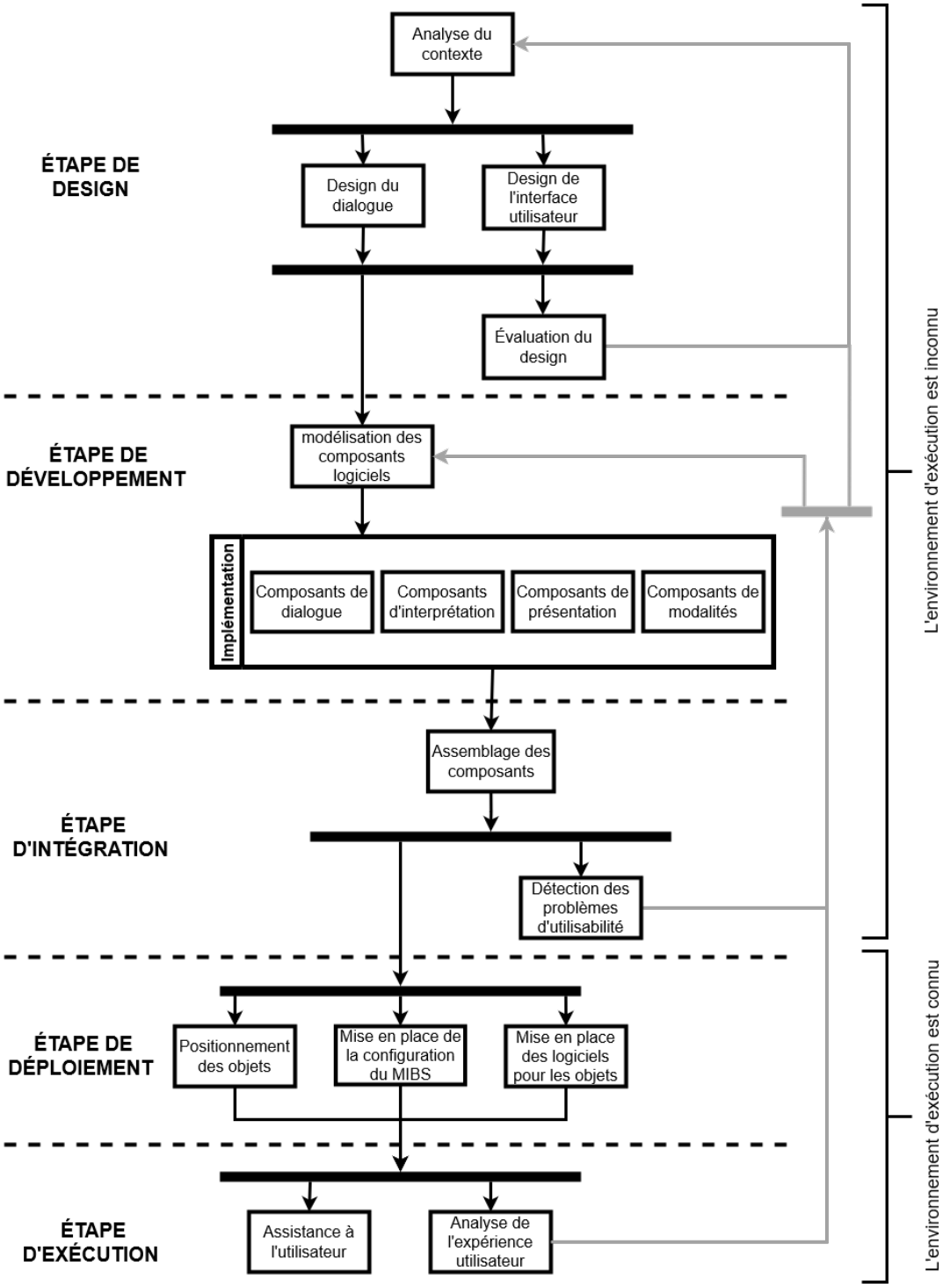


FIGURE 4.3 – Vue globale du cycle de vie des MIBS de la spécification des exigences à l'exécution. Les tâches d'évaluation des étapes de design, d'intégration et d'exécution permettent de reprendre le cycle à partir des étapes de design ou développement (flux en gris).

4.2.1 Étape de design

Selon [114] et [71], l'étape de design des MIBS englobe principalement l'analyse du contexte, le design du dialogue et le design de l'interface utilisateur. De plus, Wechsung [162] suggère que des évaluations formatives (i.e. évaluations pour apprendre et améliorer un système tout en le créant) peuvent aider à valider ces designs. Ce processus de design est illustrée en figure 4.3.

Analyse du contexte

Nous avons vu dans les chapitres 1.4 et 2.1.3 que le contexte est un aspect important des MIBS, et que le fonctionnement des applications dépend généralement de 4 types d'entités : **les dispositifs physiques**, **l'environnement**, **l'utilisateur** et **le système**. Par conséquent, la première étape consiste généralement pour les designers à analyser et à définir les informations contextuelles qui sont utilisées tout au long du cycle de vie des MIBS, comme l'explique Augusto [12].

En particulier, les informations environnementales sont essentielles pour réaliser des systèmes invisibles (C12 dans le tableau 1.1), ou pour assurer l'interaction dans des situations de mobilité (C10). Dans le processus de design de Lemmelä et al. pour les systèmes multimodaux avec des dispositifs mobiles [95], les designers définissent l'impact des environnements possibles sur les capacités perceptives et cognitives des utilisateurs. De plus, les designers et les architectes du bâtiment peuvent travailler ensemble pour modéliser les environnements physiques, comme le proposent Pittarello et al. dans leur ontologie [128]. Cela suppose toutefois que ces environnements sont déjà connus.

La représentation du contexte peut être décidée sur la base d'observations des comportements du public cible, de l'expérience des concepteurs ou de l'interprétation de résultats d'expérimentations de prototypes faisant intervenir des utilisateurs (voir sections 4.2.3 et 4.2.5). Les designers doivent aussi collaborer avec les autres acteurs pour définir les informations de contexte relatives à l'application désirée. Par exemple, Oviatt et Cohen [120] expliquent que les designers doivent identifier comment les utilisateur interagissent quand il ont à leur disposition un ensemble données de modalités, et quelles techniques d'interaction les utilisateurs souhaitent utiliser. Ces informations liées au domaine d'activité doivent ensuite être utilisées pour concevoir le modèle de tâches, qui est défini dans [35] comme la représentation des informations à collecter dans le dialogue.

Dans l'application de réservation de salles de réunion, l'utilisateur peut, entre autres,

sélectionner un créneau avec des commandes vocales ou avec des commandes gestuelles. Pour faciliter la sélection d'une alternative, les designers peuvent modéliser les microphones connectés comme des dispositifs pouvant fournir des interfaces vocales, et les smartphones comme des dispositifs pouvant fournir des interfaces graphiques et vocales. De plus, les gestes de pointage pour sélectionner des pièces nécessitent une représentation de l'environnement, ainsi que des positions et rotations des smartphones par rapport à celui-ci. On peut également prendre en compte la situation d'encombrement de la salle.

Design du dialogue

Une fois que toutes les informations contextuelles sont modélisées, des designers peuvent définir le dialogue. Ils ont accès aux exigences et aux informations contextuelles pour décrire les tâches d'interaction que les utilisateurs peuvent effectuer, ainsi que leur séquencement. Les méthodes utilisées pour définir les tâches d'interaction comprennent le cheminement de l'utilisateur à partir d'observations, le storyboarding [95] ou la participation d'un expert du domaine à la conception [16]. Dans la représentation de la tâche, les interfaces utilisateur doivent être abstraites car il est impossible de prévoir quels objets connectés seront utilisés dans le cadre des MIBS. Par exemple, dans le cadre de référence CAMELEON [38], les modèles de tâches d'interaction sont définis sans spécifier aucune information sur l'interface utilisateur.

Un designer UX peut aussi définir les alternatives de techniques d'interaction dont le dialogue pourrait avoir besoin. Cela permet d'adapter les interfaces aux contraintes imposées par le contexte, ce qui est particulièrement important pour concevoir des interactions implicites. En effet, la conception d'une interaction implicite requiert une gestion du degré d'attention nécessaire à l'interaction et de l'initiative (entre le système et l'utilisateur) selon [83]. Cela demande donc d'adapter les techniques d'engagement et de désengagement à l'état d'esprit de l'utilisateur.

Dans notre exemple, des designers UX peuvent représenter le dialogue de l'application de réservation de salle de réunion en quatre tâches d'interaction successives : engager le dialogue depuis l'espace réservé à l'interaction, sélectionner la salle de réunion et le créneau, partager l'invitation, et confirmer le résultat. Les deuxième et troisième tâches peuvent être réalisées dans n'importe quel ordre, mais les deux sont nécessaires pour passer à la dernière.

Design de l'interface utilisateur

Une fois le modèle de dialogue conçu, les interfaces utilisateurs (UI) correspondantes peuvent être définies. Le modèle transversal de Vanderdonckt [157] sur les interfaces distribuées peut être utilisé pour concevoir les interfaces des MIBS : "A UI distribution concerns the repartition of one or many **elements** from one or many **user interfaces** in order to support one or many **users** to carry out one or many **tasks** on one or many **domains** in one or many **contexts of use**, each context of use consisting of users, platforms, and environments". (*La distribution d'une interface utilisateur concerne la répartition d'un ou plusieurs* ***éléments*** *d'une ou plusieurs* ***interfaces utilisateur*** *afin d'aider un ou plusieurs* ***utilisateurs*** *à effectuer une ou plusieurs* ***tâches*** *dans un ou plusieurs* ***domaines*** *dans un ou plusieurs* ***contextes d'usage****, chaque contexte d'utilisation étant constitué d'utilisateurs, de plates-formes et d'environnements.*) Ce modèle permet de mettre en avant la place du contexte, et plus particulièrement du profil utilisateur dans la création d'interface. Cependant, les contextes dans lesquels les utilisateurs finaux interagiront ne sont pas forcément connus au moment de la conception, bien qu'ils puissent être en partie anticipés. Ainsi, les approches existantes dans la littérature proposent plusieurs solutions pour résoudre ce problème.

Les designers UI peuvent définir les différentes UI susceptibles d'être utilisées et sélectionner celle qu'ils préfèrent pendant le déploiement ou l'exécution. Cette démarche offre un bon contrôle sur les interfaces, et facilite l'évaluation du système. Toutefois, la capacité d'adaptation de l'UI est limitée à l'ensemble des interfaces ainsi conçues.

Les designers UI peuvent également concevoir des éléments d'interface utilisateur élémentaires qui peuvent être combinés avec un processus de génération d'interface utilisateur. Par exemple, le paradigme *comet* [39] permet de définir des interfaces graphiques en sélectionnant et assemblant des widgets configurables en fonction du contexte. En particulier, le style de chaque *comet* peut être modifié lors de la génération de l'interface concrète. Comme le proposent Sasse et al. [141], l'assemblage peut être réalisé automatiquement durant une session ou être personnalisable entre l'arrêt d'une session et le démarrage d'une nouvelle. L'automatisation de la génération des interfaces rend l'anticipation et l'évaluation de l'UI plus difficile. Néanmoins, l'adaptation au contexte peut être d'autant plus pertinente que le nombre de combinaisons possibles augmente. Cela permet de se servir au mieux de la diversité et du nombre de modalités dans un contexte de multimodalité massive (C14).

Quatre types de dispositifs peuvent fournir une interface utilisateur dans notre applica-

tion de réservation de salles de réunion : les écrans connectés, les enceintes, les smartphones et les montres intelligentes des utilisateurs. Pour concevoir les interfaces utilisateur, les designers peuvent dessiner ce que serait l'interface graphique élémentaire correspondant à chaque tâche. Ils peuvent également définir les commandes vocales permettant de demander à l'utilisateur de lancer l'interaction, de sélectionner une salle de réunion ou de confirmer le succès du processus de réservation.

Évaluation du design

Le dialogue modélisé et ses interfaces utilisateur répondent théoriquement aux exigences. Cependant, il n'est pas toujours possible lors de la conception de considérer toutes les situations auxquelles ces modèles peuvent être confrontés, et une de ces situations pourrait mettre en défaut ces modèles. Dans un premier temps, les designers peuvent détecter les défauts du modèle de dialogue. Ensuite, ils peuvent évaluer la cohérence des modèles par rapport aux exigences initiales.

Certaines règles et métriques peuvent être utilisées pour prédire les défauts des modèles de dialogue à l'aide de méthodes analytiques telles que l'analyse des chemins d'interaction [22] ou la viabilité des UI multimodales [45]. Dans une enquête, Abdulwakel et Nabil [1] ont proposé une classification des conflits de services basés sur l'IoT avec une méthode de détection des conflits *a priori*. Cette méthode consiste à identifier, à partir d'une analyse formelle, les conflits possibles entre plusieurs règles régissant le comportement du système. Par exemple, leur méthode permet d'identifier que la règle `SI l'utilisateur est present, ALORS allumer la lumiere` et la règle `SI la lumiere est allumee, ALORS l'utilisateur est present` peuvent mener à la création d'une boucle infinie. Cette méthode peut être utilisée pour évaluer l'adaptation de l'interface utilisateur basée sur des règles.

Pour vérifier que les modèles de dialogue et d'interface utilisateur sont conformes aux exigences initiales, Oviatt et Cohen [120] suggèrent que les designers expérimentent ces modèles avec des maquettes ("low-fidelity prototypes"). Par exemple, les expérimentations peuvent être réalisés à l'aide de cheminements cognitifs ("cognitive walkthrough") ou de revue de guidelines sur le système conçu, comme le présentent Bernhaupt et al. [22].

Pour réaliser un cheminement cognitif, une équipe composée de designers et d'experts du domaine conçoit un scénario d'usage défini par une suite d'action à accomplir. Cette équipe parcourt ensuite ce scénario, et se pose un ensemble de questions (ex. "L'utilisateur va-t-il comprendre l'action à accomplir pour passer à la suite ?") pour chaque action. Les

réponses à ces questions permettent de trouver les failles du système conçu.

En comparaison, une revue de guidelines consiste à faire noter le système dans sa globalité par un groupe d'experts du domaine selon une liste de critères, ou guidelines. Les guidelines compatibles avec le MIBS peuvent être des recommandations relatives à la conception d'entrées mobiles multimodales adaptatives comme dans [62], ou des normes du domaine. Par exemple, les normes ISO [1] sur le test et l'évaluation de systèmes de suivi peuvent être utilisées pour évaluer l'application de guidage.

Une alternative aux méthodes précédentes a été proposée dans le cadre du projet européen VUMS [123, 108]. Ce projet avait pour objectif de fournir des méthodes et standards pour modéliser et simuler des utilisateurs. L'alternative proposée est une méthode itérative pour modéliser et évaluer des systèmes interactifs avant l'étape de développement. Cette méthode a été utilisée pour réaliser des interactions multimodales, et elle suit un cycle de 3 étapes :

— un designer UX conçoit les modèles utilisateurs, de tâches et d'environnements sous formes d'ontologies ;
— un designer 3D crée des représentations 3D de l'environnement et des utilisateurs ;
— les modèles sont évalués dans une simulation.

Cette approche permet d'éviter des coûts supplémentaires causés par des erreurs de conceptions. En effet, elle facilite l'évaluation des modèles dans différents contextes. Cela permet d'éviter ainsi des problèmes situationnels qui n'aurait été rencontrés qu'à l'étape d'intégration (voir section 4.2.3). Néanmoins, cela ne recouvre pas toutes les situations possibles : chaque interaction entre l'agent et le système doit être modélisée. De plus, l'utilisation d'utilisateurs virtuels ne permet de prendre en compte que les qualités pragmatiques du système, ce qui ne permet pas de juger des qualités hédoniques telles que la satisfaction. Nous considérons donc cette approche comme une méthodologie d'évaluation itérative à considérer pour des applications ayant des coûts de développement plus élevés que des coûts de modélisation 3D.

Une fois que les conceptions sont jugées satisfaisantes, elles sont prêtes à être mises en œuvre.

Dans notre cas d'usage d'application de réservation, les designers peuvent évaluer les interfaces de l'application par revue des guidelines proposées par le W3C [2]. Par exemple, pour les interfaces vocales, le W3C recommande l'utilisation de mots-clés car les technolo-

1. `https://www.iso.org/obp/ui/#iso:std:iso-iec:18305:ed-1:v1:en`
2. `https://www.w3.org/TR/mmi-suggestions/`

gies de reconnaissance vocale actuelles ne permettent pas de comprendre aussi rapidement et précisément des phrases entières en langage naturel. Les designers doivent donc fournir un vocabulaire simple pour pouvoir sélectionner vocalement un créneau. Ensuite, les designers peuvent construire des maquettes du système. Pour cela, ils peuvent fournir plusieurs interfaces graphiques dessinées et des fichiers audio. Ces maquettes peuvent ensuite être utilisées pour réaliser des expérimentations sur des scénarios d'usage. Les testeurs de ces expérimentations peuvent ensuite remplir des questionnaires d'utilisabilité comme le MMQQ [162].

4.2.2 Étape de développement

Après l'étape de conception, l'étape de développement consiste à définir les composants logiciels (voir chapitre 2.1.3) et à les mettre en œuvre.

Modélisation des composants logiciels

À partir des interfaces utilisateur et des modèles de dialogue, les développeurs peuvent définir les composants nécessaires à la construction des chaînes d'interaction. Il existe différentes approches pour catégoriser ces composants, et chacune fournit des langages de modélisation pour définir les interfaces d'entrée et de sortie des composants, ainsi que leurs caractéristiques. Ces composants peuvent suivre la représentation utilisée en multimodalité (voir chapitre 2.1.1) comme dans notre approche présentée dans le chapitre 3, mais ce n'est pas la seule approche pour les MIBS. Par exemple, nous avons vu que dans DAME [131], les chaînes d'interaction comprennent des composants *contrôleurs de média* qui abstraient les capacités d'interaction des dispositifs ou d'un ensemble de dispositifs, et des composants *contrôleurs de langage* qui servent de pont entre ces composants spécifiques aux dispositifs et les composants de dialogue indépendants des dispositifs.

Pour définir quels composants concevoir, les développeurs peuvent s'appuyer sur des modèles conceptuels tels que le modèle WWHT ("What-Which-How-Then") de Rousseau et al.[139]. Ce modèle décrit le processus de conception d'une présentation multimodale. En particulier, il présente le cycle de vie d'une présentation comme un processus en 4 étapes :

— définir l'information à présenter ;
— choisir les modalités ;
— choisir la mise en forme de l'information présentée sur ces modalités ;

— préparer la future évolution de l'interface ainsi créée.

Les développeurs utilisent fréquemment des plateformes logicielles existantes lorsqu'ils travaillent avec les écosystèmes IoT de divers fournisseurs, et des librairies logicielles pour les algorithmes d'interprétation spécifiques à la modalité. Ainsi, des définitions rigides des composants peuvent compliquer le travail des développeurs lorsqu'ils veulent inclure ces logiciels. C'est l'une des raisons pour lesquelles Cronel et al. [55] proposent de définir dans un diagramme les technologies à utiliser (ex. librairies logicielles, API de services en ligne) dans les composants nécessaires avant de les développer.

Dans l'application de réservation de salle de réunion, les écrans, les microphones, les enceintes et les capteurs de présence dans chaque salle de réunion peuvent avoir leurs propres composants, comme dans notre approche. Pour modéliser comment présenter la liste de créneaux, le modèle WWHT peut être utilisé. L'information à présenter est la liste et elle peut être affichée sur un écran (sortie visuelle) ou énoncée en message audio (sortie vocale). Cette interface n'a pas besoin d'évoluer tant que l'utilisateur n'a pas sélectionné l'un des créneaux. Chacune de ces étapes peut être attribuée à un type de composant différent dans notre framework : le composant du dialogue définit l'information à présenter, un composant de fission choisit la modalité à utiliser, des composants de présentation réifient l'information pour que les objets puissent la présenter, et les composants d'interprétation détectent la sélection d'un créneau par l'utilisateur.

Développement des composants logiciels

La tâche suivante des développeurs consiste à mettre en œuvre les composants issus de leurs conceptions. Les composants sont généralement implémentés dans le même logiciel de développement (*Integrated Development Environment*, ou IDE), mais les fonctionnalités intégrées peuvent être différentes en fonction de leur proximité avec les dispositifs ou les techniques d'interaction. Par exemple, certaines librairies logicielles facilitent l'utilisation de services de reconnaissance vocale en ligne.

Les composants logiciels qui correspondent au modèle de dialogue sont généralement au centre des contributions sur l'architecture de systèmes multimodaux. Le dialogue peut être implémenté comme un ensemble de composants qui font partie de la chaîne d'interaction. Ces composants peuvent être génériques comme dans Openinterface [92], ce qui permet une plus grande liberté dans la modélisation du dialogue au dépend d'un support limité. En comparaison, les approches proposant des composants dédiés au dialogue comme dans la plateforme DAME [131] ont un formalisme plus strict. Cela permet toutefois d'assurer

des interfaces similaires entre ces composants, et donc d'assurer leur réutilisabilité. Le dialogue peut également être intégré dans la partie du système qui gère la composition elle-même. Par exemple, le gestionnaire du dialogue dans DynaMo [14] crée en fonction de modèles d'interaction des chaînes de composants liant les capteurs aux actionneurs.

Le développement des composants d'interprétation et de présentation dépend fortement de la modélisation de ces composants dans les étapes précédentes. Par exemple, les composants d'interprétation sont séparés des composants de fusion dans SIAM-DP [114]. Les premiers composants transforment les données des dispositifs en une représentation sémantique, tandis que les seconds ne font que fusionner les informations sémantiques. L'approche DAME [131] représente avec les mêmes composants les capacités d'entrée et de sortie. Ainsi, elle considère la dépendance potentielle entre les modalités d'entrée et de sortie.

Par exemple, les composants de l'application de réservation peuvent être développés depuis l'IDE Eclipse pour profiter des outils d'aide à la programmation. Le code peut ensuite être intégré dans l'IDE OpenInterface pour tester le fonctionnement des composants une fois assemblés.

4.2.3 Étape d'intégration

Au stade de l'intégration, tous les composants logiciels nécessaires ont été produits, et les développeurs et les designers d'interaction peuvent assembler des prototypes entièrement fonctionnels avec ceux-ci. Cette tâche d'assemblage est nécessaire pour les approches statiques, où les applications sont adaptées manuellement entre les sessions en fonction du contexte cible. Cependant, les approches dynamiques (i.e. adaptation au contexte durant l'exécution) peuvent fournir des prototypes sans assembler manuellement les composants, car cela sera fait automatiquement.

Dans les deux cas, les prototypes peuvent ensuite être évalués par des designers UX et des ergonomes dans des environnements réalistes. Comme le proposent Taib et Ruiz [154] dans leur méthode de conception par *Magicien d'Oz*, des prototypes partiels peuvent être créés pour effectuer ces évaluations, ou pour prospecter de futures techniques ou modalités d'interaction non disponibles pour le moment. En outre, Vilimek [161] affirme dans son étude du processus de création de systèmes multimodaux que le fait de travailler avec des prototypes partiels réduit le temps de réaction pour corriger les problèmes liés aux composants, ce qui réduit leurs coûts de production.

Enfin, lorsqu'un prototype est satisfaisant, il peut être préparé ("packaging") et fourni

aux administrateurs en vue de son déploiement.

Assemblage des composants

L'assemblage fait référence à la composition des composants logiciels développés pour produire les chaînes d'interaction. Le processus d'assemblage peut être confié à des designers d'interaction ou à des développeurs, selon les expertises en design et en développement requises par l'outil choisi, comme détaillé dans la section 4.2.2.

Dans l'application de réservation de salle de réunion, les designers peuvent tester la tâche d'interaction de sélection de salle de réunion avec un smartphone. Pour ce faire, ils connectent un composant associé au smartphone, un composant d'interprétation pour la reconnaissance vocale, le composant de fusion présenté précédemment, un composant de dialogue, et un composant de présentation pour transformer un message en une partie élémentaire d'interface graphique. Des exemples de chaînes de composants pour l'application de réservation se trouvent en annexe A.

Détection des problèmes d'utilisabilité

L'équipe de designers, d'ergonomes et d'analystes (appelée par la suite *équipe d'évaluation*) peut évaluer les prototypes afin de détecter les défauts du MIBS (i.e. les problèmes dans les chaînes d'interaction, la mise en œuvre des composants ou les modèles de conception), les modèles dans les comportements des utilisateurs, ou si les prototypes sont toujours conformes aux directives présentées dans la section 4.2.1. Ainsi, la détection de problème peut amener à des changements dans les étapes précédentes. Selon [131], les méthodes d'évaluation ne fournissent pas d'analyses exhaustives et complètes mais constituent l'observation la plus proche de l'utilisation anticipée du système jusqu'à présent. Elles peuvent être séparées entre les méthodes formelles (i.e. basées sur les modèles) et les tests utilisateurs.

Le premier type est basé sur la simulation du système avec des agents virtuels et des environnements modélisés. Par rapport aux autres approches, les simulations sont bon marché, flexibles, et peuvent créer des situations et des environnements difficiles, voire impossibles à reproduire dans la réalité [131, 120]. Cependant, elles sont des versions simplifiées, voire déformées, de leurs homologues du monde réel, ce qui réduit la validité des résultats et introduit des biais. De plus, les utilisateurs finaux ont une capacité à trouver les limites et les défauts des systèmes que les méthodes d'évaluation par simulation n'ont pas [131]. Des méthodes de tests automatiques peuvent aussi être utilisées, mais

leur intérêt est limité. En effet, les tests automatiques ne sont pas conseillés sur des systèmes très personnalisables tels que les MIBS, et ne sont pas suffisants pour réaliser des tests sur l'expérience utilisateur, comme le suggèrent Garousi et Mäntylä dans leur revue de littérature sur les outils des pratiques de tests automatiques. L'étude réalisée par Nass et al. [68, 112] sur les tests automatiques d'interface graphiques (GUI) semble confirmer que ces méthodes d'évaluation sont encore limitées. De plus, ces méthodes demandent généralement des connaissances en programmation pour définir les tests à réaliser. Toutefois, elles peuvent être utilisées pour réaliser des tests partiels sur des critères objectifs. Un des critères possibles est la performance d'un composant d'interprétation en fonction des données placées en entrée.

Le deuxième type consiste en des expérimentions menées avec des utilisateurs. Comme l'expliquent Silva et al. [147], un prototype du système interactif est testé par des utilisateurs dans des environnements contrôlés (ex. espaces d'expérimentation de laboratoires) ou dans l'environnement cible. Ces tests permettent d'avoir un aperçu de l'utilisabilité du système et de collecter le ressenti des utilisateurs. L'équipe d'évaluation peut évaluer au préalable les prototypes pour vérifier s'ils sont toujours conformes aux directives considérées dans la section 4.2.1. L'avantage est d'évaluer les alternatives de composition des composants dans des prototypes partiellement ou totalement fonctionnels [161, 22] en s'appuyant sur des normes fiables et sur l'expérience des designers. Ensuite, de vrais utilisateurs peuvent expérimenter les prototypes. Les comportements des participants sont analysés [161, 120] et leurs retours d'expérience sont enregistrés, comme dans le processus de prototypage rapide de Lemmelä et al. [95]. Avec un prototype complet entre les mains d'utilisateurs réels, ces expérimentations sont les plus proches de la vérité du terrain, mais leurs coûts, les limites des environnements que les expérimentateurs peuvent créer et la rigidité de leurs configurations sont leurs inconvénients [161, 120]. On notera qu'il est aussi possible de réaliser des tests pilote. Ces tests consistent à installer le système dans de vrais environnements pour étudier son influence dans la routine des utilisateurs [147].

Dans l'application de réservation de salles de réunion, c'est au cours de ces expérimentations que les designers peuvent être confrontés à des problèmes pratiques. Par exemple, les rayons lumineux de l'après-midi peuvent éblouir l'utilisateur faisant face à l'écran, ce qui peut rendre l'affichage d'une des salles de réunion envisagées impossible à utiliser.

Il convient de noter que la simulation est parfois suivie d'expériences avec les utilisateurs pour obtenir le meilleur des deux mondes [85, 130]. En effet, le fait de commencer par des simulations permet d'éliminer rapidement et à faible coût les problèmes anticipés

par l'équipe d'évaluation. Des expérimentations peuvent être réalisées par la suite pour détecter les problèmes que seule l'analyse du comportement des utilisateurs finaux permet de trouver. Cependant, le coût et la durée des deux méthodes utilisées consécutivement reste élevé.

4.2.4 Étape de déploiement

Carzaniga et al. [42] définissent le déploiement d'un logiciel comme une étape complexe qui commence par la mise en circulation du logiciel et se termine quand celui-ci est désinstallé. Cependant, nous nous concentrerons sur l'étape d'installation. Cette étape nécessite qu'un administrateur récupère le système et le configure pour qu'il soit utilisable dans son environnement. En particulier, la configuration de systèmes ubiquitaires requiert une gestion de 3 aspects selon [5] :

— le positionnement des ressources (i.e. les objets connectés) ;
— la configuration de services qui agrègent les données de plusieurs capteurs, et qui envoient des informations à plusieurs ressources ;
— l'association de ressources aux services.

Les systèmes ubiquitaires doivent donc gérer la disposition spatiale des dispositifs, s'occuper de la configuration du MIBS et connecter les dispositifs associés au MIBS. L'ordre dans lequel les trois tâches précédentes sont effectuées est arbitraire et peut changer si le système peut s'adapter à un changement de position des dispositifs, ou s'il y a un ordre entre le système et le démarrage des dispositifs.

Positionnement des objets

Les appareils connectés étant de plus en plus courants dans notre environnement, ils sont généralement déjà installés. Cependant, les administrateurs peuvent avoir besoin de davantage d'appareils pour une application particulière. Ils peuvent aussi déplacer certains de ceux qui sont installés. Selon Burzagli et al. [37], cela implique un placement minutieux de ces appareils, car ils doivent se fondre dans l'environnement de l'utilisateur. Cela a aussi une grande importance pour le suivi de l'utilisateur dans des situations de mobilité. Le positionnement relatif entre les dispositifs et la géométrie de l'environnement sont donc également importants.

Bien qu'il n'y ait pas à notre connaissance de méthode pour placer les objets en fonction des applications, les administrateurs peuvent suivre les instructions des recommandations

d'installation. Cependant, de multiples essais et erreurs peuvent être nécessaires avant d'obtenir un positionnement satisfaisant.

Par exemple, le geste de pointage pour sélectionner un créneau dans l'application de réservation peut être capté par une caméra. Cependant, le placement d'une caméra pour ce type d'interaction n'est pas évident. En effet, la caméra ne doit pas être trop prêt ou trop loin de l'écran qui affiche les créneaux pour voir le geste, aucun obstacle ne doit se trouver entre la caméra et l'utilisateur et la position de la caméra ne doit pas gêner les occupants de la pièce.

Configuration du MIBS

La géométrie de l'environnement, les caractéristiques des dispositifs d'interaction disponibles et la qualité du réseau font partie des préoccupations du MIBS. Ces considérations contextuelles ne sont connues qu'au moment du déploiement, et sont difficilement récupérables par les méthodes automatiques de détection de contexte. Ainsi, les administrateurs peuvent configurer le système en fonction du contexte d'utilisation avant de démarrer une session. Par exemple, Burzagli et al. [37] proposent d'intégrer un processus de configuration au démarrage permettant de spécifier les modalités à utiliser, l'emplacement des composants d'interprétation et la façon d'utiliser les modalités (i.e. les techniques d'interaction). Pour les approches statiques, ils peuvent également sélectionner les chaînes d'interaction avec les dispositifs disponibles, comme dans ICARE [30].

Le besoin d'adapter un système ubiquitaire aux besoins de l'utilisateur se retrouve dans le paradigme du développement par l'utilisateur ("end-user development") [17]. Dans ce type d'approche, l'utilisateur agit comme un développeur non-expert pour configurer son système. Les méthodes existantes sont principalement basées sur des règles de type IFTTT ou sur de la programmation visuelle (i.e. relier des composants par des traits). Les règles peuvent décrire des comportements sans avoir à spécifier les objets à utiliser, les approches par règles sont donc préférables pour les systèmes capables d'adaptation dynamique, Au contraire, la programmation visuelle est plus souvent utilisée pour décrire les chaînes d'interaction [56], cette méthode est donc plus fréquente dans les systèmes plus statiques. Il existe aussi des méthodes non-programmatoires, comme la méthode de configuration par démonstration présentée dans [18]. Cette méthode permet de sélectionner ses modalités préférées en montrant la technique d'interaction au système.

Dans notre cas d'utilisation, nous considérons que les responsables de l'équipement informatique (i.e. l'administrateur d'un bâtiment d'entreprise, ou l'utilisateur final pour

une application domestique) choisissent les dispositifs. Ainsi, ils peuvent sélectionner une chaîne d'interaction parmi celles recommandées par les designers. Par exemple, ils peuvent choisir la chaîne d'interaction qui utilise des microphones et des capteurs de présence, puis associer les appareils connectés de chaque salle de réunion à l'application.

Mise en place des pilotes des objets connectés

Enfin, les dispositifs connectés doivent être démarrés, configurés et connectés au système en fonction des capacités de découverte et d'enregistrement de l'architecture.

En ce qui concerne le processus de communication, Rodriguez et Moissinac [137] expliquent que bien que des travaux de standardisation menés par des consortiums industriels ont été réalisés [3], il n'existe pas de protocole unique dans l'IoT, car chaque fournisseur possède généralement son propre écosystème de réseau. Par conséquent, le déploiement des dispositifs n'est généralement pas abordé dans la littérature relative aux MIBS, et les approches existantes (voir chapitre 2.2) ne se placent que dans des environnements où les objets sont déjà placés et configurés. Néanmoins, Rodriguez et Moissinac proposent dans leurs travaux un modèle de composant de modalité et un processus de découverte et d'enregistrement qui pourraient aboutir à un consensus dans la communauté IoT.

Les administrateurs peuvent également spécifier ici la localisation des appareils s'ils n'ont aucun moyen de déterminer automatiquement leur emplacement dans l'environnement. Plusieurs technologies de localisation peuvent être utilisées, et chacune a ses avantages et limites [33, 65]. On peut en particulier différencier ces techniques selon le type d'information de position récupéré : absolues (ex. GPS), relatives (ex. en face de la porte d'entrée) et de proximité (ex. proche d'une balise).

Dans notre cas d'utilisation, les smartphones sont accessibles via les réseaux Wifi ou mobiles, tandis que les capteurs de présence peuvent n'être disponibles qu'avec Z-wave [4] ou d'autres protocoles et technologies de bas niveau. Ainsi, les responsables informatiques peuvent utiliser un outil tel que OpenHAB [5] pour permettre au MIBS de communiquer avec tous les appareils via un réseau et un protocole uniques.

3. LoRa Alliance (`https://lora-alliance.org/`) et CSA `https://csa-iot.org/fr/` sont des exemples de groupes travaillant sur la standardisation dans l'IoT

4. `https://www.z-wave.com/`

5. `https://www.openhab.org/`

4.2.5 Étape d'exécution

Une fois que le MIBS fonctionne, les utilisateurs finaux peuvent enfin interagir avec lui. Bien que le processus création du MIBS soit terminé, les utilisateurs finaux peuvent avoir besoin d'assistance pour mieux comprendre et contrôler le comportement et les causes de dysfonctionnement des MIBS. Nous verrons Section 4.2.5 le type d'assistance que les MIBS peuvent offrir.

L'évaluation du système peut aussi être approfondie à cette étape. En effet, l'utilisation du MIBS déployé dans les environnements d'exécution permet d'identifier plus précisément ce que les utilisateurs finaux aiment ou non dans l'interaction avec le système.

Analyse de l'expérience utilisateur

L'évaluation peut être séparée en deux parties : l'observation de l'utilisateur et le retour de l'utilisateur. Pour ce dernier point, des techniques telles que les propositions d'amélioration de l'utilisateur [16] ou les questionnaires [22] peuvent être utilisées pour détecter des défauts qui n'avaient pas été détectés auparavant. Les préférences des utilisateurs [22] peuvent également être détectées, par exemple à l'aide de l'analyse des fichiers de logs des observations sur le terrain.

Ces méthodes d'évaluation sont similaires aux expériences basées sur l'utilisateur dans l'étape d'intégration, mais l'absence de paramètres contrôlés, le plus grand nombre d'utilisateurs et la diversité potentiellement plus grande des environnements permettent d'obtenir des informations plus approfondies sur le système. Toutefois, cela se fait au prix d'un coût plus élevé et cela peut influencer la réputation du produit [120, 40]. De plus, les dispositifs d'interaction peuvent être cachés dans les MIBS. Par conséquent, les utilisateurs sont moins conscients de la manière dont le système collecte les données les concernant et peuvent être moins disposés à partager leurs données.

Dans notre cas d'utilisation, l'administrateur peut être le seul à avoir accès au taux d'utilisation de l'application de réservation de salles de réunion par les utilisateurs. En outre, les données peuvent être rendues anonymes pour respecter la confidentialité des données des utilisateurs.

Le résultat de l'analyse des données peut être pris en compte à n'importe quelle étape. En effet, il peut conduire à de nouvelles exigences, amorçant ainsi un tout nouveau cycle. Des problèmes de mise en œuvre peuvent être découverts dans des situations spécifiques. De nouvelles chaînes d'interaction peuvent être déduites des commentaires des utilisateurs,

tandis que les administrateurs peuvent modifier certains paramètres pour mieux adapter le système aux besoins des utilisateurs.

Assistance à l'utilisateur

Nous identifions 2 manières d'aider l'utilisateur à gérer un MIBS : l'intégration de méta-IHM et l'utilisation de technologies compagnons.

Coutaz et Nigay [52] définissent une méta-IHM comme "l'ensemble des fonctions (et leur IHM) nécessaire et suffisant afin que les utilisateurs puissent contrôler et évaluer l'état de leur espace interactif". Ainsi, pour améliorer l'utilisabilité du système, les MIBS peuvent intégrer un système de méta-IHM. Par exemple, Vanderhulst et al. [159] proposent avec Meta-STUD un framework de référence pour les méta-IHM permettant de gérer le fonctionnement d'applications interactives. Leur modèle permet d'interdire, d'autoriser voire de forcer l'usage d'un objet connecté, Il est aussi possible de démarrer, mettre en pause ou arrêter une application. Une méta-IHM peut aussi être utilisée pour mieux percevoir le fonctionnement d'un MIBS. En effet, Brudy et al. [33] identifient 4 type de fonctionnalités de visualisation qu'une méta-IHM peut intégrer pour aider un utilisateur :

- donner l'accès aux informations sur les interactions selon les objets et l'espace de test ;
- supporter la visualisation des données captées par les objets ;
- permettre le suivi du traitement des données par feedbacks ;
- faciliter la visualisation du lien entre les objets.

Par exemple, une méta-IHM peut donner l'accès aux positions des objets sur une carte en vue du dessus, supporter la visualisation des retours de caméras, permettre l'affichage du squelette de l'utilisateur et faciliter à l'aide d'un code couleur la visualisation de l'association de plusieurs écrans à une même UI. Cependant, la création de tels services demande une compréhension des applications en cours d'exécution qui peut être difficile à obtenir dans des systèmes aussi dynamiques que les MIBS. De plus, la visualisation peut être rapidement surchargée dans un contexte de multimodalité massive.

Le paradigme de technologie compagnon peut être une solution pour assister l'utilisateur dans l'exécution de tâches complexes. L'objectif de cette technologie est d'adapter l'aide en fonction des connaissances de l'utilisateur, des tâches à effectuer et de l'équipement à disposition. Par exemple, un système compagnon conçu selon l'architecture proposée par Bercher et al. [21] peut utiliser les capteurs et actionneurs dans un environnement d'exécution pour aider l'utilisateur à interagir avec une application. Pour cela, ce

système a besoin du modèle de dialogue de l'application, ainsi que de textes, d'images et de vidéos à présenter à chaque étape de ce modèle.

4.2.6 Analyse des rôles dans la conception des MIBS

Comme nous l'avons vu précédemment, la création d'un MIBS est un travail d'équipe pluridisciplinaire où chaque tâche requiert des connaissances et compétences particulières. De plus, Augusto et al. [11] expliquent qu'il y a plusieurs parties prenantes dans la gestion des environnements ubiquitaires : "Sensors available in an intelligent environment may be owned and managed by different organizations or people" (*les capteurs disponibles dans un environnement intelligent peuvent appartenir et être gérés par des organisations ou des personnes différentes*). C'est donc au système interactif d'être adapté à l'équipement en place, et seules certaines personnes en charge de l'environnement peuvent modifier la disposition de cet équipement, voire ajouter de nouveaux objets.

Cette pluridisciplinarité et cette séparation des responsabilités font qu'il est important de définir les rôles intervenants à chaque étape. Cela permet plus particulièrement de définir la cible des méthodes et outils existants pour chacune des étapes (voir section 4.3). Toutefois, la description des rôles (i.e. tâches à réaliser, domaines d'expertise et degrés d'expertise) peut changer d'une organisation (i.e. éditeur de logiciels et client) à une autre. C'est pour cela que nous nous basons sur la norme ISO sur l'ingénierie logicielle et les systèmes [80] pour présenter les rôles qui sont généralement considérés dans la création de MIBS, et dont les domaines d'expertise ne se chevauchent pas. Ainsi, nous identifions 6 rôles que nous détaillons plus en détail par la suite : designer UI, designer UX, développeur, ergonome, administrateur et utilisateur final. L'association des rôles aux tâches est synthétisée Figure 4.4.

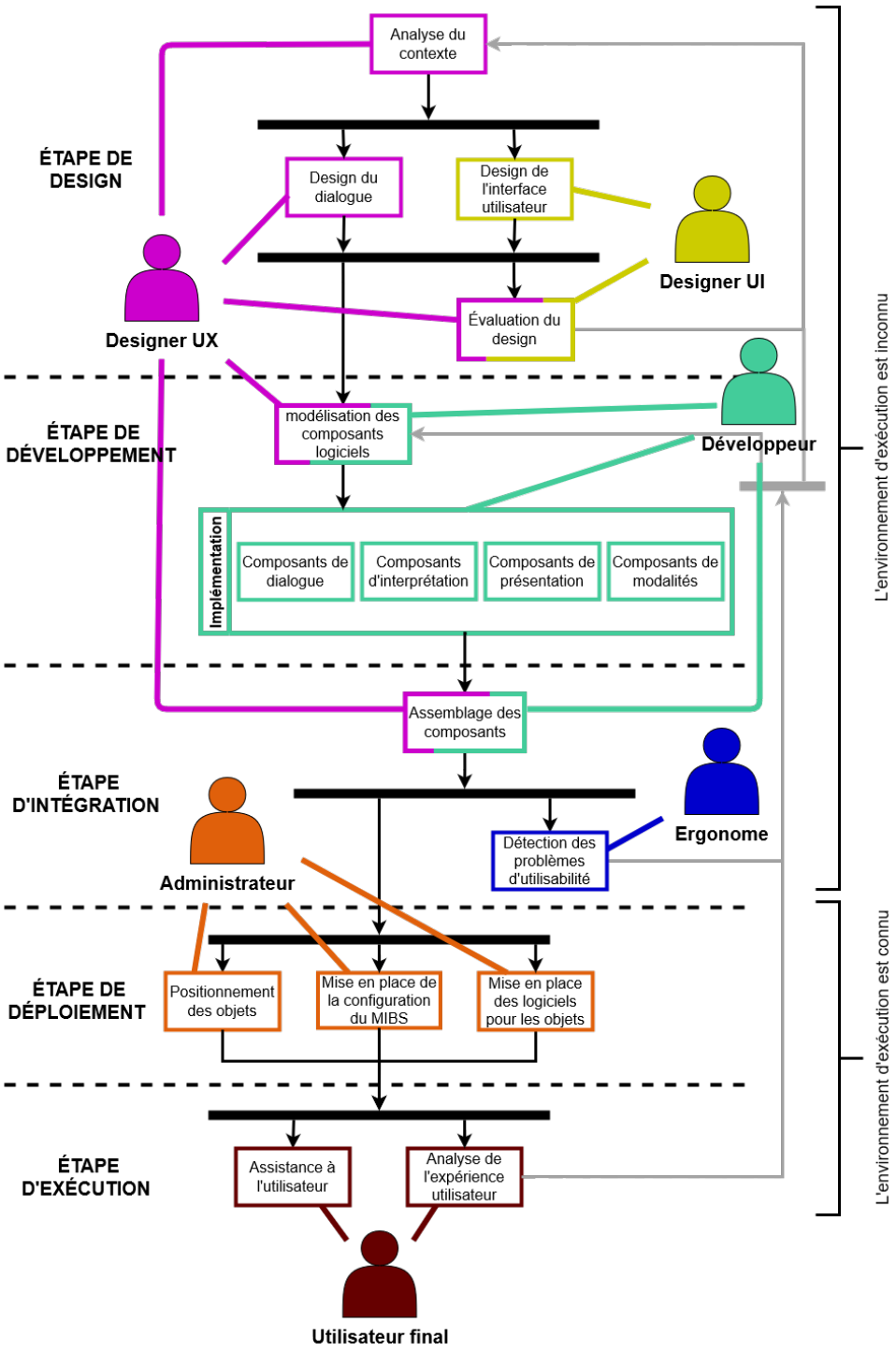


FIGURE 4.4 – Association des 6 rôles aux tâches du cycle de vie des MIBS. Les tâches d'évaluation des étapes de design, de modélisation des composants logiciels et d'assemblage des composants requièrent la collaboration de deux rôles.

On notera que la notion d'expert est aussi utilisée comme rôle à part entière [58]. Néanmoins, elle est principalement utilisée pour décrire un niveau d'expertise d'un des acteurs présentés précédemment. Ainsi, nous utilisons le terme d'expert pour décrire les acteurs quand la distinction sera nécessaire (ex. ergonome expert). Similairement, le rôle d'analyste n'intervient que pendant les phases de définitions des exigences, et parfois dans l'évaluation des prototypes. Nous considérerons donc ce rôle comme externe à la création de MIBS, à l'exception de sa tâche de référent expert lors de l'évaluation de MIBS. On notera aussi que deux des avantages des MIBS sont de pouvoir utiliser des objets installés pour d'autres usages, et des services en ligne (ex. reconnaissance vocale de Google). On considère donc que les fournisseurs de solutions IoT et de services en ligne ne sont pas des acteurs dans la création de MIBS, mais l'interfaçage de ces ressources externes fait partie du travail des acteurs considérés.

Designer UI et designer UX

La création de MIBS commence par une phase de modélisation du contexte, du dialogue et des interfaces utilisateur. Les modèles ainsi créés s'appuient sur une représentation du monde qui vient alimenter la base de connaissances du système. Ils ont aussi un impact sur la conception des composants logiciels et leur assemblage. Ces tâches peuvent donc être réalisées par un designer. En effet, l'ISO [80] définit le design ainsi : "The process of defining the software architecture, components, modules, interfaces, and data for a software system to satisfy specified requirements" (*le processus de définition de l'architecture logicielle, des composants, des modules, des interfaces et des données d'un système logiciel pour satisfaire des exigences spécifiées*). Cela inclue donc les tâches présentées précédemment.

De plus, la séparation entre la conception des interfaces et du dialogue permet de définir deux sous-catégories de designers, à savoir le **designer UX** ("**U**ser e**X**perience") chargé du dialogue et le **designer UI** en charge de la conception des interfaces utilisateur.

Le désigner UX identifie le contexte général du MIBS à partir des besoins des utilisateurs finaux. Il modélise ensuite le dialogue pour proposer une expérience satisfaisante en toute situation. On peut retrouver parmi les critères d'évaluation des qualités hédoniques tels que la facilité d'utilisation, l'agréabilité, et la qualité d'interaction [162].

Le designer UI réalise l'organisation et l'esthétique des éléments des interfaces pour des raisons d'utilisabilité et de satisfaction utilisateur. Il peut aussi prendre en compte les mêmes critères que le designer UX pour juger de l'expérience utilisateur fournie par

chaque interface. Ce rôle peut être d'avantage séparé selon les modalités considérées. Par exemple, la création d'interfaces graphiques (GUI) requiert diverses connaissances qu'aurait un designer graphique [6]) (ex. HTML/CSS), tandis que la conception d'interfaces vocales (VUI) demande une toute autre gestion dans la communication d'information [7]. De plus, la sélection et l'organisation des modalités [133] dans les systèmes multimodaux prennent en compte des critères d'UI design, ce designer peut donc intervenir dans la conception de ces tâches.

Les designers UX et UI échangent ensuite avec les développeurs pour vérifier que le produit développé (i.e. produit fini ou prototype [120]) répond bien aux besoins identifiés au départ.

On notera que la séparation n'est pas toujours clairement définie (ex. un designer peut remplir les deux rôles dans une petite équipe), et le design UI est même parfois considéré comme une partie du design UX [8], ce qui est contraire au principe d'indépendance entre les modalités et le dialogue.

Développeur

Les modèles conçus par les designers UX et UI doivent ensuite être intégrés dans des composants logiciels. Cette tâche est généralement attribuée au rôle de développeur dans la littérature, bien qu'elle puisse être réalisée en collaboration avec les designers pour assurer que les choix techniques n'influencent pas en mal l'interaction (par exemple un délai dans la gestion du dialogue[61]). Ce sera par exemple le cas avec la plateforme DAME [131].

L'ISO [80] donne la définition suivante pour le rôle de développeur de produit logiciel : "The person or organization that manufactures a software product" (*La personne ou l'organisation qui fabrique le produit logiciel*). La structure par composants utilisée pour créer les MIBS fait que la fabrication du produit logiciel se limite à la création et l'assemblage de ces composants.

Nous avons vu que les types de composants diffèrent entre chaque approche. Néanmoins, on attend de ces composants qu'ils définissent les 4 éléments des systèmes multimodaux : les modalités, le processus d'interprétation, le dialogue et le processus de

6. https://en.99designs.fr/blog/tips/types-of-graphic-design/
7. https://careerfoundry.com/en/blog/ux-design/how-to-become-a-voice-designer/#what-does-a-vui-designer-actually-do
8. https://uiuxtrend.com/ui-design-vs-ux-design-vs-interaction-design/

présentation. Par exemple, un développeur peut intégrer un algorithme d'interprétation de gestes dans le composant "MIBO definition Expert" du framework MIBO [125].

Il n'y a pas vraiment de distinctions strictes entre les développeurs, bien que le besoin d'expertise varie selon les 4 éléments cités précédemment. De plus, chaque modalité requiert un traitement particulier qui demande une maîtrise suffisante. Par exemple, l'utilisation d'un microphone demande d'utiliser des algorithmes de reconnaissance vocale et d'interprétations sémantiques des énoncés, tandis que l'utilisation de caméra pour interagir avec des gestes passe par des algorithmes de traitement d'image et reconnaissance de gestes. Nous séparerons par la suite les tâches de développement selon le type de composants à développer pour distinguer le ou les domaines d'expertise à considérer.

Utilisateur final

Comme défini par l'ISO [80], les utilisateurs finaux sont la cible du système : "the person or persons who will ultimately be using the system for its intended purpose" (*La personne ou les personnes qui vont utiliser le système pour l'usage prévu à la fin*). Augusto [12] fait aussi la distinction entre les utilisateurs primaires et les autres bénéficiaires. Cette distinction est utile dans les systèmes tels que les assistants de personnes vulnérables où le personnel soignant (utilisateur secondaire) a besoin de superviser l'état des personnes assistées (utilisateurs primaires).

Ainsi, dans la plupart des méthodologies de création de MIBS, le rôle d'utilisateur final se limite à l'utilisation du système. Toutefois, les approches centrées utilisateur font aussi participer les utilisateurs au processus de création, comme l'expliquent Oviatt et Cohen [120]. Par exemple, ils peuvent aider à spécifier les besoins et à tester le système lors d'expérimentations. Il est aussi possible de leur fournir des outils pour leur permettre d'observer, de contrôler le système interactif, voire de fournir des commentaires aux autres acteurs sur le fonctionnement du système dans leurs environnements.

Administrateur

Les rôles présentés dans les sections précédentes ne suffisent pas à prendre en compte l'ensemble des besoins de notre cas d'usage. En effet, il manque un rôle d'**administrateur du système** (appelé par la suite "administrateur") pour gérer son déploiement (ISO : "installation and checkout phase") et sa maintenance. Ce rôle se retrouve dans MIBO [125] sous le rôle de "Facility Manager", et Pruvost [131] utilise le terme d'*administrateur d'ambiant*.

En effet, les étapes de déploiement et de maintenance nécessitent un accès au site, ce qui n'est pas toujours possible pour les designers et les développeurs (ex. sous-traitance du design/développement ou comme ici, relation fournisseur-client). De plus, les éditeurs de logiciels interactifs ne proposent pas toujours un service d'aide à l'installation. Ainsi, les designers et développeurs qui n'ont pas connaissance de tous les environnements d'exécution ne peuvent pas réaliser cette tâche. De plus, la gestion de l'équipement est attribuée à des employés particuliers qui ne sont pas toujours les utilisateurs finaux. En effet, l'utilisateur final n'a pas toujours les qualifications suffisantes ou les autorisations pour configurer le système interactif. En plus du niveau d'accréditation possiblement insuffisant, l'accès à des informations possiblement privées appartenant aux utilisateurs finaux, ainsi que le temps à accorder à la maintenance du système sont des éléments qui dépassent souvent le cadre de travail des utilisateurs finaux.

A noter que dans certains cas l'utilisateur final peut réaliser les tâches de déploiement. C'est par exemple le cas pour les applications domotiques où une personne utilise elle-même le système qu'elle administre.

Ergonome

La création de MIBS passe par l'intégration de plusieurs composants qui peuvent avoir été réalisés séparément. Vérifier la cohérence et l'utilisabilité d'une application selon certains environnements d'interaction types avant le déploiement (au directement dans l'environnement cible) permet de valider en partie le travail réalisé en amont. Une meilleure validation nécessiterait de réaliser les tests dans les environnements de déploiement réels, ce qui n'est pas toujours possible. Cette tâche ne relève pas d'un rôle précédemment présenté, mais plutôt du rôle d'ergonome. Ce rôle est aussi présent dans le framework MIBO ("Usability Engineer"). DAME [131] s'appuie aussi sur le rôle "d'ergonome d'ambiant", bien qu'il intervient aussi pendant l'étape de design pour définir des règles de comportement, en coopération avec les designers.

Un ergonome peut aussi intervenir pour valider la configuration d'une application dans son environnement d'exécution si celui-ci est disponible. Cela permet d'assurer que cette application remplit correctement ses objectifs.

Analyse du cycle de vie des MIBS

Les tâches à effectuer tout au long du cycle de vie des MIBS peuvent être réparties en 5 étapes : le design, le développement, l'intégration, le déploiement et l'exécution. Nous avons identifié les méthodes utilisées pour réaliser chaque tâche, ce qui nous a permis d'identifier les 6 rôles requis pour ces tâches : le designer UI, le designer UX, le développeur, l'ergonome, l'administrateur et l'utilisateur final.

4.3 Analyse des méthodes et outils par tâches

Nous avons vu dans la section précédente que les 5 étapes pour créer des MIBS peuvent être décomposées en plusieurs tâches distinctes. Ces tâches peuvent exiger des connaissances avancées en matière de conception et de développement, peuvent prendre du temps et sont parfois répétitives. De plus, ces tâches sont réalisées par différents rôles qui ont des points de vue différents sur le processus de création. Par conséquent, des outils dédiés sont nécessaires pour faciliter le travail des différentes parties prenantes.

Selon l'étude de Dahl sur les normes multimodales dans les systèmes distribués [57], les outils logiciels facilitent et accélèrent le processus de développement des applications multimodales, mais leur utilité dépend des normes de conception représentant le système. Par conséquent, nous présentons dans cette section les outils logiciels qui ont été utilisés pour créer des systèmes multimodaux et qui sont compatibles avec les systèmes basés sur l'IoT, en fonction des tâches qu'ils aident à accomplir à chaque étape.

4.3.1 Outils de design

Comme détaillé dans la section 4.2.1, l'étape de conception du MIBS consiste à définir les informations contextuelles, le modèle de dialogue et les interfaces utilisateur. Une grande partie des outils de conception utilisés pour les MIBS sont des outils utilisés pour les systèmes multimodaux. En effet, à cette étape, les exigences supplémentaires par rapport à un système multimodal sont de fournir des interfaces utilisateur pour un ensemble plus large de dispositifs interactifs (C14) et de créer un dialogue sensible au contexte (C7) qui soit aussi indépendant de la notion de modalités (C15). Or les outils de conception pour les systèmes multimodaux sont souvent suffisamment flexibles pour le

permettre. Les outils sont synthétisés dans le tableau 4.1.

OUTILS	RÔLES ASSOCIÉS	TÂCHES			
		Analyse du contexte	Modélisation du dialogue	Modélisation de l'UI	Évaluation du design
Éditeurs d'ontologies (ex. [131])	Designer (ontologies)	✓			
CTTe [110]	Designer et développeur		✓		Dialogue
Outils de modélisation du dialogue [82, 113, 32]	Designer UX		✓		Dialogue
Outils de design web (ex. Figma (11)	Designer Web			Web	
Markup validator (12)	Designer Web				Web
Outil de design d'interfaces graphiques distribuées [73]	Designer ou développeur UI			GUI distribuée	
Éditeur de GUI distribuées [98]	Designer UI			GUI distribuée	
	Développeur UI				GUI distribuée
TERESA [121]	Designer UI			GUI, VUI	
Sketch (13)	Designer UI			Apple GUI	
Éditeur de graphes vocaux (ex. SUEDE [86])	Designer UI			VUI	VUI
Feelix [86]	Designer UI			Tangible	
d.tool [75]	Développeur			Interfaces expérimentales	
Éditeur d'UI multimodale (ex. MOSTe [140])	Designer UI	règles d'adaptation		✓	
Spin [13]	Designer UX				Dialogue
SIHMM [25]	Designer UX				✓
MuMoWOz [25]	Designer UX				GUI, VUI mobiles
CrossWeaver [148]	Designer UX				✓

TABLE 4.1 – Synthèse des outils qui supportent la phase de design.

Outils de modélisation du contexte

Comme présenté dans le chapitre 4.2.1, il existe différentes classes d'informations contextuelles (i.e. utilisateur, plateforme, environnement). Par conséquent, la représentation doit être suffisamment flexible pour prendre en charge la diversité des contextes. Nous avons vu dans le chapitre 2.1.3 que les données dans les MIBS sont généralement

représentées par des ontologies pour assurer cette flexibilité. Nous présentons donc ici les outils qui facilitent la modélisation du contexte par ontologies.

Bien qu'il n'y a pas de consensus sur l'organisation du contexte [12], plusieurs ontologies ont été proposées dans le domaine de l'informatique ubiquitaire comme par exemple l'ontologie utilisée dans CoBrA ("Context Broker Architecture") [48] ou BOnSAI ("Smart Building Ontology for Ambient Intelligence") [150]. Certaines approches considèrent aussi la cohabitation de plusieurs ontologies conçues par différents parties prenantes. Ces ontologies peuvent être sémantiquement incompatibles, donc elles sont maintenues séparément, et possiblement alignées (i.e. mise en correspondance des entités de plusieurs ontologies). Par exemple, chaque entité (i.e. les objets, les utilisateurs, les applications) a une ontologie qui la représente dans le cadre du projet ATRACO (**A**daptive and **TR**usted **A**mbient e**CO**logies) [70]. Ces ontologies sont ensuite alignées par une ontologie pivot.

Des outils tels que l'éditeur d'ontologie Protégé [9] permettent de gérer facilement des ontologies standard génériques. Protégé a notamment été utilisé pour représenter des bases de connaissances dans des systèmes multimodaux [105, 131] et supporte le standard de langage du W3C [10]. Cependant, Pruvost [131] affirme que cet outil est complexe à prendre en main et trop permissif. Il a donc construit sur son API l'outil "Describe" [131] pour simplifier l'édition des ontologies, et interdire l'édition de leurs ontologies de base d'architecture.

Les outils existants sont donc suffisants pour faciliter la modélisation du contexte par ontologies.

Outils de modélisation du dialogue

Une fois le contexte défini, de nombreux outils sont proposés pour modéliser les tâches du dialogue [118]. Par exemple, CTTe [110] aide les designers d'interaction à spécifier le dialogue comme des tâches d'interaction basées sur une représentation en arbre des tâches. Cet outil offre également une fonctionnalité de maquettage [120] pour vérifier si le fonctionnement dynamique du modèle de tâches correspond aux exigences de l'interaction. Par exemple, la complémentarité entre la tâche de sélection de salle de réunion et la tâche de commande de réservation dans l'application de réservation de salle de réunion peut facilement être décrite comme des sous-tâches liées par un opérateur temporel.

Le dialogue conçu peut être analysé indépendamment du reste du système afin de

9. `https://protege.stanford.edu/`
10. `https://www.w3.org/TR/owl2-overview/`

prévoir ses défauts. Similairement à la capacité de simulation de CTTe, deux autres outils peuvent aider à faciliter l'analyse du système. Silva et al. [146] utilisent les outils Petshop [113] et Colored Petri-net [82] pour la vérification formelle des chemins d'interaction. MIGTool [32] aide à transformer les spécifications du scénario en chemins d'interaction pour analyser et comparer les modèles d'interaction abstraits en fonction de paramètres tels que la flexibilité ou la cohérence de l'interface utilisateur.

Ainsi, pour une représentation du dialogue comme un arbre de tâches ou de chemins d'interaction, CTTe et MIGTool apportent le support nécessaire à la conception et à l'évaluation de modèles de dialogue.

Outils de modélisation des UI

Une fois les tâches définies et validées, les designers peuvent modéliser les UI. Suivant le type d'interface, différents outils peuvent être utilisés. Les outils de conception de GUI forment la majorité des outils de conception d'interfaces, et quelques outils pour concevoir des VUI ont été proposés avec la démocratisation des assistants vocaux.

Les GUI [131, 114] dans les systèmes multimodaux sont conçues avec des langages de description d'UI et des outils associés. Pour les interfaces basées sur le Web, des outils tels que Figma [11] prennent en charge la génération de codes HTML et CSS pour des sites Web et sur mobile (iOS, Android), et divers outils tels que l'application de validation du balisage [12] complètent les éditeurs par des fonctionnalités utiles. Grundy et Yang [73] ont développé un éditeur dans lequel les designers (ou développeurs) d'UI peuvent modéliser des interfaces graphiques dans plusieurs vues de mise en page (i.e. affichage de l'interface concrète, de la structure hiérarchique et du texte en XML). L'outil fournit aussi des balises spéciales qui permettent de définir les interfaces élémentaires qui peuvent être distribuées sur différents dispositifs. Similairement, Lozano et al. [98] présentent deux outils intégrés à Eclipse pour designers UI et développeurs UI. Ces éditeurs permettent de définir des modèles d'interfaces graphiques distribuées et d'évaluer leurs capacités de distribution (i.e. distribuable, divisible) ainsi que leurs états courants. TERESA [121] permet de construire des UI à partir d'un modèle de tâche composé d'éléments graphiques et vocaux. Certains outils dépendent de la plateforme d'exécution, bien que cela puisse nuire à l'interopérabilité des éléments d'interface. Par exemple, Sketch [13] est un outil

11. `https://www.figma.com/design/`
12. `https://validator.w3.org/`
13. `https://www.sketch.com/`

permettant de créer des GUI exclusivement pour l'écosystème Apple.

Pour les VUI, l'interface ne repose pas sur la distribution spatiale de composants élémentaires, mais sur la succession dans le temps d'éléments de dialogue. Par conséquent, l'interface peut être considérée comme le fonctionnement d'une modalité et une partie du modèle de dialogue. L'outil SUEDE [86] permet de concevoir des modèles conversationnels en trois temps :

— des exemples de conversation sont définis puis assemblés sous forme de graphe de messages vocaux ;
— le modèle est parcouru par des utilisateurs à l'aide d'un magicien d'Oz ;
— les designers analysent le parcours des utilisateurs pour faire évoluer le modèle.

Cette méthode de conception se retrouve dans plusieurs outils commercialisés, tel que Voiceflow [14] et BotSociety [15].

En dehors des GUI et VUI, les méthodes de conception d'interfaces sont moins normalisées que les deux précédentes. Par conséquent, les premiers outils qui ont été proposés ne sont compatibles qu'avec des ensembles limités de dispositifs, ou nécessitent des connaissances techniques [111]. Cependant, le domaine a mûri depuis, et des outils faciles à utiliser et non techniques commencent à apparaître. Par exemple, l'outil graphique Feelix [119] propose un processus de conception d'interfaces utilisateur haptiques simple : il ne nécessite aucune connaissance technique préalable pour concevoir des interfaces utilisateur et comprend un éditeur de croquis intuitif. De nouveaux objets connectés arrivent régulièrement sur le marché, il faut donc pouvoir les intégrer facilement. Pour éviter un délai élevé entre la mise en service des objets et leur intégration, une possibilité est d'anticiper leurs usages à l'aide de prototypes. Pour cela, un outil comme *d.tool* [75] permettrait d'intégrer facilement des prototypes de dispositifs physiques dans des MIBS. Cet outil fournit un environnement pour développer, tester et analyser des dispositifs physiques sur trois couches : la couche physique (ex. les composants électroniques), la couche de communication et la couche logicielle.

Des outils ont aussi été conçus pour supporter le design d'interfaces multimodales. L'outil graphique MOSTe [140, 139] aide à spécifier tous les composants d'interaction (i.e. les composants du dispositif) dans une représentation abstraite, et fournit même un éditeur pour définir les règles de sélection dynamique des UI. Cette stratégie de sélection de l'interface basée sur des règles est également présente dans l'outil Behave [131] et dans

14. `https://www.voiceflow.com/`
15. `https://botsociety.io/`

l'environnement de création de Ghiani et al. [69].

En résumé, la tâche de conception d'interface est suffisamment outillée pour les interfaces classiques (GUI, VUI), mais il n'y a pas assez de support pour les autres types d'interfaces que l'on peut retrouver dans les MIBS.

Outils d'évaluation des modèles

Les designers peuvent évaluer les modèles de manière formelle ou expérimentale avec des utilisateurs finaux.

Les outils d'analyse formelle évaluent les modèles selon un ensemble de critères tels que que les critères pragmatiques et les critères hédoniques proposés par Wechsung [162].Par exemple, un designer peut utiliser l'outil de recherche d'erreurs de design Spin [16]. Cet outil permet de simuler des logiciels distribués et vérifier à la volée l'exactitude ("correctness") des modèles selon le langage Promela. Il a été utilisé, entre autres, pour améliorer la fiabilité d'environnements intelligents [13]. Dans cette approche, le modèle à étudier intègre des informations sur le comportement de plusieurs entités : l'environnement, les utilisateurs, les dispositifs physiques (capteurs et actionneurs) et les unités de calculs (i.e. l'application, ce qui inclut le dialogue).

Les designers peuvent aussi simuler des utilisateurs dans des environnements virtuels pour tester le système. Par exemple, le simulateur SIHMM [25] permet d'observer des utilisateurs virtuels (i.e. des agents virtuels) interagissant avec un système multimodal au travers d'appareils mobiles simulés dans un environnement modélisé. Le dialogue est représenté par une machine à états, où chaque état est une action du système vers l'utilisateur, et les transitions correspondent aux actions de l'utilisateur vers le système. Le contexte est décrit ici selon 9 concepts : la luminosité ambiante, le bruit, la présence de voix interférentes, le contexte affectif, la pression temporelle, l'importance du but, la mobilité, l'activité double (i.e. l'utilisateur réalise une autre tâche en même temps), la ressource cognitive (i.e. charge cognitive). Pour réaliser une simulation, les designers doivent modéliser les propriétés multimodales des appareils, les perturbations physiques et sociales que les entités de l'environnement peuvent générer, le scénario à suivre et le modèle d'utilisateur dictant ses préférences d'interaction.

L'avantage des méthodes par simulation est de contextualiser les expériences dans des environnements proches du réel, dans les limites du moteur physique et de la modélisation géométrique et comportementale des utilisateurs, de l'environnement et des objets. On

16. `https://spinroot.com/spin/whatispin.html`

notera que la qualité des modèles est très importante dans les domaines d'application (ex. industrie, médical) où la marge d'erreur tolérable est basse. En effet, expérimenter un système dans des environnements simulés très proches du réel permet de plus facilement valider les performances qu'aura le système dans l'environnement réel.

La dernière méthode d'évaluation des modèles consiste à faire participer des utilisateurs finaux à des échanges ou des expérimentations. MuMoWOz [6] est un outil permettant d'évaluer les choix de conception à l'aide d'expériences du Magicien d'Oz. Cependant, les designers doivent définir le contenu concret à fournir aux utilisateurs lors des expériences, et cet outil ne couvre que les systèmes multimodaux mobiles. Un designer UX peut également évaluer le système sans tout modéliser au préalable, comme dans l'outil CrossWeaver [148]. Ici, les designers d'interaction créent un storyboard en définissant l'état du système sous forme de dessins, les transitions possibles en fonction des entrées de l'utilisateur, et les retours du système aux utilisateurs. Ensuite, les expériences des utilisateurs finaux sont réalisées pour produire des journaux que les designers peuvent visualiser et rejouer.

Ainsi, les outils d'analyse formelle et d'assistance à l'expérimentation fournissent un socle suffisant pour faciliter cette l'évaluation des MIBS à l'étape de design.

4.3.2 Outils de développement

L'étape de développement consiste à définir et développer les composants nécessaires au fonctionnement du système désiré. Les développeurs peuvent aussi tester les composants réalisés pour détecter des erreurs et comportement indésirables.

L'indépendance recherchée entre les composants facilite ici le travail des développeurs. En effet, chaque développeur peut travailler sur des composants en lien avec son domaine d'expertise. Cela permet en outre de laisser au développeur le choix de l'outil de développement qui lui convient le mieux. Ce point est important car chaque outil couvre un domaine d'expertise particulier, à l'exception des outils génériques qui requièrent souvent d'installer des extensions pour ajouter de nouvelles fonctionnalités (ex. les plugins d'Eclipse ou de PyCharm) qui ne correspondent pas toujours aux besoins des développeurs. Les outils sont synthétisés dans le tableau 4.2.

OUTILS	RÔLES ASSOCIÉS	TÂCHES	
		Modélisation des composants	Implémentation des composants
SKEMMI [92]	Designer et développeur	✓	
Extensions Eclipse (ex. AsTeRICS [160])	Développeur		✓
Plateformes IoT (ex. OpenHAB (5)	Développeur		IoT
Éditeur de dialogue (ex. IMBuilder [31])	Designer et développeur UX		dialogue
Interface builder (20)	Développeur UI		Apple
Outils de développement Web (ex. Bootstrap (21)	Développeur Web		Web
Outils d'analyse (ex. SoapUI (22)	Développeur Web		Web
Éditeur d'assistants vocaux (ex. Wit.ai (24)	Développeur		VUI

TABLE 4.2 – Synthèse des outils qui supportent la phase de développement.

Outils de modélisation de composants

Le processus de définition des entrées, sorties et caractéristiques des composants avant la mise en œuvre effective permet de clarifier l'objectif de chaque composant (voir section 4.2.2), autant sur sa finalité que sur son fonctionnement. Des méthodes et outils génériques de visualisation d'architectures logicielles peuvent être utilisés. Par exemple, le modèle C4 [17] permet de représenter une architecture selon 4 niveaux de granularité qui sont :

- celui du **C**ontexte, qui correspond à la place du système logiciel (i.e. MIBS) dans son environnement d'exécution ;
- celui des **C**onteneurs, où les relations entre les applications, ou conteneurs, et l'environnement d'exécution sont définies ;
- celui des **C**omposants, qui représente le niveau de granularité qui peut être utilisé par des designers et développeurs à cette étape du développement ;
- celui du **C**ode, qui n'est pas l'implémentation concrète de chaque composant, mais une description (ex. UML) de ces éléments (ex. les classes, les interfaces, les fonctions).

Similairement, les designers de systèmes multimodaux peuvent utiliser SKEMMI [92] pour décrire les composants avec une représentation simplifiée de leurs fonctionnalités, tandis que les développeurs peuvent travailler sur ces composants à leur niveau spécifique. La possibilité de travailler sur les mêmes composants sur des vues différentes pour les designers de systèmes et les développeurs est bénéfique pour la coopération, mais, à notre connaissance, cette fonctionnalité n'est présente que dans cet outil logiciel.

17. le modèle C4`https://c4model.com/` est implémenté dans l'outil de modélisation ArchiMate `https://www.archimatetool.com/`

Ainsi, ArchiMate et SKEMMI facilitent la compartimentation des différentes technologies utilisées par les MIBS à plusieurs niveaux de granularité. La tâche de modélisation de composants est donc suffisamment outillée.

Outils de développement de composants

Pour développer les composants logiciels, les approches existantes proposent souvent le même éditeur et exigent de suivre des modèles de code pour correspondre au modèle et aux fonctionnalités de leurs composants. Dans SIAM-DP [114], ils proposent des plugins pour Eclipse afin d'aider les développeurs à générer des modèles de code, à visualiser le flux de dialogue et à prévisualiser l'interface graphique. De manière identique, la boîte à outils AsTeRICS [18] fournit également des modèles avec des extensions Eclipse pour mettre en œuvre des composants compatibles.

Ensuite, c'est aux développeurs de remplir les modèles de composants avec le code nécessaire, en utilisant les API de services et les SDK. Pour d'autres approches (ex. dans MIBO [125]), des plateformes tel que OpenHAB [19] sont utilisées pour interagir avec les objets connectés.

Avant d'implémenter les composants de dialogue, les développeurs peuvent transformer le modèle en un dialogue lisible par une machine. Pour ce faire, les développeurs ont accès à divers éditeurs de systèmes interactifs. IMBuilder [31], FSM translator [44] et MyUI editor [123] sont quelques-uns de ces outils qui permettent l'implémentation du dialogue comme une machine à états finis (FSM). Petshop [113] fournit le même service, mais avec une représentation en réseau de Petri.

La méthode de développement des UI est souvent liée à l'approche utilisée pour modéliser les interfaces.

Par exemple, un développeur peut utiliser l'outil Apple Interface Builder [20] pour intégrer les interfaces conçus sur l'outil Sketch d'Apple.

De même, les développeurs Web favoriseront des outils comme Bootstrap [21] pour intégrer les fichiers CSS et HTML générés précédemment dans leurs programmes. Chi et al. [49] proposent l'outil Weave pour développer des interfaces WIMP (Window, Icon, Menu, Pointing device [61]) distribuées sur des objets portés (ex. smartphones, montres intelligentes). Cet éditeur permet de programmer des interfaces Web indépendamment

18. AsTeRICS[160] est disponible sur le site `https://www.asterics.eu/`.
19. `https://www.openhab.org/`
20. `https://developer.apple.com/xcode/interface-builder/`
21. `https://getbootstrap.com/`

des spécificités des objets, puis de tester ces interfaces sur des objets réels ou émulés. Les interfaces Web peuvent être testée par la suite avec des outils d'analyse automatique tels que SoapUI [22] ou postman [23].

Les outils pour concevoir les assistants vocaux (ex. Wit.ai [24], assistant Watson [25]) peuvent être utilisés pour concevoir des VUI [64]. Ces outils suivent à peu près la même méthode : un développeur entraîne un modèle de langage avec un corpus de phrases annotées. L'avantage de cette méthode est de faciliter l'extension du vocabulaire connu. Cela permet au système de détecter un plus large panel d'intentions, et cela rend possiblement le système plus efficace à détecter l'intention dans l'énoncé de l'utilisateur.

Ainsi, pour chaque technologie à intégrer, les développeurs ont le choix entre des outils spécifiques à ces technologies et des éditeurs génériques améliorés par des plugins.

4.3.3 Outils d'intégration

À ce stade, l'équipe qui crée le MIBS assemble les composants développés pour créer des prototypes. Comme nous avons vu précédemment, cet assemblage peut être réalisé statiquement ou dynamiquement à l'exécution. Pour les deux approches, des outils dédiés permettent d'assembler les composants plus rapidement et simplement. La plupart de ces outils adoptent une représentation graphique des composants, et les composants peuvent être liés les uns aux autres par leurs interfaces. Les outils sont synthétisés dans le tableau 4.3.

OUTILS	RÔLES ASSOCIÉS	TÂCHES		
		Assemblage statique des composants	Assemblage dynamique des composants	Évaluation de prototypes
Éditeurs de chaînes d'interaction statique ([60, 30])	Designer UX	✓		
Éditeurs de chaînes d'interaction statique (ACS(27), [92])	Designer UX et développeur	✓		
Éditeurs de chaînes d'interaction dynamique (ex. hciλ2 [145])	développeur		✓	
Outils de collecte de données (ex. EagleView [34])	Designer, ergonome et analyste			Vérification partielle
"Simulate" [131]	Designer et ergonome			Vérification partielle

TABLE 4.3 – Synthèse des outils qui supportent la phase d'intégration.

22. `https://www.soapui.org/getting-started/rest-testing/`
23. `https://www.postman.com/`
24. `https://wit.ai/`
25. `https://cloud.ibm.com/docs/watson-assistant`

Outils d'intégration statique

ICON [60] est un éditeur graphique qui aide à créer des chaînes d'interaction. Cependant, les interfaces des composants sont représentées par les variables qu'ils utilisent en entrée, et qu'il fournissent en sortie. Cette représentation similaire à un langage de programmation peut être difficile à maîtriser rapidement. ICARE [30] améliore ce concept et permet la création sans besoin de connaissances approfondies grâce à la représentation du paradigme CARE [54] présenté dans le chapitre 1, ce qui est plus intuitif que de travailler avec des variables telles que les positions x ou y des curseurs. ACS [26] (et sa version web WebACS [27]) est un autre outil graphique pour le développement d'applications d'assistance similaire à SKEMMI [92]. Il permet de visualiser la même chaîne d'interaction du point de vue du designer ou du développeur. De plus, ACS fournit un éditeur GUI basé sur le composant instancié.

Les outils d'intégration statique proposent donc pour la plupart des méthodes graphique d'assemblage des composants, ce qui facilite cette méthode d'intégration.

Outils d'intégration dynamique

L'outil de prototypage de [143] diffère des outils précédents. Premièrement, il permet aux développeurs de modifier la composition des composants en cours d'exécution. Deuxièmement, il intègre les attributs des composants dans leur représentation graphique, ce qui facilite le processus de configuration. Similairement, l'atelier de construction de systèmes de Shen et al. [145] permet comme les outils précédents de définir des chaînes d'interaction avec un outil de débogage. Cependant, son architecture de type publish-subscribe permet l'association complexe et dynamique de composants.

Outils d'évaluation de prototypes

L'équipe d'évaluation peut tester l'utilisabilité des prototypes développés (i.e. version du système) avant de mettre en circulation une version du système. Pour cela, les prototypes peuvent être évalués par simulation ou par expérience utilisateur (voir section 4.2.3). On notera aussi que les caractéristiques des différents outils d'évaluation formelle disponibles doivent être comparées (comme dans [67]) avant de sélectionner un outil : tout changement a posteriori de ce genre d'outil est coûteux [68].

26. `https://www.asterics.eu/get-started/Overview.html#acs`
27. `https://www.asterics.eu/manuals/WebACS/`

L'équipe d'évaluation s'appuie sur leur expérience personnelle pour l'évaluation de la conformité aux directives. Ils peuvent néanmoins utiliser des outils d'analyse de données multimédias. EXCITE [101] et EagleView [34] sont deux outils de visualisation de données qui permet à un analyste d'étudier le comportement des utilisateurs à partir de données captées et d'enregistrements vidéo. Ils fournissent des représentations graphiques d'événements spatialisés et temporalisés (i.e. timeline et visualisation de proxémiques sur les vidéos). En parallèle, les designers peuvent observer les utilisateurs et recueillir leurs commentaires à l'aide de questionnaires tels que le SUS [28] ou le MMQQ [162]). Les outils de déploiement et d'exécution (voir sections 4.3.4 et 4.3.5) peuvent également être utilisés dans l'étape d'intégration lors d'expériences sur le terrain ou en laboratoire.

En comparaison, les outils d'évaluation par simulation sont relativement limités. Par exemple, "Simulate" [131] est un outil qui permet d'évaluer des prototypes de MIBS entièrement développés. Avec "Simulate", un designer peut simuler le comportement du système avec différents ensembles et emplacements d'objets et de personnes simulés. Les objets et les personnes sont représentés sous forme d'icônes, tandis que l'environnement est représenté par de multiples zones schématiques. Cette représentation est facile à créer mais est loin de la condition réelle.

En résumé, les outils d'intégration statiques et dynamiques supportent l'assemblage des composants par des méthode graphiques simples. Cependant, les outils d'évaluation de prototype ne permettent pas de pleinement supporter l'évaluation des MIBS face à la diversité des environnements. Les outils d'aide aux expérimentations utilisateurs se limitent à de la supervision, et les méthodes par simulation se limitent à des représentations schématiques d'environnement, ce qui ne permet pas d'obtenir le comportement du système dans un cadre réaliste.

4.3.4 Outils de déploiement

Le prototype final est envoyé à l'administrateur pour le déploiement. Certaines méthodes d'installation de MIBS présentent l'étape de déploiement comme étant aussi simple que le démarrage d'un logiciel. Un exemple est l'outil ARE [29] qui fournit un menu simple pour gérer l'exécution du système. Même s'il existe des systèmes qui sélectionnent dynamiquement leurs dispositifs d'interaction, la plupart des approches nécessitent toujours un administrateur pour placer le matériel, démarrer et configurer le système interactif et les

28. `https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html`
29. `https://www.asterics.eu/get-started/Overview.html#are`

dispositifs. Pour ce faire, les administrateurs peuvent être assistés dans leurs tâches. Nous présentons par la suite les outils permettant de configurer les MIBS, et nous discutons sur le manque d'outil pour faciliter le placement des objets et la mise en place des pilotes d'objets connectés. Les outils sont synthétisés dans le tableau 4.4.

OUTILS	RÔLES ASSOCIÉS	TÂCHES		
		Configuration de MIBS	Placement d'objets	Configuration d'objets
Outils graphiques pour la configuration et le test d'interactions multimodales ([103, 30, 100, 126])	Administrateur	Assemblage de composants logiciels		
Outils graphiques pour la configuration et le test des interactions spatialisées ([24, 131, 59, 101, 69])	Administrateur	visualisation spatiale de la configuration	Positionnement dans des modèles simplifiés des environnements d'exécution	
Outil graphique pour guider l'installation d'objets connectés ([66])	Administrateur			Installation et configuration d'objets connectés

TABLE 4.4 – Synthèse des outils qui supportent la phase de déploiement.

Outils de configuration de MIBS

La configuration de MIBS consiste à sélectionner et configurer les applications à installer. L'éditeur graphique dans [103] permet de décrire par programmation visuelle comment réaliser un service. Pour cela, l'utilisateur peut intégrer les objets, ressources multimédia (ex. playlist de musiques) et conditions (ex. température, heure) nécessaires à la contextualisation du service.

Les outils d'assemblage présentés dans l'étape d'intégration (par exemple, ICON [60]) peuvent également être utilisés ici. Cependant, les utilisateurs doivent avoir des connaissances sur la multimodalité et la représentation des données pour pouvoir les utiliser. De plus, ils doivent encore savoir comment installer les dispositifs dans l'environnement, car aucune information spatiale n'est fournie. L'outil de personnalisation dynamique dans [100] fournit un outil moins technique que ICON pour distribuer différentes parties de l'interface entre les appareils. Les utilisateurs peuvent également modifier la distribution de l'interface tout en testant le système. Cependant, les techniques d'interaction ne peuvent pas être sélectionnées indépendamment des objets disponibles. Avec MIBO IDE [126], les techniques d'interaction peuvent être assemblées à partir de composants faciles à comprendre, sans compétences en programmation. Ici, les représentations des objets que l'utilisateur manipule se limitent aux modalités que ces objets fournissent, et les différentes configurations sont comparées pour trouver d'éventuels conflits. De plus, cet outil affiche les événements générés par le système en cours d'exécution.

Dans leur approche, Biehl & Bailey [24] fournissent un outil permettant d'associer des applications à des écrans et des tablettes positionnés en utilisant une vue de dessus de la salle modélisée. Ils démontrent qu'avec leur représentation spatiale, les utilisateurs ont une meilleure compréhension et de meilleures performances qu'avec un outil plus simple sans information spatiale. Cependant, cet outil est limité au processus de configuration. Pour aider les ergonomes à configurer et à évaluer le MIBS, Pruvost [131] propose trois outils : l'éditeur d'ontologie "Describe" pour modéliser l'environnement, l'éditeur de règles "Behave" pour définir le comportement du système, et l'outil de simulation "Simulate" présenté précédemment. Son approche d'évaluation par simulation permet d'évaluer un plus grand panel de configuration à un faible coût de développement. Néanmoins, cette approche nécessite des connaissances sur les ontologies et les règles logiques pour configurer les MIBS. Similairement, Di Mauro propose avec le framework PHASER [58] (Chapitre 2.2.5) une suite d'outils pour configurer et évaluer les MIBS en 3 étapes. Tout d'abord, un outil permet de choisir et positionner des représentations 2D de dispositifs physiques et d'éléments de décor dans un modèle 2D de l'environnement réel. Ensuite, un outil graphique permet de paramétrer chaque nœud. Enfin, un outil permet de tester la configuration en simulant dans une modélisation 3D de l'environnement (réalisée à partir de la configuration réalisée en 2D) des agents autonomes qui se déplacent et interagissent avec le MIBS. Les interactions se limitent toutefois à des échanges de messages entre les agents autonomes et les nœuds pour simuler des interfaces vocales.

Des travaux récents sur la notion de proxémique [46] fournissent un support partiel pour configurer et évaluer les MIBS. Par exemple, la distance et le mouvement d'une tablette par rapport à une caméra peuvent être utilisés pour définir le comportement de ces objets. Le "Proximity toolkit" [102, 101] est un outil de supervision de systèmes multimodaux construit sur ce paradigme pour mieux saisir le lien entre le comportement du système et les emplacements des objets et des utilisateurs. Il suit toutes les entités (c'est-à-dire les objets, le mobilier, les utilisateurs) et fournit des informations sur leurs distances relatives, angles relatifs et vitesses relatives. Elle peut rejouer des séquences enregistrées à partir de ce qui a été mesuré par les capteurs. L'outil graphique de [69] fournit un vocabulaire plus simple pour la proxémique (par exemple "près de" ou "lorsque l'utilisateur se déplace") pour adapter l'interface distribuée aux entités proxémiques.

Par conséquent, les outils existants fournissent un certain soutien à l'étape de déploiement mais n'effectuent pas toutes les tâches identifiées dans la section 4.2.4 : les tâches de placement des appareils et de configuration de leurs programmes logiciels ne sont pas

abordées.

Méthodes d'aide au placement d'objets connectés

Nous avons vu que le positionnement des objets dans l'espace a une grande importance pour les MIBS. Pourtant, le support se limite à des recommandations et à l'expérience de l'administrateur. L'absence d'outil peut s'expliquer par la difficulté de proposer des choix d'objet et de placement qui soient adaptés à l'environnement considéré, aux réglementations en vigueur et aux besoins et préférences des utilisateurs.

Méthodes d'aide à la mise en place des pilotes des objets connectés

Les MIBS peuvent requérir les informations de placement des objets dans l'environnement physique selon les applications considérées.

Des plateformes utilisées pour gérer les objets connectés peuvent être utilisées ici. Par exemple, Frigo et al. [66] proposent un outil pour configurer de bout en bout un environnement IoT. Avec cet outil, les administrateurs manipulent depuis une interface graphique des blocs représentant les différentes entités (i.e. objets, réseaux, applications), ainsi qu'un guide d'installation. Bien que cet outil permette donc de faciliter la coopération entre les développeurs et l'administrateur, il ne propose pas de mécanismes qui permettent de réutiliser les composants développés.

4.3.5 Outils d'exécution

L'étape d'exécution finalise une itération du cycle de vie des MIBS. Nous avons présentés dans la section 4.3.3 des outils qui permettent d'exécuter le système. Toutefois, l'étape d'exécution ne se limite pas à cela. Des outils d'analyse du système peuvent être utilisés pour fournir aux développeurs et designers des points d'évolution possibles. Le système peut aussi évoluer localement en permettant aux utilisateurs et administrateurs de maintenir le système fonctionnel. Les outils sont synthétisés dans le tableau 4.5.

OUTILS	RÔLES ASSOCIÉS	TÂCHES	
		Surveillance et analyse	Assistance à l'utilisateur
Outils de logs (ex. MMWA [115])	Utilisateur final	Logs	
Outils de collecte de feedback (ex. DynEaaS [124])	Utilisateur final	Questionnaires	

Table 4.5 – Synthèse des outils qui supportent la phase d'exécution.

Outils de supervision et analyse

Une fois le système démarré, les administrateurs et les éditeurs de logiciels peuvent superviser l'usage du système pour détecter des modèles de comportement, des préférences ou des failles non détectées jusque-là. Par exemple, dans l'environnement de création MMWA [115], les designers peuvent accéder au journal des interactions des utilisateurs. Les outils de supervision en Réalité Augmentée ou en Réalité Virtuelle dans un jumeau numérique de l'environnement réel tels que présentés par Lacoche et al. [90] peuvent également être adaptés aux systèmes multimodaux. En effet, l'immersion peut aider à mieux comprendre l'état du système.

Les utilisateurs peuvent aussi être amenés à remplir des questionnaires en fonction du comportement de l'utilisateur et du contexte. La plateforme d'évaluation dynamique DynEaaS [124, 147] permet par exemple à l'équipe d'évaluation de transmettre des questionnaires aux utilisateurs à l'aide d'une interface Web. Ces questionnaires sont proposés à l'utilisateur selon des plans d'évaluation déclenchés à certains moments ou selon certaines conditions, et distribués sur les objets choisis.

De nos jours, il est courant de recueillir des données d'usage auprès des utilisateurs pour hâter le développement des correctifs avec les mises à jour du réseau, mais les exigences de confidentialité et de sécurité dépendent de l'application et du public cible. Il semble donc que la supervision du système à l'exécution soit généralement limitée à la stratégie d'évaluation du prototypage dans l'étape d'intégration, et donc le support est limité aux outils présentés en section 4.3.3.

Il existe donc plusieurs outils qui permettent de faciliter le suivi des MIBS dans l'environnement d'exécution, bien que leurs fonctionnalités puissent être limitées par rapport à la variété de données que les éditeurs de logiciels voudraient analyser.

Outils d'assistance à l'utilisateur

Nous n'avons pas identifié d'outil permettant d'assister l'utilisateur pendant qu'il interagit avec un MIBS. Cependant, il peut être intéressant d'assurer l'interopérabilité entre ces systèmes et les systèmes déjà présents pour faciliter le passage d'un service à un autre pour l'utilisateur. Par exemple, les assistants virtuels (i.e. Microsoft Cortana, Apple Siri, Amazon Alexa et Google Home) embarqués dans des enceintes connectées, les smartphones et les ordinateurs peuvent servir de méta-IHM pour contrôler les MIBS. De plus, ces systèmes IoT offrent déjà des interfaces multimodales. Le temps de développement

peut donc être réduit en utilisant les briques logicielles de ces systèmes.

Ce ne sont pour l'instant que des usages restreints à des plateformes propriétaires. Des travaux supplémentaires seraient nécessaires pour standardiser ce type d'approche.

Analyse des méthodes et outils par tâche

La plupart des tâches du cycle de vie des MIBS sont correctement outillées. Toutefois, les méthodes pour assurer le processus d'adaptation des MIBS aux environnements d'exécution ne sont pas suffisantes, que l'adaptation soit automatique ou configurable. Pour les MIBS adaptant automatiquement leurs interfaces au contexte, aucun outil ne permet de vérifier qu'ils fonctionnent en toute situation. Pour les MIBS adaptant leur interface selon des configurations manuelles, le support est limité. En effet, les méthodes d'évaluation par expérimentations utilisateurs dans des environnements de tests et les méthodes par simulation ne permettent pas de s'assurer que le système fonctionnera correctement dans chaque environnement d'exécution.

4.4 Discussion

Au travers des analyses réalisées pour chaque tâche, nous pouvons constater que certains aspects ne sont pas encore suffisamment pris en compte. Le résultat de ces analyses est synthétisé dans le tableau 4.6. Nous discutons ci-dessous des tâches qui pourraient être mieux prises en charge.

4.4.1 Modélisation des UI

Les représentations des dispositifs, qui définissent leurs capacités d'interaction, et leurs composants correspondants sont correctement gérés lorsqu'ils fournissent des modalités graphiques ou vocales, mais les autres modalités, qui sont nombreuses dans les systèmes IoT, sont plutôt considérées comme des fournisseurs ou des consommateurs d'événements sans autre support. Des travaux tels que l'ontologie DOG [28] pourraient être utilisés comme base pour fournir des représentations standard des appareils connectés, et ainsi améliorer le support des outils.

4.4.2 Détection des problèmes d'utilisabilité

Les tests de systèmes en étape d'intégration manquent encore d'un outil permettant d'évaluer un système multimodal dans une simulation réaliste avec de vrais utilisateurs. En effet, les expériences sur le terrain et en laboratoire sont difficiles et coûteuses à réaliser, mais elles garantissent des résultats proches de la vérité du terrain. La simulation est une bonne stratégie pour tester rapidement plusieurs chaînes de configuration, mais les outils existants reposent sur des utilisateurs modélisés, de sorte que les résultats sont susceptibles d'être biaisés.

4.4.3 Positionnement et configuration des objets connectés

Les outils de déploiement ne prennent en compte que la configuration du logiciel, il leur manque un moyen d'aider l'administrateur à placer les dispositifs d'interaction sans s'appuyer sur des essais et des erreurs. Peut-être qu'un système compagnon tel que celui présenté par Bercher et al. [21] peut aider les administrateurs pendant l'installation : des repères visuels localisés dans un environnement simulé pourraient permettre de mieux comprendre le positionnement à effectuer. En particulier, la visualisation des zones d'effet des objets (ex. le champ de vue des caméras) et des contraintes comme le bruit ou la luminosité faciliterait grandement la configuration des MIBS. Cela signifie également que nous avons besoin d'une méthode pour décrire au préalable l'environnement réel et sa capacité à accueillir les applications souhaitées. La capture de l'environnement et la configuration du dispositif en RA proposées par Soedji et al. [149] pourraient résoudre ce problème.

4.4.4 Assistance à l'utilisateur

À l'exécution, les utilisateurs n'ont souvent à leur disposition que des manuels d'utilisation. Or, ces manuels ne permettent pas d'anticiper les différences de comportement qu'aura un MIBS en fonction de l'environnement d'exécution. Les méta-IHM et les technologies compagnon permettent de réaliser ce support, mais ces méthodes ne sont pas assez outillées. Il n'y a donc pas d'outil qui peut s'adapter à l'environnement pour fournir une assistance personnalisée à l'utilisateur.

4.4.5 Analyse de l'expérience utilisateur

Enfin, il n'existe que quelques supports pour la supervision des systèmes multimodaux, et l'aide apportée se limite à la visualisation de données brutes ou sur un plan 2D. Or cette fonctionnalité de supervision peut être utile pour les applications multimodales nécessitant un accès Internet. En effet, certaines applications multimodales peuvent utiliser des services en ligne ou des systèmes IoT accessibles de partout. Ainsi, les outils de supervision en ligne pourraient être utilisés pour la maintenance en cas d'interruption de service ou de dysfonctionnement de l'IoT, de plus pour la détection des défauts de l'application. La solution Smart Space de Softengi [30], qui facilite la supervision de l'IoT grâce à la Réalité Augmentée, peut être adaptée au paradigme multimodal.

4.5 Synthèse

TABLE 4.6 – Résumé des tâches du cycle de vie des MIBS qui ne sont pas bien prises en charge par les outils logiciels. Les couleurs verte, orange et rouge représentent respectivement les tâches qui sont bien supportées, qui pourraient être améliorées ou qui ne sont pas supportées.

Design	Développent	Intégration	Déploiement	Exécution
Analyse du contexte	Modélisation des composants	Assemblage des composants	Positionnement des objets	Analyse de l'expérience utilisateur
Modélisation du dialogue	Implémentation des composants	Détection de problèmes d'utilisabilité	Mise en place des pilotes des objets connectés	
Modélisation des UI			Configuration du MIBS	
Évaluation du design				

Notre identification des tâches et des outils logiciels associés au SDLC des systèmes multimodaux utilisant des dispositifs connectés comme supports interactifs conduit à une proposition des outils qui pourraient être conçus pour combler le vide dans le processus identifié. Ceci est synthétisé dans le tableau 4.6.

La conception d'interfaces utilisateur autres que graphiques ou vocales nécessite des outils plus normalisés et plus faciles à utiliser que ceux qui existent déjà. L'étape d'intégration peut bénéficier de meilleurs outils de simulation pour évaluer des systèmes dépendant

30. https://softengi.com/projects/public-sector/smart-space-solution-for-the-softengi-office/

de l'environnement sans avoir à recourir à des expériences coûteuses en laboratoire ou sur le terrain. De plus, cela permettrait de tester les MIBS dans une plus grande variété d'environnement à faible coût. Le déploiement n'est pas non plus bien exploré, et améliorer les outils d'installation pourraient faciliter cette partie du processus. Un travail pourrait être réalisé sur la création d'outils permettant d'identifier et de placer facilement les dispositifs connectés dans l'environnement. Des outils d'aide à la supervision plus intuitifs pourraient également être bénéfiques pour les systèmes multimodaux où les dispositifs d'interaction sont distribués.

Outre l'identification des manques, les rôles considérés pour chaque outil ont été identifiés. Cela permet d'associer pour chaque acteur les tâches dans lesquelles il peut intervenir.

Le processus identifié et notre analyse pourraient guider la création d'un cadre et de nouveaux outils pour les futurs chercheurs dans le domaine de l'interaction multimodale qui souhaitent utiliser les appareils connectés comme supports d'interaction.

En particulier, l'absence de méthodes ou d'outils de configuration et évaluation lors du déploiement est un verrou important à débloquer pour répondre à notre problématique de recherche. La simulation des environnements d'exécution avec de vrais utilisateurs peut apporter la qualité de configuration des expériences sur le terrain en gardant la facilité d'utilisation de l'approche par simulation, et la réalité virtuelle (RV) peut être une technologie prometteuse dans cet aspect. C'est pour cette raison que nous proposons dans le chapitre suivant une méthodologie utilisant la RV pour faciliter la configuration d'un MIBS en fonction des environnements d'exécution.

Chapitre 5

Proposition : Méthode de configuration et évaluation de MIBS

Comme nous avons vu précédemment, il est possible d'adapter le comportement des MIBS automatiquement à partir d'un raisonnement sur les informations contextuelles. Toutefois, cela implique d'anticiper le plus de situations possibles sans avoir la certitude que cela répondra réellement aux attentes de chaque utilisateur. Pour reprendre l'exemple de l'application de guidage, il n'est pas possible d'anticiper toutes les structures de bâtiments, et l'utilisateur pourrait préférer un chemin qui passe en extérieur plutôt que le chemin le plus court. Pour dépasser les limites de l'identification *a priori* des usages attendus, Coutaz et Crowley [51] proposent de faire intervenir les utilisateurs dans cette phase d'adaptation : "il convient d'inventer de nouveaux langages permettant à l'utilisateur non-expert de commander ("programmer") pour qu'il puisse fabriquer ses choses à lui, s'approprier l'espace ambiant, inventer les usages que les concepteurs ne sauraient prévoir". Il est donc important de fournir une méthode de configuration simple des MIBS lors du déploiement [33]. Nous définissons les configurations par les techniques d'interaction utilisées, les dispositifs connectés associés et leur emplacement dans un environnement.

De plus, un point clé de l'informatique ambiante est d'intégrer dans nos environnements un nombre extensible de capteurs et actionneurs pouvant être utilisé dans diverses applications [63]. Cela se traduit ici par la possibilité d'ajouter de nouveaux objets connectés *prêt-à-l'emploi* (i.e. aucun processus d'installation n'est nécessaire) à l'environnement et d'associer les objets disponibles aux MIBS.

Enfin, la fiabilité d'un système sensible au contexte ne peut être évaluée qu'une fois le système placé dans les conditions identiques à l'environnement d'exécution [11]. Pour assurer l'adaptation d'un MIBS à un environnement, il faut donc permettre de réaliser des tests réalistes.

Les administrateurs doivent généralement tester diverses configurations, en recourant à des outils d'installation [155] et à des documentations [107] pour les guider, jusqu'à ce qu'ils trouvent une configuration satisfaisante. Ce processus nécessite généralement des connaissances techniques, prend du temps, et le retour d'information provenant des systèmes basés sur l'IoT peuvent être difficile à comprendre pour un administrateur. Pour faciliter le travail de configuration et de test des applications interactives dans des environnements ubiquitaires, des travaux récents ont étudié l'utilisation d'outils de Réalité Virtuelle (RV) et de Réalité Augmentée (RA) [153, 89]. De cette manière, les utilisateurs peuvent avoir une meilleure compréhension spatiale du processus de configuration, mais il n'existe pas de support pour le déploiement d'interactions plus élaborées et plus longues dans le temps, comme l'on peut retrouver avec les MIBS.

C'est pourquoi nous proposons une nouvelle méthodologie basée sur la RV et un outil permettant aux administrateurs de configurer et d'évaluer les MIBS dans des environnements modélisés réalistes avant leur déploiement. Les administrateurs sont immergés dans une modélisation 3D [72] de l'environnement cible. Cette modélisation comprend des informations sur la géométrie des environnements cibles ainsi que des informations sur les caractéristiques et les comportements des appareils connectés disponibles. Ainsi, ils peuvent observer et se déplacer pendant la configuration et le processus de test de manière similaire à la réalité.

Un outil associé facilite le processus de déploiement des MIBS pour les administrateurs. Premièrement, il permet aux administrateurs de sélectionner les services, les techniques d'interaction et de manipuler les dispositifs simulés simplement et sans effort. Deuxièmement, il est possible d'évaluer dans la RV les services configurés pour fournir une expérience similaire à celle de la réalité, sans avoir besoin d'un dispositif réel ou d'un accès à l'environnement réel quand celui-ci n'est pas accessible. Cela permet de configurer facilement des MIBS même si l'environnement réel est occupé, en cours de construction ou loin géographiquement. Cela explique notre choix de la RV plutôt que de la RA. Enfin, les administrateurs seront en mesure de fournir un retour d'information aux producteurs de MIBS (par exemple, les développeurs ou les designers) ou de générer des données facilitant l'installation des services configurés dans un environnement réel.

Dans la section 5.1, nous présentons une revue des travaux connexes. Puis, dans la section 5.2, nous présentons la configuration et l'évaluation du MIBS dans MCEV, l'outil de RV que nous avons créé. Enfin, nous concluons et présentons les perspectives d'évolution de notre méthodologie.

Soumission

Ce chapitre a fait l'objet d'une soumission à la conférence internationale HUCAPP 2023.

5.1 Étude des outils immersifs pour la configuration et le test des services MR et IoT

Le déploiement d'un système ubiquitaire dans un environnement est une étape importante du cycle de vie de ce système, et dépend généralement de son administrateur. Pour les aider, la plupart des approches fournissent des outils pour faciliter les processus de configuration et de test des systèmes. Toutefois, nous avons vu dans le chapitre 4.3.4 que les outils existants n'offrent qu'un support limité. La plupart des outils de déploiement ne prennent pas en charge les interactions spatialisées, et ceux qui le font sont limités. En effet, les utilisateurs n'ont accès avec ces outils qu'à des représentations 2D des environnements ou à des informations textuelles pendant le processus de configuration. Cependant, disposer d'une représentation 3D peut être bénéfique à ce processus car certaines informations ne peuvent pas être visualisées en 2D, ou conceptualisées à partir d'informations textuelles uniquement. Par exemple, lors de la configuration d'une interaction avec des caméras, il faut faire attention à leurs angles de vue horizontaux et verticaux, ainsi qu'aux distances minimales et maximales requises pour un fonctionnement correct. Pour répondre à ce besoin de représentation 3D, nous étudions dans la suite de cette section des travaux qui exploitent le potentiel immersif de la RV et de la RA pour faciliter le déploiement de services IoT.

Le développement des technologies RA et RV a apporté de nouvelles possibilités dans les représentations 3D, l'immersion, et plus particulièrement dans la configuration et l'interaction avec les appareils connectés. En effet, les MIBS peuvent être configurés et testés à l'échelle 1 dans un environnement réel (RA) ou dans son jumeau numérique (RV). Dans notre cas, la modélisation 3D d'un environnement (proche d'un jumeau numérique) comprend des informations sur la géométrie de l'environnement ainsi que des informations sur les appareils connectés. Par exemple, ExProtoVAR [127] fournit une méthodologie pour prototyper à partir d'images panoramiques 360 un système interactif en réalité mixte (RM) interagissant avec des objets connectés. La prise en charge d'interfaces spatialisées,

ainsi que les fonctionnalités d'annotation et d'enregistrement, facilitent la création et le partage de nouveaux designs par les producteurs du système. Suzuki et al. [153] ont proposé l'outil de RA ReallifeEngine où les objets peuvent être reliés par des interfaces de programmation visuelle. Ainsi, il est possible de créer des scénarios d'automatisation sans connaissances techniques tout en considérant l'objet dans l'environnement réel. Le pipeline introduit dans [90] fournit également un support pour la gestion des objets connectés. Il inclut la création du jumeau numérique de l'environnement avec la RA et la RV. De plus, les jumeaux numériques des objets peuvent simuler des comportements réalistes, ainsi le système peut être testé en RV indépendamment de l'environnement réel. De plus, Lacoche et al. [89] ont présenté un outil de création de RV qui peut aider les utilisateurs non techniques à adapter les éléments de contenu interactifs dans les applications de RA en fonction des informations recueillies par les objets connectés. Une autre utilisation de la RV pour configurer des systèmes est la mise en service virtuelle (VC) pour l'industrie 4.0 [94]. Le VC consiste en l'observation et la validation du comportement des systèmes d'automatisation par la simulation matérielle et logicielle, et il pourrait être étendu aux systèmes interactifs. Par exemple, Metzner et al. [106] présentent une méthode permettant d'intégrer un opérateur humain pour tester des automates programmables (PLC). Bien qu'elle soit destinée aux producteurs d'automates et à leurs partenaires commerciaux en cours de production, elle fournit une méthode de test des objets intuitive et réaliste. La RA et la RV sont des technologies puissantes qui permettent de convertir la représentation logique des systèmes interactifs en informations intuitives et faciles à visualiser. Cependant, les outils existants sont limités au prototypage de services de Réalité Mixte ou à l'automatisation de l'IoT, ce qui ne permet pas de gérer des systèmes interactifs multimodaux plus complexes et centrés sur l'utilisateur. De plus, l'environnement réel peut être indisponible si, par exemple, il est en cours de construction ou de rénovation, trop éloigné ou déjà utilisé. La RA n'est donc pas adaptée dans ces situations.

Les outils immersifs pour la configuration et le test des services MR et IoT

Les outils immersifs pour la configuration et le test ont pour avantage de recréer un cadre d'expérimentation similaire à celui des expérimentations dans l'environnement réel tout en gardant la simplicité des méthodes d'évaluation par simulation. Néanmoins, les outils existants sont destinées aux services MR et l'automatisation de l'IoT, et ne sont donc pas adapté aux systèmes interactifs multimodaux plus complexes et centrés sur l'utilisateur. De plus, les solutions utilisant la RA ont un usage plus restreint que les solutions utilisant la RV car l'accès à l'environnement physique est nécessaire en RA, ce qui n'est pas toujours possible.

5.2 Proposition : Méthodologie MCEV

Il n'existe actuellement aucun outil qui prenne entièrement en charge la configuration et l'évaluation des MIBS.

Nous proposons donc la méthodologie MCEV (**M**IBS **C**onfiguration and **E**évaluation in **V**R) : une méthode et un outil de configuration et d'évaluation de MIBS en RV pour aider à configurer et à tester efficacement ces systèmes pendant le processus d'installation dans divers environnements.

Notre méthodologie permet de définir une configuration d'un MIBS (i.e. l'environnement, les dispositifs, les services et les techniques d'interaction), ainsi que de réaliser des simulations de MIBS pour des tests immersifs. De plus, plusieurs fonctionnalités de retour d'information sont fournies pour partager les configurations et les résultats d'évaluation.

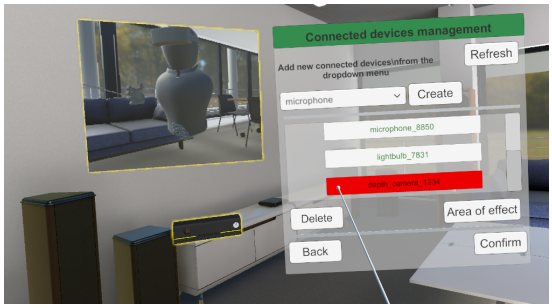


FIGURE 5.1 – Menu de gestion des objets

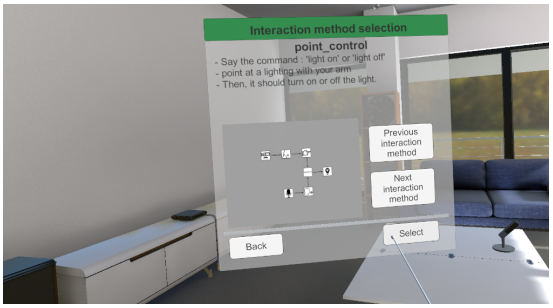


FIGURE 5.2 – Menu pour sélectionner une technique d'interaction

Dans l'outil, la plupart des fonctions de configuration sont accessibles par des interfaces utilisateur 2D positionnées dans l'environnement 3D. Ces interfaces utilisateur et les objets sont interactifs grâce à une technique classique de sélection et de manipulation basée sur des rayons 3D. La navigation dans les environnements virtuels est facilitée par une fonctionnalité classique de téléportation consistant à pointer une position souhaitée pour s'y déplacer instantanément.

Il convient de noter que les ergonomes peuvent appliquer la méthodologie MCEV pour évaluer les MIBS pendant leur production, de manière similaire à la méthodologie de simulation de [97].

5.2.1 Sélection de l'environnement

Avec notre méthodologie, les administrateurs importent tous leurs environnements dans la base de connaissances, puis en sélectionnent un depuis l'outil. Ainsi, ils n'ont pas besoin des environnements réels potentiellement indisponibles, ce qui facilite la transition entre les différentes zones à équiper dans des bâtiments parfois grands. Ces environnements peuvent être des environnements bac à sable, comme des environnements simplistes ou des environnements de démonstration fournis par les producteurs de MIBS, ou des environnements générés à partir des jumeaux numériques de l'environnement physique cible. La création de jumeaux numériques d'objets et d'environnements pourrait être considérée comme une contrainte, mais nous pensons que ce n'est pas un obstacle de nos jours : les bâtiments modernes sont souvent décrits dans des modèles d'information du bâtiment (i.e. *Building information modeling*, ou BIM [1]), et des outils récents tels que l'outil de capture RA proposé dans [90] peuvent simplifier ce processus. Par conséquent, les administra-

1. `https://www.iso.org/fr/standard/68078.html`

teurs peuvent configurer et tester des MIBS dans des environnements virtuels multiples et éventuellement énormes sans délai.

Dans notre évaluation présentée dans le chapitre 6.1, le modèle 3D de l'environnement a été créé à partir du fichier de modélisation (BIM) du bâtiment, puis affiné par un infographiste 3D.

5.2.2 Gestion des objets connectés

Ces environnements 3D peuvent déjà comporter des appareils connectés. Cependant, pour se conformer à certaines exigences de service, les administrateurs peuvent vouloir déplacer ces objets ou en ajouter de nouveaux. Ces modifications de l'environnement dans la RV peuvent également être utiles pour anticiper les changements dans un environnement réel. Pour cette raison, les administrateurs peuvent instancier des objets simulés (par exemple, un thermomètre, un haut-parleur connecté) avec l'outil MCEV (voir Figure 5.1). Ces objets sont des jumeaux numériques d'objets réels qui comprennent des représentations 3D des objets et des scripts qui simulent le comportement des objets selon leurs fiches techniques. Par exemple, la caméra Kinect de la figure 5.1 a le même champ de vision qu'une vraie caméra Kinect.

Les objets instanciés doivent ensuite être placés dans l'environnement virtuel. Pour cela, leurs capacités interactives doivent être prises en compte. En effet, la plupart des capteurs et des actionneurs (par exemple, les capteurs de mouvement, les écrans) ont leurs propres *auras* [20], qui sont les zones dans lesquelles ils peuvent détecter ou être détectés. La position des objets dans un environnement peut influencer ces *auras*. Par exemple, les capteurs de mouvement ont des angles de vue spécifiques qui peuvent être obstrués par des piliers ou des meubles. Ces *auras* sont difficiles à visualiser dans la réalité, et le seul outil actuel qui supporte la visualisation des informations proxémiques [102] ne tient pas compte des capacités de l'objet. Par conséquent, nous proposons une méthode pour visualiser les capacités d'interaction des objets connectés afin d'aider à l'identification et au placement de ces objets. Avec l'outil MCEV, les administrateurs peuvent observer visuellement ou oralement (lorsque cela a du sens) les *auras* des objets. Cela permet de placer intuitivement les objets en considérant les spécificités de l'environnement (ex. le type et la position du mobilier, des points d'accroche, les prises électriques). Par exemple, un administrateur peut placer une caméra devant des chaises pour observer ce qu'elle peut voir avec son champ de vision, comme l'illustre la figure 5.5. Grâce à cette représentation, les administrateurs peuvent comprendre rapidement et intuitivement l'impact

de l'emplacement d'un objet sur ses performances, et trouver plus facilement un emplacement satisfaisant. Actuellement, notre outil comprend plusieurs modèles d'objets avec leurs *auras* qui correspondent aux objets réels en notre possession, mais cela peut être étendu à n'importe quel type d'objets. C'est particulièrement intéressant pour des objets indisponibles (ex. chers) ou non-existants (pour un usage prospectif).

Les objets peuvent aussi avoir des paramètres qui leurs sont spécifiques (ex. la taille des caractères pour le texte sur un écran). Ces paramètres peuvent être fixés par l'application, mais des valeurs par défaut peuvent être définies grâce aux scripts de comportement. C'est pour cela que l'outil MCEV permet de modifier ces paramètres avec des interfaces 2D attachées aux objets virtuels. L'outil intègre aussi un Monde-en-Miniature interactif [152] de l'environnement. Cela permet à l'utilisateur d'avoir une vue plus complète de l'environnement et se déplacer facilement dans de grands environnements.

5.2.3 Sélection des services, des techniques d'interaction et des objets

Pour configurer un MIBS, l'administrateur doit sélectionner pour chaque application une technique d'interaction associée et des objets qui fournissent les modalités nécessaires. Dans les approches actuelles de configuration des MIBS, les objets sont sélectionnés à partir d'une liste [131, 126], ainsi l'identification des objets souhaités à partir de leurs identifiants pourrait être difficile, en particulier dans les environnements avec un grand nombre d'objets. Les outils de configuration de l'IoT dans la RV, comme dans [90], fournissent des méthodes pour associer les objets en liant leurs modèles 3D entre eux. Cette métaphore du lien est difficile à maintenir car le nombre d'objets peut être important dans les systèmes multimodaux, et plusieurs objets peuvent être utilisés de manière redondante. Par exemple, la détection d'un geste de l'utilisateur peut être captée par plusieurs caméras, et une métaphore de lien nécessiterait de relier une à une les caméras.

Pour chaque service qu'il souhaite configurer, l'administrateur sélectionne une technique d'interaction. Comme le montre la figure 5.2, les techniques d'interaction sont décrites et sont illustrées par une représentation graphique des chaînes de composants logiciels associées. Ainsi, l'administrateur n'a pas besoin de connaissances techniques pour comprendre comment tester une configuration, mais peut obtenir des détails techniques en cliquant sur les différentes icônes représentant le composant. L'administrateur peut également déterminer rapidement quelles sont les modalités nécessaires à l'interaction en

regardant simplement les extrémités des chaînes. Ces chaînes d'interaction peuvent être développées avec l'un des outils graphiques de composition de composants pour systèmes multimodaux, tels que Squidy [88] ou SKEMMI [93].

Ensuite, pour chaque modalité requise par la technique d'interaction, l'administrateur doit sélectionner les objets à utiliser (comme dans la Figure 5.3). Chaque modalité requise peut être associée à plusieurs objets, soit à partir de la liste d'une interface 2D (voir Figure 5.4) soit directement en pointant sur les objets. La liste des objets que l'administrateur peut associer à une technique d'interaction est limitée aux objets qui peuvent fournir l'une des modalités souhaitées. L'administrateur peut faire le lien entre le nom d'un objet de la liste et son modèle 3D dans l'environnement en sélectionnant l'un des deux éléments, ce qui met en surbrillance l'autre élément. Par conséquent, l'administrateur peut sélectionner les objets de manière intuitive en les pointant directement. Cependant, l'administrateur dispose également d'une vue centralisée des associations qui simplifient la modification de la configuration et la sélection de plusieurs objets.

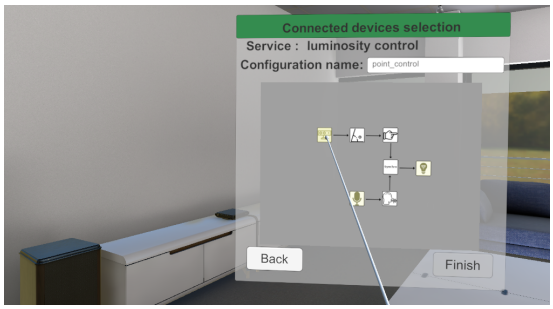


FIGURE 5.3 – Menu général pour associer des objets selon les besoins d'une chaîne d'interaction sélectionnée

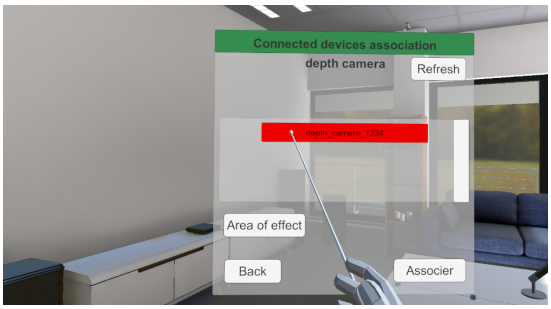


FIGURE 5.4 – Menu pour associer des objets à un besoin spécifique d'une chaîne d'interaction

5.2.4 Évaluation du MIBS

Une fois le MIBS configuré, il est important de le tester. En effet, l'expérience utilisateur offerte par certains services ou certaines techniques d'interaction peut varier en fonction de la topologie des environnements et du choix des objets et de leur emplacement. Par exemple, l'utilisation de certaines techniques d'interaction basées sur les gestes peut être fastidieuse pour passer rapidement d'une chaîne de télévision à une autre. Par conséquent, notre outil prend en charge les tests immersifs de MIBS dans la RV pour l'évaluation de la configuration.

FIGURE 5.5 – l'*auras* de la caméra (Kinect 2).

Lorsqu'un administrateur souhaite évaluer ses configurations, l'outil s'appuie sur un moteur d'exécution pour exécuter les services configurés. Le moteur d'exécution s'inspire des architectures à base de composants de l'état de l'art telles que l'architecture AM4I [4] pour leur approche modulaire (une meilleure séparation entre les différentes fonctionnalités de MIBS est recommandée [14]), leur facilité d'utilisation et leur prise en compte du contexte.

Ensuite, les capteurs simulés génèrent des informations et les actionneurs agissent en fonction des commandes reçues du système. Par conséquent, l'administrateur peut interagir dans la RV comme il le ferait dans les expériences sur le terrain.

Lorsque l'administrateur rencontre des problèmes pendant le test, l'outil fournit une console (voir Figure 5.6) dans laquelle l'administrateur peut trouver les événements reçus des objets, des services et des composants de traitement des techniques d'interaction. Elle permet d'avoir un historique du dialogue et ainsi de mieux comprendre l'origine d'un problème. Cette console est attachée à la main non dominante, et peut être repliée.

5.2.5 Révision collaborative

Bien que nous ne fournissions pas un environnement virtuel collaboratif en temps réel, nous visons à aider à améliorer les étapes précédentes (c'est-à-dire la conception et le développement) et suivantes (c'est-à-dire le déploiement) du cycle de vie du MIBS. En

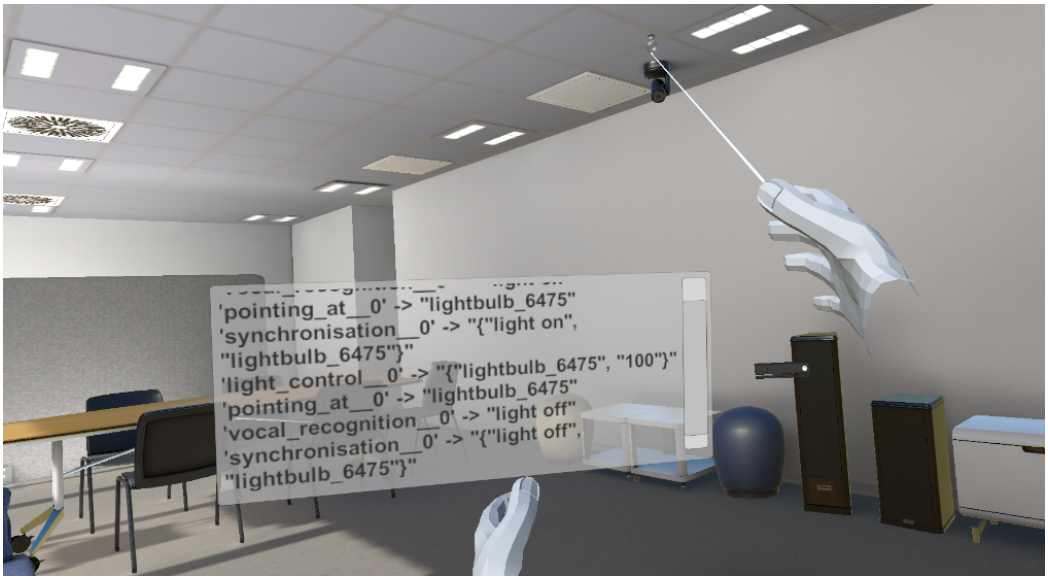


FIGURE 5.6 – La console attachée à la main. Les événements du MIBS sont affichés ici.

particulier, nous voulons soutenir deux types de collaboration :

— Le retour d'information aux producteurs de MIBS. Les administrateurs peuvent vouloir partager avec les développeurs et les designers MIBS certains problèmes rencontrés afin de les résoudre. Cependant, les problèmes peuvent provenir d'erreurs de mise en œuvre, des limites des capacités des appareils, de techniques d'interaction inadéquates ou de contraintes spatiales. Ainsi, il pourrait être difficile de contextualiser les problèmes rencontrées.
— La transition des MIBS de la RV à l'environnement réel. La configuration réalisée en RV sert de guide d'installation du système dans l'environnement réel, mais porter un équipement de RV pendant le déploiement serait peu pratique. Le passage d'une configuration dans la RV à l'environnement réel doit être abordé.

Dans leur revue de littérature [136], Rodrigues et Silva identifient plusieurs catégories de méthodes permettant d'aider à l'utilisation de systèmes interactifs. On retrouve par exemple l'usage de textes, d'images, et de vidéos expliquant l'utilité d'un élément d'interface ou de l'usage de cet élément. Nous nous sommes inspirés de ces approches pour proposer trois fonctionnalités de révision collaborative : l'accès et la création de notes contextualisées, l'enregistrement et relecture d'actions d'un utilisateur de l'outil MCEV, la prise de capture d'images. Ces fonctionnalités sont décrites par la suite.

Gestion de notes contextualisées

Les administrateurs peuvent rédiger des notes colocalisées et horodatées tout au long du processus MCEV : ils peuvent les rédiger à l'étape de la configuration pour décrire la configuration globale, ou pendant un test pour signaler une situation spécifique. Un exemple de note se trouve en figure 5.7.

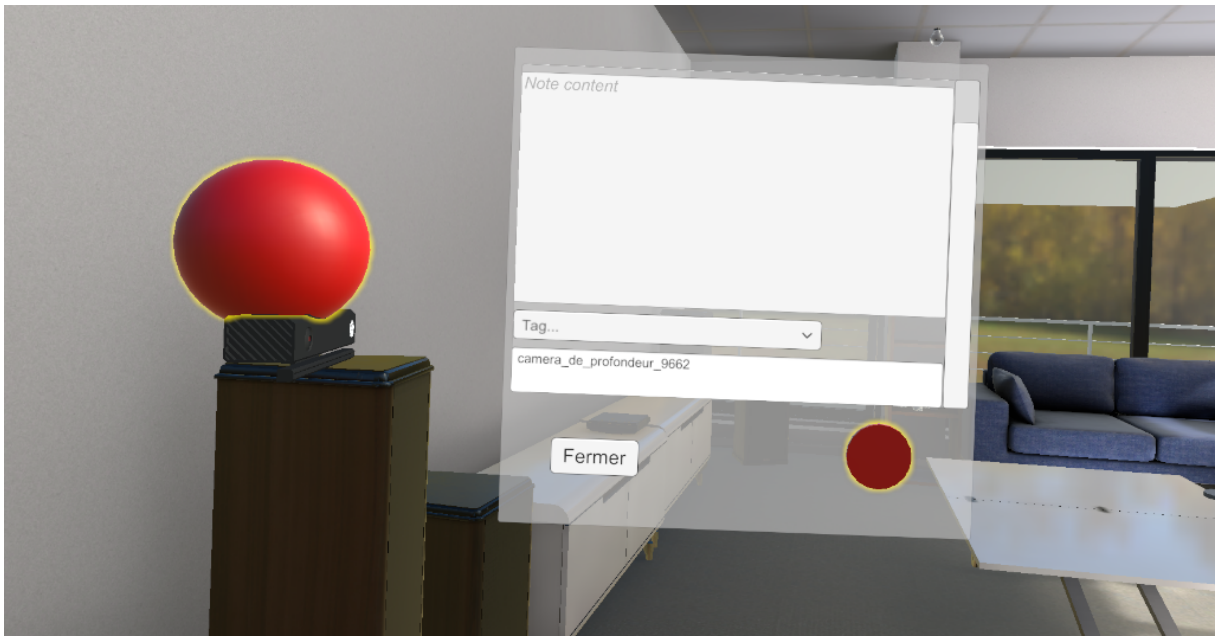


FIGURE 5.7 – Interface d'une note en RV. La sphère représente la position de la note dans l'environnement. La couleur rouge signifie que la note est attachée à un objet (la Kinect). Ainsi, la position de la note sera fixe (i.e. colocalisation) par rapport à l'objet.

Les informations contextuelles sur l'environnement actuel, telles que la position des appareils, l'état du service et la position de la note sont automatiquement enregistrées dans chaque note. De plus, les notes peuvent être attachées à des objets ou simplement placées à un endroit spécifique de l'environnement testé. Ainsi, les administrateurs peuvent facilement fournir des commentaires descriptifs aux producteurs de MIBS. Ces notes peuvent également être utilisées pour fournir des conseils ou des avertissements lors de l'installation d'une configuration dans l'environnement réel. Ces notes peuvent être rédigées par des ergonomes à partir de guidelines telles que présentées dans le chapitre 4.

De plus, les notes, les objets et les configurations créés peuvent être sauvegardés dans des fichiers de configuration et peuvent être chargés pour démarrer à partir d'un MIBS préconfiguré. Par conséquent, des configurations alternatives pour chaque environnement

peuvent être facilement partagées.

Enregistrement et relecture d'actions

Les administrateurs MIBS peuvent enregistrer et rejouer leurs actions dans les environnements testés. Ainsi, ils peuvent vérifier l'impact de l'utilisation du MIBS d'un point de vue externe. Par exemple, la gêne occasionnée par des commandes vocales pourrait être plus facilement remarquée de cette façon (ex, les gestes de la main ou les commandes vocales peuvent gêner les personnes qui partagent le même espace comme des collègues de bureau). De plus, les actions enregistrées peuvent être réutilisées pour tester des changements mineurs (par exemple, le changement de modèle de microphone n'a pas d'impact sur le processus d'interaction) sans effort. En plus de la possibilité de rejouer les tests des administrateurs, les services, les techniques d'interaction et les objets peuvent être modifiés pour produire de nouveaux résultats avec le même comportement de l'utilisateur.

Fonctionnalités de capture d'images : première vue et vue d'ensemble

Les administrateurs peuvent produire des vues en 2D de leurs configurations pour aider ceux qui sont chargés du déploiement. En effet, ils peuvent prendre des captures d'écran depuis n'importe quel endroit ou angle de l'environnement, et générer automatiquement une carte 2D en vue de dessus de tout l'espace avec les emplacements des objets marqués. Les captures d'écran affichent également les notes localisées sur lesquelles on peut cliquer pour les lire. Ensuite, un outil comme l'éditeur Pervasive Maps [158] peut importer ces captures d'écran et la carte pour recréer de manière plus fiable la configuration virtuelle.

Méthodologie MCEV (MIBS Cconfiguration and Eévaluation in VR)

La méthodologie MCEV permet à un utilisateur de faciliter le processus de configuration et d'évaluation des MIBS. Un ergonome peut évaluer facilement et rapidement des MIBS dans une grande diversité de configuration. Il peut ensuite partager les résultats de leurs analyses grâce à trois fonctionnalités de révision collaborative. Au déploiement, un administrateur peut tester plusieurs configurations évaluées par les ergonomes dans une modélisation 3D de son environnement. Il peut ainsi choisir la configuration qui lui convient sans avoir besoin de réaliser des tests sur le terrain.

5.3 Architecture

Notre outil s'intègre à l'architecture présentée dans le chapitre 3. Il est séparé en quatre composants principaux, comme l'illustre la figure 5.8) :

— le simulateur d'environnements prend en charge la transition entre les jumeaux numériques des environnements (gérés par la base de contexte) et les simulations 3D de ces environnements ;

— le gestionnaire de notes fournit les fonctionnalités de collaboration proposées, et gère la création et le stockage des notes ;

— le gestionnaire de dispositifs simulés simplifie la gestion des dispositifs : il centralise la création, la suppression et l'association des dispositifs aux applications, et l'association est ensuite transmise au gestionnaire de configuration ;

— le gestionnaire de configuration permet de créer de nouvelles configurations, ainsi que charger, et mettre à jour les configurations existantes. Pour cela, il prend en charge le processus d'association entre les services, les techniques d'interaction et les dispositifs. Il a aussi accès aux interfaces des composants du MIBS pendant son exécution, ce qui lui permet d'agréger les rapports publiés (ex. messages d'erreur).

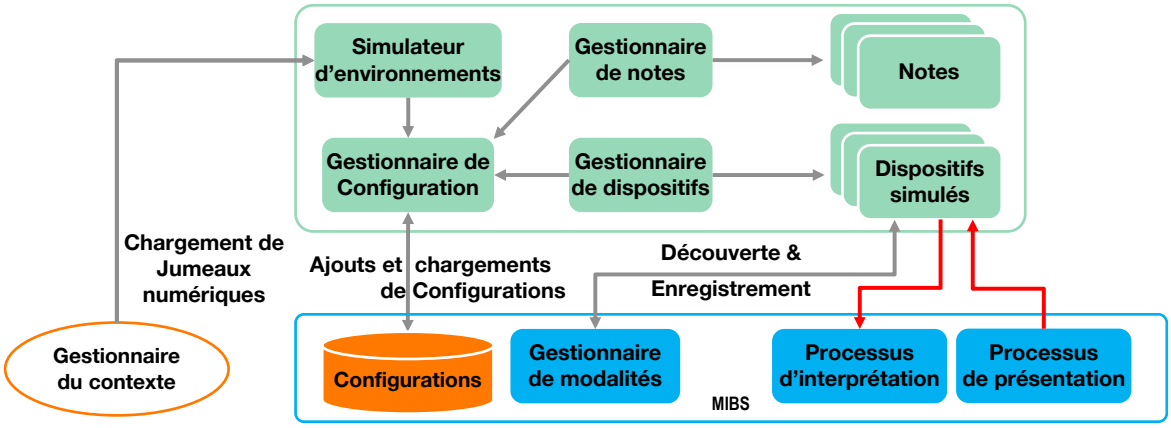


FIGURE 5.8 – Architecture de l'outil MCEV. Les flèches rouges représentent les canaux de communication entre les dispositifs simulés et les applications configurés pendant l'exécution du MIBS dans la phase d'évaluation.

On notera que les jumeaux numériques des objets connectés sont récupérés depuis la base de contexte, ce qui implique que la modélisation 3D de l'environnement doit avoir été effectuée lors de la phase d'analyse du contexte. En effet, les connaissances techniques requises pour créer les composants de modalités peuvent inclure des informations sur la

géométrie de l'objet. Par exemple, la position relative entre les deux objectifs d'une caméra stéréoscopique est indispensable dans la plupart des processus d'interprétation utilisant une information de profondeur. Les modèles peuvent aussi servir pour des méthodes d'évaluation de MIBS dans les phases de design et développement comme dans [123].

5.4 Synthèse

Dans ce chapitre, nous avons proposé une méthodologie et un outil logiciel basés sur la RV pour faciliter la configuration et l'évaluation des MIBS. Grâce aux interactions 3D, notre méthode vise à faciliter la sélection et le positionnement des dispositifs et leur association avec les techniques et services d'interaction multimodale nouvellement configurés. Les MIBS créés peuvent ensuite être évalués en immersion sans avoir besoin de l'environnement réel. Enfin, les configurations évaluées peuvent être partagées avec les fonctionnalités collaboratives que nous proposons pour améliorer le MIBS ou faciliter son installation dans l'environnement réel. Dans le chapitre suivant, nous décrivons une expérimentation réalisée pour comparer l'utilité de notre méthodologie par rapport à une installation directement dans l'environnement réel. De plus, nous illustrons la méthodologie MCEV au travers de deux cas d'usage sur les applications de réservation et de guidage présentées en introduction.

Chapitre 6

Application de la méthode MCEV

Dans ce chapitre, nous allons mettre en pratique la méthode MCEV décrite dans le chapitre précédent, dans le but notamment de l'évaluer. Dans un premier temps (6.1), nous décrivons une expérimentation que nous avons menée auprès de 24 utilisateurs, qui consistait à comparer l'outil MCEV à un équivalent sans réalité virtuelle.Dans un second temps (chapitre 6.2), nous illustrons l'usage de la méthode MCEV sur les deux cas d'usage présentés en introduction, à savoir la réservation de salle de réunion et le guidage.

6.1 Validation de la méthode MCEV

Nous proposons donc d'évaluer notre solution pour atteindre deux objectifs principaux.

Premièrement, nous voulons déterminer si notre outil MCEV peut aider à configurer et à tester les MIBS plus rapidement qu'avec un outil graphique 2D lancé sur un ordinateur dans les expériences directement sur le terrain, sans dégrader le résultat. Deuxièmement, nous voulons évaluer la charge cognitive et la facilité d'utilisation de notre méthodologie MCEV par rapport aux expériences sur le terrain. En effet, même si la plupart des gens sont plus habitués aux interactions souris-clavier, nous pensons que la RV peut fournir un moyen plus naturel et efficace de configurer le MIBS grâce aux interactions 3D et à la navigation à l'échelle un.

Pour valider ces objectifs, nous avons réalisé une expérience utilisateur sur un scénario de contrôle de la luminosité d'une pièce. Dans la suite de cette section, nous détaillons le déroulement et les résultats de cette expérimentation. Tout d'abord, nous introduisons l'outil graphique 2D (appelé par la suite *outil graphique*) que nous avons développé pour permettre de comparer les deux méthodes de configurations. Ensuite, nous décrivons le cadre de l'expérimentation réalisée, ainsi que les résultats et les observations que nous avons obtenus. Enfin, nous discutons de la validation des hypothèses en fonction des résultats.

Soumission

Ce chapitre a fait l'objet d'une soumission à la conférence internationale HUCAPP 2023.

6.1.1 Outil graphique 2D pour la comparaison entre un déploiement directement sur le terrain et la méthodologie MCEV

Pour évaluer l'outil MCEV, nous avions besoin d'un outil graphique comparable à notre outil, et qui supporte les expériences sur le terrain sur MIBS. Les outils graphiques existants ne fournissent qu'un support limité. Par exemple, la position des dispositifs ne peut pas être gérée avec l'IDE MIBO [126], et l'outil Proximity [102] ne prend en charge que le processus de test. Nous avons donc développé un *outil graphique* pour les expériences sur le terrain qui prend en charge la configuration et le test des interactions spatialisées et multimodales. Pour garantir une configuration similaire à celle de la RV , l'*outil graphique* suit le même processus de configuration que l'outil MCEV, avec une interface 2D presque identique. Cependant, il existe des différences notables entre les fonctionnalités de l'outil MCEV et celles de l'*outil graphique*.

Premièrement, la plupart des appareils connectés ne peuvent pas connaître par eux-mêmes leur emplacement dans l'environnement, et il n'existe pas de méthode universelle et automatique pour localiser chaque appareil connecté [33]. Par conséquent, les emplacements des dispositifs sont indiqués manuellement avec l'*outil graphique*. Il fournit une carte 2D générée à partir du jumeau numérique de l'environnement, comme l'illustre la figure 6.1. Sur cette carte, les représentations iconiques des dispositifs ajoutés peuvent être déplacées et pivotées pour correspondre aux positions des dispositifs réels. Cette méthode de suivi manuel de la position ignore la hauteur et les deux degrés de liberté en rotation, mais comme ces dimensions n'étaient pas utilisées dans notre scénario, les capacités des fonctions de positionnement de l'*outil graphique* étaient suffisantes pour l'expérience.

Deuxièmement, les dispositifs de notre outil MCEV peuvent être mis en évidence pour les reconnaître à partir de leurs identifiants, mais cette fonctionnalité ne peut pas être utilisée directement dans l'expérience sur le terrain avec l'outil 2D. Cependant, les capteurs peuvent afficher leur retour d'information, et les actionneurs peuvent se comporter de manière perceptible. Ainsi, chaque dispositif de notre expérience avait un comportement spécifique que l'*outil graphique* était capable de déclencher lorsqu'une identification était

nécessaire (par exemple, le retour de la caméra, le clignotement de la lumière).

Troisièmement, les *auras* des appareils dans l'*outil graphique* sont représentées par leurs vues 2D. Des visualisations 3D telles que celles de [102] ont également été envisagées, mais elles nécessitent l'utilisation de dispositifs de suivi supplémentaires, ce qui créerait une trop grande différence entre une expérience sur le terrain et en RV. Dans l'*outil graphique*, les participants pouvaient afficher l'*aura* dans la carte en vue de dessus, comme illustré dans la figure 6.1.

Enfin, les participants pouvaient associer des dispositifs en sélectionnant les icônes sur la carte 2D. Ainsi, le processus d'association est similaire à la méthode de la RV en fournissant une méthode de sélection spatialisée.

En résumé, les différences de fonctionnement entre l'*outil graphique* et l'outil MCEV s'expliquent par les différences entre une installation sur le terrain et en RV (contrairement à un objet réel, la position et l'état d'un objet virtuel peuvent être récupérés à tout moment). Pour proposer des expériences comparables entre les deux outils, nous avons intégrés dans l'*outil graphique* des fonctionnalités qui réduisent ces différences sans pour autant utiliser de matériel supplémentaire car cela s'éloignerait trop des pratiques existantes de déploiement.

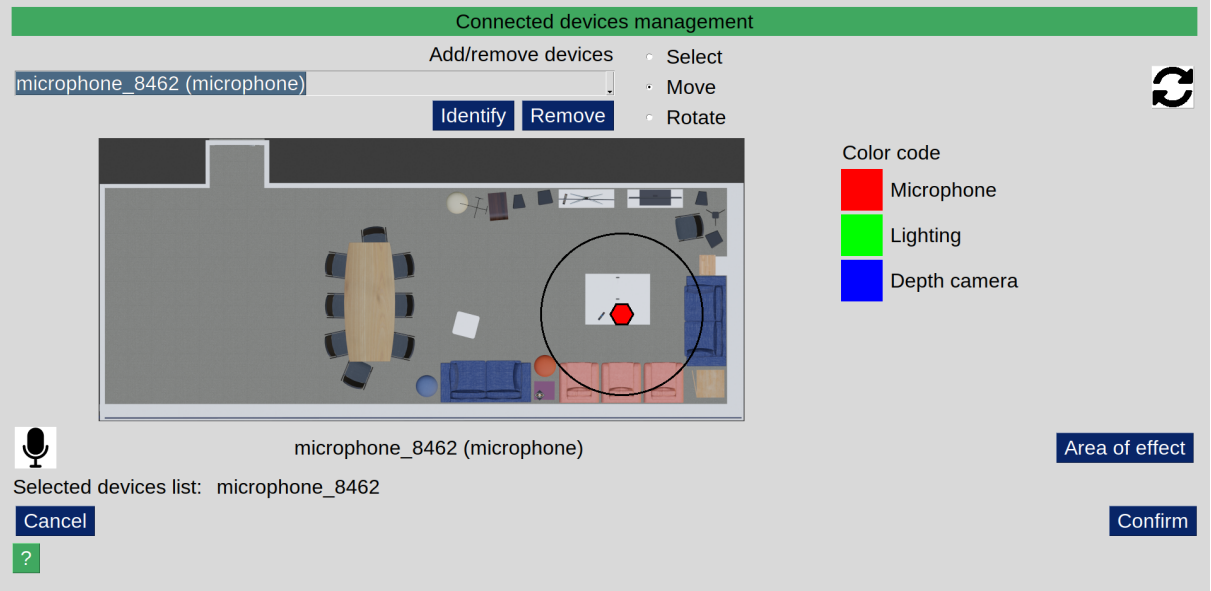


FIGURE 6.1 – Menu de gestion des objets connectés dans l'outil graphique 2D (sur le terrain). Les participants peuvent identifier les objets disponibles et les placer dans une représentation en vue du dessus de la pièce. Le cercle noir représente l'*aura* du microphone.

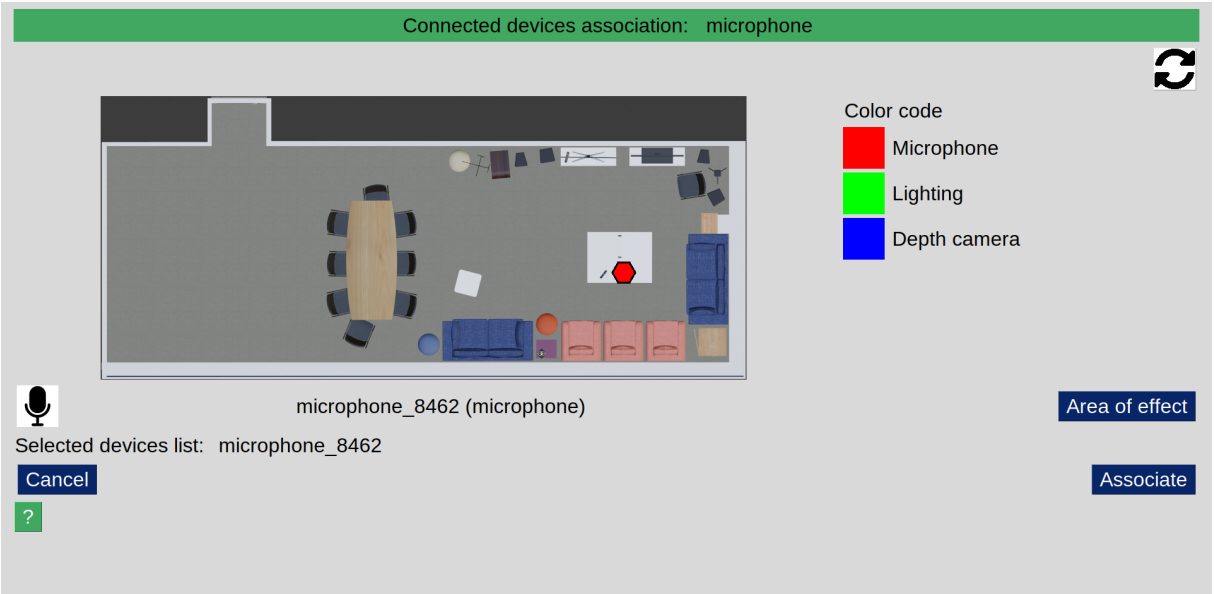


FIGURE 6.2 – Menu d'association des objets connectés à un service dans l'outil graphique 2D (sur le terrain). Les participants peuvent sélectionner parmi les objets positionnés fournissant la modalité requise pour le service (ici le microphone permet d'écouter l'utilisateur) ceux que l'utilisateur souhaite utiliser.

6.1.2 Contexte d'expérimentation

Pour comparer notre méthodologie MCEV aux expériences sur le terrain, nous avons demandé aux participants de configurer et de tester un service de gestion de la lumière avec les deux méthodologies. Comme le montrent les figures 6.3a et 6.3b, l'environnement dans la RV était le même que dans l'expérience sur le terrain.

Scénario

Notre exigence pour l'expérience était de fournir un scénario qui présente l'utilisation d'interactions multimodales où l'emplacement d'au moins un dispositif connecté a un impact sur le résultat. Les interactions multimodales complexes peuvent être difficiles à utiliser au début pour la plupart des utilisateurs, et fournir de nombreuses alternatives de modalité est une tâche répétitive qui n'est pas nécessaire pour une expérience. Les participants devaient donc configurer dans la RV et directement dans l'environnement réel (sur le terrain) un service simple de gestion de la lumière. Ce scénario couvre un cas d'utilisation adapté à divers environnements (par exemple, maisons, bureaux) [114].

Ce service consiste à contrôler par des commandes vocales et des gestes deux ampoules connectées placées aux mêmes endroits dans l'environnement réel et dans la RV (voir Figure 6.3c). Pour représenter les cas d'utilisation dans lesquels les positions de certains dispositifs connectés sont déjà définies et non modifiables, les ampoules sont déjà positionnées et ne peuvent pas être déplacées. Seules deux modalités en entrée (vocale et gestuelle) et une en sortie (luminosité) étaient possibles afin de fournir une interaction multimodale sans répétitions inutiles dans le processus de configuration. Les participants devaient utiliser un microphone et une caméra de profondeur pour les commandes vocales et gestuelles. Dans l'expérience RV , les deux dispositifs ont dû être instanciés et placés. Pour recréer ce besoin de positionner les objets dans l'expérience sur le terrain, les dispositifs étaient dans la pièce, mais placés délibérément dans de mauvaises positions (voir la figure 6.5b. De plus, les deux appareils étaient connectés par USB à un ordinateur portable qui exécutait l'*outil graphique*. La caméra était fixée à un trépied professionnel réglable en angle et en hauteur.

De plus, d'autres appareils étaient disponibles, en plus de ceux qui devaient être utilisés pendant l'expérience.. En effet, nous avons voulu recréer un scénario où des appareils connectés extérieurs à l'environnement considéré sont disponibles et détectés. Ainsi, une ampoule électrique et un microphone ont été simulés dans les deux expériences et étaient visibles dans la liste des identifiants des dispositifs, mais les participants ne pouvaient pas trouver ces dispositifs autour d'eux.

Pour faciliter la comparaison du positionnement des dispositifs, l'espace d'interaction a été limité à une zone spécifique : les participants ont été invités à configurer le service de gestion de la lumière afin de pouvoir l'utiliser depuis les trois chaises beiges (voir la figure 6.3c).

Trois techniques d'interaction étaient disponibles pour ce service :

T1 : contrôle de l'éclairage avec seulement la commande vocale "lumière" pour allumer ou éteindre les éclairages.

T2 : contrôle de l'éclairage avec un geste de pointage et une commande vocale où l'utilisateur doit commander oralement d'"allumer" ou d'"éteindre" une lumière spécifique en la pointant. Nous l'appelons la technique du "pointage".

T3 : contrôle de l'éclairage avec un mouvement de la main et une commande vocale comme déclencheur pour démarrer ou arrêter en considérant la position de la main pour changer l'intensité de l'éclairage. C'est la technique dite du "haut en bas".

T1 était une technique d'interaction monomodale pour apprendre le processus de confi-

guration tandis que les participants ont testé T2 et T3 avec les deux outils.

Pour fournir des conditions similaires entre l'expérience de RV et la réalité, l'environnement de RV était une réplique haute-fidélité de l'environnement réel. Cet environnement était une salle de réunion dédiée aux expériences des utilisateurs qui reproduit un salon. Dans l'environnement RV , les participants pouvaient utiliser une technique de téléportation pour naviguer qui consiste à pointer avec la manette la position desiré pour y être déplacer. Les participants étaient incarnés par un avatar composé de deux mains et d'un corps (voir le retour caméra dans la figure 5.1). L'objectif de l'utilisation d'un corps était d'aider les participants à se situer lorsqu'ils utilisaient le retour caméra, de manière similaire au retour caméra réel.

(a) (b) (c)

FIGURE 6.3 – Les deux environnements utilisés pour l'expérimentation : (a) l'environnement réel et (b) l'environnement virtuel. Les étoiles vertes dans (c) la vue du dessus de l'environnement représentent les positions des ampoules connectées, et le rectangle rouge représente l'espace d'interaction.

Procédure

Les participants devaient expérimenter successivement les deux outils. Ils commençaient par une explication de la procédure d'expérimentation. Ensuite, pour chaque outil, ils avaient une étape d'apprentissage pour se familiariser avec le processus et les spécificités des outils (par exemple, les commandes dans la RV). Les participants recevaient un guide de configuration étape par étape avec des explications (voir Annexe B) pour configurer et tester le service de gestion de la lumière avec la technique d'interaction T1. Ils n'avaient besoin que du microphone à ce stade. Une fois formés à l'utilisation d'un outil, ils étaient invités à configurer et à tester le service avec les techniques d'interaction T2 ou T3 sans instructions détaillées. T2 et T3 nécessitaient toutes deux une caméra de profondeur pour fonctionner correctement. L'ensemble de l'expérience durait jusqu'à 2

heures, avec une moyenne de 1h16.

L'objectif était d'éviter les biais d'apprentissage et les biais causés par les différences de consignes et de difficultés. Ainsi, les participants ont été répartis en 4 groupes présentés dans le tableau 6.1. Ces groupes ont été nommés en fonction de la technique d'interaction et de la méthode de configuration qu'ils devaient utiliser au départ. Par exemple, le groupe qui commençait par l'expérience sur le terrain (SLT) en utilisant la méthode T2 était "T2_SLT_first".

	Commencer avec T2	Commencer avec T3
Commencer par l'expérience SLT	T2_SLT_first	T3_SLT_first
Commencer en RV	T2_VR_first	T3_VR_first

TABLE 6.1 – Les 4 groupes de participants

Participants

L'objectif de notre expérience était de préserver entre les groupes une diversité similaire d'âge, de sexe et d'expérience en RV. Ainsi, les 4 groupes étaient composés de 6 personnes. Chaque groupe comptait 5 hommes et 1 femme. La moyenne (m) et la déviation standard (sd) de l'âge par groupe étaient :

— m=37.7, sd=16.8 pour "T2_SLT_first" ;
— m=35,8, sd=15,8 pour "T2_VR_first" ;
— m=42.3, sd=12.3 pour "T3_SLT_first" ;
— m=34.8, sd=15.7 pour "T3_VR_first".

Les groupes qui ont commencé en réel avaient le même nombre d'experts (c'est-à-dire des participants ayant des heures d'expérience en RV) et de non-experts en RV (6 personnes), alors qu'il y avait 7 experts pour 5 non-experts pour les deux autres groupes.

L'expérimentateur et les participants étaient des designers d'interface utilisateur, des développeurs et des chercheurs de la même entreprise. Notre laboratoire n'a pas de comité d'éthique, mais nous avons fait de notre mieux pour respecter les principes éthiques : les participants ont été informés des principes de l'expérience, ils ont dû donner leur consentement écrit et ils pouvaient arrêter l'expérience à tout moment. En outre, les données collectées ont été conservées de manière anonyme et l'expérience n'a pas mis les participants dans une situation dangereuse.

Implémentation et équipement

Notre outil MCEV est développé avec Unity 2019.4 LTS `https://unity3d.com/`. Nous avons utilisé l'API de reconnaissance vocale de Google pour reconnaître les commandes vocales. L'outil MCEV était exécuté sur un casque Oculus Quest 2 connecté à un ordinateur portable (RTX 2080, Intel Core I9-9900K, 32Go de RAM) avec le mode "link" (avec fil).

L'*outil graphique* a été développé avec la bibliothèque Python PySimpleGUI [1] et était exécuté sur un ordinateur portable (RTX 2070, Intel Core I7-10750H, 16Go RAM) pour l'expérience sur le terrain. Une caméra Kinect (sur un trépied de caméra) et un microphone USB étaient connectés à ce PC. Les ampoules étaient des ampoules connectées Philips.

Données collectées

Pour comparer les performances temporelles de l'expérience sur le terrain et de la RV , les temps de réalisation pour chaque mode (configuration ou test) et au total ont été enregistrés. De plus, pour comparer la qualité des configurations créées avec les deux outils, nous avons enregistré le positionnement des dispositifs chaque fois que les participants testaient leur configuration.

Nous avons mis en place un système de notation en pourcentage pour imposer une qualité de configuration minimale sur le positionnement des dispositifs et la couverture de l'espace d'interaction illustré par le carré rouge de la figure 6.3c. Le système de score est détaillé dans l'encart ci-dessous. Une configuration était considérée comme acceptable si le score pour chaque dispositif était suffisamment élevé (supérieur à 75 En pratique, il a été facilement obtenu tant que les deux dispositifs n'étaient pas trop éloignés des chaises et que la caméra avait les chaises dans son champ de vision. Ainsi, il n'y avait pas de positionnement optimal spécifique qui pouvait être déduit du score.

1. `https://www.pysimplegui.org/en/latest/`

Considérons $d_{j,i}$ la distance entre le point d'intérêt j et le dispositif i, et $d_{i_{max}}$ la distance maximale à laquelle le dispositif i pourrait fonctionner correctement (par exemple, la distance maximale avant que le microphone ne puisse plus écouter).
Nous proposons comme formule de score pour la distance $S_{d,j,i}$ $S_{d,j,i} = \frac{\exp c - \exp \frac{c * d_{j,i}}{d_{i_{max}}}}{\exp c - 1}$ où c est une constante déterminée pour avoir un score de 75% lorsque $d_{j,i}$ est autour de 75% de $d_{i_{max}}$, c'est-à-dire pour $c = 6$.
Pour un angle horizontal $\theta_{j,i}$ entre le point d'intérêt j et le dispositif i, $\theta_{i_{max}}$ son champ de vision horizontal, avec $\theta_{i_{half}}$ sa moitié (c'est-à-dire $\theta_{i_{half}} = \frac{\theta_{i_{max}}}{2}$). On obtient avec une formule similaire le score de l'angle $S_{\theta,j,i}$ $S_{\theta,j,i} = \frac{\exp k - \exp \frac{k * \theta_{j,i}}{\theta_{i_{half}}}}{\exp k - 1}$ où k est une constante déterminée pour avoir un score de 75% lorsque $\theta_{j,i}$ se situe aux alentours de 95% de $\theta_{i_{max}}$, c'est-à-dire pour $k = 27.726$.
Ainsi, pour n points d'intérêt, le score final $S_{i,total}$ est $S_{i,total} = \frac{min(\sum_{j=1}^{n} S_{d,j,i}, \sum_{j=1}^{n} S_{\theta,j,i})}{n}$.

À la fin de chaque expérience, les participants devaient répondre au questionnaire TLX (Task Load Index) de la NASA [74], et pouvaient proposer des méthodes pour accélérer l'ensemble du processus. De plus, les participants devaient remplir le questionnaire SSQ (simulator sickness questionnaire) [84] avant et après l'expérience en RV afin de vérifier que les participants n'avaient pas souffert du mal des simulateurs, ce qui peut affecter les résultats. L'objectif de ces questionnaires était d'évaluer et de comparer les coûts cognitifs globaux des deux méthodes avec des données quantitatives. De plus, nous voulions comparer la facilité d'utilisation des deux méthodologies pour gérer des appareils. Ainsi, nous avons demandé aux participants de noter sur une échelle de 7-Likert les 3 affirmations présentées dans le tableau 6.2. Les questionnaires AttrakDiff [2] et SUS [3] ont également été envisagés pour l'évaluation de l'utilisabilité mais ont été mis de côté. L'expérience a duré près de 2 heures pour certains participants. Ajouter de nouveaux questionnaires aurait augmenté la durée de l'expérience ainsi que la fatigue des participants, ce qui aurait pu entraîner des biais dans les résultats.

À la fin, les participants pouvaient commenter l'expérience globale, ce qui nous a permis de mieux comprendre les résultats de l'expérimentation.

2. `http://www.attrakdiff.de/index-en.html`
3. `https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html`

Id	Affirmations
FQ1	Le placement et repositionnement des objets connectés est plus rapide ...
FQ2	Le placement et repositionnement des objets connectés est plus permissif ...
FQ3	L'identification des objets connectés est plus facile ...

TABLE 6.2 – Les affirmations utilisées pour comparer la gestion des objets connectés dans l'expérience sur le terrain (SLT) et en RV. L'évaluation est entre -3 ("sur le terrain") et 3 ("en réalité virtuelle")

.

Hypothèses

Avec cette expérience, notre objectif est de prouver que notre méthodologie MCEV pourrait être une alternative fiable à la configuration et au test des MIBS sur le terrain. En particulier, nous pensons que notre outil MCEV est plus rapide qu'un outil graphique 2D, et nous prévoyons un positionnement similaire des dispositifs dans les deux expériences. De plus, nous pensons que les avantages de la simulation et de l'immersion apportés par la RV réduisent la pénibilité du processus de configuration et de test. Nous pensons que c'est particulièrement vrai lors de la manipulation de dispositifs connectés dans des interactions multimodales et spatialisées. Ainsi, nos hypothèses sont les suivantes :

H1) La configuration de MIBS avec l'outil MCEV est plus rapide qu'avec l'*outil graphique*, sans dégradation du résultat.

H2) L'outil MCEV induit chez l'utilisateur une charge de travail cognitive moins importante que l'*outil graphique*.

H3) L'outil MCEV est plus facile à utiliser que l'*outil graphique* pour identifier, placer ou déplacer les dispositifs.

6.1.3 Résultats

Temps de configuration

Afin de comparer le temps nécessaire pour configurer et tester le service de gestion de la lumière, nous avons demandé aux participants d'effectuer les tâches dans un délai convenable (i.e. le plus rapidement possible sans que cela nuise à la qualité de la configuration). Les participants pouvaient arrêter les expériences dès que le score était supérieur à 75% pour chaque appareil. Le temps nécessaire à la configuration et au test a été enregistré. Les résultats sont détaillés dans la figure 6.4. Comme nous pouvons le constater,

les participants ont eu besoin de plus de temps pour configurer le service avec l'*outil graphique* qu'avec l'outil MCEV (une différence moyenne de 195s). Chaque participant a expérimenté à la fois dans la réalité et dans la RV, les mesures sont donc appariées. Comme les résultats ne suivent pas une distribution normale, nous évaluons la signification du résultat avec un test de Wilcoxon. Le résultat (Z=-2.23, p=0.013) indique que cette différence de temps moyen est significative, ce qui confirme l'hypothèse initiale selon laquelle il est plus rapide de configurer MIBS avec l'outil MCEV qu'avec l'*outil graphique*.

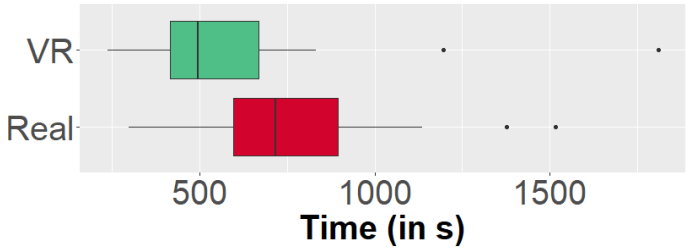
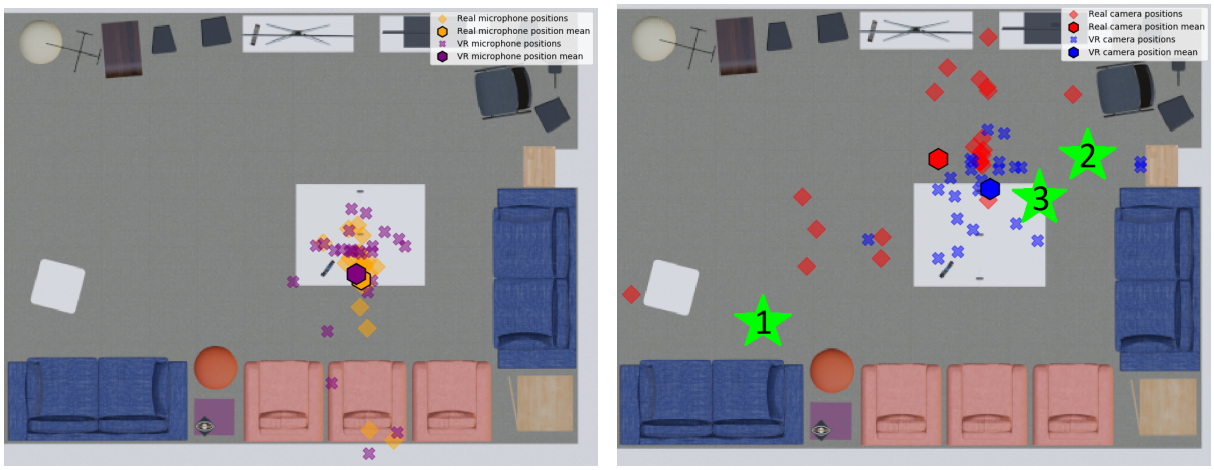


FIGURE 6.4 – Temps total pour trouver une configuration acceptable

Qualité de la configuration

La deuxième partie de l'hypothèse H1 consiste à s'assurer que les configurations réalisées avec l'outil MCEV sont de la même qualité que celles réalisées avec l'*outil graphique*. Plus précisément, le critère de qualité est basé sur les différences possibles dans les positions des dispositifs dans la RV et la réalité. Pour cela, les positions finales des dispositifs projetés sur le sol ont été enregistrées, comme le montrent les figures 6.5a et 6.5b. Dans ces figures, les positions finales de chaque dispositif sont représentées par des losanges transparents, tandis que les positions moyennes sont des hexagones. Nous avons mesuré une distance moyenne (par paire) entre les positions 2D dans l'expérience sur le terrain et dans la RV de 0,44 m pour le microphone, et de 1,10 m pour la caméra. En moyenne, il ne semble pas y avoir de différences dans le positionnement du microphone. Pour la caméra de profondeur, les participants ont adopté des positions finales plus diverses, surtout dans la réalité. Néanmoins, les positions moyennes de la caméra de profondeur dans l'expérience sur le terrain et dans la RV étaient à peu près au même endroit : devant les trois chaises. En résumé, il n'y a pas de dégradation notable de la qualité de la configuration car les configurations réalisées dans les deux expérience sont globalement similaires.

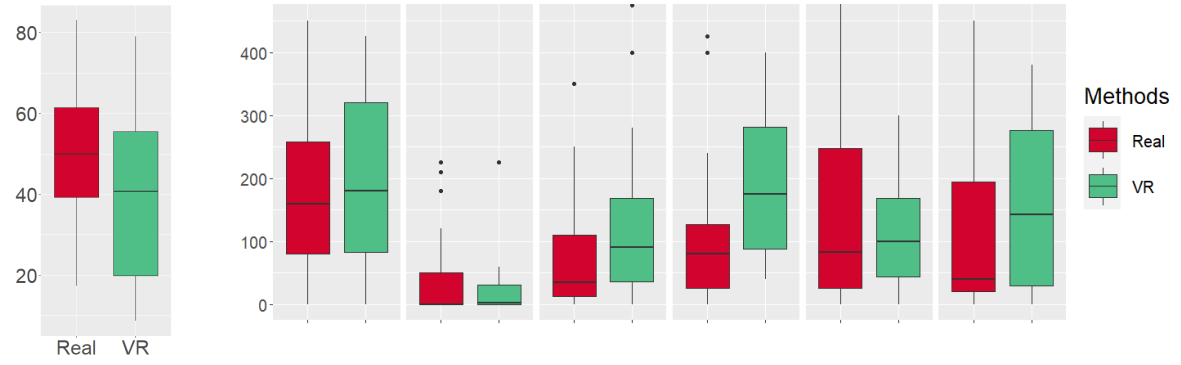


(a) (b)

FIGURE 6.5 – (a) Les positions du microphone et (b) les positions de la caméra de profondeur pour toutes les configurations finales des participants, ainsi que les positions moyennes de ces capteurs. Les étoiles vertes numérotées dans la figure (b) sont respectivement les positions initiales de la caméra, du microphone et de l'ordinateur portable dans l'expérience sur le terrain (SLT).

Charge de travail

Pour évaluer la charge de travail des utilisateurs, nous avons recueilli les résultats des questionnaires NASA-TLX remplis après chaque expérience. Les résultats détaillés dans la figure 6.6 montrent que la charge de travail semble légèrement inférieure en RV qu'en réalité. Les résultats sont appariés de la même manière que les mesures du temps de configuration, De plus, les données ne suivaient pas une loi normale. Nous avons donc effectué un test de Wilcoxon pour évaluer la signification de ce résultat. En conséquence, la différence de charge de travail globale n'est pas significative (Z=-1,34, p=0,09). Pour une analyse plus approfondie, nous avons comparé les résultats pour chaque facteur NASA-TLX, comme détaillé dans la Figure 6.6. Aucune tendance n'a été observée sur la plupart des paramètres (p>0,05), sauf pour le critère d'estimation de la performance (Z=-2,51, p=0,006) qui semble montrer que la performance induit davantage de charge mentale en RV qu'en réel.



(a) (b)

FIGURE 6.6 – (a) La notation globale de la charge cognitive et (b) la notation pour chacun des critères.

SSQ

Pour évaluer si les participants n'ont pas eu le mal des simulateurs, nous avons comparé les résultats du SSQ avant et après cette expérience. Les mesures n'ont pas suivi une distribution normale, nous avons donc utilisé un test de rang signé par paire appariée de Wilcoxon pour évaluer la signification des résultats du questionnaire. Même si le score SSQ a significativement (Z=-3.45, p=$2.8 * 10^{-4}$) augmenté pendant l'expérience de RV (c'est-à-dire qu'il est passé d'une moyenne de 4.99 à une moyenne de 18.23), il est resté faible.

Utilisabilité

Pour l'évaluation de la convivialité de l'outil MCEV par rapport à l'*outil graphique*, nous avons analysé les résultats des déclarations sur une échelle de Likert à 7 valeurs, détaillés dans la figure 6.7. Pour les 3 notes, le résultat est en faveur de l'expérience RV . La distribution des notes n'est pas normale, nous validons donc la signification de ce résultat avec un test de rang signé de Wilcoxon sous l'hypothèse nulle que les rangs sont inférieurs ou égaux à 0. Comme les notes sont significativement supérieures à 0 (FQ1 : Z=-3. 43, p=$3,0 * 10^{-4}$, FQ2 : Z=-3,62, p=$1,5 * 10^{-4}$; FQ3 : Z=-3,31, p=$4,6 * 10^{-4}$), cela semble confirmer que l'outil MCEV offre une meilleure utilisabilité que l'*outil graphique* lors de la manipulation d'appareils connectés (H3).

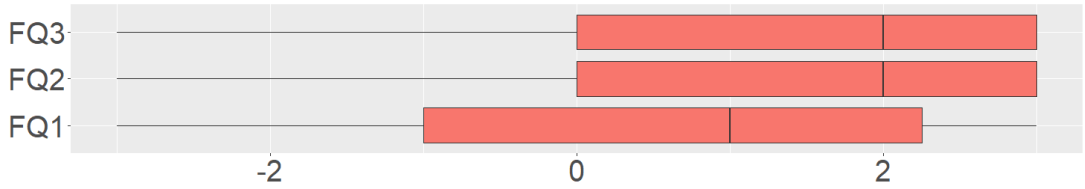


FIGURE 6.7 – Évaluation des affirmations du tableau 6.2, entre -3 et 3

6.1.4 Observations et retours des participants

Tout d'abord, la plupart des participants ont signalé des difficultés à placer la caméra dans l'expérience sur le terrain, et ont perdu du temps à le faire. En effet, nous avions expressément autorisés les participants à déplacer l'ordinateur portable hébergeant l'*outil graphique* pour pouvoir manipuler la caméra tout en ayant son retour affiché sur l'ordinateur. Malgré cela, nous avons observé que la plupart des participants ne déplaçaient pas l'ordinateur, et faisaient des allers-retours entre la caméra pour la placer, et l'ordinateur portable pour le retour visuel. Il y avait aussi des participants qui n'ont pas utilisé le retour visuel de la caméra au début, car ils ont déclaré qu'ils étaient confiants dans leurs placements de caméra. En comparaison, nous avons observé que la plupart des participants n'ont pas eu de difficultés à placer la caméra dans l'outil MCEV, et qu'ils ont été relativement rapides à trouver une position apparemment acceptable. Deuxièmement, certains participants ont fait part de leurs inquiétudes quant au réalisme du positionnement de l'appareil dans la RV. Par exemple, il était possible de placer les appareils (par exemple le microphone) sous ou "dans" les meubles dans la RV. Certains participants ont essayé de tels placements pendant l'expérience, mais ils ont fini par se contenter d'emplacements plus réalistes. Troisièmement, la plupart des utilisateurs ont eu des difficultés dans la RV avec les interactions gestuelles car ils ont tendance à oublier que la caméra simulée a un *aura* similaire à celui de la caméra réelle. En effet, ils pensaient que les positions de leur corps et de leurs mains étaient captées en permanence grâce au tracking RV, même s'ils n'étaient pas parfaitement en face de la caméra. Enfin, lorsqu'ils déplaçaient la caméra réelle, certains participants ont oublié de mettre à jour sa position sur la carte en vue de dessus de l'*outil graphique*.

6.1.5 Discussion

Performance de l'outil MCEV

Nos résultats montrent que notre première hypothèse H1 est validée. En effet, les participants étaient plus rapides avec notre outil MCEV, et les dispositifs étaient positionnés à peu près aux mêmes endroits. Nos observations indiquent que la possibilité de manipuler un dispositif et de visualiser son retour en même temps dans l'environnement virtuel (comme dans les figures 5.1 et 5.5) pourrait permettre de gagner du temps. De même, le fait que les emplacements des dispositifs soient directement accessibles dans l'environnement virtuel pourrait aider à éviter les erreurs et les approximations, ce qui pourrait également avoir un impact positif sur les performances temporelles. Il pourrait être intéressant d'observer la différence de performance dans des cas d'utilisation plus complexes (environnements plus grands, plus de dispositifs, services plus élaborés), comme un service pour guider les nouveaux arrivants dans un bâtiment du tertiaire connecté. Nous pensons que le gain de temps pour configurer des MIBS serait encore plus grand en RV pour ces scénarios exigeants.

Cependant, en RV certains participants ont essayé des positions irréalistes, et se sont inquiétés du processus de reproduction des configurations RV dans l'environnement réel. En effet, le réalisme du positionnement des appareils n'a pas été mesuré ou contraint, et notre expérience s'est appuyée sur les participants pour définir ce qui était acceptable et ce qui ne l'était pas. De plus, la qualité des configurations n'a été mesurée qu'en prenant en compte l'emplacement des dispositifs sur les dimensions 2D. Nous avons fixé cette limite afin de comparer le positionnement des dispositifs dans la réalité et en environnement virtuel avec des informations similaires. Cependant, les facteurs environnementaux tels que les obstacles 3D ou l'environnement sonore peuvent avoir un impact sur la qualité des configurations. Ainsi, le réalisme des environnements virtuels et la qualité du support fourni par l'outil MCEV pourraient avoir un impact sur le temps nécessaire à l'installation du MIBS, et cela pourrait conduire à des expériences plus approfondies.

Néanmoins, l'outil MCEV pourrait être amélioré par l'ajout d'une fonctionnalité permettant de générer automatiquement des avertissements et des recommandations sur le positionnement des dispositifs à partir des informations sur les dispositifs et l'environnement.

Charge de travail

Contrairement à notre hypothèse H2, nos résultats suggèrent que les *outils graphiques* et MCEV ne présentent pas de différences vérifiées en termes de charge cognitive, bien que les participants se soient trouvés en moyenne moins performants en RV que sur le terrain.

Cette absence de différence d'utilisabilité pourrait s'expliquer par la simplicité du cas d'usage de notre expérimentation. En effet, comme nous savions que nous allions avoir des participants aux profils très différents, certains étant très peu familiers avec la Réalité Virtuelle, nous avons limité notre expérience à un scénario simple dans une seule pièce, et nous n'avons pas expérimenté les fonctionnalités collaboratives de notre système. Comme nous nous y attendions, certains des commentaires des participants suggèrent que l'avantage de la RV pourrait devenir plus significatif avec des situations à plus grande échelle et complexité. La différence de performance dans le NASA-TLX pourrait s'expliquer par les difficultés de la plupart des utilisateurs à comprendre que la caméra virtuelle et la caméra réelle partagent la même *aura*.

Utilisabilité de la gestion des objets avec MCEV

Notre hypothèse H3 a été validée, ce qui signifie que l'identification et le déplacement des dispositifs semblent plus faciles avec l'outil MCEV.

Les difficultés à gérer des dispositifs avec une aura non triviale, comme la caméra de profondeur dans notre expérience, semblent être la raison principale de ces différences entre les deux expériences. Cela suggère que la méthodologie MCEV est avantageuse avec les MIBS qui reposent fortement sur des capteurs unidirectionnels. La pertinence du positionnement du dispositif dans la RV pourrait être évaluée plus avant dans des scénarios plus complexes. En effet, les dispositifs simulés et réels ont été placés à peu près à la même position en moyenne. Cependant, notre scénario était limité à un environnement relativement simple, sans perturbations ni difficultés considérables.

Pour montrer la pertinence de notre approche pour des cas plus complexe, nous présentons dans la section suivante des exemples d'utilisation de l'outil MCEV pour configurer les applications de réservation et de guidage.

Validation de la méthodologie MCEV

Nous avons comparé dans une expérimentation notre méthode MCEV à une méthode d'installation sur le terrain. Pour cela, nous avons développé un outil graphique 2D d'installation sur le terrain pour pouvoir comparaison les deux méthodes. Les résultats montrent que les utilisateurs réalisent des configurations similaires avec les deux méthodes, mais que le processus de configuration et d'évaluation est réalisé plus rapidement avec la méthode MCEV.

6.2 Applications aux cas d'usage

L'expérimentation présentée dans la section précédente n'a donné qu'un aperçu limité des fonctionnalités de notre méthodologie MCEV. En effet, l'application se limitait à une interaction, seulement 4 objets connectés étaient utilisables, l'environnement d'exécution était relativement simple et toutes les fonctionnalités de collaboration étaient désactivées.

Pour montrer le plein potentiel de notre approche, nous présentons dans cette section la manière dont la méthodologie MCEV pourrait être utilisée pour adapter les applications de réservation et de guidage aux environnements présentés en introduction.

Nous reprenons dans ces exemples les composants présentés dans le chapitre 3.5 (voir figures 6.8 et 6.9).

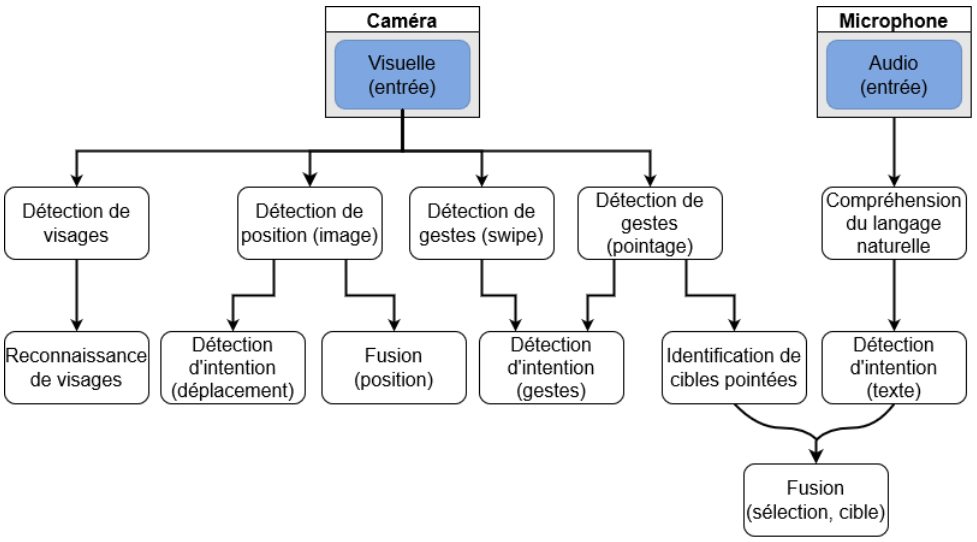


FIGURE 6.8 – Composants CPS et composants d'interprétation qui peuvent être utilisés pour générer les chaînes d'interaction.

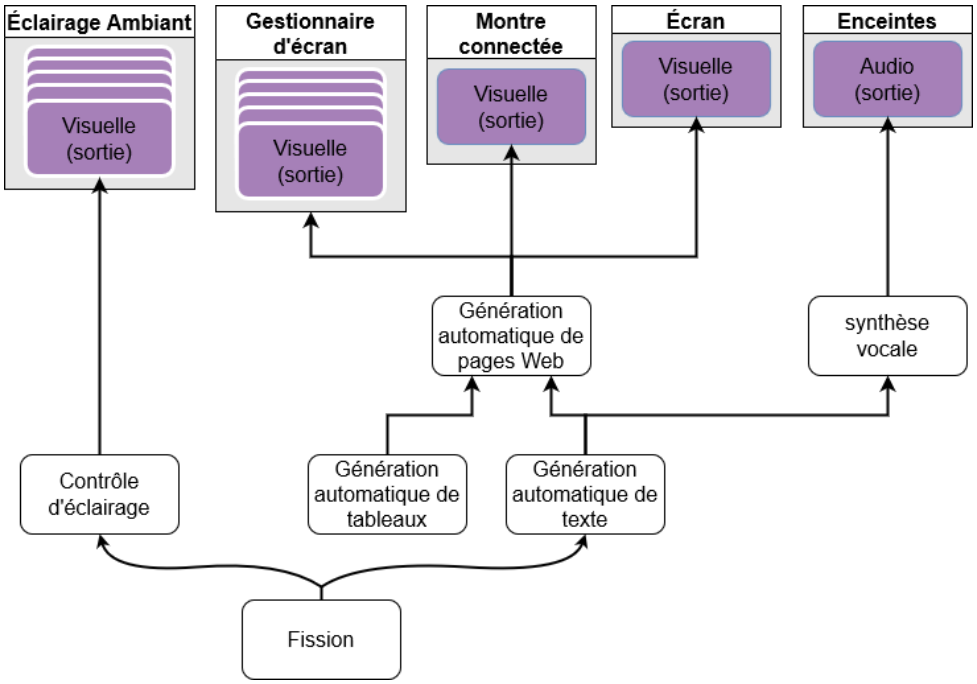


FIGURE 6.9 – Composants CPS et composants de présentation qui peuvent être utilisés pour générer les chaînes d'interaction.

FIGURE 6.10 – Graphes représentant les composants (interprétation, présentation et CPS) réalisés pour les deux applications. Les composants d'interprétation sont dans le graphe 6.8, et les composants de présentation sont dans le graphe 6.9. Les flèches entre les composants représentent les compositions possibles. Les flèches avec accolades représentent le lien entre les entrées des composants de fusion et les sorties des composants de fission.

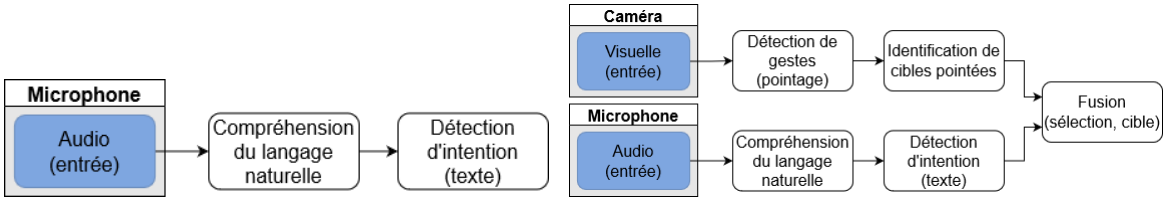
Pour chacun des cas d'usage, nous détaillons le travail d'évaluation qu'un ergonome pourrait réaliser lors de la phase d'intégration. Nous présentons ensuite comment les recommandations de cet ergonome pourraient aider l'administrateur d'un bâtiment à installer l'application.

6.2.1 Cas d'usage de réservation

Phase d'intégration : évaluation et recommandations

Pendant l'intégration, un ergonome évalue les différentes techniques d'interaction développées, et l'impact du positionnement des objets sur l'espace d'interaction.

Par exemple, pour sélectionner un élément dans la liste de créneaux, l'ergonome envisage deux alternatives de technique d'interaction. Les portions des chaînes d'interaction représentant ces deux alternatives sont représentées en figures 6.11a et 6.11b. La première consiste à réserver un créneau uniquement par commande vocale (ex. "je veux réserver le créneaux de 14h heure à 16 heure le Jeudi 12 Janvier"). La deuxième technique permet d'identifier par un geste de pointage le créneau désiré, et par commande vocale l'intention de réserver.

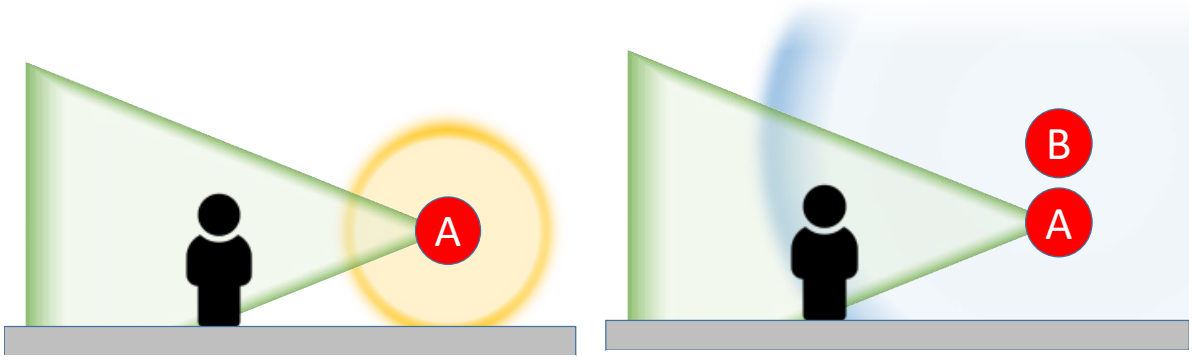


(a) (b)

FIGURE 6.11 – Les deux portions des chaînes d'interaction pour l'application de réservation de salles de réunion. La chaîne de composants (a) permet de sélectionner un créneau avec une seule commande vocale. La chaîne de composants (b) permet de sélectionner un créneau en fusionnant une commande vocale et un geste de pointage.

Pour évaluer les différentes alternatives, l'ergonome utilise un casque de RV pour lancer l'outil MCEV. Son premier objectif est de valider si les techniques d'interaction fonctionnent bien avec l'application de réservation. Il n'a donc pas besoin d'un modèle d'environnement complexe. C'est pour cela que l'ergonome sélectionne depuis l'outil l'environnement virtuel vide disponible par défaut. Pour tester les deux techniques d'interaction, l'ergonome a besoin d'une caméra et d'un microphone. Il choisit parmi les modèles

3D d'objets connectés une webcam équipée d'un microphone. Il place la webcam en face de lui pour être dans l'*aura* de la caméra, comme illustré figure 6.12a.



(a) (b)

FIGURE 6.12 – Schéma du placement des modèles 3D des objets par rapport à l'avatar de l'ergonome. La webcam placée à la position A et le microphone en B. L'*aura* de la caméra est représentée en vert, l'*aura* du microphone intégré est représentée en jaune, et l'*aura* du microphone sur pied est représentée en bleu.

Cependant, l'*aura* du microphone intégré n'est pas très grande, et il faudrait être à moins d'un mètre pour être entendu. Or, les bras ne peuvent pas être captés par la caméra si l'utilisateur est à cette distance. C'est pour cela que l'ergonome explique dans une note que l'utilisation d'une webcam avec la technique d'interaction de pointage est à déconseiller, puis il place cette note sur le modèle 3D de la webcam pour que la note soit associée à l'usage de webcams.

Pour pouvoir tester la technique d'interaction par pointage, il ajoute un microphone qu'il place au dessus de la webcam. Comme illustré figure 6.12b, cette configuration permet bien à l'utilisateur d'être dans les deux zones.

Il configure ensuite l'application pour utiliser la technique d'interaction par pointage avec les deux objets installés, et teste le système pour vérifier que l'application ainsi configurée fonctionne. Il sauvegarde sa configuration (ce qui inclue l'environnement et les objets virtuels), et organise une expérience utilisateur en envoyant cette configuration à un groupe de testeurs. Ces testeurs ont tous un équipement de RV, ce qui permet de réaliser toutes les séances en parallèle. Le comportement des utilisateurs dans le monde virtuel, les rapports du système et les notes réalisées dans l'outil sont collectés, et envoyés

à l'ergonome. Des questionnaires comme celui présenté dans [162] sont aussi complétés par les participants pour obtenir leur ressenti sur l'interaction. Avec ce processus, l'ergonome n'a pas besoin de préparer un espace d'expérimentation ou de faire venir des testeurs, car tout peut se faire à distance avec un équipement unique.

Ensuite, l'ergonome peut rapidement créer de nouvelles configurations à partir de cette première phase. Contrairement à une expérimentation dans un environnement réel, il peut tester facilement et rapidement différents modèles ou positions de caméra. De plus, il peut passer instantanément d'un environnement à un autre, et cela sans avoir à déplacer de matériel de test. L'ergonome peut donc évaluer dans une même séance avec des testeurs des configurations très différentes. Préparer une grande variété de configurations à plusieurs avantages :

— changer d'objets permet d'identifier les objets les plus adaptés pour l'interaction ;
— changer de technique d'interaction permet d'estimer l'interaction préférée des utilisateurs selon leurs profils ;
— changer de positionnement des objets permet d'identifier le rapport entre le placement des objets et la qualité de l'interaction ;
— changer d'environnement permet d'identifier l'impact d'un changement d'espace d'interaction disponible sur la qualité de l'interaction.

Par exemple, dans la phase 3 du dialogue, l'ergonome peut vérifier si les testeurs préfèrent avoir la liste des créneaux transmise par messages audio ou par affichage graphique.

Les problèmes rencontrés lors de ces tests (ex. dysfonctionnement du système) peuvent être transmis à l'équipe de design et développement. Les configurations et les données collectées pendant les tests facilitent l'identification du problème.

Ensuite, l'ergonome identifie parmi les configurations testées celles qui pourraient servir d'exemple au déploiement. Il ajoute pour chacune de ces configurations les notes contenant des conseils et avertissements définis à partir des évaluations réalisées. Par exemple, l'ergonome conseille de n'utiliser l'interaction par caméra de profondeur qu'à 5 conditions :

— que la caméra de profondeur soit de bonne qualité (1280 x 720, ex. Realsense D435) ;
— que la caméra soit au dessus ou en dessous de l'écran, et que l'écran soit à hauteur du regard ;
— que l'espace d'interaction soit suffisamment profond (entre 2m et 5m) ;
— qu'il y ait une bonne luminosité ;

— qu'il n'y ait aucune fenêtre orientée dans le champ de vision de l'utilisateur.

En résumé, l'ergonome a pu évaluer l'application de réservation de salles de réunion dans une grande variété de configurations. Pour cela, il n'avait besoin que de jumeaux numériques d'objets connectés, de modèles 3D d'environnements et de l'équipement de Réalité Virtuelle pour lui et ses testeurs. Il n'a donc pas perdu de temps à installer du matériel physique dans des espaces de test, et il n'a pas eu besoin d'organiser des séances d'expérimentation pour chaque testeur.

Phase de déploiement : configuration et tests

L'administrateur du bâtiment télécharge l'application de réservation. Ensuite, l'administrateur lance l'outil MCEV pour définir les dispositifs physiques et les techniques d'interaction que l'administrateur souhaite utiliser. Il a accès à un jumeau numérique de son bâtiment qu'il peut charger dans l'outil, ainsi que les différents environnements de tests qui illustrent les configurations possibles de l'application.

Depuis l'outil, l'administrateur sélectionne l'application de réservation de salle de réunion, et parcourt les chaînes d'interaction pour avoir une vue d'ensemble des alternatives possibles. Après avoir comparé toutes les configurations, il hésite encore sur un point : la sélection d'un créneau de réservation par commande vocale lui semble simple à réaliser, mais la technique d'interaction par geste de pointage est peut-être plus rapide. Pour départager les deux méthodes, il sélectionne un environnement de test qui intègre la première configuration. La méthode est simple, mais les phrases sont longues à dire.

Il teste ensuite la deuxième configuration dans l'environnement de test associé, et la méthode de pointage lui convient parfaitement.

Il décide donc de tester la configuration qui utilise une caméra de profondeur, un écran, un microphone et des enceintes (configuration du scénario présenté en introduction) dans toutes les salles de réunion. Les salles de réunion n'ont pas l'équipement nécessaire pour cette application. Il ajoute donc ces objets dans une des salles du jumeau numérique du bâtiment, et il les place dans un endroit facile d'accès mais discret. Pour cela, il s'inspire des configurations et des recommandations ergonomiques qui se trouvaient dans les environnements de test. En particulier, les 5 critères ergonomiques sur le choix et le positionnement d'une caméra étaient marquées dans une note associée à la caméra.

Ensuite, l'administrateur sélectionne les techniques d'interaction pour les 4 tâches d'interaction successives (voir chapitre 4.2.1), et associe les objets virtuels nécessaires pour chaque technique.

Une fois la configuration réalisée, l'administrateur teste l'application dans l'environnement virtuel. Pour chaque problème dans l'interaction, l'administrateur peut repositionner les objets (ex. modifier l'angle de la caméra pour observer plus en hauteur) ou changer de technique d'interaction.

Après avoir validé sa configuration, l'administrateur utilise les fonctionnalités de l'outil pour générer plusieurs livrables :

— il rédige des notes qu'il associe à chaque objet, et prend des captures d'écran (i.e. vue du dessus de la pièce et vue à la première personne). Les captures lui permettent de montrer où les objets doivent être placés, et les notes donnent des informations supplémentaires sur ce placement (ex. la caméra doit être connectée à un ordinateur, un support mural doit être installé pour l'écran) ;

— il s'enregistre pendant la phase de test pour générer un double de son avatar qui interagit avec le système. Cela lui permet d'enregistrer une vidéo montrant son double interagir avec le système depuis le point de vue externe de l'avatar que l'administrateur contrôle. Cette vidéo peut ensuite être partagée avec les utilisateurs finaux pour leur montrer comment utiliser le système. Le double peut aussi être importé dans un logiciel tiers de Réalité Augmentée pour montrer l'usage de l'application dans l'environnement réel.

Enfin, l'administrateur répète les phases de configuration et génération de livrables pour toutes les salles de réunion dans lesquelles l'application de salle de réunion doit être installée.

6.2.2 Cas d'usage de guidage

La phase d'intégration et de déploiement de l'application de guidage reprend la plupart des étapes de l'application de réservation. Il y a toutefois plusieurs points de différences que nous présentons par la suite.

Phase d'intégration : évaluation et recommandations

L'ergonome doit s'assurer que l'application permet bien de guider un utilisateur dans des bâtiments de topologies différentes, voire au travers de plusieurs bâtiments et étages. Un bâtiment peut par exemple être représentés comme un graphe 3D constitué de pièces, d'escaliers, d'ascenseurs, de couloirs et de portes [156]. Contrairement à l'application de réservation, les environnements testés doivent donc être complexes. Par exemple, l'er-

gonome peut tester l'application dans un modèle 3D de labyrinthe. Un labyrinthe est souvent constitué de couloirs, virages et croisements. L'ergonome peut donc tester pour ces 3 éléments les modalités les plus adaptés. Par exemple, il peut réaliser une expérience pour savoir si les utilisateurs préfèrent être guidé par messages vocales ou à l'aide de textes affichés sur des écrans. L'ergonome peut aussi vérifier pour chaque configuration si l'utilisateur perçoit bien les indications. En effet, un message vocal transmis par une enceinte trop éloignée du chemin, ou avec un volume trop faible peut être problématique. De même, un environnement fréquenté peut distraire l'utilisateur. Il faut donc pouvoir tester l'application dans ces conditions.

Pour avoir les mêmes conditions dans toutes les expériences utilisateurs, la technique d'enregistrement et re-jeu du comportement des utilisateurs est utilisé ici. En effet, l'ergonome peut s'enregistrer pendant qu'il parcourt l'environnement de test. En rejouant ces enregistrements, l'ergonome crée ainsi des agents se déplaçant selon un trajet prédéfini. Ces agents peuvent prendre le rôle d'éléments perturbateurs (ex. un agent qui s'arrête devant un écran et gêne la vue de l'utilisateur), mais peuvent aussi représenter l'utilisateur de l'application. Par exemple, les testeurs peuvent être amenés à suivre un enregistrement de personne guidée pour obtenir leur ressenti (ex. la gêne occasionnée par un usage individuel de l'éclairage et des écrans).

Phase de déploiement : configuration et tests

Chaque instance de l'application de guidage est associée à un utilisateur final (i.e. un nouvel arrivant dans le bâtiment), et non à une pièce comme dans l'application de réservation. L'administrateur n'installe donc l'application qu'une seule fois pour tout le bâtiment. Le bâtiment en question est équipé d'écrans, d'enceintes et d'ampoules connectés, et l'administrateur souhaite les utiliser pour guider l'utilisateur. Avant de déployer l'application, l'administrateur décide de tester la configuration dans une modélisation 3D du bâtiment. Son test consiste à partir du portique d'entrée pour atteindre des bureaux situés à plusieurs endroits du bâtiment. C'est un test qui aurait été difficile à réaliser en réel : cela aurait dérangé les occupants du site, surtout si le système de guidage dysfonctionne. Les tests étant satisfaisant, il peut maintenant installer l'application dans l'environnement réel.

Application de la méthodologie aux cas d'usage de réservation et de guidage

L'application de la méthode MCEV à la configuration des applications de réservation et de guidage permet d'illustrer l'efficacité et la simplicité de notre approche pour configurer et évaluer des MIBS dans des environnements complexes. Les configurations requises par notre framework peuvent être générées par l'outil MCEV sans nécessiter de connaissance technique, et la visualisation des *auras* permet d'anticiper plus facilement les zones d'interaction. Enfin, les fonctionnalités de rejeu des comportements et de "monde en miniature" permettent de tester le fonctionnement du système en présence de plusieurs occupants du site, et cela sans recourir à des agents autonomes ou des expérimentations utilisateurs.

6.3 Synthèse

Dans ce chapitre, nous avons étudié l'application de la méthode MCEV dans des cas pratiques.

Tout d'abord, nous avons mené une expérience utilisateur pour comparer l'efficacité et la convivialité de notre outil à celles d'un *outil graphique* que nous avons développé pour adapter notre processus méthodologique aux expériences sur le terrain. Les résultats montrent que le processus de configuration et d'évaluation est plus efficace avec notre outil de RV, et qu'il a au moins la même facilité d'utilisation que l'*outil graphique*. Ainsi, nous pensons que notre méthodologie et notre outil basés sur la RV proposent une alternative valable aux expériences sur le terrain et pourraient participer à l'arrivée de ces systèmes dans notre quotidien.

Ensuite, nous avons présenté comment la méthode MCEV peut être appliquée aux cas d'usage de réservation et de guidage. Nous avons illustré comment un ergonome peut travailler sur des prototypes des MIBS pour faciliter leur déploiement. Nous avons aussi montré comment un administrateur d'un bâtiment du tertiaire peut facilement configurer et tester les MIBS pour qu'ils soient adaptés à l'environnement d'exécution.

Conclusion et perspectives

Conclusion

Dans cette thèse, nous avons vu que les MIBS (i.e. systèmes multimodal utilisant des objets connectés comme interfaces d'interaction) permettent de profiter au mieux des objets connectés qui nous entourent pour fournir des interactions accessibles et naturelles dans notre quotidien. Toutefois, la création de tels systèmes pour une grande variété d'environnements est encore un challenge. C'est pour cette raison que nous proposons un framework, une méthodologie et des outils associés pour faciliter l'adaptation d'un MIBS aux spécificités des environnements. Nous avons illustré nos travaux au travers de deux cas d'usage : une application de réservation de salle de réunion, et une application de guidage d'un utilisateur dans un bâtiment.

Notre analyse de l'état de l'art nous a permis de définir les 15 exigences auxquelles ces systèmes doivent répondre. Nous avons distingué en particulier 6 exigences qui régissent la capacité d'adaptation des MIBS aux environnements. Ces 6 exigences sont : l'abstraction des dispositifs physiques, la perception du contexte, le support de l'interopération spontanée, le choix libre de modalités et combinaison de modalités, la gestion de la multimodalité massive et l'abstraction des modalités.

Ensuite, nous avons montré au travers d'une étude des standards architecturaux et des frameworks existants qu'il n'y a pas d'approche pour répondre à toutes les exigences. En particulier, nous avons établi qu'aucune solution ne permet de satisfaire les 6 exigences liées à l'adaptation des MIBS aux environnements, et qu'une intervention humaine est nécessaire pour s'assurer de l'utilisabilité du système au déploiement. Nous avons ensuite présenté et développé un framework qui s'inspire des différentes solutions aux exigences apportées par la littérature. Ce framework s'intègre aux solutions logicielles fournies par Orange, et il nous sert par la suite de base logicielle pour proposer une méthodologie facilitant le processus d'adaptation.

Notre analyse des méthodes et outils utilisés pour concevoir et développer les MIBS nous a permis de définir le cycle de vie d'un MIBS qui prend en compte son adaptation aux environnements d'exécution. Cette étude nous a aussi permis d'identifier les tâches, acteurs, outils et moments opportuns pour effectuer cette adaptation de la manière la plus simple possible. En particulier, nous avons identifié que le processus de déploiement par un administrateur du bâtiment (i.e. l'environnement d'exécution) n'était pas assez outillé pour pouvoir efficacement configurer les MIBS.

Notre analyse nous a permis de proposer une méthode de configuration et d'évalua-

tion en réalité virtuelle (MCEV). Cette méthode facilite les expérimentations de plusieurs configurations d'applications (ex. choix des objets, des techniques d'interaction) en utilisant un casque de réalité virtuelle pour placer l'utilisateur dans un jumeau numérique de l'environnement réel dans lequel il souhaite déployer un MIBS. Ainsi, notre méthode offre aux utilisateurs une solution complète pour aider un administrateur à configurer un MIBS en fonction de l'environnement d'exécution. L'usage de la Réalité Virtuelle (RV) permet de ne pas avoir besoin de tester le système dans l'environnement réel, et donc d'éviter les contraintes matérielles que les tests sur le terrain impliquent (ex. accès à l'environnement, gestion de l'équipement). De plus, un administrateur n'a pas besoin de compétences avancées en programmation, design ou ergonomie pour configurer le système. Au contraire, il peut tester dans des environnements virtuels des configurations réalisées et validées par des ergonomes, ce qui peut guider l'administrateur dans ses choix.

L'expérience utilisateur réalisée permet de valider que la méthode MCEV permet de réaliser des configurations similaires à une installation directement dans l'environnement réel, et que le processus de configuration et d'évaluation est plus efficace en réalité virtuelle que sur le terrain. L'application de notre méthodologie aux cas d'usage de réservation et de guidage montre à quel point il est facile d'adapter des MIBS à une grande variété d'environnements.

Ainsi, nous montrons au travers de cette thèse une approche permettant de faciliter l'adaptation des MIBS aux spécificités des environnements. Pour cela, nous intégrons au processus de création de MIBS une méthodologie pour permettre de facilement configurer les MIBS, ce que les outils et frameworks existants n'abordent pas assez.

Perspectives

Les perspectives de recherche envisagées concernent l'évolution de notre outil de configuration et d'évaluation pour utiliser au mieux la capacité d'immersion de l'utilisateur dans une modélisation 3D de l'environnement réel.

Extension et amélioration de l'outil MCEV

L'utilisabilité de l'outil MCEV lui-même est encore perfectible, et plusieurs améliorations peuvent y être apportées. Par exemple, la description des techniques d'interaction dans le menu de configuration était sous un format textuel. Le visionnage de courtes vi-

déos démontrant ces techniques d'interaction pourrait être plus intuitif et agréable pour l'utilisateur.

De plus, nous n'avons développé que les composants CPS d'interprétation, de présentation et de dialogue requis pour notre expérimentation et pour interagir avec les objets connectés à notre disposition. Or, pour réaliser la phase de test avec l'outil MCEV, le MIBS doit utiliser des composants existants. Pour pouvoir explorer d'autres cas d'usage, de nouveaux composants doivent donc être développés.

Assistance au placement d'objet

Le processus de configuration peut être facilité lors du placement des objets dans l'environnement . En effet, une piste d'évolution de l'outil MCEV est de proposer des zones de recommandations (et avertissements) dans l'environnement virtuel quand un utilisateur souhaite ajouter de nouveaux objets. La détection et visualisation de ces zones permettrait de faire gagner du temps à la configuration, et d'aider à réaliser des configurations plus fiables. Les travaux sur l'optimisation du déploiement de *réseaux de capteurs sans fil* peuvent être utilisés pour proposer une méthode d'ajout de dispositifs dans de grands espaces. Par exemple, Afghantoloee et Mostafavi [2] proposent une méthode pour optimiser le placement de capteurs dans des modèles 3D d'environnements d'intérieur. Pour cela, Leur algorithme utilise les zones que chaque capteur peut percevoir (i.e. les *auras*).

Exploration de l'utilisation de la RA

Nous utilisons la RV dans notre méthodologie de configuration car cela permet de se passer de l'environnement réel, ce que ne permet pas la RA. Toutefois, la RA permet de configurer directement les dispositifs physiques tout en profitant des fonctionnalités présentées dans le chapitre 5. Ainsi, la configuration n'a pas besoin d'être reportée de l'environnement virtuel au monde réel.

La RA pourrait aussi être utilisé pour faciliter cette transition. En effet, la configuration réalisée avec l'outil MCEV pourrait être colocalisée dans l'environnemt réel, et l'administrateur n'aurait plus qu'à superposer la position des dispositifs physiques aux modèles 3D des objets virtuels.

Étude de la collaboration entre acteurs

Nous n'avons évalué notre méthodologie qu'au travers d'une expérimentation sur la phase de configuration avec des utilisateurs qui n'avaient, pour la plupart, jamais installé ou utilisé d'applications multimodales ou ubiquitaires. Nous n'avons donc pas étudié l'impact de cette méthode sur la collaboration entre les acteurs intervenant dans le processus de création de MIBS. Une perspective d'expérimentation serait de regrouper des designers, développeurs, ergonomes et administrateurs, et de leur demander de concevoir, développer et installer un MIBS en utilisant notre méthodologie.

Annexe A : Cas d'usage sur l'application de réservation de salle de réunion

Dans cette annexe, nous montrons comment chaque phase du scénario présenté dans l'introduction peut être réalisée avec un MIBS créé avec notre framework (voir chapitre 3). Plus spécifiquement, nous montrons des exemples de besoins du composant de dialogue, et de chaînes d'interaction qui peuvent être générées. Nous considérons pour cela que les composants utilisés ont été développés au préalable.

Les 5 étapes du scénario de réservation de salle de réunion sont détaillées dans les sections suivantes. Pour chaque étape, les besoins du composant de dialogue et la chaîne d'interaction choisie par le gestionnaire de dialogues sont présentés.

A.1 Phase 1 : engagement

Déroulement de la phase 1 dans le scénario

Dominique entre dans la salle de réunion et s'arrête devant l'interface murale.

```
{
"application_reservation" :
 {
  "user-hi":
   {
    "optionnel" : Faux,
    "source" : "entrée",
    "préférence" : {},
    "requis" : {"id_utilisateur"},
    "canal" : 0
   }
 }
}
```

FIGURE A.1 – Besoins du composant de dialogue pour la phase 1, exprimés au format Json.

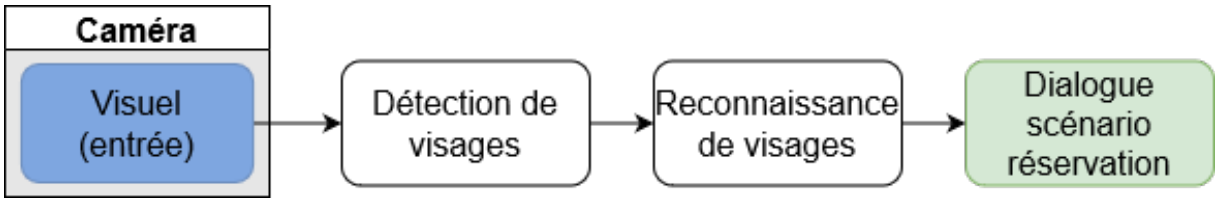


FIGURE A.2 – Exemple de chaîne d'interaction pour la phase 1.

A.2 Phase 2 : détection de l'intention de réservation

Déroulement de la phase 2 dans le scénario

L'écran s'allume sur un message de bienvenue, et le système l'interpelle vocalement :
- "Bonjour Dominique. Que souhaitez-vous faire ?"
- "je voudrais réserver cette salle début octobre. Indique-moi les jours où elle est libre toute la journée".

```
{
{
"application_reservation" :
 {
  "request":
   {
    "optionnel" : Faux,
    "source" : "entrée",
    "préférence" : {},
    "requis" : {"nom_service"},
    "canal" : 0
   },
  "greetings":
   {
    "optionnel" : Faux,
    "source" : "sortie",
    "préférence" : {"modalité" : "audio"},
    "requis" : {"cible" : "id_utilisateur"},
    "canal" : 0
   },
  "greetings":
   {
    "optionnel" : Vrai,
    "source" : "sortie",
    "préférence" : {"modalité" : "visuelle"},
    "requis" : {"cible" : "id_utilisateur"},
    "canal" : 1
   }
 }
}
```

FIGURE A.3 – Besoins du composant de dialogue pour la phase 2, exprimés au format Json.

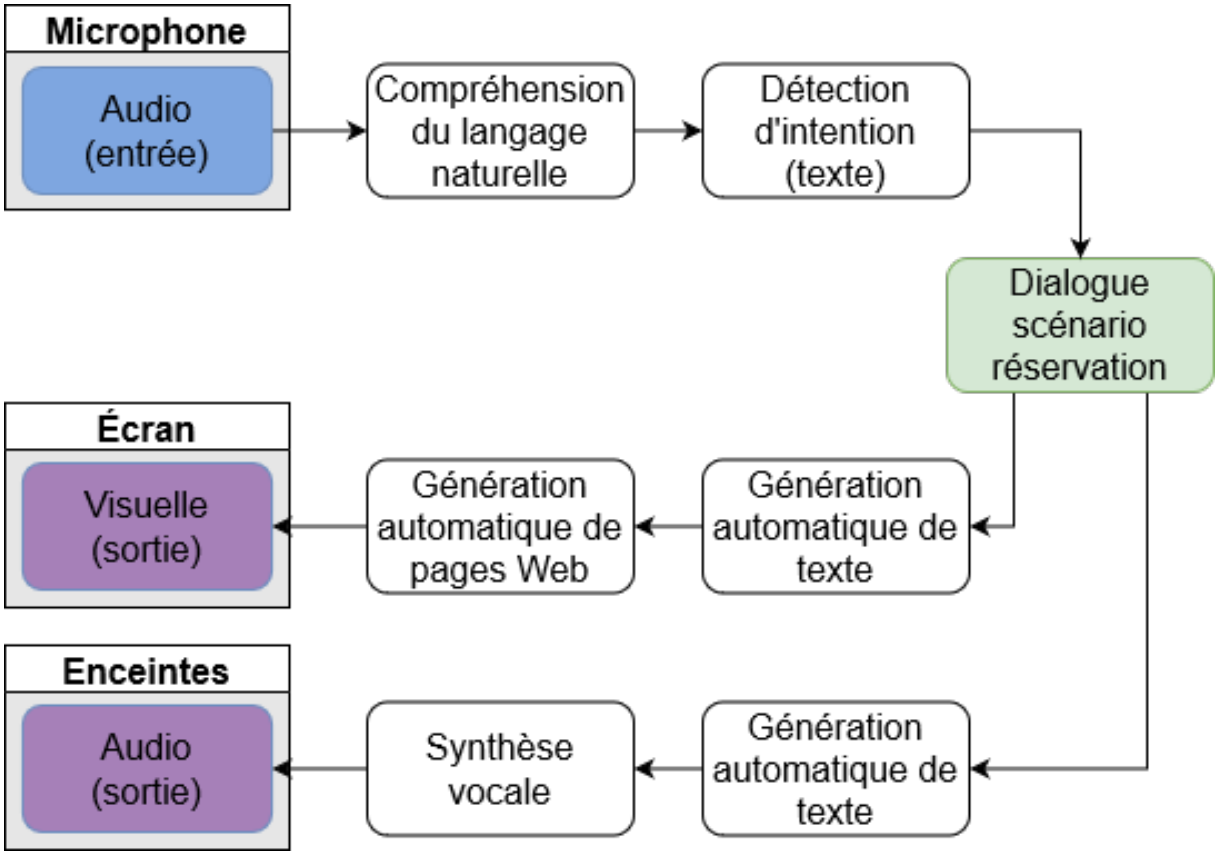


FIGURE A.4 – Exemple de chaîne d'interaction pour la phase 2.

A.3 Phase 3 : sélection du créneau horaire

Déroulement de la phase 3 dans le scénario

- "Voici les créneaux disponibles". Une liste de créneaux s'affiche sur l'écran d'affichage.
Dominique fait défiler la liste de mouvement de balayage de la main, puis pointe du doigt un créneau libre en date du 5 Octobre et dit :
- "Celui-ci."

```
{
"application_reservation" :
 {
  "reqalts":
   {
    "optionnel" : Faux,
    "source" : "entrée",
    "préférence" : {},
    "requis" : {"cible" : ["précédent", "suivant"]},
    "canal" : 0
   },
  "user-confirm":
   {
    "optionnel" : Faux,
    "source" : "entrée",
    "préférence" : {},
    "requis" : {"cible"},
    "canal" : 1
   },
  "inform":
   {
    "optionnel" : Faux,
    "source" : "sortie",
    "préférence" : {"modalité" : "audio"},
    "requis" : {"type" : "créneaux", "disponible" : Vrai},
    "canal" : 0
   },
  "offer":
   {
    "optionnel" : Faux,
    "source" : "sortie",
    "préférence" : {"modalité" : "visuelle"},
    "requis" : {"type" : "créneau"},
    "canal" : 1
   }
 }
}
```

FIGURE A.5 – Besoins du composant de dialogue pour la phase 3, exprimés au format Json.

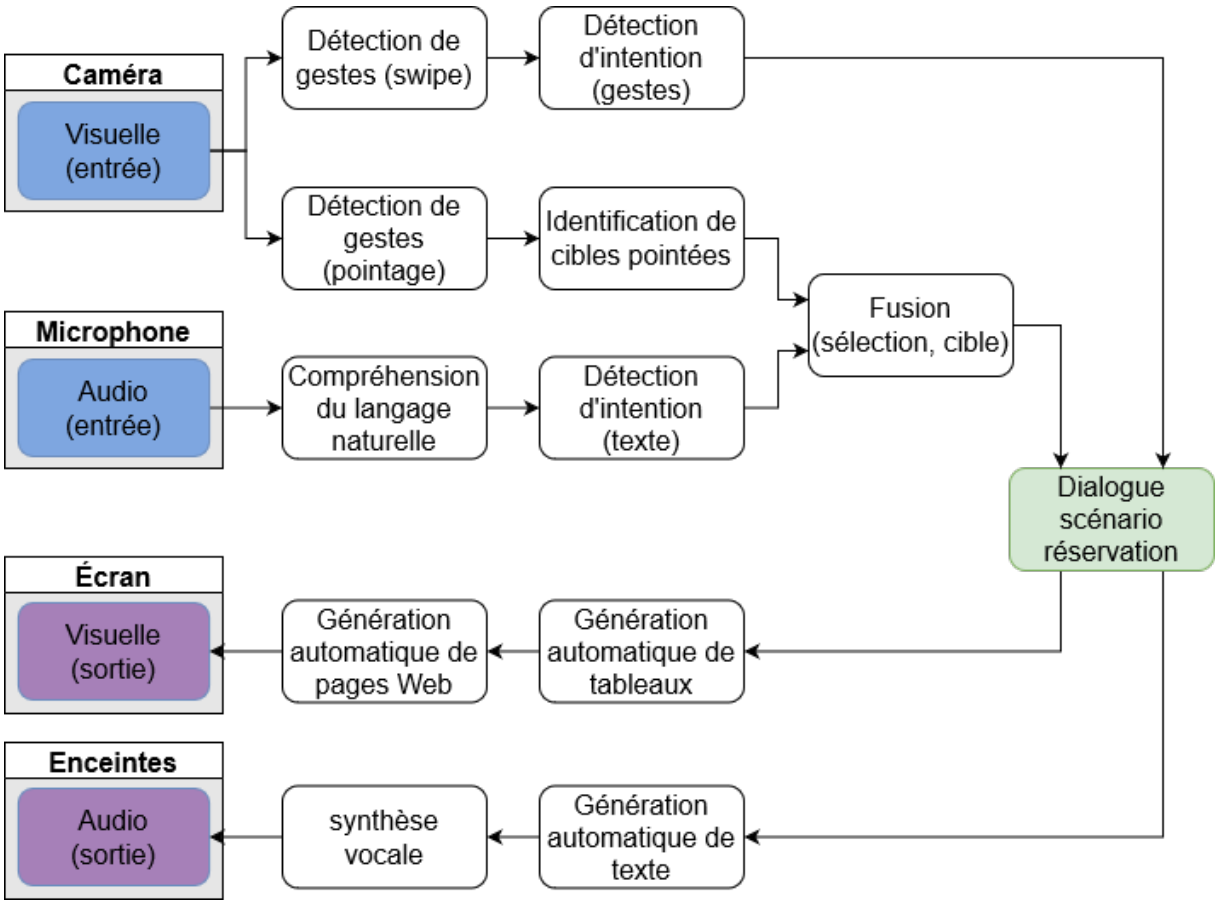


FIGURE A.6 – Exemple de chaîne d'interaction pour la phase 3.

A.4 Phase 4 : partage de l'invitation

Déroulement de la phase 4 dans le scénario

- "Très bien. Veuillez compléter l'invitation sur votre appareil personnel."
Dominique ouvre depuis son smartphone l'invitation générée pour y ajouter l'objet de la réunion ainsi que les adresses mails des autres participants.

```
{
"application_reservation" :
 {
  "inform":
   {
    "optionnel" : Faux,
    "source" : "sortie",
    "préférence" : {},
    "requis" : {},
    "canal" : 0
   }
 }
}
```

FIGURE A.7 – Besoins du composant de dialogue pour la phase 4, exprimés au format Json.

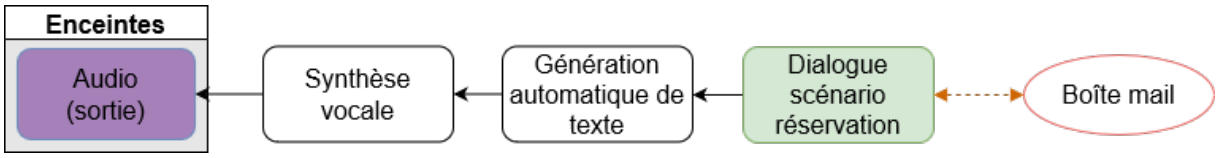


FIGURE A.8 – Exemple de chaîne d'interaction pour la phase 4.

A.5 Phase 5 : désengagement

Déroulement de la phase 5 dans le scénario

Après avoir envoyé l'invitation, Dominique quitte le champ de la caméra. En réponse, l'écran s'éteint, et le système énonce :
- "Rendez-vous ici le 5 Octobre. Bonne journée!"

```
{
"application_reservation" :
 {
  "bye":
   {
    "optionnel" : Faux,
    "source" : "entrée",
    "préférence" : {},
    "requis" : {},
    "canal" : 0
   },
  "bye":
   {
    "optionnel" : Faux,
    "source" : "sortie",
    "préférence" : {},
    "requis" : {},
    "canal" : 0
   }
 }
}
```

FIGURE A.9 – Besoins du composant de dialogue pour la phase 5, exprimés au format Json.

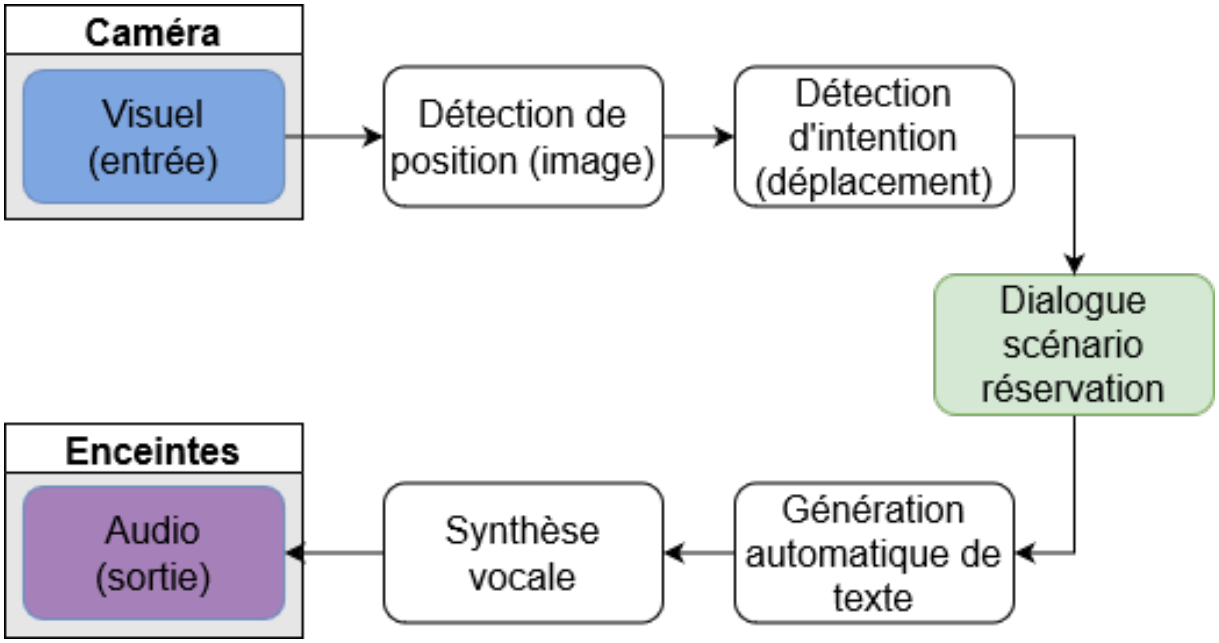


FIGURE A.10 – Exemple de chaîne d'interaction pour la phase 5.

Annexe B : Instructions pour la phase d'apprentissage de l'expérimentation dans l'environnement réel

L'expérimentation réalisée sur la méthode MCEV était en deux parties : une partie en RV, et l'autre partie directement dans l'environnement réel. La suite de cette annexe contient les instructions que les participants avaient dans la phase d'apprentissage de la partie dans l'environnement réel.

Phase d’apprentissage dans l’environnement réel

Pendant cette phase d’apprentissage, votre objectif est de **configurer** puis **tester** un service permettant d’allumer ou éteindre depuis les fauteuils orange les ampoules connectées de la pièce où vous êtes. Vous utiliserez un microphone pour écouter les instructions de l’utilisateur.

Vous aurez en bas de l’interface un bouton d’aide qui ouvre une fenêtre donnant un numéro. Ces numéros font références aux instructions associées à chaque étape dans l’annexe INSTRUCTIONS.

Dans la phase de **test**, un rappel de la méthode d’interaction, les scores de placement des objets, et les événements survenant au cours de la phase de test s’afficheront sur une pop-up.

I Configuration des objets connectés

La première étape est de sélectionner les objets connectés que vous allez utiliser pendant l’interaction, et les placer dans l’environnement

•Dans le *Menu principal*, cliquez sur *Gestion des objets connectés*. Cela va vous permettre d’identifier les objets connectés disponibles et de définir leurs emplacements dans l’environnement.

•Vous avez besoin des deux ampoules connectés et du microphone présents dans la pièce. Dans le menu déroulant, veuillez choisir une des ampoules connectées. Il est possible que certains objets n’ont pas été synchronisés, cliquez sur le bouton en haut à droite pour rafraîchir la liste.

•L’ampoule sélectionnée a un identifiant. En revanche vous ne savez pas à quelle ampoule cet identifiant correspond. Pour cela, on va utiliser une méthode d’identification. Chaque objet à sa propre méthode d’identification. Dans le cas des ampoules, il s’agit de faire varier leurs intensités. Utilisez maintenant le bouton *identifier* et repérez l’ampoule qui clignote. S’il s’agit d’une ampoule de la pièce, il faut l’ajouter à la carte pour pouvoir l’utiliser ultérieurement dans une configuration, pour cela cliquez sur *ajouter*. Sinon, essayez une autre ampoule.

•Il faut ensuite rentrer dans l’outil l’emplacement de l’objet ajouté. Certains services ont besoin de savoir où sont les objets connectés pour fonctionner correctement, cette étape est donc essentielle. Placez-vous dans le mode *déplacement*, puis faites glisser l’icône représentant l’ampoule (l’hexagone vert) pour être au même endroit que l’ampoule réelle (celle qui vient de clignoter).

•Vous pouvez aussi définir l’orientation des objets avec le mode *rotation*, ce qui sera utile plus tard. En effet, un score s’affichera en début de phase de **test**. Ce score représente la validité théorique du placement des objets (position et angles). Le résultat sera jugé satisfaisant si le score de chaque objet est supérieur à 75 %. Il est conseillé de se servir des zones d’action pour améliorer le score.

•Faites de même pour l’autre ampoule de la pièce.

•Il est possible que l’on souhaite ajouter et placer de nouveaux objets dans l’environnement. Dans notre cas, le microphone est un ajout par rapport à l’équipement préexistant de la pièce. Placez le microphone de telle façon que votre voix soit bien détectée. Pour cela, aidez-vous de l’outil d’identification qui affiche l’évolution de l’intensité du signal sonore dans le temps.

•Reportez la position du microphone dans l’outil comme pour les ampoules connectées.

•*Validez* pour revenir dans le menu principal un fois terminé.

II Configuration d’un service

La deuxième étape est d’ajouter une configuration au *service de contrôle de la luminosité*. Pour cela, il faut choisir le service, choisir une méthode d’interaction adaptée aux objets connectés désirés, puis associer ces objets à cette configuration.

•Allez dans *Gestion des services*. Vous trouverez dans cette fenêtre toutes les configurations que vous allez ajouter.

•Vous n’avez pas de configuration pour l’instant, et vous devez en ajouter une nouvelle. Dans le menu *Gestion des services*, cliquez sur *nouveau*.

•Il n’y a qu’un seul service disponible ici, celui que l’on souhaite mettre en place. Dans *Ajout d’un service*, choisissez *contrôle de la luminosité*, puis faites S*uivant*.

•Chaque service peut offrir différentes fonctionnalités. Il y a donc un descriptif de ce qu’il est capable de faire. Lisez le descriptif du service, puis faites *Configurer*.

•Vous avez maintenant le choix entre plusieurs méthodes d’interaction, et chacune fournit une fonctionnalité et requiert différents types d’objets connectés. Pour naviguer entre les différentes méthodes d’interaction, cliquez sur *configuration précédente ou configuration suivante.* Choisissez la méthode *contrôle par commande vocale simple* qui permet de réaliser la configuration désirée : piloter les ampoules simplement par commande vocale. Cliquez sur *Sélectionner* dès que vous avez atteint la bonne méthode d’interaction.

III Associer le service avec les objets connectés

•Vous allez maintenant pouvoir renommer votre configuration pour pouvoir l’identifier par la suite. Donnez-lui un nom en remplissant le champ en haut de la fenêtre.

•Il faut maintenant choisir les objets connectés que vous utiliserez pour cette configuration. Une couleur orange ou verte met en avant les objets connectés nécessaires à la méthode d’interaction. La couleur orange signifie qu’il n’y a pas d’objet associé et vert que l’association est faite. Cliquez sur l’icône orange représentant l’ampoule connectée pour réaliser l’association.

•Vous vous retrouvez dans une fenêtre similaire à celle rencontrée plus tôt. L’objectif ici est d’associer les ampoules connectées de la pièce avec le service désiré. Veuillez vous assurer d’être dans le mode *sélection*, puis cliquez sur tous les hexagones représentant les ampoules désirées. Vous avez en dessous de la carte les noms de tous les objets sélectionnés pour l’association. Enfin, validez votre choix en cliquant sur *Associer*.

•L’icône de l’ampoule connecté est devenue verte, l’association a donc réussi !

Il ne vous reste plus qu’à faire de même pour les autres objets connectés, puis cliquer sur T*erminer* pour finaliser la **configuration** et être renvoyé au menu principal.

IV Tester le service

Le bouton vert dans le menu principal permet de lancer/arrêter les configurations installées. **Testez** votre configuration. Vous pouvez supprimer cette configuration pour éviter qu'elle rentre en conflit avec d'autres configurations depuis le menu *Gestion des services*.

Vous venez de finir la phase d'apprentissage. Vous pouvez passer à la phase d'expérimentation si dessous.

Phase d'expérimentation dans l'environnement réel

L'objectif de l'outil de configuration est de rapidement **configurer** et **tester** plusieurs configurations jusqu'à obtenir un résultat satisfaisant.

Pour cette phase, vous aurez donc à fournir rapidement une configuration pour le besoin suivant :

« Je veux pouvoir contrôler toutes les lumières de la pièce depuis les fauteuils orange en les pointant individuellement du doigt et en leur demandant de s'allumer ou s'éteindre »

Pour cela, vous allez avoir besoin d'ajouter une caméra de profondeur (Kinect), et de repositionner le microphone déjà ajouté. Le repositionnement des objets ne sera pris en compte que dans les phases de configuration, mais vous pouvez toutefois réaliser plusieurs cycles de configurations et tests pour obtenir de meilleurs résultats.

Il ne faut pas déplacer ou ajouter d'ampoules : ce ne sera pas possible dans l'environnement final.

BIBLIOGRAPHIE

[1] Hamada ABDULWAKEL et emad nabil emad, « A conflicts' classification for IoT-based services : a comparative survey », in : *PeerJ Computer Science* 7 (avr. 2021), DOI : 10.7717/peerj-cs.480.

[2] Ali AFGHANTOLOEE et Mir Abolfazl MOSTAFAVI, « A Local 3D Voronoi-Based Optimization Method for Sensor Network Deployment in Complex Indoor Environments », en, in : *Sensors* 21.*23* (jan. 2021), Number : 23 Publisher : Multidisciplinary Digital Publishing Institute, p. 8011, ISSN : 1424-8220, DOI : 10.3390/s21238011, URL : https://www.mdpi.com/1424-8220/21/23/8011 (visité le 23/09/2022).

[3] Antonio ALBERTI et al., « Platforms for Smart Environments and Future Internet Design : A Survey », in : *IEEE Access* PP (oct. 2019), p. 1-1, DOI : 10.1109/ACCESS.2019.2950656.

[4] Nuno ALMEIDA et al., « The AM4I Architecture and Framework for Multimodal Interaction and Its Application to Smart Environments », eng, in : *Sensors (Basel, Switzerland)* 19.*11* (juin 2019), ISSN : 1424-8220, DOI : 10.3390/s19112587.

[5] Oscar ARDAIZ, F. FREITAG et Leandro NAVARRO, « On Service Deployment in Ubiquitous Computing », in : (mars 2002).

[6] Carmelo ARDITO et al., « A tool for Wizard of Oz studies of multimodal mobile systems », in : juin 2009, p. 344-347, DOI : 10.1109/HSI.2009.5091003.

[7] Luigi ATZORI, Antonio IERA et Giacomo MORABITO, « The Internet of Things : A survey », en, in : *Computer Networks* 54.*15* (oct. 2010), p. 2787-2805, ISSN : 1389-1286, DOI : 10.1016/j.comnet.2010.05.010, URL : http://www.sciencedirect.com/science/article/pii/S1389128610001568 (visité le 07/01/2021).

[8] Mirjam AUGSTEIN et Thomas NEUMAYR, « A Human-Centered Taxonomy of Interaction Modalities and Devices », in : *Interacting with Computers* 31.*1* (jan. 2019), p. 27-58, ISSN : 0953-5438, DOI : 10.1093/iwc/iwz003, URL : https://doi.org/10.1093/iwc/iwz003 (visité le 23/04/2021).

[9] J. Augusto et al., « A Smart Environments Architecture (Search) », in : *Applied Artificial Intelligence* 34.*2* (jan. 2020), Publisher : Taylor & Francis _eprint : https ://doi.org/10.1080/08839514.2020.1712778, p. 155-186, ISSN : 0883-9514, DOI : 10.1080/08839514.2020.1712778, URL : https://doi.org/10.1080/08839514.2020.1712778 (visité le 07/08/2022).

[10] J. Augusto et al., « The user-centred intelligent environments development process as a guide to co-create smart technology for people with special needs », en, in : *Universal Access in the Information Society* 17.*1* (mars 2018), p. 115-130, ISSN : 1615-5297, DOI : 10.1007/s10209-016-0514-8, URL : https://doi.org/10.1007/s10209-016-0514-8 (visité le 26/08/2022).

[11] Juan C. Augusto et al., « Intelligent Environments : a manifesto », en, in : *Human-centric Computing and Information Sciences* 3.*1* (juin 2013), p. 12, ISSN : 2192-1962, DOI : 10.1186/2192-1962-3-12, URL : https://doi.org/10.1186/2192-1962-3-12 (visité le 28/07/2022).

[12] Juan Carlos Augusto, « Contexts and Context-awareness Revisited from an Intelligent Environments Perspective », in : *Applied Artificial Intelligence* 36.*1* (déc. 2022), Publisher : Taylor & Francis _eprint : https ://doi.org/10.1080/08839514.2021.2008644, p. 2008644, ISSN : 0883-9514, DOI : 10.1080/08839514.2021.2008644, URL : https://doi.org/10.1080/08839514.2021.2008644 (visité le 06/08/2022).

[13] Juan Carlos Augusto et Miguel J. Hornos, « Software simulation and verification to increase the reliability of Intelligent Environments », en, in : *Advances in Engineering Software* 58 (avr. 2013), p. 18-34, ISSN : 0965-9978, DOI : 10.1016/j.advengsoft.2012.12.004, URL : https://www.sciencedirect.com/science/article/pii/S0965997813000033 (visité le 31/08/2022).

[14] Pierre-Alain Avouac, Philippe Lalanda et Laurence Nigay, « Service-oriented autonomic multimodal interaction in a pervasive environment », en, in : *Proceedings of the 13th international conference on multimodal interfaces - ICMI '11*, Alicante, Spain : ACM Press, 2011, p. 369, ISBN : 978-1-4503-0641-6, DOI : 10.1145/2070481.2070552, URL : http://dl.acm.org/citation.cfm?doid=2070481.2070552 (visité le 02/12/2019).

[15] Syafril Bandara et al., « Towards a standard API design for open services in smart buildings », in : *2016 TRON Symposium (TRONSHOW)*, déc. 2016, p. 1-7, DOI : 10.1109/TRONSHOW.2016.7842883.

[16] Barbara R. Barricelli et al., *Designing Pervasive and Multimodal Interactive Systems : An Approach Built on the Field*, English, chap., Archive Location : designing-pervasive-multimodal-interactive-systems ISBN : 9781605663869 Library Catalog : www.igi-global.com Publisher : IGI Global, 2009, URL : https://www.igi-global.com/gateway/chapter/35891 (visité le 08/06/2021).

[17] Barbara Rita Barricelli et Stefano Valtolina, « Designing for End-User Development in the Internet of Things », en, in : *End-User Development*, sous la dir. de Paloma Díaz et al., Lecture Notes in Computer Science, Cham : Springer International Publishing, 2015, p. 9-24, ISBN : 978-3-319-18425-8, DOI : 10.1007/978-3-319-18425-8_2.

[18] Zouhir Bellal et al., « Integrating Mobile Multimodal Interactions based on Programming By Demonstration », in : *International Journal of Human–Computer Interaction* 37.5 (mars 2021), Publisher : Taylor & Francis _eprint : https ://-doi.org/10.1080/10447318.2020.1823688, p. 418-433, ISSN : 1044-7318, DOI : 10.1080/10447318.2020.1823688, URL : https://doi.org/10.1080/10447318.2020.1823688 (visité le 08/06/2021).

[19] Yacine Bellik et D Teil, « Définitions terminologiques pour la communication multimodale », in : *Proceeding of IHM* (jan. 1992).

[20] Steve Benford et Lennart Fahlén, « A Spatial Model of Interaction in Large Virtual Environments », en, in : *Proceedings of the Third European Conference on Computer-Supported Cooperative Work 13–17 September 1993, Milan, Italy ECSCW '93*, sous la dir. de Giorgio de Michelis, Carla Simone et Kjeld Schmidt, Dordrecht : Springer Netherlands, 1993, p. 109-124, ISBN : 978-94-011-2094-4, DOI : 10.1007/978-94-011-2094-4_8, URL : https://doi.org/10.1007/978-94-011-2094-4_8 (visité le 21/07/2022).

[21] Pascal Bercher et al., « A companion-system architecture for realizing individualized and situation-adaptive user assistance », in : 2018, DOI : 10.18725/OPARU-11023.

[22] Regina Bernhaupt et al., « Model-Based Evaluation : A New Way to Support Usability Evaluation of Multimodal Interactive Applications », en, in : *Maturing Usability : Quality in Software, Interaction and Value*, sous la dir. d'Effie Lai-Chong Law, Ebba Thora Hvannberg et Gilbert Cockton, Human-Computer Interaction Series, London : Springer, 2008, p. 96-119, ISBN : 978-1-84628-941-5,

DOI : 10.1007/978-1-84628-941-5_5, URL : https://doi.org/10.1007/978-1-84628-941-5_5 (visité le 15/06/2021).

[23] Niels Ole BERNSEN, « Multimodality Theory », en, in : *Multimodal User Interfaces* (2008), Publisher : Springer, Berlin, Heidelberg, p. 5-29, DOI : 10.1007/978-3-540-78345-9_2, URL : https://link.springer.com/chapter/10.1007/978-3-540-78345-9_2 (visité le 25/03/2020).

[24] Jacob T. BIEHL et Brian P. BAILEY, « Improving interfaces for managing applications in multiple-device environments », in : *Proceedings of the working conference on Advanced visual interfaces*, AVI '06, Venezia, Italy : Association for Computing Machinery, mai 2006, p. 35-42, ISBN : 978-1-59593-353-9, DOI : 10.1145/1133265.1133273, URL : https://doi.org/10.1145/1133265.1133273 (visité le 21/03/2022).

[25] Laurent BODIC et al., « SIHMM : Simulateur de l'Interaction Homme Machine Multimodale », in : (jan. 2006).

[26] Dan BOHUS et Eric HORVITZ, « Situated interaction », in : *The Handbook of Multimodal-Multisensor Interfaces : Language Processing, Software, Commercialization, and Emerging Directions*, Association for Computing Machinery et Morgan & Claypool, juill. 2019, p. 105-143, ISBN : 978-1-970001-75-4, URL : https://doi.org/10.1145/3233795.3233800 (visité le 02/08/2022).

[27] Richard A. BOLT, « "Put-that-there" : Voice and gesture at the graphics interface », en, in : *ACM SIGGRAPH Computer Graphics* 14.*3* (juill. 1980), p. 262-270, ISSN : 00978930, DOI : 10.1145/965105.807503, URL : http://portal.acm.org/citation.cfm?doid=965105.807503 (visité le 02/12/2019).

[28] Dario BONINO et Fulvio CORNO, « DogOnt - Ontology Modeling for Intelligent Domotic Environments », in : t. 5318, oct. 2008, p. 790-803, ISBN : 978-3-540-88563-4, DOI : 10.1007/978-3-540-88564-1_51.

[29] Geeth BORALESSA et al., « Towards Robust Ubicomp : A Comprehensive Review on the Grand Challenges of Ubiquitous Computing », in : juill. 2021.

[30] Jullien BOUCHET, Laurence NIGAY et Thierry GANILLE, « ICARE software components for rapidly developing multimodal interfaces », en, in : *Proceedings of the 6th international conference on Multimodal interfaces - ICMI '04*, State College, PA, USA : ACM Press, 2004, p. 251, ISBN : 978-1-58113-995-2, DOI : 10.

1145/1027933.1027975, URL : http://portal.acm.org/citation.cfm?doid=1027933.1027975 (visité le 02/12/2019).

[31] Marie-Luce BOURGUET, « A toolkit for creating and testing multimodal interface designs », in : (jan. 2002).

[32] Giorgio BRAJNIK et Simon HARPER, « Measuring interaction design before building the system : a model-based approach », in : *Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '16, Brussels, Belgium : Association for Computing Machinery, juin 2016, p. 183-193, ISBN : 978-1-4503-4322-0, DOI : 10.1145/2933242.2933246, URL : https://doi.org/10.1145/2933242.2933246 (visité le 09/07/2021).

[33] Frederik BRUDY et al., « Cross-Device Taxonomy : Survey, Opportunities and Challenges of Interactions Spanning Across Multiple Devices », in : *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, Glasgow, Scotland Uk : Association for Computing Machinery, mai 2019, p. 1-28, ISBN : 978-1-4503-5970-2, DOI : 10.1145/3290605.3300792, URL : https://doi.org/10.1145/3290605.3300792 (visité le 17/02/2022).

[34] Frederik BRUDY et al., « EagleView : A Video Analysis Tool for Visualising and Querying Spatial Interactions of People and Devices », in : *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*, ISS '18, Tokyo, Japan : Association for Computing Machinery, nov. 2018, p. 61-72, ISBN : 978-1-4503-5694-7, DOI : 10.1145/3279778.3279795, URL : https://doi.org/10.1145/3279778.3279795 (visité le 30/08/2022).

[35] Trung BUI, « Multimodal dialogue management-state of the art », in : *Surface Science - SURFACE SCI* (fév. 2006).

[36] Harry BUNT et al., « Dialogue Act Annotation with the ISO 24617-2 Standard », en, in : *Multimodal Interaction with W3C Standards : Toward Natural User Interfaces to Everything*, sous la dir. de Deborah A. DAHL, Cham : Springer International Publishing, 2017, p. 109-135, ISBN : 978-3-319-42816-1, DOI : 10.1007/978-3-319-42816-1_6, URL : https://doi.org/10.1007/978-3-319-42816-1_6 (visité le 13/08/2022).

[37] Laura Burzagli, Pier Luigi Emiliani et Francesco Gabbanini, « Ambient Intelligence and Multimodality », en, in : *Universal Access in Human-Computer Interaction. Ambient Interaction*, sous la dir. de Constantine Stephanidis, Lecture Notes in Computer Science, Berlin, Heidelberg : Springer, 2007, p. 33-42, isbn : 978-3-540-73281-5, doi : 10.1007/978-3-540-73281-5_4.

[38] Gaëlle Calvary et al., « A Unifying Reference Framework for multi-target user interfaces », en, in : *Interacting with Computers*, Computer-Aided Design of User Interface 15.*3* (juin 2003), p. 289-308, issn : 0953-5438, doi : 10.1016/S0953-5438(03)00010-9, url : https://www.sciencedirect.com/science/article/pii/S0953543803000109 (visité le 22/09/2021).

[39] Gaëlle Calvary et al., « Towards a New Generation of Widgets for Supporting Software Plasticity : The "Comet" », en, in : *Engineering Human Computer Interaction and Interactive Systems*, sous la dir. de David Hutchison et al., t. 3425, Berlin, Heidelberg : Springer Berlin Heidelberg, 2005, p. 306-324, isbn : 978-3-540-26097-4 978-3-540-31961-0, doi : 10.1007/11431879_21, url : http://link.springer.com/10.1007/11431879_21 (visité le 03/12/2019).

[40] Victoria Carlsson et Bernt Schiele, « Towards systematic research of multimodal interfaces for non-desktop work scenarios », in : *CHI '07 Extended Abstracts on Human Factors in Computing Systems*, New York, NY, USA : Association for Computing Machinery, avr. 2007, p. 1715-1720, isbn : 978-1-59593-642-4, url : https://doi.org/10.1145/1240866.1240889 (visité le 13/09/2021).

[41] Rainara Maia Carvalho et al., « Quality characteristics and measures for human–computer interaction evaluation in ubiquitous systems », en, in : *Software Quality Journal* 25.*3* (sept. 2017), p. 743-795, issn : 1573-1367, doi : 10.1007/s11219-016-9320-z, url : https://doi.org/10.1007/s11219-016-9320-z (visité le 26/08/2022).

[42] Antonio Carzaniga et al., « A Characterization Framework for Software Deployment Technologies », in : (mai 1998).

[43] Maria Chiara Caschera et al., « Multimodal Systems : An Excursus of the Main Research Questions », in : t. 9416, oct. 2015, p. 546-558, doi : 10.1007/978-3-319-26138-6_59.

[44] Jaeseung CHANG et Marie-Luce BOURGUET, « Usability Framework for the Design and Evaluation of Multimodal Interaction », in : *Proceedings of the 22Nd British HCI Group Annual Conference on People and Computers : Culture, Creativity, Interaction - Volume 2*, BCS-HCI '08, event-place : Liverpool, United Kingdom, Swindon, UK : BCS Learning & Development Ltd., 2008, p. 123-126, ISBN : 978-1-906124-06-9, URL : http://dl.acm.org/citation.cfm?id=1531826.1531857 (visité le 12/12/2019).

[45] Shih-Fu CHANG et al., « Report of 2017 NSF Workshop on Multimedia Challenges, Opportunities and Research Roadmaps », in : *arXiv :1908.02308 [cs]* (août 2019), arXiv : 1908.02308, URL : http://arxiv.org/abs/1908.02308 (visité le 29/05/2020).

[46] Khadidja CHAOUI, Sabrina BOUZIDI-HASSINI et Yacine BELLIK, « Spatial User Interaction : What Next ? », in : mars 2022, p. 163-170, ISBN : 978-989-758-555-5, URL : https://www.scitepress.org/PublicationsDetail.aspx?ID=piqbh29ZppA=&t=1 (visité le 16/03/2022).

[47] Abiy Biru CHEBUDIE, Roberto MINERVA et Domenico ROTONDI, « Towards a definition of the Internet of Things (IoT) », thèse de doct., août 2014.

[48] Harry CHEN, Tim FININ et Anupam JOSHI, « An ontology for context-aware pervasive computing environments », en, in : *The Knowledge Engineering Review* 18.*3* (sept. 2003), Publisher : Cambridge University Press, p. 197-207, ISSN : 1469-8005, 0269-8889, DOI : 10.1017/S0269888904000025, URL : https://www.cambridge.org/core/journals/knowledge-engineering-review/article/abs/an-ontology-for-contextaware-pervasive-computing-environments/AA9BDD7E22480F3478A523014AC6B794 (visité le 28/08/2022).

[49] Pei-Yu (Peggy) CHI et Yang LI, « Weave : Scripting Cross-Device Wearable Interaction », in : *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, Seoul, Republic of Korea : Association for Computing Machinery, avr. 2015, p. 3923-3932, ISBN : 978-1-4503-3145-6, DOI : 10.1145/2702123.2702451, URL : https://doi.org/10.1145/2702123.2702451 (visité le 22/03/2022).

[50] Cristiano Andre da COSTA, Adenauer Correa YAMIN et Claudio Fernando Resin GEYER, « Toward a General Software Infrastructure for Ubiquitous Computing »,

in : *IEEE Pervasive Computing* 7.*1* (jan. 2008), Conference Name : IEEE Pervasive Computing, p. 64-73, ISSN : 1558-2590, DOI : 10.1109/MPRV.2008.21.

[51] Joëlle COUTAZ et James L CROWLEY, « Plan « intelligence ambiante » : défis et opportunités », fr, in : (oct. 2008), p. 75.

[52] Joëlle COUTAZ et Laurence NIGAY, « Multimodalité et plasticité des interfaces homme-machine en informatique ambiante : concepts et espaces de conception », in : *Information Interaction Intelligence - Le point sur le i (3)*, sous la dir. de Florence SÈDES, Jean-Marc OGIER et Pierre MARQUIS, Cépaduès Editions, juin 2012, p. 179-214, URL : https://hal.archives-ouvertes.fr/hal-00752068 (visité le 07/02/2020).

[53] Joëlle COUTAZ et al., « Context is key », in : *Commun. ACM* 48 (mars 2005), p. 49-53, DOI : 10.1145/1047671.1047703.

[54] Joëlle COUTAZ et al., « Four Easy Pieces for Assessing the Usability of Multimodal Interaction : The Care Properties », en, in : *Human—Computer Interaction*, sous la dir. de Knut NORDBY et al., Boston, MA : Springer US, 1995, p. 115-120, ISBN : 978-1-5041-2898-8 978-1-5041-2896-4, DOI : 10.1007/978-1-5041-2896-4_19, URL : http://link.springer.com/10.1007/978-1-5041-2896-4_19 (visité le 02/12/2019).

[55] Martin CRONEL et al., « MIODMIT : A Generic Architecture for Dynamic Multimodal Interactive Systems », en, in : *Human-Centered Software Engineering*, sous la dir. de Cristian BOGDAN et al., Lecture Notes in Computer Science, Cham : Springer International Publishing, 2019, p. 109-129, ISBN : 978-3-030-05909-5, DOI : 10.1007/978-3-030-05909-5_7.

[56] Fredy CUENCA et al., « Graphical Toolkits for Rapid Prototyping of Multimodal Systems : A Survey », in : *Interacting with Computers* 27.*4* (juill. 2015), Conference Name : Interacting with Computers, p. 470-488, ISSN : 1873-7951, DOI : 10.1093/iwc/iwu003.

[57] Deborah A. DAHL, *Standards for Multimodal Interaction*, en, chap., ISBN : 9781605663869 Library Catalog : www.igi-global.com Pages : 409-429 Publisher : IGI Global, 2009, DOI : 10.4018/978-1-60566-386-9.ch022, URL : https://www.igi-global.com/chapter/standards-multimodal-interaction/www.igi-global.com/chapter/standards-multimodal-interaction/35900 (visité le 15/09/2021).

[58] Dario Di Mauro, *Human-Computer Interaction in Intelligent Environments*, it, Tesi di dottorato, Library Catalog : www.fedoa.unina.it Num Pages : 142 Publisher : Università degli Studi di Napoli Federico II, déc. 2017, URL : http://www.fedoa.unina.it/12241/ (visité le 07/08/2022).

[59] Dario Di Mauro et Francesco Cutugno, « A Framework for Interaction Design in Intelligent Environments », in : sept. 2016, p. 246-249, DOI : 10.1109/IE.2016.56.

[60] Pierre Dragicevic et Jean-Daniel Fekete, « Input Device Selection and Interaction Configuration with ICON », en, in : *People and Computers XV—Interaction without Frontiers*, sous la dir. d'Ann Blandford, Jean Vanderdonckt et Phil Gray, London : Springer London, 2001, p. 543-558, ISBN : 978-1-85233-515-1 978-1-4471-0353-0, DOI : 10.1007/978-1-4471-0353-0_34, URL : http://link.springer.com/10.1007/978-1-4471-0353-0_34 (visité le 02/12/2019).

[61] Bruno Dumas, Denis Lalanne et Sharon Oviatt, « Multimodal Interfaces : A Survey of Principles, Models and Frameworks », in : *Human Machine Interaction*, sous la dir. de Denis Lalanne et Jürg Kohlas, t. 5440, Berlin, Heidelberg : Springer Berlin Heidelberg, 2009, p. 3-26, ISBN : 978-3-642-00437-7, DOI : 10.1007/978-3-642-00437-7_1, URL : http://link.springer.com/10.1007/978-3-642-00437-7_1 (visité le 02/12/2019).

[62] Bruno Dumas, María Solórzano et Beat Signer, « Design Guidelines for Adaptive Multimodal Mobile Input Solutions », in : août 2013, DOI : 10.1145/2493190.2493227.

[63] Deborah Estrin et al., « Connecting the Physical World with Pervasive Networks », in : *Pervasive Computing, IEEE* 1 (fév. 2002), p. 59-69, DOI : 10.1109/MPRV.2002.993145.

[64] Michael Feld, Robert Neβelrath et Tim Schwartz, « Software platforms and toolkits for building multimodal systems and applications », in : *The Handbook of Multimodal-Multisensor Interfaces : Language Processing, Software, Commercialization, and Emerging Directions*, Association for Computing Machinery et Morgan & Claypool, juill. 2019, p. 145-190, ISBN : 978-1-970001-75-4, URL : https://doi.org/10.1145/3233795.3233801 (visité le 15/06/2022).

[65] Daniel Fernandes et al., « Combining IoT and Users' Profiles to Provide Contextualized Information and Services », in : *Journal of Internet Social Networking and Virtual Communities* 2020(2020) (jan. 2020), p. 1-10, DOI : 10.5171/2020.663286.

[66] Matheus Frigo et al., « A Toolbox for the Internet of Things - Easing the Setup of IoT Applications », in : oct. 2020.

[67] Heidilyn Gamido et Marlon Gamido, « Comparative Review of the Features of Automated Software Testing Tools », in : *International Journal of Electrical and Computer Engineering* 9 (oct. 2019), p. 4473-4478, doi : 10.11591/ijece.v9i5.pp4473-4478.

[68] Vahid Garousi et Mika Mäntylä, « When and what to automate in software testing ? A multi-vocal literature review », in : *Information and Software Technology* 76 (avr. 2016), doi : 10.1016/j.infsof.2016.04.015.

[69] Giuseppe Ghiani, Marco Manca et Fabio Paternò, « Authoring context-dependent cross-device user interfaces based on trigger/action rules », in : *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*, MUM '15, Linz, Austria : Association for Computing Machinery, nov. 2015, p. 313-322, isbn : 978-1-4503-3605-5, doi : 10.1145/2836041.2836073, url : https://doi.org/10.1145/2836041.2836073 (visité le 14/09/2021).

[70] Christos Goumopoulos, « A Middleware Architecture for Ambient Adaptive Systems », in : jan. 2016, p. 1-35, isbn : 978-3-319-23451-9, doi : 10.1007/978-3-319-23452-6_1.

[71] Christos Goumopoulos et al., *Next Generation Intelligent Environments*, en, sous la dir. de Stefan Ultes et al., Cham : Springer International Publishing, 2016, isbn : 978-3-319-23452-6, doi : 10.1007/978-3-319-23452-6, url : http://link.springer.com/10.1007/978-3-319-23452-6 (visité le 12/02/2020).

[72] Michael Grieves et John Vickers, « Digital Twin : Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems », in : août 2017, p. 85-113, isbn : 978-3-319-38754-3, doi : 10.1007/978-3-319-38756-7_4.

[73] John Grundy et Biao Yang, « An environment for developing adaptive, multi-device user interfaces », in : *Proceedings of the Fourth Australasian user interface conference on User interfaces 2003 - Volume 18*, AUIC '03, Adelaide, Australia : Australian Computer Society, Inc., fév. 2003, p. 47-56, isbn : 978-0-909925-96-3, (visité le 24/09/2021).

[74] Sandra G. Hart et Lowell E. Staveland, « Development of NASA-TLX (Task Load Index) : Results of Empirical and Theoretical Research », en, in : *Advances in Psychology*, sous la dir. de Peter A. Hancock et Najmedin Meshkati, t. 52, Human Mental Workload, North-Holland, jan. 1988, p. 139-183, doi : 10.1016/S0166-4115(08)62386-9, url : https://www.sciencedirect.com/science/article/pii/S0166411508623869 (visité le 04/04/2022).

[75] Björn Hartmann et al., « Reflective physical prototyping through integrated design, test, and analysis », in : *Proceedings of the 19th annual ACM symposium on User interface software and technology*, UIST '06, Montreux, Switzerland : Association for Computing Machinery, oct. 2006, p. 299-308, isbn : 978-1-59593-313-3, doi : 10.1145/1166253.1166300, url : https://doi.org/10.1145/1166253.1166300 (visité le 31/08/2022).

[76] Gerd Herzog et Alassane Ndiaye, « Building Multimodal Dialogue Applications : System Integration in SmartKom », en, in : *SmartKom : Foundations of Multimodal Dialogue Systems*, sous la dir. de Wolfgang Wahlster, Cognitive Technologies, Berlin, Heidelberg : Springer, 2006, p. 439-452, isbn : 978-3-540-36678-2, doi : 10.1007/3-540-36678-4_28, url : https://doi.org/10.1007/3-540-36678-4_28 (visité le 07/04/2021).

[77] Frank Honold, Felix Schüssel et Michael Weber, « Adaptive probabilistic fission for multimodal systems », in : *Proceedings of the 24th Australian Computer-Human Interaction Conference*, OzCHI '12, Melbourne, Australia : Association for Computing Machinery, nov. 2012, p. 222-231, isbn : 978-1-4503-1438-1, doi : 10.1145/2414536.2414575, url : https://doi.org/10.1145/2414536.2414575 (visité le 12/08/2022).

[78] Frank Honold et al., « Context Models for Adaptive Dialogs and Multimodal Interaction », in : *2013 9th International Conference on Intelligent Environments*, ISSN : null, juill. 2013, p. 57-64, doi : 10.1109/IE.2013.54.

[79] Urs Hunkeler, Hong Linh Truong et Andy Stanford-Clark, « MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks », in : *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, jan. 2008, p. 791-798, doi : 10.1109/COMSWA.2008.4554519.

[80] « ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary », in : *ISO/IEC/IEEE 24765 :2017(E)* (août 2017), Conference Name : ISO/IEC/IEEE 24765 :2017(E), p. 1-541, DOI : 10.1109/IEEESTD.2017.8016712.

[81] Hyun-Su JANG et al., « Agent-based Intelligent Middleware for User-Centric Services in Ubiquitous Computing Environments », in : *J. Inf. Sci. Eng.* 27 (sept. 2011), p. 1729-1746.

[82] Kurt JENSEN, Lars Michael KRISTENSEN et Lisa WELLS, « Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems », en, in : *International Journal on Software Tools for Technology Transfer* 9.*3* (juin 2007), p. 213-254, ISSN : 1433-2787, DOI : 10.1007/s10009-007-0038-x, URL : https://doi.org/10.1007/s10009-007-0038-x (visité le 19/09/2021).

[83] Wendy JU et Larry LEIFER, « The Design of Implicit Interactions : Making Interactive Systems Less Obnoxious », in : *Design Issues* 24.*3* (juill. 2008), p. 72-84, ISSN : 0747-9360, DOI : 10.1162/desi.2008.24.3.72, URL : https://doi.org/10.1162/desi.2008.24.3.72 (visité le 29/08/2022).

[84] Robert S. KENNEDY et al., « Simulator Sickness Questionnaire : An Enhanced Method for Quantifying Simulator Sickness », in : *The International Journal of Aviation Psychology* 3.*3* (juill. 1993), Publisher : Taylor & Francis _eprint : https ://-doi.org/10.1207/s15327108ijap0303_3, p. 203-220, ISSN : 1050-8414, DOI : 10.1207/s15327108ijap0303_3, URL : https://doi.org/10.1207/s15327108ijap0303_3 (visité le 04/04/2022).

[85] Pierre Taner KIRISCI et al., « SUPPORTING INCLUSIVE PRODUCT DESIGN WITH VIRTUAL USER MODELS AT THE EARLY STAGES OF PRODUCT DEVELOPMENT », en, in : *DS 68-9 : Proceedings of the 18th International Conference on Engineering Design (ICED 11), Impacting Society through Engineering Design, Vol. 9 : Design Methods and Tools pt. 1, Lyngby/Copenhagen, Denmark, 15.-19.08.2011* (2011), p. 80-90, URL : https://www.designsociety.org/publication/30786/SUPPORTING+INCLUSIVE+PRODUCT+DESIGN+WITH+VIRTUAL+USER+MODELS+AT+THE+EARLY+STAGES+OF+PRODUCT+DEVELOPMENT (visité le 04/06/2021).

[86] Scott R. KLEMMER et al., « Suede : a Wizard of Oz prototyping tool for speech user interfaces », in : *Proceedings of the 13th annual ACM symposium on User interface software and technology*, UIST '00, San Diego, California, USA : Association for

Computing Machinery, nov. 2000, p. 1-10, ISBN : 978-1-58113-212-0, DOI : 10.1145/354401.354406, URL : https://doi.org/10.1145/354401.354406 (visité le 31/08/2022).

[87] Michael KNAPPMEYER et al., « Survey of Context Provisioning Middleware », in : *IEEE Communications Surveys Tutorials* 15.*3* (2013), p. 1492-1519, ISSN : 2373-745X, DOI : 10.1109/SURV.2013.010413.00207.

[88] Werner A. KONIG, Roman RADLE et Harald REITERER, *Interactive Design of Multimodal User Interfaces Reducing technical and visual complexity*, en, Library Catalog : www.semanticscholar.org, 2009, URL : https://www.semanticscholar.org/paper/Interactive-Design-of-Multimodal-User-Interfaces-Konig-Radle/fce4e431da4b53544011aeb1c8676a12dc841c98 (visité le 22/04/2022).

[89] Jérémy LACOCHE et Eric VILLAIN, « Prototyping Context-aware Augmented Reality Applications for Smart Environments inside Virtual Reality », in : *GRAPP 2022*, Online, Portugal, fév. 2022, URL : https://hal.archives-ouvertes.fr/hal-03559175 (visité le 11/03/2022).

[90] Jérémy LACOCHE et al., « Model and Tools for Integrating IoT into Mixed Reality Environments : Towards a Virtual-Real Seamless Continuum », in : *ICAT-EGVE 2019 - International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments*, Tokyo, Japan, sept. 2019, URL : https://hal.archives-ouvertes.fr/hal-02332096 (visité le 03/12/2019).

[91] Philippe LALANDA, Laurence NIGAY et Morgan MARTINET, « Multimodal Interactions in Dynamic, Service-Oriented Pervasive Environments », in : *2014 IEEE International Conference on Services Computing*, ISSN : null, juin 2014, p. 251-258, DOI : 10.1109/SCC.2014.41.

[92] Jean-Yves LAWSON et al., « An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components », in : jan. 2009, p. 245-254, DOI : 10.1145/1570433.1570480.

[93] Jean-Yves LAWSON et al., « Component-Based High Fidelity Interactive Prototyping of Post-WIMP Interactions. », in : nov. 2010, p. 47, DOI : 10.1145/1891903.1891961.

[94] Tobias LECHLER et al., « Virtual Commissioning – Scientific review and exploratory use cases in advanced production systems », in : *Procedia CIRP* 81 (juin 2019), p. 1125-1130, DOI : 10.1016/j.procir.2019.03.278.

[95] Saija LEMMELÄ et al., « Designing and evaluating multimodal interaction for mobile contexts », in : *Proceedings of the 10th international conference on Multimodal interfaces*, ICMI '08, Chania, Crete, Greece : Association for Computing Machinery, oct. 2008, p. 265-272, ISBN : 978-1-60558-198-9, DOI : 10.1145/1452392.1452447, URL : https://doi.org/10.1145/1452392.1452447 (visité le 19/05/2021).

[96] Irma LINDT, « Adaptive 3D-User-Interfaces », in : 2009.

[97] Pierre DE LOOR et al., « Un simulateur d'usage pour l'évaluation des systèmes interactifs multimodaux », fr, in : 7 (2006), p. 33.

[98] María LOZANO et al., « Distributed user interfaces and multimodal interaction », in : 8541 (jan. 2014), p. 581-582.

[99] Zaigham MAHMOOD, éd., *Guide to Ambient Intelligence in the IoT Environment*, en, Springer, 2019, ISBN : 978-3-030-04172-4, URL : https://doi.org/10.1007/978-3-030-04173-1 (visité le 15/06/2022).

[100] Marco MANCA et Fabio PATERNÒ, « Customizable dynamic user interface distribution », in : *Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '16, Brussels, Belgium : Association for Computing Machinery, juin 2016, p. 27-37, ISBN : 978-1-4503-4322-0, DOI : 10.1145/2933242.2933259, URL : https://doi.org/10.1145/2933242.2933259 (visité le 22/03/2022).

[101] Nicolai MARQUARDT, Frederico SCHARDONG et Anthony TANG, « EXCITE : EXploring Collaborative Interaction in Tracked Environments », en, in : *Human-Computer Interaction – INTERACT 2015*, sous la dir. de Julio ABASCAL et al., Lecture Notes in Computer Science, Cham : Springer International Publishing, 2015, p. 89-97, ISBN : 978-3-319-22668-2, DOI : 10.1007/978-3-319-22668-2_8.

[102] Nicolai MARQUARDT et al., « The proximity toolkit : prototyping proxemic interactions in ubiquitous computing ecologies », in : *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, Santa Barbara, California, USA : Association for Computing Machinery, oct. 2011, p. 315-

326, ISBN : 978-1-4503-0716-1, DOI : 10.1145/2047196.2047238, URL : https://doi.org/10.1145/2047196.2047238 (visité le 16/03/2022).

[103] Simon MAYER et al., « Configuration of smart environments made simple : Combining visual modeling with semantic metadata and reasoning », in : *2014 International Conference on the Internet of Things (IOT)*, oct. 2014, p. 61-66, DOI : 10.1109/IOT.2014.7030116.

[104] Tomasz MAZURYK et Michael GERVAUTZ, « Virtual Reality - History, Applications, Technology and Future », in : (déc. 1999).

[105] Hildeberto MENDONÇA et al., « A fusion framework for multimodal interactive applications », in : *Proceedings of the 2009 international conference on Multimodal interfaces*, ICMI-MLMI '09, Cambridge, Massachusetts, USA : Association for Computing Machinery, nov. 2009, p. 161-168, ISBN : 978-1-60558-772-1, DOI : 10.1145/1647314.1647344, URL : https://doi.org/10.1145/1647314.1647344 (visité le 19/09/2021).

[106] Maximilian METZNER et al., « Intuitive, VR- and Gesture-based Physical Interaction with Virtual Commissioning Simulation Models », in : jan. 2020, p. 11-20, ISBN : 978-3-662-61754-0, DOI : 10.1007/978-3-662-61755-7_2.

[107] Andrei MICLAUS, Till RIEDEL et Michael BEIGL, « End-user installation of heterogeneous home automation systems using pen and paper interfaces and dynamically generated documentation », in : *2014 International Conference on the Internet of Things (IOT)*, oct. 2014, p. 19-24, DOI : 10.1109/IOT.2014.7030109.

[108] Markus MODZELEWSKI et al., « Creative Design for Inclusion Using Virtual User Models », in : t. 7382, juill. 2012, p. 288-294, ISBN : 978-3-642-31521-3, DOI : 10.1007/978-3-642-31522-0_43.

[109] Sebastian MOLLER et al., « A taxonomy of quality of service and Quality of Experience of multimodal human-machine interaction », in : *2009 International Workshop on Quality of Multimedia Experience*, juill. 2009, p. 7-12, DOI : 10.1109/QOMEX.2009.5246986.

[110] G. MORI, F. PATERNÒ et Carmen SANTORO, « CTTE : Support for Developing and Analyzing Task Models for Interactive System Design », in : *IEEE Trans. Software Eng.* (2002), DOI : 10.1109/TSE.2002.1027801.

[111] Camille Moussette, *Tangible interaction toolkits for designers*, en, Library Catalog : www.semanticscholar.org, 2007, URL : https://www.semanticscholar.org/paper/Tangible-interaction-toolkits-for-designers-Moussette/69fcf4959218be260cdf581219c88d6e8d077425 (visité le 06/10/2021).

[112] Michel Nass, Emil Alégroth et Robert Feldt, « Why many challenges with GUI test automation (will) remain », en, in : *Information and Software Technology* 138 (oct. 2021), p. 106625, ISSN : 0950-5849, DOI : 10.1016/j.infsof.2021.106625, URL : https://www.sciencedirect.com/science/article/pii/S0950584921000963 (visité le 30/08/2022).

[113] David Navarre et al., « ICOs : A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability », in : *ACM Trans. Comput.-Hum. Interact.* 16 (nov. 2009).

[114] Robert Nesselrath, « SiAM-dp », thèse de doct., jan. 2016.

[115] Americo Talarico Neto et Renata Pontin de Mattos Fortes, « Improving multimodal interaction design with the MMWA authoring environment », in : *Proceedings of the 28th ACM International Conference on Design of Communication*, New York, NY, USA : Association for Computing Machinery, sept. 2010, p. 151-158, ISBN : 978-1-4503-0403-0, URL : https://doi.org/10.1145/1878450.1878476 (visité le 06/01/2021).

[116] Laurence Nigay et Joëlle Coutaz, « A design space for multimodal systems : concurrent processing and data fusion », en, in : *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '93*, Amsterdam, The Netherlands : ACM Press, 1993, p. 172-178, ISBN : 978-0-89791-575-5, DOI : 10.1145/169059.169143, URL : http://portal.acm.org/citation.cfm?doid=169059.169143 (visité le 02/12/2019).

[117] Laurence Nigay et Joëlle Coutaz, « Espaces conceptuels pour l'interaction multimédia et multimodale », in : *Technique Et Science Informatiques - TSI* 15 (jan. 1996).

[118] Laurence Nigay, Yann Laurillau et Frédéric Jourde, « Description of tasks with multi-user multimodal interactive systems : existing notations », fr, in : *Journal d'Interaction Personne-Système* Volume 3, Numéro 3, Numéro Spécial : Modèles de Tâches (juill. 2015), Publisher : Episciences.org, DOI : 10.46298/jips.658, URL : https://jips.episciences.org/658/pdf (visité le 05/10/2021).

[119] Anke van Oosterhout, Miguel Bruns et Eve Hoggan, « Facilitating Flexible Force Feedback Design with Feelix », in : *Proceedings of the 2020 International Conference on Multimodal Interaction*, ICMI '20, Virtual Event, Netherlands : Association for Computing Machinery, oct. 2020, p. 184-193, isbn : 978-1-4503-7581-8, doi : 10.1145/3382507.3418819, url : https://doi.org/10.1145/3382507.3418819 (visité le 05/10/2021).

[120] Sharon Oviatt et Philip R. Cohen, « The Paradigm Shift to Multimodality in Contemporary Computer Interfaces », in : *Synthesis Lectures on Human-Centered Informatics* 8.*3* (avr. 2015), Publisher : Morgan & Claypool Publishers, p. 1-243, issn : 1946-7680, doi : 10.2200/S00636ED1V01Y201503HCI030, url : https://www.morganclaypool.com/doi/abs/10.2200/S00636ED1V01Y201503HCI030 (visité le 18/05/2020).

[121] F. Paternò et al., « Authoring pervasive multimodal user interfaces », in : *Int. J. Web Eng. Technol.* (2008), doi : 10.1504/IJWET.2008.018099.

[122] Shachi Paul, Rahul Goel et Dilek Hakkani-Tür, *Towards Universal Dialogue Act Tagging for Task-Oriented Dialogues*, rapp. tech. arXiv :1907.03020, arXiv :1907.03020 [cs] type : article, arXiv, juill. 2019, doi : 10.48550/arXiv.1907.03020, url : http://arxiv.org/abs/1907.03020 (visité le 26/09/2022).

[123] Matthias Peissner et al., « Interim Report on VUMS cluster standardisation », en, in : (juin 2011), p. 59, url : https://manualzz.com/doc/o/v1opy/myui--mainstreaming-accessibility-through-synergistic-user-context-management-approach.

[124] Carlos Pereira, António Teixeira et Miguel Oliveira e Silva, « Live evaluation within ambient assisted living scenarios », in : *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '14, Rhodes, Greece : Association for Computing Machinery, mai 2014, p. 1-6, isbn : 978-1-4503-2746-6, doi : 10.1145/2674396.2674414, url : https://doi.org/10.1145/2674396.2674414 (visité le 31/08/2022).

[125] Sebastian Peters, « MIBO - A Framework for the Integration of Multimodal Intuitive Controls in Smart Buildings », in : 2016.

[126] Sebastian Peters, Jan Ole Johanssen et Bernd Bruegge, « An IDE for multimodal controls in smart buildings », in : *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, ICMI '16, Tokyo, Japan : Association for Computing Machinery, oct. 2016, p. 61-65, isbn : 978-1-4503-4556-9, doi : 10.1145/2993148.2993162, url : https://doi.org/10.1145/2993148.2993162 (visité le 14/09/2021).

[127] Thies Pfeiffer et Nadine Pfeiffer-Lessmann, « Virtual Prototyping of Mixed Reality Interfaces with Internet of Things (IoT) Connectivity », in : *i-com* 17.*2* (2018), p. 179-186, issn : 1618-162X, doi : 10.1515/icom-2018-0025, url : https://www.degruyter.com/view/j/icom.2018.17.issue-2/icom-2018-0025/icom-2018-0025.xml (visité le 03/12/2019).

[128] Fabio Pittarello et Augusto Celentano, « Deployment of Multimodal Services : an Ontology Driven Architecture », in : *IEEE International Conference on Pervasive Services*, juill. 2007, p. 267-274, doi : 10.1109/PERSER.2007.4283925.

[129] Fabrice Poirier et al., « Interactive Multimodal System Characterization in the Internet of Things Context », in : *HUCAPP 2022*, Conférence Virtuelle en ligne, France, fév. 2022, url : https://hal.archives-ouvertes.fr/hal-03451478 (visité le 22/03/2022).

[130] Davy Preuveneers et Paulo Novais, « A survey of software engineering best practices for the development of smart applications in Ambient Intelligence », en, in : *Journal of Ambient Intelligence and Smart Environments* 4.*3* (jan. 2012), Publisher : IOS Press, p. 149-162, issn : 1876-1364, doi : 10.3233/AIS-2012-0150, url : https://content.iospress.com/articles/journal-of-ambient-intelligence-and-smart-environments/ais150 (visité le 30/08/2022).

[131] Gaëtan Pruvost, « Modélisation et conception d'une plateforme pour l'interaction multimodale distribuée en intelligence ambiante », fr, thèse de doct., Université Paris Sud - Paris XI, fév. 2013, url : https://tel.archives-ouvertes.fr/tel-00805487 (visité le 24/02/2020).

[132] Ismo Rakkolainen et al., « Technologies for Multimodal Interaction in Extended Reality—A Scoping Review », en, in : *Multimodal Technologies and Interaction* 5.*12* (déc. 2021), Number : 12 Publisher : Multidisciplinary Digital Publishing Institute, p. 81, issn : 2414-4088, doi : 10.3390/mti5120081, url : https://www.mdpi.com/2414-4088/5/12/81 (visité le 20/07/2022).

[133] Leah M. Reeves et al., « Guidelines for multimodal user interface design », in : *Communications of the ACM* 47.*1* (jan. 2004), p. 57-59, ISSN : 0001-0782, DOI : 10.1145/962081.962106, URL : https://doi.org/10.1145/962081.962106 (visité le 15/09/2021).

[134] Gianna Reggio et al., « What are IoT systems for real ? An experts' survey on software engineering aspects », en, in : *Internet of Things* 12 (déc. 2020), p. 100313, ISSN : 2542-6605, DOI : 10.1016/j.iot.2020.100313, URL : https://www.sciencedirect.com/science/article/pii/S254266052030144X (visité le 05/07/2022).

[135] Ana Rocha et al., « An Accessible Smart Home Based on Integrated Multimodal Interaction », in : *Sensors* 21 (août 2021), p. 5464, DOI : 10.3390/s21165464.

[136] Pedro Rodrigues et Jose Luis Silva, « Help through demonstration and automation for interactive computing systems : A survey of recent works », en, in : *International Journal of Electrical and Computer Engineering (IJECE)* 11.*2* (avr. 2021), Number : 2, p. 1549-1560, ISSN : 2722-2578, DOI : 10.11591/ijece.v11i2.pp1549-1560, URL : https://ijece.iaescore.com/index.php/IJECE/article/view/21962 (visité le 25/09/2022).

[137] Berha Helena Rodriguez et Jean-Claude Moissinac, « Discovery and Registration : Finding and Integrating Components into Dynamic Systems », en, in : (2017), Publisher : Springer, p. 325, DOI : 10.1007/978-3-319-42816-1_15, URL : https://hal.telecom-paris.fr/hal-02287487 (visité le 03/10/2021).

[138] Léon Rothkrantz et al., « Multimodal Dialog Management », in : (mars 2012).

[139] Cyril Rousseau et al., « A framework for the intelligent multimodal presentation of information », en, in : *Signal Processing*, Special Section : Multimodal Human-Computer Interfaces 86.*12* (déc. 2006), p. 3696-3713, ISSN : 0165-1684, DOI : 10.1016/j.sigpro.2006.02.041, URL : https://www.sciencedirect.com/science/article/pii/S0165168406001423 (visité le 16/02/2021).

[140] Cyril Rousseau et al., « Multimodal output simulation platform for real-time military systems », in : (jan. 2005).

[141] Angela Sasse, David Thevenin et Joëlle Coutaz, « Plasticity of User Interfaces : Framework and Research Agenda », in : (fév. 2000).

[142] Albrecht SCHMIDT, « Interactive context-aware systems interacting with ambient intelligence », in : *Ambient intelligence* (jan. 2005), p. 159-178.

[143] Ronny SEIGER et al., « A framework for rapid prototyping of multimodal interaction concepts », in : *CEUR Workshop Proceedings* 1380 (jan. 2015), p. 21-28.

[144] Pallavi SETHI et Smruti R. SARANGI, « Internet of Things : Architectures, Protocols, and Applications », en, in : *Journal of Electrical and Computer Engineering* 2017 (jan. 2017), Publisher : Hindawi, e9324035, ISSN : 2090-0147, DOI : 10.1155/2017/9324035, URL : https://www.hindawi.com/journals/jece/2017/9324035/ (visité le 14/11/2022).

[145] Jie SHEN, Wenzhe SHI et Maja PANTIC, « HCIλ2 Workbench : A development tool for multimodal human-computer interaction systems », in : avr. 2011, p. 766-773, DOI : 10.1109/FG.2011.5771346.

[146] José SILVA et al., « Analysis of WIMP and Post WIMP Interactive Systems based on Formal Specification », in : (jan. 2013).

[147] Samuel SILVA et al., « Design and Development of Multimodal Applications : A Vision on Key Issues and Methods », en, in : *Universal Access in Human-Computer Interaction. Access to Today's Technologies*, sous la dir. de Margherita ANTONA et Constantine STEPHANIDIS, Lecture Notes in Computer Science, Cham : Springer International Publishing, 2015, p. 109-120, ISBN : 978-3-319-20678-3, DOI : 10.1007/978-3-319-20678-3_11.

[148] Anoop SINHA et James LANDAY, « Capturing user tests in a multimodal, multidevice informal prototyping tool », in : jan. 2003, p. 117-124, DOI : 10.1145/958432.958457.

[149] Barnabé Edoh Barnabé SOEDJI, Jérémy LACOCHE et Eric VILLAIN, « Creating AR Applications for the IOT : a New Pipeline », in : *VRST '20 : 26th ACM Symposium on Virtual Reality Software and Technology*, Virtual Event Canada, France : ACM, nov. 2020, p. 1-2, DOI : 10.1145/3385956.3422088, URL : https://hal.archives-ouvertes.fr/hal-03005941 (visité le 06/10/2021).

[150] Thanos G. STAVROPOULOS et al., « BOnSAI : a smart building ontology for ambient intelligence », in : *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics*, WIMS '12, Craiova, Romania : Association for Computing Machinery, juin 2012, p. 1-12, ISBN : 978-1-4503-0915-8, DOI :

10.1145/2254129.2254166, URL : https://doi.org/10.1145/2254129.2254166 (visité le 28/08/2022).

[151] Stefanos STAVROTHEODOROS et al., « A Smart-Home IoT Infrastructure for the Support of Independent Living of Older Adults », en, in : t. AICT-520, Springer International Publishing, mai 2018, p. 238, DOI : 10.1007/978-3-319-92016-0_22, URL : https://hal.inria.fr/hal-01821303 (visité le 02/06/2021).

[152] Richard STOAKLEY, Matthew J. CONWAY et Randy PAUSCH, « Virtual reality on a WIM : interactive worlds in miniature », in : *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, Denver, Colorado, USA : ACM Press/Addison-Wesley Publishing Co., mai 1995, p. 265-272, ISBN : 978-0-201-84705-5, DOI : 10.1145/223904.223938, URL : https://doi.org/10.1145/223904.223938 (visité le 26/04/2022).

[153] Ryohei SUZUKI, Katsutoshi MASAI et Maki SUGIMOTO, *ReallifeEngine : A Mixed Reality-Based Visual Programming System for SmartHomes*, en, Accepted : 2019-09-11T05 :43 :10Z ISSN : 1727-530X, The Eurographics Association, 2019, ISBN : 978-3-03868-083-3, DOI : 10.2312/egve.20191287, URL : https://diglib.eg.org:443/xmlui/handle/10.2312/egve20191287 (visité le 22/03/2022).

[154] Ronnie TAIB et Natalie RUIZ, « Wizard of Oz for Multimodal Interfaces Design : Deployment Considerations », in : juill. 2007, p. 232-241, ISBN : 978-3-540-73104-7, DOI : 10.1007/978-3-540-73105-4_26.

[155] Farshid TAVAKOLIZADEH, Sisay Adugna CHALA et Hanbing ZHANG, « An Interactive Interface for Bulk Software Deployment in IoT », in : *Proceedings of the 9th International Conference on the Internet of Things*, IoT 2019, Bilbao, Spain : Association for Computing Machinery, oct. 2019, p. 1-4, ISBN : 978-1-4503-7207-7, DOI : 10.1145/3365871.3365912, URL : https://doi.org/10.1145/3365871.3365912 (visité le 27/07/2022).

[156] Jean-Claude THILL, Diep DAO et Yuhong ZHOU, « Traveling in the three-dimensional city : Applications in route planning, accessibility assessment, location analysis and beyond », in : *Journal of Transport Geography - J TRANSP GEOGR* 19 (mai 2011), p. 405-421, DOI : 10.1016/j.jtrangeo.2010.11.007.

[157] Jean VANDERDONCKT, « Distributed User Interfaces : How to Distribute User Interface Elements across Users, Platforms, and Environments », in : (jan. 2010).

[158] Geert VANDERHULST, Kris LUYTEN et Karin CONINX, « Pervasive maps : Explore and interact with pervasive environments », in : *2010 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, mars 2010, p. 227-234, DOI : 10.1109/PERCOM.2010.5466973.

[159] Geert VANDERHULST et al., « Edit, inspect and connect your surroundings : a reference framework for meta-UIs », in : *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, EICS '09, Pittsburgh, PA, USA : Association for Computing Machinery, juill. 2009, p. 167-176, ISBN : 978-1-60558-600-7, DOI : 10.1145/1570433.1570466, URL : https://doi.org/10.1145/1570433.1570466 (visité le 21/11/2022).

[160] Christoph VEIGL et al., « Model-based design of novel human-computer interfaces — The Assistive Technology Rapid Integration & Construction Set (AsTeRICS) », in : *2013 ISSNIP Biosignals and Biorobotics Conference : Biosignals and Robotics for Better and Safer Living (BRC)*, ISSN : 2326-7844, fév. 2013, p. 1-7, DOI : 10.1109/BRC.2013.6487539.

[161] Roman VILIMEK, « More Than Words : Designing Multimodal Systems », en, in : *Usability of Speech Dialog Systems : Listening to the Target Audience*, sous la dir. de Thomas HEMPEL, Signals and Commmunication Technologies, Berlin, Heidelberg : Springer, 2008, p. 123-145, ISBN : 978-3-540-78343-5, DOI : 10.1007/978-3-540-78343-5_6, URL : https://doi.org/10.1007/978-3-540-78343-5_6 (visité le 23/06/2021).

[162] Ina WECHSUNG, *An Evaluation Framework for Multimodal Interaction : Determining Quality Aspects and Modality Choice*, en, T-Labs Series in Telecommunication Services, Cham : Springer International Publishing, 2014, ISBN : 978-3-319-03809-4, DOI : 10.1007/978-3-319-03810-0, URL : http://link.springer.com/10.1007/978-3-319-03810-0 (visité le 07/02/2020).

[163] Mark WEISER, « The Computer for the 21 st Century », in : *Scientific American* 265.*3* (1991), Publisher : Scientific American, a division of Nature America, Inc., p. 94-105, ISSN : 0036-8733, URL : https://www.jstor.org/stable/24938718 (visité le 28/09/2021).

COLLEGES BRETAGNE LOIRE SCIENCES POUR L'INGENIEUR ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Les interactions multimodales dans le contexte de l'internet des objets

Mot clés : Interactions Multimodales, Informatique ambiante, Objets connectés, Internet des Objets, Réalité Virtuelle

Résumé : Les MIBS sont des systèmes multimodaux utilisant des objets connectés comme interfaces d'interaction. Il faut des méthodes et des outils pour faciliter l'adaptation de ces systèmes aux spécificités des environnements d'exécution. D'un environnement à l'autre, les objets connectés sont différents, placés à des endroits différents, avec des disponibilités variables. Un état de l'art a permis d'identifier les exigences à atteindre pour réaliser des MIBS, puis de déterminer les fonctionnalités essentielles et les principales limitations des frameworks permettant de réaliser des MIBS. Cela nous a permis de proposer et développer notre propre framework. À ce stade une intervention humaine est nécessaire au processus d'adaptation des MIBS. Nous avons ensuite proposé une étude du cycle de vie de ces systèmes pour identifier les tâches, rôles et outils intervenant dans ce processus d'adaptation. Pour outiller l'étape de déploiement des MIBS nous avons proposé une méthode et un outil proposant d'utiliser la Réalité Virtuelle (RV) pour configurer et évaluer des MIBS dans des modèles 3D d'environnements avant de passer au déploiement dans les environnements réels. Une expérimentation utilisateur a confirmé l'intérêt de cette approche.

Title: Multimodal interactions in the context of the Internet of Things

Keywords: Multimodal Interactions, Ambiant systems, Connected Objects, Internet of Things, Virtual Reality

Abstract: MIBS are multimodal systems using connected objects as interaction interfaces. Methods and tools are needed to facilitate the adaptation of these systems to the specificities of the execution environments. From one environment to another, the connected objects are different, placed in different places, with variable availability. A state of the art made it possible to identify the requirements to be met in order to create MIBS, then to determine the essential functionalities and the main limitations of the frameworks allowing the creation of MIBS. This allowed us to propose and develop our own framework. At this stage, human intervention is necessary for the MIBS adaptation process. We then proposed a study of the life cycle of these systems to identify the tasks, roles and tools involved in this adaptation process. To equip the MIBS deployment stage, we have proposed a method and a tool proposing to use Virtual Reality (VR) to configure and evaluate MIBS in 3D models of environments before moving on to deployment in real environments. A user experiment confirmed the interest of this approach.