THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *l'Université Toulouse III - Paul Sabatier*
Discipline ou spécialité : *Informatique*

Présentée et soutenue par *Wafaa ABOU MOUSSA*
Le *18/12/2008*

Titre : Prototypage moyenne fidélité incrémental et basé sur la simulation des systèmes interactifs mixtes

---

JURY

Rapporteurs :	Pascal GUITTON
	Stéphane DONIKIAN
Examinateurs :	Rémi BASTIDE
	Jean-Yves TIGLI
	Sylvain MICHELIN
Directeur de thèse :	Jean-Pierre JESSEL
Encadrants :	Emmanuel DUBOIS
	Patrice TORGUET

Ecole doctorale : *MITT*
Unité de recherche : *Institut de Recherche en Informatique de Toulouse*
Directeur de Thèse : *Jean-Pierre JESSEL*

Résumé

Les systèmes interactifs mixtes s'appuient sur l'utilisation d'objets de l'environnement physique de l'utilisateur comme support à l'interaction de cet utilisateur avec le système. En alliant les capacités de traitement des systèmes informatiques aux capacités de l'utilisateur à interagir de façon intuitive avec son environnement physique, ces systèmes proposent des solutions nouvelles à des problèmes divers et variés. L'émergence de ces nouvelles situations d'interaction intégrées dans l'environnement d'activité de l'utilisateur, conduit cependant à des problématiques de conception et de réalisation particulières.

Les approches existantes de développement sont souvent partagées entre l'optimisation des couches logicielles et des processus de développement d'une part, et l'amélioration de l'utilisabilité des systèmes de l'autre. Ces deux visions mènent souvent à des processus de développement incompatibles et divergents, faisant intervenir de grandes variations d'expertises. Indépendamment de la méthodologie choisie, l'évaluation des systèmes et de l'interaction de l'utilisateur s'appuie souvent sur le prototypage de solutions intermédiaires. Les méthodes de prototypage de basse et de haute fidélité appliquées aux systèmes interactifs classiques se révèlent alors inadaptées aux environnements associant numérique et physique.

Cette thèse propose un prototypage des systèmes interactifs basé sur la simulation de l'environnement et de l'interaction dans un environnement de réalité virtuelle. Le prototypage par simulation proposé est évolutif, et permet un passage incrémental entre les systèmes simulés et leurs réalisations finales. Nos travaux proposent alors un certain nombre de services de simulation et d'interaction adaptés au prototypage incrémental, ainsi qu'un espace de modélisation qui permet la conduite d'un processus de prototypage et de développement dirigé par les modèles. Ce processus permet l'intégration des différentes variations d'expertise.

Deux cas d'études nous serviront à illustrer la méthodologie adoptée ainsi que les différents outils utilisés.

Mots-clés: Prototypage incrémental, simulation de l'interaction, approche dirigée par les modèles.

Abstract

Mixed interactive systems rely on the use of objects belonging to the user's physical environment to support user's interaction with the system. Allying digital processing potential of computer systems with users' natural ability to manipulate physical entities allows these systems to propose new solutions to a variety of different problems and contexts. The emergence of these new interaction situations however raises particular design and development issues.

Addressing these problems often follows two different approaches : on one hand, the engineering approach that tries to optimize system architecture and development processes, and on the other, interaction-centered development that favors improving system interactivity and end-user experience. These approaches often lead to incompatible development processes and involve large variations of expertise. Independently of the approach adopted for development, evaluation of system and interaction properties often relies on frequent prototyping of intermediate solutions. Existing low and high fidelity prototyping approaches for classic interactive systems are often ill-adapted to systems combining physical and digital entities.

This thesis proposes prototyping of mixed interactive systems based on the simulation of the interaction and its environment in a virtual space. The proposed simulation-based prototyping supports incremental evolution of simulated prototypes into the final systems. Our work proposes a set of tools and services adapted for simulation-based incremental prototyping, as well as a model-driven development process that supports incremental evolution of prototypes and the integration of the various expertise variations.

Two case studies allow us to illustrate the development methodology and the different tools involved.

Keywords: Simulation-based prototyping, incremental evolution, model-driven development.

Table des matières

Chapitre 1 Introduction **1**

1.1 Proposition 3

1.2 Structure de cette thèse 4

Partie I Historique et état de l'art **7**

Chapitre 1 Introduction **9**

1.1 Historique et définitions 9

1.2 Spécificités des systèmes interactifs mixtes 11

1.2.1 En sortie 12

1.2.2 En entrée 12

1.2.3 Interactivité 13

Chapitre 2 Outils pour le développement des SIM **15**

2.1 Les outils logiciels 17

2.1.1 Couverture 17

2.1.2 Les contraintes architecturales 19

2.1.3 Services de l'environnement d'exécution 25

2.1.4 Services aux développeurs 31

2.1.5 Limitations des approches purement logicielles 32

2.2 Outils d'analyse et de conception 34

2.2.1 Taxonomies 35

2.2.2 Modèles et modélisation des systèmes mixtes 38

2.3 Prototypage des systèmes interactifs 47

2.3.1 Caractérisation du prototypage 47

2.3.2 Prototypage et systèmes interactifs mixtes 50

Chapitre 3 Analyse de l'état de l'art **55**
3.1 Intégration des choix de conception dans des processus de développement performants . . . 56
3.2 Accommodation des variations d'expertise . . . 57
3.3 Inadéquation des approches de prototypage classiques . . . 58

Partie II SIMBA **61**

Chapitre 1 Objectifs, difficultés et outils **63**
1.1 Objectifs . . . 63
1.1.1 Ancrage dans les choix de conception . . . 64
1.1.2 Prototypage évolutif et incrémental . . . 65
1.1.3 Adaptation à différents domaines d'expertise . . . 66
1.2 Proposition . . . 66
1.3 Identification des problèmes à résoudre par rapport aux objectifs . . . 67
1.3.1 Intégration des choix de conception . . . 67
1.3.2 Prototypage par simulation et cycle itératif . . . 68
1.3.3 Prototypage par simulation et processus incrémental . . . 68
1.3.4 Conformité du prototype avec les modèles conceptuels . . . 69
1.3.5 Intégration de services applicatifs . . . 69
1.3.6 Gestion de la distribution des utilisateurs . . . 70
1.3.7 Gestion de la distribution des dispositifs . . . 70
1.3.8 Gestion de la distribution des traitements et calculs . . . 71
1.4 Aspects traités et non traités par la simulation . . . 73
1.5 Outils de développement . . . 75
1.5.1 Le modèle de conception . . . 75
1.5.2 Environnement de modélisation . . . 76
1.5.3 Modèle d'implémentation . . . 76

Chapitre 2 Architecture de SIMBA **79**
2.1 Approche globale . . . 80
2.1.1 Approche orientée composants . . . 80
2.1.2 Distribution transparente . . . 81
2.2 Services de simulation . . . 81
2.2.1 Séparation entre les représentations et les aspects fonctionnels . . . 82
2.2.2 Gestionnaire graphique . . . 83
2.2.3 Horloge . . . 92

2.3 Services d'interaction 94
2.3.1 Adaptation au prototypage incrémental par simulation 94
2.3.2 Couche d'accès uniforme 98
2.3.3 Couche de capteurs logiques 100
2.3.4 Couche des métaphores 102
2.3.5 Modes de communication 105
2.3.6 Intégration des pilotes et bibliothèques 108

Chapitre 3 Approche dirigée par les modèles : méta-modèle, outils, et processus de développement 111

3.1 Le méta-modèle SIMBA 112
3.1.1 Éléments génériques du méta-modèle SIMBA 113
3.1.2 Éléments représentants les services de simulation 115
3.1.3 Éléments relatifs aux services d'interaction 117
3.1.4 Composants miroirs 118
3.1.5 Éléments de persistance des modèles conceptuels 120
3.2 Règles et outils 121
3.2.1 Règles 122
3.2.2 Transformations et interpréteurs de modèles 125
3.3 Processus de développement 128
3.3.1 Les concepteurs de l'interaction 128
3.3.2 Intégrateurs des systèmes 130
3.3.3 Les développeurs de composants 132
3.3.4 Les acteurs divers 133

Chapitre 4 Applications 135

4.1 Musée augmenté 135
4.1.1 Assistance 136
4.1.2 Simulation 137
4.1.3 Processus de développement 138
4.1.4 Itérations 141
4.2 Système d'assistance à la maintenance aéronautique nomade 145
4.2.1 Contexte général 145
4.2.2 Assistance 145
4.2.3 Simulation 147
4.2.4 Processus de développement 148

Chapitre 5 Conclusions et perspectives **155**

5.1 Récapitulatifs 155

5.2 Limitations et perspectives 158

Bibliographie **161**

Table des figures

1.1 Continuum de Milgram 10
1.2 Classification selon l'objet de la tâche et le type d'augmentation 10

2.1 Boites à outils 18
2.2 Principe Hollywood 20
2.3 Couche décisionnelle 24
2.4 Interaction pilotée par le système 28
2.5 Interaction pilotée par l'utilisateur 29
2.6 Défilement temporel 32
2.7 Processus de DWARF 33
2.8 Développement par composition 34
2.9 Développement par décomposition 35
2.10 Processus itératifs 46

3.1 Récapitulatif 59

1.1 Interactions individuelles 71
1.2 Distribution des dispositifs 72

2.1 Modèle ASUR d'un système de localisation par caméras 84
2.2 Simulation de conduite 86
2.3 Simulation d'un accident 87
2.4 Modélisation dynamique 88
2.5 Composant GM avec mise en évidence de l'interface de sélection 93

2.6 Les trois couches de l'architecture des interactions 96
2.7 Composant de la CM pour un personnage articulé simple 103
2.8 Composant de la CM pour un personnage articulé complexe 104
2.9 Exemple d'un dispositif physique de pilotage 105
2.10 Composant de la CM pour un véhicule piloté 106
2.11 Mode de communication synchrone 107
2.12 Mode de communication synchrone partiel 108
2.13 Mode de communication asynchrone 109
2.14 Mode de communication asynchrone périodique 110

3.1 Elements génériques du méta-modèle SIMBA 115
3.2 Modèle SIMBA du système à un stade d'évolution intermédiaire 119
3.3 Modèle SIMBA du système utilisant un composant miroir 120
3.4 Modèle miroir 121
3.5 Séparation des aspects dans GME 122
3.6 Modèle vide initial 126
3.7 Processus de développement 129
3.8 Modélisation ASUR dans GUIDE-ME 130
3.9 Modèle vide initial 131
3.10 Modèle miroir 132
3.11 Composants complétés, avant l'étape de connexion 133

4.1 Modèle ASUR par rapport à la tâche de localisation 137
4.2 Modèle ASUR par rapport à la tâche de contacter le guide 138
4.3 Modèle vide initial 139
4.4 Modèle Miroir généré par la TAS 140
4.5 Modèle SIMBA complet 141
4.6 Modèle SIMBA complet, avec une caméra non simulée 142
4.7 Plateforme matérielle faisant tourner la simulation avec une caméra et un visiteur non simulés 143
4.8 Modèle SIMBA final sans simulation 143

4.9 Capture d'écran de la simulation à un stade d'évolution intermédiaire 144
4.10 Modèle ASUR partiel du système d'assistance 149
4.11 Modèle vide généré par la TAS 150
4.12 Modèle miroir généré par la TAS 151
4.13 Modèle SIMBA final, avec le GPS simulé via une caméra ARToolkit 152
4.14 Caméra remplaçant un composant GPS pour le suivi à l'extérieur 153
4.15 Capture d'écran de l'interaction dans un des hangars 153
4.16 Capture d'écran sur le terrain extérieur aux hangars 154

1

Introduction

Il est désormais acquis que les interactions entre un utilisateur et un système informatique, ne sont plus uniquement limitées au traditionnel triptyque clavier, écran, souris. Des évolutions technologiques, logicielles, et de communication, favorisent le développement de techniques d'interaction avancées, comme celles mises en place dans les environnements de réalité virtuelle (RV). Il en résulte ainsi l'émergence de nouvelles situations d'interaction, favorisant leur intégration dans l'environnement d'activité de l'utilisateur.

A partir des premiers systèmes définis il y a plus d'un quart de siècle, les avancées technologiques et logicielles ont rapidement installé la réalité virtuelle et la réalité augmentée en tant que domaines de recherche et d'innovation à part entière. L'évolution rapide de ces domaines, étroitement liée à l'évolution de la technologie, a initialement conduit à privilégier les solutions rapides au détriment de la rigueur de la démarche de développement. Afin de prendre en compte les différentes facettes de ces systèmes et de rationaliser les approches ad-hoc initiales, différents formalismes et plateformes ont été développés. Cependant, adapter les méthodologies de conception et de développement aux multiples domaines d'application potentiels requiert une prise en compte méthodique des technologies nécessaires, ainsi que des variations d'expertise des acteurs impliqués.

Dans ce contexte, nous nous intéressons aux systèmes interactifs mixtes (SIM), des systèmes qui visent à améliorer l'expérience interactive en mélangeant entités et traitements numériques aux éléments de l'environnement physique de l'utilisateur. En apportant les capacités numériques à l'environnement "naturel" d'interaction, les systèmes interactifs mixtes élargissent leur champ

d'application à des activités industrielles, ludiques ou éducatives.

Le manque de standards suffisants gouvernant la mise en place des configurations logicielles et matérielles et le besoin croissant de définir des techniques d'interaction adaptées à des situations nouvelles, mènent souvent à des cycles de développement itératifs. En plus, dans le cas particulier des SIM, ces cycles se révèlent extrêmement longs et coûteux, du fait des acquisitions matérielles répétitives et des coûts et complexités qui sont associés à leur déploiement. Les démarches de prototypage classiques, censées accélérer et baisser les coûts des cycles itératifs, se révèlent insuffisantes voire totalement inadaptées. En effet, les prototypes de basse fidélité, ou "papier-crayon", peinent à représenter les aspects de mobilité, d'ubiquité, ou la dynamicité des systèmes et de leurs interfaces. En revanche, les prototypes de haute fidélité ne suppriment pas le besoin d'acquisitions et de déploiements matériels.

En dépit des méthodes de prototypage utilisées, l'intégration des choix de conception dans les processus de développement demeure une problématique centrale. Les approches basées sur des plateformes et architectures de services peinent à conserver les choix conceptuels à travers les cycles répétitifs de modifications et d'implémentations. Contrairement à ces méthodes qualifiées d'ascendantes, les approches descendantes consistent à baser le développement sur la spécialisation des choix conceptuels, et donc respectent au mieux ces derniers. Ces méthodologies sont complexes et peu convergentes, et par conséquent souvent peu opérationnalisées dans des démarches efficaces. La pluridisciplinarité des équipes de développement requise par les SIM afin de s'adapter à des contextes et à des domaines d'application différents ne fait qu'amplifier la complexité des cycles de développement. Les variations d'expertise concernées sont alors souvent cloisonnées, avec peu de mécanismes de collaboration, mis à part quelques notations et formalismes abstraits peu adaptés à la pluridisciplinarité des équipes de développement.

La réalité virtuelle, tout en partageant un certain nombre de propriétés et de technologies avec les systèmes interactifs mixtes (interactivité, environnement tridimensionnel, ubiquité, etc.) bénéficie d'un effort important de standardisation des technologies et des outils de développement. Les investissements immenses par les différents acteurs du domaine de la RV (fabriquants de cartes graphiques, éditeurs de logiciels et de jeux, laboratoires de recherches, comités de standardisation, etc.) ont servi à faire baisser ses coûts d'exploitation, et à étendre son champs d'application à des secteurs d'activité divers. La réalité virtuelle est alors souvent utilisée pour

le prototypage de systèmes complexes, comme par exemple dans les domaines du bâtiment ou des infrastructures. Son usage participe alors à la coordination du travail des acteurs concernés par les projets en cours de réalisation, ou à la démonstration et visualisation des résultats de conception.

L'évolution continue des recherches dans les domaines des systèmes interactifs mixtes et de la réalité virtuelle montre néanmoins les limites du rapprochement entre les deux domaines. Alors que l'essentiel des moyens en réalité virtuelle est investi pour améliorer le rendu, la modélisation des objets et des mouvements ou la qualité de l'immersion, les systèmes interactifs mixtes tirent profit de ces avancées mais concentrent leurs efforts sur l'intégration des différentes technologies mises en oeuvre et sur l'interaction des systèmes résultants avec les utilisateurs et avec l'environnement de cette interaction. Cette divergence émane surtout des incertitudes liées aux environnements des systèmes interactifs mixtes comparés aux environnements virtuels, en général bien contrôlés.

1.1 Proposition

A la frontière des deux disciplines, nous souhaitons tirer profit des avantages de la réalité virtuelle comme une technologie bien rodée, et de la proximité des environnements virtuels avec les environnements mixtes, pour apporter une contribution au niveau des méthodologies de conception et d'évaluation des interactions dans les systèmes interactifs mixtes. Cette thèse propose pour cela de contribuer à l'amélioration du processus de développement pour prendre en compte les diversités d'approches et d'expertises. Cette contribution s'articule autour d'une approche de prototypage que nous jugeons bien adaptée au contexte des SIM : le prototypage par simulation. Alors que le terme "simulation" est souvent utilisé dans le contexte du prototypage pour signifier l'émulation de la plateforme matérielle, la simplification du fonctionnement de certaines composantes ou l'imitation d'une logique applicative, nous proposons d'intégrer ces différentes facettes dans une démarche de simulation de l'environnement de l'interaction dans sa globalité. Le but d'un tel prototypage est de permettre d'éviter de déployer des configurations matérielles et logicielles complexes, tant que les choix de conception n'ont pas suffisamment convergé. Nous montrerons aussi que le prototypage par simulation permet la réutilisation des architectures des

prototypes simulés pour le développement des systèmes finaux. Les prototypes deviennent alors moins coûteux à implémenter et permettent aux différents acteurs de communiquer leurs visions des systèmes avec un support concret et compréhensible, complétant ainsi les formalismes déclaratifs abstraits.

1.2 Structure de cette thèse

Cette thèse est divisée en deux grandes parties. La première partie explore l'état de l'art des approches existantes de développement des systèmes mixtes. Un premier chapitre de cette partie présente l'historique et les définitions principales du domaine des SIM, ainsi que les particularités de ces systèmes par rapport aux applications interactives "classiques". Un second chapitre présente les outils existants d'analyse, de conception et de développement des SIM, selon l'approche globale utilisée et le degré et la nature de l'aide apportée aux développeurs. Ces méthodes conduisant souvent à des processus de développement itératifs, les approches de prototypage existantes utilisées pour faire baisser les coûts des implémentations successives sont alors présentées. Une analyse de l'état de l'art nous permet finalement d'identifier les limitations des approches actuelles.

La deuxième partie de cette thèse présente notre contribution. Un premier chapitre présente les objectifs de cette thèse, et identifie les principales difficultés associées à la réalisation de ces objectifs. Les outils utilisés pour opérationnaliser les différents concepts et idées avancés dans cette thèse sont également présentés dans ce premier chapitre. Un second chapitre présente le premier volet de SIMBA, notre outil support à l'opérationnalisation de l'approche de prototypage par simulation. Il s'agit donc en particulier de présenter l'architecture de services de simulation et d'interaction que nous avons déployée. L'architecture des services est définie pour mieux répondre aux objectifs de notre travail et contourner les principales difficultés associées à la méthodologie adoptée. Le deuxième volet de SIMBA est présenté dans le troisième chapitre de notre contribution. Il consiste en un espace de modélisation qui permet de compléter les choix de conception, et définit un certain nombre de règles et d'outils qui facilitent le processus de développement des systèmes interactifs mixtes. Le processus de développement résultant ainsi que les différents rôles des acteurs concernés sont également présentés dans ce chapitre. Le

quatrième chapitre applique SIMBA à deux cas d'études afin d'illustrer la méthodologie et les concepts présentés tout au long de cette thèse. Ces cas d'études nous permettent par la suite d'identifier un certain nombre de perspectives pour faire évoluer SIMBA, présentées dans le chapitre final de cette thèse.

Première partie

Historique et état de l'art

1

Introduction

1.1 Historique et définitions

Le premier système de réalité augmentée a été décrit par Ivan Sutherland en 1968 [Sut68], un système où les entités du monde physique sont augmentées par des informations et des annotations numériques. Le terme “réalité augmentée” fut introduit plus tard par Tom Caudell en 1990 [CM92]. Contrairement à la réalité virtuelle, la réalité augmentée vise non pas à remplacer mais à enrichir l'environnement de l'utilisateur. L'augmentation a pour but d'aider à la réalisation d'une tâche, ou d'enrichir l'interaction [FMS93].

Selon Azuma [Azu97], un système de réalité augmentée doit répondre à trois critères :

- combiner physique et numérique ;
- l'interaction doit se faire en temps réel ;
- l'interaction doit avoir lieu dans un espace tridimensionnel ;

[Pon04] identifie quatre caractéristiques des systèmes de réalité augmentée, qu'ils partagent d'ailleurs avec l'ensemble du domaine des GVAR (Game, Virtual and Augmented Reality) :

- son et graphique 3D ;
- performance en temps réel ;
- interaction utilisateur-machine placée au centre du système ;
- simulation de personnages tridimensionnels ;

D'autres définitions de la réalité augmentée [ABB+01] ne limitent pas l'augmentation à l'aspect visuel, mais l'étendent également aux autres sens. Cette définition est moins restrictive, et permet

d'inclure toute augmentation numérique des informations perçues par les sens humains.

Le terme réalité mixte fait référence à des systèmes qui mélangent des entités numériques et physiques au sein d'une seule et même représentation. Milgram [MK94] définit alors un continuum linéaire qui va du physique au numérique, et qui englobe la réalité augmentée, mais aussi la virtualité augmentée (figure 1.1).

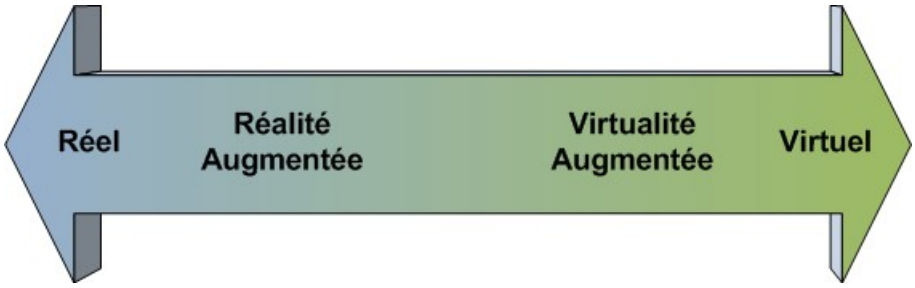

FIGURE 1.1 – Continuum de Milgram

Les systèmes interactifs mixtes (SIM) [Dub01] s'appuient sur l'utilisation de l'environnement physique de l'utilisateur comme support à l'interaction entre cet utilisateur et le système informatique. Ces systèmes couplent les capacités de traitement et de stockage informatiques aux capacités naturelles des utilisateurs à manipuler et interagir avec des entités physiques. Le continuum de Milgram est alors remplacé par une taxonomie basée sur l'objet de la tâche et le type d'augmentation (figure 1.2).

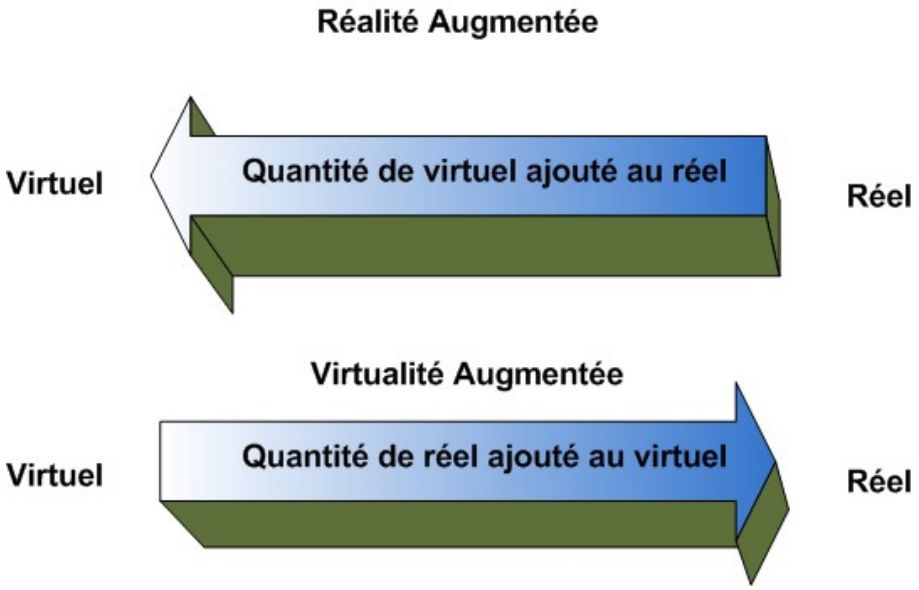

FIGURE 1.2 – Classification selon l'objet de la tâche et le type d'augmentation

Cette définition englobe toute augmentation de l'interaction associée à une tâche particulière, même si la perception de cette augmentation n'est pas centrale au déroulement de l'interaction.

Le développement de l'aspect interactif des systèmes mixtes requiert une prise en compte particulière des multiples facettes qui peuvent influencer cette interactivité [MFFM98] :

- les types de données fournies aux utilisateurs, allant du simple texte aux graphiques tridimensionnels ou au retour haptique ;
- les cibles physiques potentielles de l'augmentation, pouvant inclure les utilisateurs, les objets physiques ou l'environnement ;
- l'adéquation des données fournies à la tâche et à l'environnement où elles seront perçues ;
- la capacité des systèmes à réduire l'écart entre les entités physiques et numériques ;

Les SIM ne cessent d'évoluer, grâce surtout à l'attractivité et à l'aspect innovant des technologies mises en jeu, ainsi que l'effet "magique" de faire coexister deux dimensions à première vue très différentes. Ces systèmes se déclinent souvent sous différentes formes selon le contexte et les outils mis en jeu. Le terme SIM englobe alors les interfaces tangibles [PTB+01], les systèmes pervasifs [BML05], les systèmes ubiquitaires [Wei95], la réalité augmentée, la réalité mixte, la virtualité augmentée, les systèmes mobiles [TYT92], etc. Néanmoins, ces systèmes partagent un certain nombre de spécificités qu'il est utile d'identifier.

1.2 Spécificités des systèmes interactifs mixtes

L'évolution rapide des systèmes mixtes ainsi que son lien étroit avec l'évolution des technologies mises en jeu, ont initialement conduit à privilégier les solutions rapides au détriment de la rigueur de la démarche de développement. Ceci est d'autant plus vrai que la plupart des systèmes mélangeant physique et numérique peinent à résoudre les problèmes techniques de localisation, de calibrage, et de réalisme de la représentation. L'absence de formalismes a longtemps privé ces systèmes de méthodologies systématiques pour piloter les processus de développement, ainsi que pour concevoir et évaluer l'interaction des utilisateurs. Développer un système interactif mixte est donc resté longtemps assimilable à une tâche complexe et non intuitive, du fait des différences importantes qu'ont ces systèmes avec les systèmes interactifs classiques, que ce soit au niveau des sorties, des entrées, ou de l'interactivité.

1.2.1 En sortie

Apporter les capacités de traitement informatique à l'environnement physique de l'utilisateur nécessite souvent de réaliser une fusion ou intégration des différents éléments et représentations physiques et numériques. La qualité du mélange dépend aussi bien du périphérique de sortie que de l'adéquation des outils et techniques logiciels aux tâches et aux environnements de l'interaction.

Les configurations de gestion de l'aspect visuel du mélange sont diverses et variées, mais bénéficient en général d'une standardisation de leurs interfaces matérielles et logicielles. Néanmoins, les concepteurs doivent porter une attention particulière aux aspects d'utilisabilité et d'ergonomie de ces périphériques.

Moins répandu que l'aspect visuel, l'aspect auditif peut aussi jouer un rôle dans l'augmentation de l'environnement. Ceci est d'autant plus important que la superposition visuelle d'informations peut s'avérer dangereuse dans la mesure où elle cacherait des élément essentiels de l'environnement. Le son 3D consiste souvent en une dissémination dans l'environnement de haut-parleurs, ou la simulation logicielle d'une telle configuration [AL00]. L'augmentation peut également porter sur les autres sens, mais ceci demeure à l'état actuel réservé à des projets de recherche exploratoires.

1.2.2 En entrée

La nature même de l'interaction dans un environnement mixte exclut l'utilisation des périphériques classiques tels le clavier ou la souris. L'expérience dans le domaine de la réalité virtuelle montre que ces périphériques ne sont adaptables que pour des cas bien particuliers, où l'interaction est limitée à certains gestes principalement bidimensionnels (jeux de tirs à la première personne ou de stratégie), et s'opérant principalement devant un écran d'ordinateur. En revanche, les situations de mobilité ou d'interaction tridimensionnelles assez communes dans les SIM, nécessitent l'usage de périphériques et d'outils adéquats. Le mélange des entités numériques et physiques dépend souvent de la position et d'autres attributs physiques de l'utilisateur et des objets, ce qui conduit à des formes d'entrées implicites qui ne nécessitent pas d'actions particulières (suivi du regard et du corps, capteurs divers, etc.)

L'absence de périphériques standards, la diversification des formes d'entrées ainsi que leur dissémination potentielle dans l'environnement, confèrent à la gestion des entrées un rôle très important dans le développement des SIM.

1.2.3 Interactivité

Les données fournies par les dispositifs sont récupérées par le coeur du système afin d'être interprétées en termes de gestes et d'actions. Souvent dans le cadre des SIM, une donnée seule ne fournit pas d'informations suffisantes sur l'action de l'utilisateur dans son intégralité, et une succession de données de différentes sources est parfois requise afin de signifier une action particulière. L'interaction de l'utilisateur correspond alors à une interprétation contextuelle des données récupérées en entrée puis fusionnées, une tâche qui peut s'avérer complexe.

La gestion des particularités des SIM nécessite des outils de conception et de développement appropriés. Ces outils se répartissent entre les outils d'aide à l'implémentation logicielle, et les outils d'aide à l'exploration de solutions conceptuelles.

2

Outils pour le développement des SIM

Les premières recherches conduites dans le domaine des systèmes mixtes ont visé l'amélioration des performances et des fonctionnalités de ces systèmes, via l'enrichissement continu des algorithmes et outils existants. Ces avancées se sont souvent concentrées sur les aspects de localisation, de reconnaissance, et de calibrage [Azu97], ainsi que sur l'amélioration des plateformes de distribution et de collaboration. La technologie arrivant à une maturation suffisante pour développer des systèmes utiles, les contraintes et besoins des utilisateurs de ces systèmes sont devenus de plus en plus pressants, nécessitant le recours à des plateformes de développement spécialisées [ABB+01].

[DGHP02] définit les caractéristiques d'une plateforme de développement de systèmes mixtes comme une architecture ouverte, nécessitant peu d'expertise, et permettant d'intégrer différents outils. L'intégration des outils permet de gérer la croissance continue du nombre de concepts et d'artefacts hétérogènes mis en oeuvre dans le développement de tels systèmes. L'ouverture permet en parallèle de développer des systèmes adaptés à divers domaines, tels la visualisation scientifique, la médecine, l'architecture, la maintenance, etc. Elle permet par exemple de traiter les contraintes spécifiques aux systèmes mobiles et ubiquitaires, liées à la mobilité, à la collaboration multi-utilisateurs, ou aux grands espaces couverts par ces systèmes. Une architecture ouverte peut également s'adapter plus facilement aux expertises requises pour la résolution de

ces contraintes.

Adapter les plateformes aux différents niveaux et domaines d'expertises est d'autant plus complexe que le développement des SIM opère sur deux fronts différents :

- d'une part la conception de l'environnement et de l'interaction, visant à créer une expérience fluide et améliorer en priorité l'utilisabilité du système. Cette notion de la conception revêt une importance majeure dans les domaines de l'amusement ou de l'éducation, appelés aussi domaines expérientiels.
- de l'autre, la notion d'ingénierie visant à travers la conception à définir un ensemble de plans, modèles et démarches cohérentes prenant en charge le développement logiciel et le déploiement des systèmes, et à en assurer le bon fonctionnement. Cette vision de la conception et du développement est souvent adoptée dans le cas de systèmes " orientés vers les tâches ".

Cette tension entre les deux approches de conception favorise souvent un axe au détriment de l'autre. Les plateformes de développement issues de la vision de l'ingénieur adressent le coeur fonctionnel du système informatique et adoptent une approche dite ascendante, qui s'appuie principalement sur un ensemble d'outils d'aide au développement logiciel. La deuxième approche porte l'attention sur les aspects d'interaction de l'utilisateur, et favorise l'amélioration de l'utilisabilité des systèmes, souvent au détriment de l'extensibilité ou la réutilisabilité des modules. Une telle approche implique souvent des conceptions globales couvrant l'intégralité de l'aspect interactif, et une spécialisation graduelle de ces conceptions menant à des démarches descendantes.

Afin de dresser une classification des outils existants, nous nous basons sur un certain nombre de caractéristiques jugées pertinentes au développement des systèmes interactifs mixtes. Les outils existants sont premièrement divisés en deux grandes familles, les outils offrant un support au développement logiciel des SIM, et les outils qui offrent un support à la conception méthodique de l'interaction dans ces systèmes. Une classification plus fine nous permet par la suite de comparer les outils au sein de chacune de ces deux catégories.

2.1 Les outils logiciels

La multiplication des dispositifs d'affichage et de localisation ainsi que la diversification des périphériques et des techniques d'interaction, nécessitent des outils de gestion et de développement performants et configurables. La maturation du domaine des systèmes mixtes a conduit à l'élaboration d'un grand nombre d'approches logicielles, couvrant les différents aspects du développement. L'objectif commun de ces approches est d'accélérer le processus de développement ainsi que d'étendre les domaines d'application des SIM, en permettant à l'application une meilleure adaptation à différents contextes, profils d'utilisateurs et configurations matérielles.

Globalement, nous pouvons identifier quatre axes selon lesquels se différencient ces outils : la couverture, les contraintes architecturales, les services de l'environnement d'exécution, et les outils fournis aux développeurs.

2.1.1 Couverture

En vue de la diversité des technologies et expertises mises en oeuvre dans le contexte des systèmes interactifs mixtes, différents outils proposent des aides couvrant plus ou moins de facettes du développement logiciel de ces systèmes. Alors que les boîtes à outils gèrent souvent les aspects liés aux entrées, aux sorties ou aux interactions, les cadres de développement (*framework*) proposent des solutions plus complètes mais plus contraignantes d'un point de vue architectural. Les boîtes à outils sont des bibliothèques de modules réutilisables, offrant un certain nombre de fonctionnalités similaires dans des domaines prédéfinis. La figure 2.1 représente ces boîtes selon les domaines couverts et le niveau d'abstraction. Les fonctionnalités sont souvent optimisées pour ces domaines, offrant de meilleures propriétés que les solutions logicielles génériques.

VRPN (Virtual Reality Peripheral Network) [RMTHS+01] prend en charge la gestion des entrées des périphériques physiques. L'interface de programmation offre un accès uniformisé aux données de ces périphériques, ce qui permet de coder des applications quasi-indépendantes des configurations matérielles particulières. Ces configurations sont définies séparément du code de l'application dans des fichiers textuels plats. Les mécanismes de gestion des entrées de VRPN sont de bas niveau et mal adaptés à des systèmes complexes et à grande échelle. En particulier, les fichiers textuels plats sont difficiles à gérer par des outils logiciels, comparés par exemple à

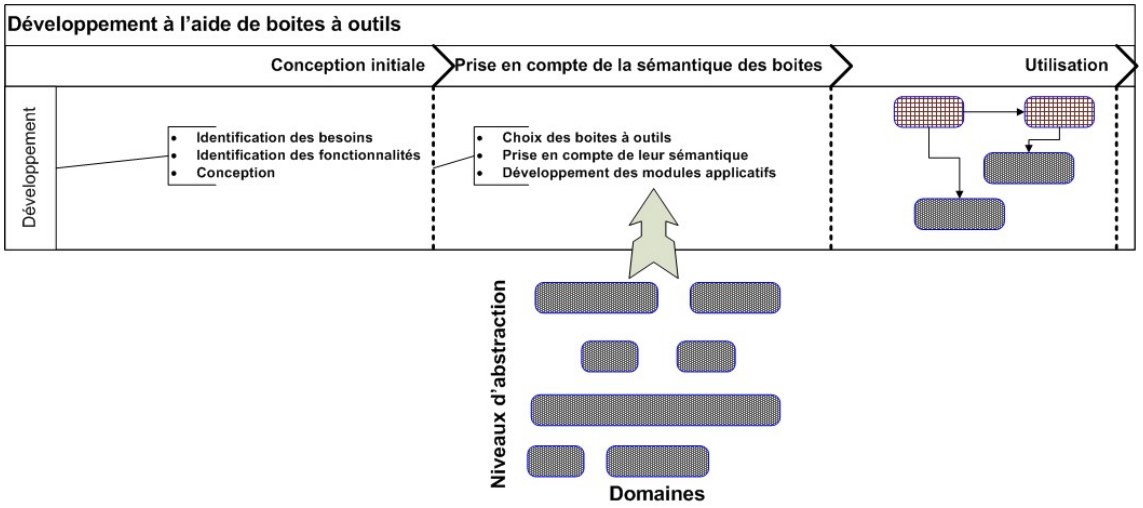

FIGURE 2.1 – Boites à outils

la flexibilité des fichiers XML.

ARToolkit [KB99] prend en charge une partie importante du pipeline des entrées qui est la localisation. Des algorithmes de vision par ordinateur assurent la localisation de marqueurs géométriques planaires par des caméras vidéo. En plus du pipeline des entrées, ARToolkit intègre GLUT [Kil96] pour une gestion partielle du pipeline de sortie. La position tridimensionnelle des marqueurs est alors utilisée pour positionner des entités graphiques en superposition au flux vidéo capté par les caméras. Néanmoins, la faible précision de ARToolkit devient assez vite une contrainte dans certains contextes particuliers. D'autres outils tels ART [Web08] proposent alors une alternative plus coûteuse mais beaucoup plus précise pour une localisation à base de caméras.

D'autres boîtes à outils proposent une gestion plus évoluée des entrées et des interactions dans les systèmes mixtes. MAPPER [FSS97] définit par exemple une solution logicielle multi-couches offrant différents niveaux d'abstraction pour l'accès aux données des périphériques. Les niveaux les plus élevés factorisent et traitent les données provenant des couches basses, conformément à des techniques et métaphores d'interaction pré-configurées. Le couplage entre les différentes couches est de bas niveau, comparé à d'autres boîtes à outils tels MRPlatform [UTS+02], conçue pour assurer un maximum de flexibilité pour la gestion des entrées des dispositifs et la localisation à base de marqueurs.

Malgré leur facilité d'intégration, les boîtes à outils sont souvent limitées à offrir des services

dans des domaines orthogonaux [Pon04]. D'autres plateformes plus complètes prennent simultanément en charge la gestion des différentes facettes des systèmes mixtes, et constituent de ce fait des cadres de développement de ces systèmes. Ces cadres proposent des solutions partiellement finalisées, complétées par les développeurs afin de générer les systèmes finaux [Rog97]. La complétude des solutions proposées impose un certain nombre de contraintes architecturales qui constituent une des caractéristiques les plus importantes des cadres de développement.

2.1.2 Les contraintes architecturales

Une architecture dans le contexte de notre travail est l'ensemble des choix décisionnels organisant les modules d'un système. Ceci inclut la structure de ces modules ainsi que les modalités de collaboration et de composition. L'utilisation d'un cadre de développement nécessite souvent d'adhérer à son architecture, une contrainte qui peut s'avérer limitative selon le contexte (*Freedom from choice vs. Freedom of choice*). Les modules de l'application sont contrôlés par les cadres eux-mêmes (figure 2.2), un principe communément appelé le principe "Hollywood" (*" Don't call us, We call you "*)[Inc08].

Contrairement aux boîtes à outils, les architectures des cadres de développement dictent celles des applications, ce qui rend l'étude de cette contrainte primordiale pour la caractérisation de ces outils. Trois grandes catégories de cadres émergent alors de cette définition : les cadres orientés objets, les cadres orientés composants, et les cadres adaptatifs.

2.1.2.1 Les cadres orientés objets

Les objets sont un paradigme d'encapsulation des données d'un système avec les traitements associés à ces données. Des motifs de conception (*design patterns*), tels la programmation à base d'interfaces, permettent de séparer les méthodes offertes par les objets des implémentations particulières de ces méthodes. A travers ses interfaces, un objet expose les méthodes qu'il offre, mais il n'existe pas de moyen standard pour expliciter les méthodes ou fonctionnalités dont dépend cet objet.

Coterie [MF96] et TinMith [PT01] sont deux cadres de développement orientés objets. Les deux systèmes définissent des architectures modulaires. L'interconnexion des différents modules permet alors de définir des applications variées. Coterie combine un langage compilé, Modula3,

FIGURE 2.2 – Principe Hollywood

et un langage interprété, Repo-3D [MF98], qui permet le couplage fonctionnel des modules développés avec Modula3. Une fois le couplage établi, les modules connectés peuvent communiquer via deux mécanismes : l'invocation de procédures et la mémoire partagée. Les mécanismes de communication et la granularité des modules demeurent largement dépendants des mécanismes des systèmes d'exploitation.

TinMith en revanche fournit uniquement des moyens de couplage structurel entre les modules. Les applications profitent d'une architecture en plusieurs couches, offrant différents niveaux d'abstraction. La communication entre les modules utilise un modèle de flux de données, avec des modules de traitement opérant sur les données échangées. Le modèle de communication est flexible, et s'adapte très bien au passage à l'échelle des systèmes développés.

Le coût des systèmes n'étant pas calculé uniquement sur la phase de développement initiale mais sur toute la durée de vie d'un système, l'évolutivité, la réutilisabilité, et la facilité de main-

tenance revêtent une importance accrue, précisément dans le cadre des systèmes complexes tels que les systèmes mixtes. En effet, ces derniers nécessitent souvent l'intégration d'outils hétérogènes [Pon04], et il est rare que l'ensemble de ces outils soit développé par une même personne du fait des grandes variations d'expertises impliquées (programmation 3D, interaction, capteurs de position et calibrage, etc.). L'évolution des technologies mises en jeu ne s'opérant pas à la même cadence, il est essentiel de pouvoir développer les sous-parties d'un système séparément. Cette évolution parallèle requiert la disponibilité de mécanismes de connexion bien définis, et l'identification explicite des dépendances des modules. Dans ce sens, les objets demeurent quelque peu limitatifs puisqu'ils ne définissent explicitement qu'une seule des deux facettes de leurs dépendances. En contre-partie, les composants explicitent clairement leurs tailles, structures et dépendances.

2.1.2.2 Cadres orientés composants

Un composant expose ses dépendances par le biais d'interfaces qui expriment les services fournis, mais aussi les services dont il dépend. Les contrats entre les différents modules d'un système sont alors mieux définis que dans le cas des objets. Un système orienté composants peut alors être comparé à un jeu de LEGO, où le système entier est assemblé à partir de ses petits composants. Ces même composants peuvent également servir à l'élaboration d'autres systèmes [DGHP02] .

Studierstube [Fuh99] combine une gestion de la facette de sortie des systèmes mixtes basée sur un graphe de scène orienté objets, et une gestion des entrées et des interactions utilisant OpenTracker [RS01] orienté composants. Le pipeline d'acquisition et de traitement des entrées est alors constitué de modules aux interfaces fortement typées, avec un mode de communication à base de messages. Ces modules sont responsables de l'acquisition des données auprès des périphériques physiques, du traitement de ces données et de leur application aux objets du graphe de scène. Cette dépendance partielle vis à vis des objets d'un graphe de scène orienté objets rend une partie de l'architecture difficilement accessible ou configurable. Des approches plus évoluées utilisent les composants pour la définition de tous leurs modules et services. Dans cette catégorie, on trouve DART [MGB+03], AMIRE [AMI04] et DWARF [SK05].

DART propose un environnement de création de systèmes mixtes sur la base des méthodes

de création de contenu multimédia. Ses composants sont principalement un ensemble d'extensions des composants de Adobe Director (anciennement Macromedia Director) [Hom08]. Les composants exposent un certain nombre d'attributs et propriétés éditables par les développeurs. DART supporte deux modes de couplage : un couplage structurel qui permet d'associer les dépendances des composants, et un couplage fonctionnel qui permet d'élaborer des fonctionnalités plus évoluées avec une composition logique de comportements plus simples. Néanmoins, l'approche orientée composants ne suffit pas à isoler les applications des mécanismes sous-jacents de Director, et il est toujours plus facile de définir des greffons (*plug-ins*) pour étendre les fonctionnalités de ce dernier que d'aborder une approche de développement de systèmes par composition.

AMIRE adresse cette limitation en proposant une architecture encore plus modulaire. Les systèmes sont définis comme un ensemble de composants englobant les fonctionnalités du système, mais aussi les entités géométriques quand ceci se révèle nécessaire. Ce couplage entre géométrie et fonctionnalités simplifie l'approche de développement par composition. Des composants complexes peuvent être assemblés à partir de composants plus simples. DART et AMIRE utilisent tous les deux un mode de communication à base d'événements. Néanmoins, AMIRE ne fournit pas de mécanismes de couplage fonctionnel. Les architectures de ces deux approches les rendent adaptées au développement de systèmes expérientiels, mais peu adéquats à la composition de systèmes complexes déployés dans des environnements industriels. En effet, l'approche globale demeure largement inspirée de la création de contenu (*authoring*) nécessitant peu d'expertise, mais offrant en même temps moins de marge pour l'intégration de nouvelles fonctionnalités.

A l'opposé de AMIRE et DART, DWARF offre une plateforme adaptée au développement des systèmes orientés vers les tâches. Le cadre de développement privilégie la flexibilité de l'architecture et la facilité d'intégration des composants pour l'élaboration des systèmes. Ces composants agissent comme des boîtes noires exposant leurs nécessités appelées "besoins", et leurs offres appelées "habilités". Ces interfaces sont typées par les données qu'elles fournissent et les relations sont gérées par l'infrastructure de DWARF. Des gestionnaires de dépendances permettent de découvrir les besoins et habilités compatibles et de les connecter. Ces connexions peuvent être modifiées pendant l'exécution de l'application, sans intervention de l'utilisateur (changement de contexte lié à la mobilité par exemple). Un système développé avec DWARF est alors constitué de trois couches différentes : les deux couches basses abritent les services bas niveau et haut

niveau de DWARF. La couche applicative abrite le coeur fonctionnel et logique du système. Cette architecture multi-couche est assez flexible, et permet l'accès de l'application à différents services de bas niveau, sans devoir passer par les couches intermédiaires. Ceci peut s'avérer très avantageux du point de vue des performances pour les systèmes à ressources limitées. L'architecture de DWARF le rend très utile pour la génération de systèmes sur la base de plateformes matérielles relativement similaires. Le gestionnaire de dépendances offre un niveau de flexibilité supérieur au couplage statique des composants adopté par AMIRE. Ce mode de couplage nécessite néanmoins une connaissance préalable des types des interfaces requises ou fournies, même si les composants qui les exposent sont découverts en temps réel. Il demeure alors difficile de développer des systèmes à partir de capteurs hétérogènes [VL06].

Contrairement à ces cadres de développement, les cadres dédiés au développement de systèmes interactifs mixtes mobiles ou ubiquitaires privilégient l'aspect " intuitif " (*seamless*) des transitions entre différents cadres d'interaction et d'exécution. Concrètement, les interfaces offertes ou requises par les composants du système et de l'environnement peuvent être découvertes pendant le déroulement de l'interaction.

2.1.2.3 Cadres adaptatifs

Ces cadres peuvent être orientés objets ou composants. Néanmoins, la découverte et la connexion des services étant entièrement gérées par le cadre lui-même, ces différences s'estompent au profit d'une automatisation totale et en temps réel de la gestion des interdépendances. Deux cadres de développement spécialisés dans ce genre de système sont le cadre proposé par Veas [VKT05] et WCOMP [CTLR06]. Le premier développe une infrastructure qui supporte non seulement l'extension de la couche applicative, mais aussi l'évolutivité de l'infrastructure elle-même, en assurant la portabilité de l'application. L'architecture est optimisée pour permettre une reconfiguration intuitive des modes d'accès aux services, en fonction de la mobilité de l'utilisateur, de ses préférences ainsi que des besoins de l'application. La découverte des services dépend d'un méta-modèle qui définit les règles de description des fonctionnalités offertes, ainsi que les modes d'échanges supportés par les composants et leurs attributs. L'implémentation est orientée objets, avec des mécanismes de réflexion [1] bas niveau remplaçant les définitions

1. Processus par lequel un programme informatique peut observer et modifier sa propre structure

explicites des dépendances.

Wcomp propose une approche plus complète en identifiant les contraintes et besoins liés aux contextes mobiles et ubiquitaires. Le cadre propose en particulier des mécanismes de gestion des ressources et du stockage d'informations dans un contexte peu stable, ainsi que des moyens d'intégration de services transversaux. Un niveau décisionnel (figure 2.3) permet de piloter les services du cadre et de l'infrastructure, et d'adapter les schémas applicatifs aux variations de contexte. Les liens entre les composants de WCOMP sont définis dans un langage d'interaction ISL qui représente les interactions comme des entités à part entières, ce qui permet de les manipuler et les modifier à travers la couche décisionnelle. Contrairement à l'approche proposée par Veas, WCOMP adopte une approche complètement orientée composants. Les mécanismes de réflexion sont alors remplacés par les fonctionnalités offertes par la plateforme d'implémentation (orientée composants) sous-jacente (.NET, J2EE ou WebServices [WQ05]).

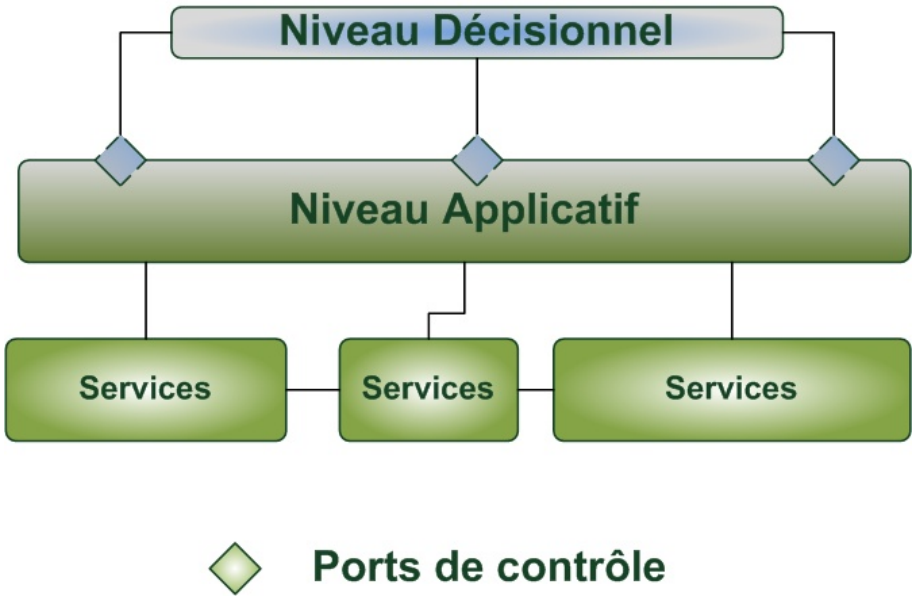

FIGURE 2.3 – Couche décisionnelle

Malgré leur importance, les contraintes architecturales ne sont pas le seul axe de différentiation des outils logiciels. Ces architectures structurent un certain nombre de services dont la richesse peut également privilégier un outil ou un autre.

2.1.3 Services de l'environnement d'exécution

Les différentes plateformes de développement offrent un certain nombre de services, dont le type et le nombre dépendent de l'orientation globale de ces plateformes (systèmes d'informations, découverte d'interfaces, etc.). Ces services articulent les applications et systèmes développés, et jouent un rôle important dans l'orientation des choix des cadres appropriés. Ces services caractérisent également la catégorie à laquelle appartient la plateforme de développement [FS97] :

- les plateformes d'infrastructure : elles facilitent le développement de l'infrastructure de l'application. Ceci inclut en général la gestion des ressources, les communications avec le système d'exploitation, la gestion du parallélisme, la synchronisation et l'accès aux bases de données.
- les plateformes de type intergiciel : elles facilitent l'intégration de différentes applications et simplifient leur portabilité sur différentes plateformes logicielles et matérielles.
- les plateformes spécifiques aux domaines : elles prennent en charge les aspects utiles au développement des applications dans des domaines particuliers.

Les outils dédiés au développement des systèmes mixtes appartiennent globalement à cette dernière catégorie et offrent un certain nombre de services spécifiques à ce domaine. Certains de ces cadres intègrent néanmoins un bon nombre de services qui facilitent l'accès aux infrastructures disponibles, et augmentent l'interopérabilité des systèmes. De tels services sont génériques vis à vis du domaine des SIM, mais peuvent se révéler très intéressants selon le contexte applicatif de ces derniers.

2.1.3.1 Les services génériques

Le déploiement des systèmes interactifs mixtes dans différents domaines applicatifs, nécessite la prise en charge par ces systèmes des fonctionnalités requises dans des contextes multiples et variés. Ces fonctionnalités sont alors implémentées par les applications ou fournies par les plateformes de développement. Souvent, la disponibilité de tels services oriente le choix vers l'outil de développement, en dépit des contraintes architecturales potentielles. Parmi ces services, nous identifions la gestion de la distribution, des modes de communication asynchrones, des transactions et de la persistance.

Distribution Une plateforme qui supporte la distribution des modules est spécialement utile dans le cadre d'un système mixte à grande échelle, et précisément dans un contexte mobile ou ubiquitaire. Dans ce contexte, la distribution permet de gérer la répartition géographique des dispositifs portés ou intégrés dans l'environnement. Elle permet également la distribution des calculs complexes, et offre un support aux interactions multi-utilisateurs ou collaboratives. Cependant, développer un système distribué nécessite généralement une prise en compte des contraintes et limitations associées à cette distribution depuis les phases de conception initiales. MAPPER et VRPN utilisent par exemple un mode de distribution bas niveau basé sur un modèle client/serveur. Le changement des hôtes abritant les entités distribuées nécessite un effort de configuration conséquent.

Certains cadres de développement offrent des mécanismes de gestion de la distribution plus ou moins transparents. La transparence permet globalement d'éviter la prise en compte des aspects liés à la distribution pendant les phases de conception (à part les paramètres de performance). Studierstube utilise par exemple un graphe de scène distribué de OpenInventor appelé DIV (Distributed Open Inventor) [HSFP99]. Ceci implique que tout module est à priori distribué, et doit dériver d'un objet du graphe de scène. La transparence de cette distribution a néanmoins un prix conséquent, puisqu'elle induit une forte charge de synchronisation entre les entités distribuées.

Coterie propose un mécanisme similaire basé sur la réplication des modules de Repo. Néanmoins, afin de pallier aux surcharges de synchronisation induites par les mises à jour répétitives, Coterie définit deux autres types d'objets : les objets locaux non distribués, et les objets répliqués dont les mises à jour se font via des invocations de méthodes distantes. Ces deux nouvelles entités limitent la transparence de la distribution, et imposent un certain nombre de choix lors des phases de conception.

TinMith utilise un mode de distribution qui nécessite la génération de fonctions assurant la sérialisation des données. Le cadre de développement fournit néanmoins les outils nécessaires pour assister la génération de ces fonctions.

DWARF définit ses composants et services comme une extension des objets CORBA [HV99]. Les gestionnaires de services sont déployés sur chaque hôte du système, et communiquent entre eux pour le partage des informations sur les ressources et services disponibles. Les gestionnaires

font alors abstraction des mécanismes CORBA tout en les utilisant pour communiquer.

Le cadre de développement adaptatif WCOMP et celui proposé par Veas, fournissent également des mécanismes de gestion de la distribution transparents. La couche décisionnelle de ces cadres fait alors abstraction des mécanismes de distribution sous-jacents.

Communications asynchrones Les plateformes de distribution ont classiquement privilégié les modes de communication synchrones, comme les invocations de procédures distantes. Ce mode de fonctionnement est simple et fiable, et permet de minimiser la complexité de la couche de distribution, en témoignent les multiples standards de distribution développés (RPC, RMI, CORBA 1.x). Cependant, les modes asynchrones sont très utiles dans un contexte interactif, puisqu'ils permettent entre autres de découpler les entités émettrices et réceptrices.

Un mode de communication par événements est un mode symétrique, où aucune des entités qui interagissent n'établit explicitement la connexion avec l'autre partie. La plateforme de développement est alors responsable de fournir les mécanismes pour développer les composants indépendamment. Les outils de déploiement et d'exécution doivent également fournir les mécanismes pour établir en différé le lien entre les entités (*late binding*).

Le mode de communication par événements permet surtout de supporter des modes d'interaction pilotés par les utilisateurs ou les dispositifs. En effet, contrairement aux débuts des interactions homme-machine fortement liés aux interfaces textuelles [Agr06] où les interactions sont pilotées par le système (figure 2.4), les interfaces interactives ont depuis les travaux du centre de recherche Xerox PARC [Xer07] en 1973 et leur popularisation par Apple en 1984 [Hor96] privilégié un mode où l'interaction est pilotée par l'utilisateur (figure 2.5). Le système n'interroge plus ce dernier mais attend ses actions avant de réagir. Dans le cas des systèmes mixtes, les dispositifs ou les services divers peuvent aussi piloter le fonctionnement du système. Les deux modèles de communication sont clairement différents du point de vue applicatif puisqu'ils imposent des approches de programmation et d'interaction différentes. Dörner [DG02] rajoute même la gestion des modes symétriques comme un pré-requis des plateformes orientées composants.

TinMith utilise un mode de communication à base de messages, où l'entité émettrice invoque des fonctions de callbacks sur l'entité réceptrice. Ce mode s'apparente quelque peu à un

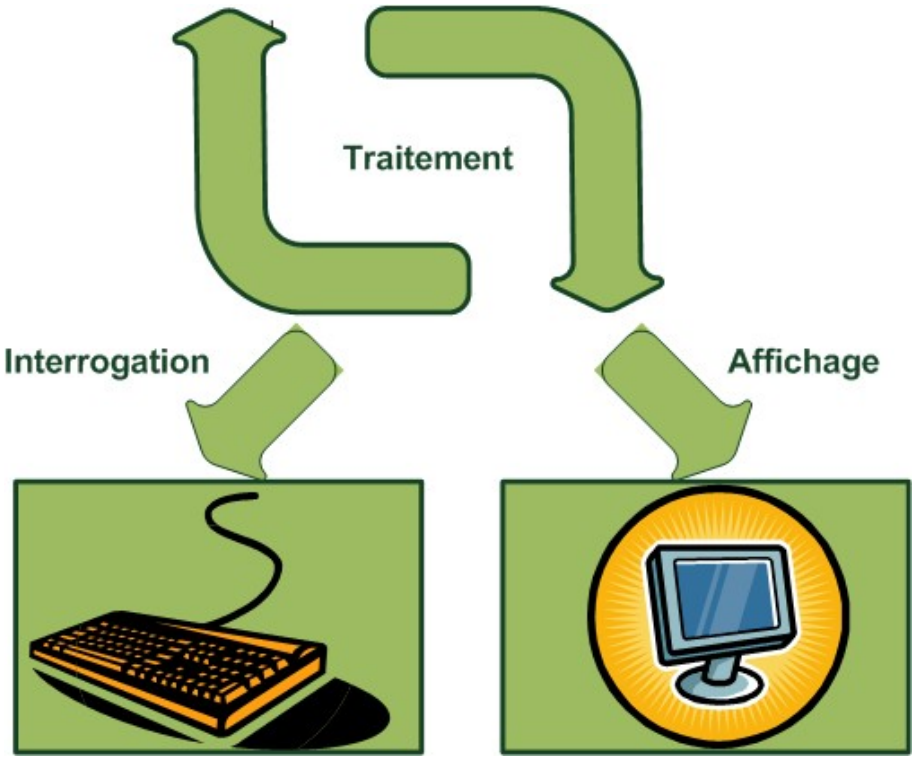

FIGURE 2.4 – Interaction pilotée par le système

mode à base d'événements, mais nécessite néanmoins la connaissance au préalable des entités réceptrices ou émettrices. Studierstube propose en contrepartie un vrai découplage des deux types d'entités afin de supporter le mode de communication requis par son interface graphique. DART, AMIRE et DWARF fournissent également des modes de communication asynchrones à base d'événements, où des services dédiés permettent d'établir le lien en différé et de gérer des files d'événements.

Services divers L'utilisation des SIM dans des contextes industriels impose à ces systèmes de pouvoir intégrer les services courants tels la persistance des états, l'accès aux bases de données ou les services de transactions. Tout comme pour la distribution, l'absence de support logiciel de la plateforme de développement impose un effort d'implémentation supplémentaire de la part des développeurs. En plus, l'intégration d'outils externes offrant ces services peut se heurter aux contraintes architecturales, lorsqu'elles existent, des outils de développement des systèmes mixtes. Dans ce contexte, on observe généralement que seuls les cadres de développement construits au-dessus des couches de services génériques offrent de telles fonctionnalités, en

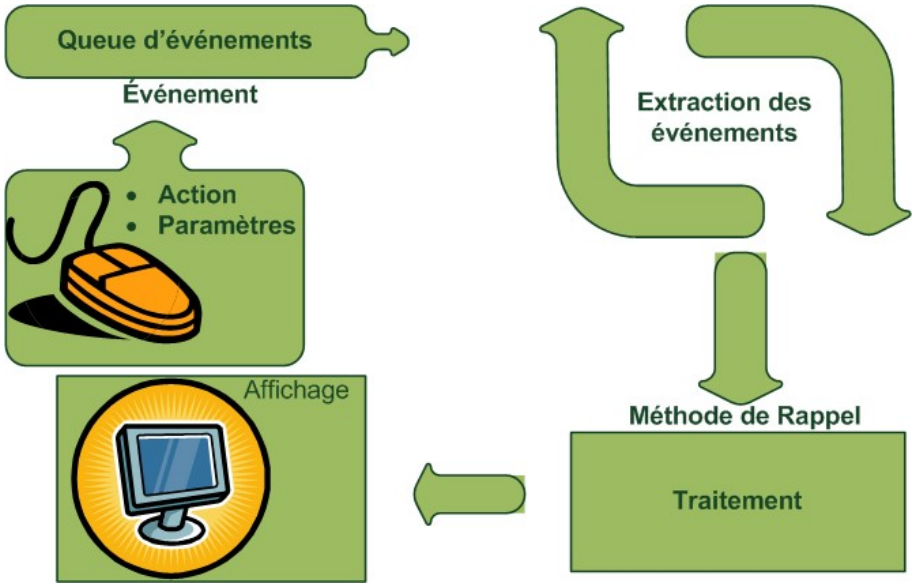

FIGURE 2.5 – Interaction pilotée par l'utilisateur

particulier DWARF qui a accès aux services de CORBA. Le cadre de développement proposé par Veas pour les systèmes mobiles et ubiquitaires fournit également des mécanismes de persistance et de transaction bas niveaux. WCOMP utilise quant à lui les services de sa plateforme d'implémentation. Tout comme pour la distribution, ces cadres fournissent des mécanismes d'abstraction des services de leurs infrastructures.

2.1.3.2 Les services spécifiques aux SIM

Le terme "services spécifiques" ne désigne pas des services réservés au développement des systèmes interactifs mixtes, mais des services et fonctionnalités qui font partie intégrante et essentielle du développement de la plupart des systèmes dans cette catégorie. Nous identifions en particulier les services de gestion du rendu, des entrées et des interactions. Ces services sont fournis par la plupart des cadres de développement des systèmes mixtes, mais leurs caractéristiques et leurs structurations varient d'un cadre à l'autre.

Coterie utilise Obliq-3D pour la gestion du rendu et l'animation des objets graphiques. L'accès à ces fonctionnalités doit être programmé en Modula3 ou en Obliq. Coterie fournit également un ensemble de modules de gestion de périphériques (lecteurs de code barre, traqueurs, calibrage, etc.).

TinMith intègre, en plus des services de gestion des périphériques physiques, un système de

localisation hybride [PATM04]. Ce système combine différentes techniques afin de permettre la localisation efficace à l'intérieur comme à l'extérieur en situation de mobilité. Côté graphique, TinMith utilise un moteur de rendu à base de graphe de scène ainsi qu'un moteur de CSG (Constuctive Solid Geometry) permettant la construction d'objets graphiques complexes à partir de primitives plus simples.

Studierstube utilise les classes de OpenInventor pour la création d'objets d'interaction tels les Widget et fenêtres 3D. Il offre aussi des mécanismes pour l'acquisition vidéo, la localisation via OpenTracker, et la gestion de dispositifs d'entrée et d'affichage divers. La gestion des entrées et des interactions est centralisée au sein d'un sous-système appelé Personal Interaction Panel-PIP.

Le système de localisation de DART utilise principalement ARToolkit, et son système de gestion de périphériques est basé sur VRPN. L'environnement permet la création de contenu ainsi que le rajout de comportements basés sur les entités "score", "stripes" et "behavior" de Macromedia Director.

AMIRE adopte une approche plus fine pour la définition de ses services. Les services de base communs à la majorité des SIM sont alors implémentés dans un répertoire de composants réutilisables appelés GEMS. On identifie par exemple des GEMS de localisation utilisant ARToolkit, de synthèse des scènes et de création de contenu utilisant OpenScenegraph [BO04], et des GEMS de gestion de périphériques d'affichage divers.

Les services de DWARF se divisent en quatre grandes catégories :

- les services de gestion de la localisation intègrent entre autre la boîte à outil ARToolkit pour une localisation à base de marqueurs ;
- les services de gestion des interfaces graphiques et du rendu des entités ;
- les services de gestion de l'interaction, utilisant notamment un moteur de tâches (*task-flow engine*) ou un moteur de réseau de Pétri ajouté dans le cadre du développement de l'application SHEEP [MSW+03] ;
- le service CAP (Context Aware Packets) qui permet de faciliter l'accès aux services extérieurs à la plateforme de DWARF ;

Les modules applicatifs des systèmes doivent être couplés aux services des outils logiciels, comme c'est le cas du service de simulation comportementale des moutons virtuels de l'application SHEEP couplé à DWARF. Les différentes étapes de couplage et de configuration sont

souvent facilitées par des outils de développement, qui constituent également un aspect important des outils de développement logiciel.

2.1.4 Services aux développeurs

Le choix de l'une ou l'autre des plateformes de développement peut dépendre des outils fournis aux développeurs qui leur permettent de concevoir et implémenter facilement les systèmes. Concrètement, ces services consistent à automatiser ou assister certaines des étapes du pipeline de développement. Une documentation accessible et précise peut également améliorer la courbe d'apprentissage et de maîtrise de l'outil. La facilité d'installation ou la vitesse d'apprentissage peuvent se révéler des facteurs déterminants dans le choix des outils appropriés. Par exemple, la complétude de DWARF et la cohérence de l'approche globale n'équilibrent pas toujours la lourdeur de l'installation du cadre de développement et des services associés [VL06]. A l'autre bout de l'échelle, les boîtes à outils ARToolkit et VRPN peuvent être installées et testées en quelques minutes. Ces outils offrent des documentations assez précises et des guides d'installation et d'utilisation qui permettent de développer assez vite les premières applications. En plus, la gratuité et la disponibilité des deux boîtes à outils les rendent idéales pour les projets à faible budget ou à caractère exploratoire. ART en revanche est plus onéreux et plus complexe à installer.

Les cadres de développement orientés composants offrent un certain nombre d'outils d'assistance au développement particulièrement intéressants. Ces cadres utilisent les propriétés des composants pour offrir des mécanismes de développement plus adaptés aux contextes d'utilisation complexes. Ces mécanismes facilitent le processus de développement, à travers des aides à la conception logicielle, au développement et au déploiement faisant abstraction des détails de bas niveau du fonctionnement logiciel [Pon04]. Parmi de telles aides, on peut citer la gestion de la portabilité, de l'interopérabilité, du cycle de vie des composants et de l'accès intuitif aux services de la plateforme.

La configuration des modules de Studierstube est liée au format de fichiers de OpenInventor. DART et AMIRE sont destinés à des non-experts/programmeurs et par conséquent offrent des outils de développement accessibles et intuitifs. Ces outils consistent en premier lieu en un environnement d'édition et de création graphique qui permet de procéder à la conception

artistique et à la composition fonctionnelle du système mixte. DART offre en plus un outil de composition de l'interaction sur un modèle de défilement temporel (figure 2.6), assez courant pour la création de contenu multimédia. Ce modèle demeure globalement inadapté aux systèmes où s'imposent des contraintes strictes, tels les systèmes orientés vers les tâches destinés à des milieux industriels. La dépendance de l'architecture de DART à Macromedia Director limite la flexibilité du cadre de développement pour répondre à d'autres contraintes que celles de la création (*authoring*).

AMIRE propose un processus de développement en deux étapes : la conception du scénario de l'interaction accomplie par l'expert du domaine, et l'identification des composants nécessaires par rapport aux besoins identifiés. Le cadre de développement fournit les outils adaptés à un tel processus, et notamment les utilitaires pour l'implémentation, l'intégration, et le déploiement des applications.

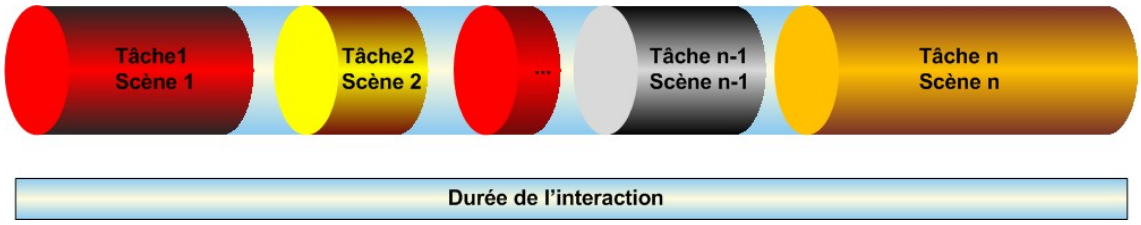

FIGURE 2.6 – Défilement temporel

La génération du code non-fonctionnel (*glue code*) adoptée par AMIRE est également utilisée par DWARF qui adapte le processus de développement au contexte orienté vers la réalisation de tâches. Ce processus identifie alors quatre acteurs : le chef de projet, le développeur du système, le développeur de composants, et l'utilisateur (figure 2.7). DWARF offre en plus des outils de génération d'interfaces graphiques 2D à partir de descriptions en CUIML [SR01]. Cette description est alors convertie en HTML ou VRML, et l'interface générée peut inter-opérer avec les services de DWARF comme la gestion des entrées de l'utilisateur. Le moteur de tâche permet également de faciliter l'implémentation de l'interaction.

2.1.5 Limitations des approches purement logicielles

Les outils et plateformes présentés définissent des approches de développement ascendantes et fournissent les services nécessaires au développement des systèmes mixtes. Les systèmes sont

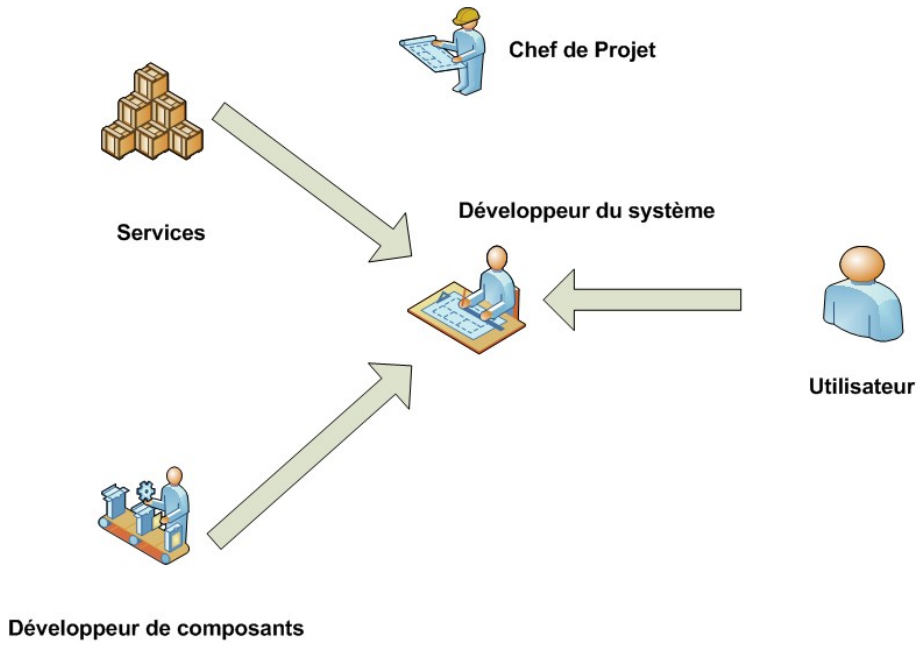

FIGURE 2.7 – Processus de DWARF

développés par composition des services de la plateforme et des modules applicatifs (figure 2.8). L'intérêt de ces plateformes vient donc des besoins du développeur des systèmes mixtes. Ce dernier peut éventuellement, s'il le juge utile, intégrer les utilisateurs dans le processus de développement, comme c'est le cas du processus proposé par DWARF. En privilégiant les besoins des développeurs, les approches ascendantes optimisent les processus de développement et d'implémentation. Elles conduisent aussi à des solutions logicielles performantes, profitant d'optimisations locales et ciblées. Il y a cependant des aspects de l'interaction entre l'utilisateur et le système ainsi que les besoins des domaines d'application qui sont mal capturés par ces approches. Les aspects du système conçu et développé via une approche ascendante demeurent par exemple peu ou pas visibles dans leur globalité, et les optimisations sont locales et restreintes aux sous-systèmes de l'application. Les développeurs, conscients de ce manque, cherchent souvent à y pallier par des méthodes informelles, puisant dans leur expertise personnelle. Ces méthodes sont peu contraignantes, mais fournissent des résultats souvent bruités et peu structurés.

Contrairement aux outils précédents, les outils d'aide à la conception offrent un support formel à l'analyse et à la conception des systèmes. Ces outils sont souvent associés à des méthodologies de développement descendantes, et sont basés sur un certain nombre de cadres conceptuels tels les taxonomies et les modèles. Le système est tout d'abord conçu dans sa globalité, puis

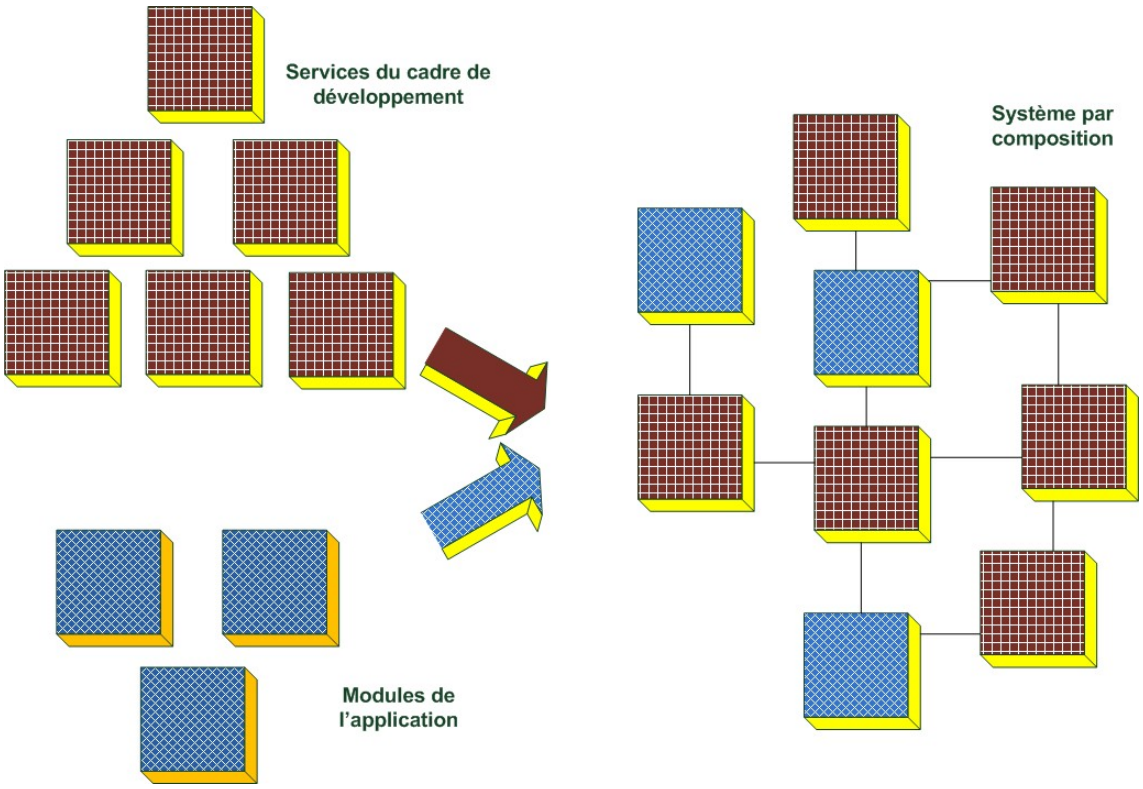

FIGURE 2.8 – Développement par composition

implémenté par décomposition et spécification des organes conceptuels. La figure 2.9 illustre un tel processus, où des services fournis par le cadre de développement sont composés avec des modules spécifiques au système en cours de développement pour obtenir le système final.

2.2 Outils d'analyse et de conception

Le développement d'un outil d'analyse et de conception formelles nécessite une identification claire des besoins des systèmes et de leurs domaines d'application, mais surtout une caractérisation des concepts et des éléments de l'espace de solutions. Une première approche à cette caractérisation dans le cadre des systèmes mixtes consiste à établir des taxonomies des différents outils et techniques mis en oeuvre dans le développement de ces systèmes. Le cadre de classification défini par les taxonomies sert de support à l'analyse de certaines facettes des systèmes, ainsi qu'à l'émergence de règles gouvernant l'utilisation des différents outils.

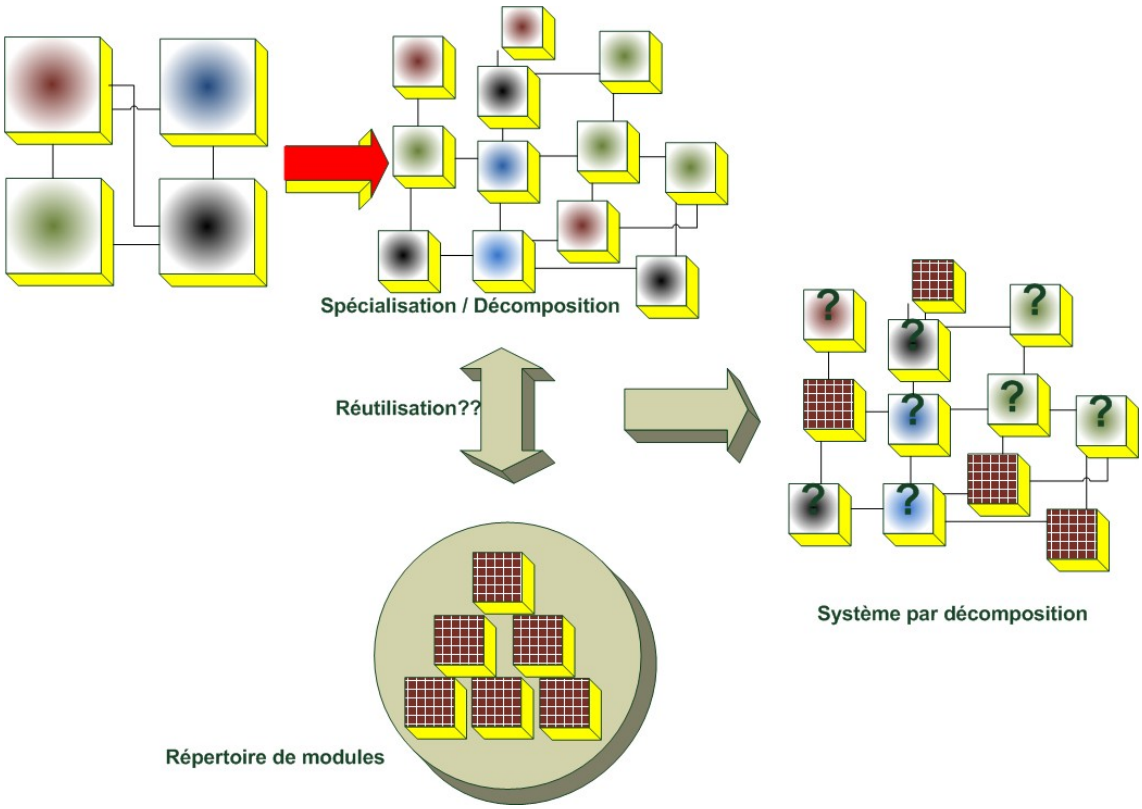

FIGURE 2.9 – Développement par décomposition

2.2.1 Taxonomies

Il existe de multiples taxonomies des outils et techniques qui interviennent dans l'élaboration des trois facettes selon lesquelles se distinguent les systèmes mixtes : les sorties, les entrées, et l'interaction. Ce paragraphe fournit un aperçu sur certaines des taxonomies les plus répandues, et permet ainsi d'aborder plus facilement les approches de conception plus évoluées.

2.2.1.1 Taxonomies des entrées

Les taxonomies des périphériques d'entrée dépendent largement des critères sélectionnés pour la classification :

- les propriétés physiques que les périphériques détectent (position, vitesse, force, etc.).
- les propriétés mécaniques des dispositifs (fixe, mobile, retour d'effort, etc.).
- les types de données que fournissent les périphériques (degrés de liberté, relatives ou absolues).
- leur contexte d'utilisation. Ceci inclut le type de manipulations à produire via ces dispo-

sitifs, l'expertise de l'utilisateur, etc.

Buxton [Bux83] utilise comme critère de classification trois types de propriétés physiques détectées par le périphérique : position, mouvement et pression. La taxonomie prend aussi en compte la dimension de la donnée mesurée (1D, 2D, 3D) ainsi que la manière dont est récupérée cette donnée (toucher ou mouvement). Le contexte d'utilisation est utilisé par Foley [FWC90] qui base sa taxonomie sur la tâche pouvant être accomplie par les périphériques.

Ces taxonomies fournissent un cadre d'analyse pour les concepteurs des interactions, en les aidant à choisir les périphériques ou classes de périphériques qui conviennent au profil de l'utilisateur ou à la tâche à réaliser. Certaines taxonomies basées sur le type de données fournies à l'application permettent en plus de définir des outils logiciels génériques pour la gestion des entrées. On note en particulier dans cette catégorie la taxonomie de GPGS [CVV77] ou de GKS3D [Hew86]. Cette dernière nous permet de définir une architecture de gestion des périphériques et des interactions dans le chapitre 2 :

- la sélection (*pick*) : cette entrée définit une sélection non graphique, une désignation d'un objet prédéfini ;
- localisation (*locator*) : une coordonnée provenant d'un périphérique de pointage ;
- bouton (*button*) : cette entrée correspond à une action de confirmation via un capteur à état binaire ;
- valeur (*valuator*) : cette donnée correspond à une valeur réelle bornée ;
- séquence (*stroke*) : elle définit une séquence de coordonnées de localisation ;
- chaîne (*string*) : définit une séquence de caractères ;

2.2.1.2 Taxonomies des sorties

Les dispositifs de sortie posent généralement moins de problèmes aux programmeurs vu la standardisation de leurs interfaces matérielles et de la flexibilité de leurs API logicielles. Ceci est moins le cas pour les concepteurs qui doivent se pencher sur les facteurs liés à l'interaction homme-machine, et qui sont très influencés par le choix des périphériques.

Il existe diverses taxonomies et recommandations générales s'appliquant surtout aux dispositifs de visualisation. Les dispositifs portés semi-transparents offrent par exemple un contact visuel direct avec l'environnement physique. Par contre, la couche semi-transparente diminue la

luminosité reçue par l'oeil et génère une sensation de flou. En plus, il demeure impossible à des objets de synthèse d'occulter des objets physiques.

Les dispositifs à base de flux vidéo offrent une meilleure opacité et coupent souvent tout contact visuel direct entre l'utilisateur et son environnement. Dans les situations où la sécurité joue un rôle prépondérant, ceci limite fortement leur utilisation au profit des casques semi-transparents ou des dispositifs tenus dans la main tels les PDA.

Les dispositifs à base de projection peuvent quant à eux servir plusieurs utilisateurs à la fois (LCD, projecteurs). Ils permettent également d'illuminer la scène, d'en modifier les textures, et sont très utilisés dans les interactions collaboratives comme [IBJU+02] par exemple. Néanmoins, il demeure difficile d'adapter le mélange physique-numérique à différents points de vues, et la mobilité reste très réduite.

Les classifications des périphériques et des dispositifs d'entrée et de sortie, malgré leur intérêt, ne renseignent pas toujours sur la nature et les caractéristiques de l'interaction. Dans ces cas, les taxonomies des tâches et techniques d'interaction peuvent s'avérer plus intéressantes.

2.2.1.3 Taxonomies de l'interaction

La compréhension approfondie des tâches pouvant être accomplies dans un système mixte ainsi que des techniques d'interaction mises en jeu, est très importante à toute approche de développement de SIM. L'hypothèse principale dans l'analyse et la classification des tâches est que ces dernières peuvent être, indépendamment de leur nature ou complexité, décomposées en des sous-tâches de base [Mun78]. On distingue par exemple la taxonomie des tâches établies dans [BKLP04], et distinguant un certain nombre de tâches de base :

- la navigation concerne principalement le mouvement de l'utilisateur mais aussi la découverte de sa trajectoire ou de sa destination.
- la sélection (*reach and grab*) représente la désignation d'un objet dans l'environnement. Dans des environnements mixtes, la sélection peut concerner des objets physiques, numériques ou hybrides.
- la manipulation spatiale rigide représente la modification de la position, la rotation ou le facteur d'échelle d'un objet de l'environnement. Tout comme la sélection, la manipulation rigide peut concerner des objets physiques et numériques.

- le contrôle du système correspond aux manipulations dont le but est de réaliser une certaine action ou de changer le mode d'interaction ou l'état du système.
- les entrées symboliques correspondent à des caractères codés ajoutés à l'environnement ou faisant partie de la communication entre plusieurs utilisateurs. Dans les systèmes mixtes, ce genre d'échange peut s'opérer dans l'environnement physique (passer une note sur un papier, etc.).
- la modélisation correspond en général à une succession de sélections, d'ajouts et de manipulations rigides des sous-composantes de l'objet à modéliser.

Une classification intéressante des techniques d'interaction est établie dans [AL00]. Elle distingue trois catégories de techniques :

- les techniques de désignation qui transposent une ou plusieurs localisations de l'espace de l'utilisateur en une ou plusieurs localisations dans l'espace de l'application.
- les sélections qui sont intimement liées à la désignation, mais font entrer en jeu plusieurs mécanismes supplémentaires tels la confirmation et l'annulation de la sélection.
- les manipulations qui consistent à faire subir à l'objet sélectionné une ou plusieurs transformations rigides dans l'espace de l'application. Les manipulations peuvent avoir une signification directe (changer de position, d'orientation) ou plus subtile (*drag and drop*).

Les taxonomies offrent un point de vue partiel sur les multiples aspects concernés par le processus d'analyse et de conception des systèmes interactifs mixtes. L'identification des tâches renseigne par exemple peu sur l'interaction en sa globalité. Une approche plus complète consiste à utiliser un langage riche pour exprimer les diverses facettes des systèmes, et permettre une exploration et une analyse méthodiques de l'espace de solution. Le langage humain demeure néanmoins difficilement interprétable dans toute sa richesse par l'outil logiciel, et le besoin de modèles adaptés à l'expression et à l'exploration des diverses facettes s'avère alors nécessaire.

2.2.2 Modèles et modélisation des systèmes mixtes

Un modèle est une abstraction qui permet de réduire la complexité associée à un système, en se focalisant sur un certain nombre d'aspects en fonction de certains objectifs ([AGLR96] et [AGG00]). Le modèle simplifie le langage naturel en un nombre limité de concepts et d'artefacts qui permettent alors de fournir un soutien logiciel aux étapes de conception, de validation et

d'interprétation. Cette simplification peut néanmoins rendre le modèle peu accessible à différents niveaux d'expertise, ou peu adapté à représenter les concepts apportés par les différents experts. On distingue en général différents types de modèles : les modèles déclaratifs, les modèles comparatifs et les modèles génératifs ou impératifs ([Men06] et [BCR06]).

Les étapes de modélisation préliminaires à toute étape de conception logicielle ou d'implémentation ont pour rôle de capturer les entités du domaine et de l'interaction, ainsi que les connaissances générales et le mode de fonctionnement du système. En ce sens, les modèles de conception doivent constituer un langage à la fois compris par les experts des domaines (sécurité, procédures, etc.), par les experts logiciels, ainsi que par les experts en interaction (cognition, IHM, etc.). Les modèles déclaratifs se révèlent souvent mieux adaptés à jouer ce rôle d'expression et d'exploration des facettes des systèmes. Coupler les modèles à des outils d'édition, d'analyse et d'interprétation permet de faciliter et d'automatiser certaines étapes du processus de conception, voire de développement.

Parmi les multiples modèles utilisés en IHM, nous distinguons en particulier deux grandes catégories : les modèles d'architecture et les modèles d'interaction. Un modèle d'architecture décrit les éléments fonctionnels de l'implémentation du système et leurs relations, et est de ce fait plus lié à la réalisation logicielle du système qu'à l'étude des propriétés de l'interaction. Un modèle d'interaction couvre les règles et les caractéristiques qui guident la conception de l'interface du système [CD04], et permet d'analyser les propriétés ergonomiques de cette interface. Trevisan [TVM02] définit un modèle d'interaction comme un ensemble de concepts, une structure de représentation et une série de primitives utilisables pour capturer explicitement les connaissances sur l'interface et le système, à un niveau d'abstraction élevé. Néanmoins, il est difficile d'établir un niveau d'abstraction de référence séparant les modèles d'interaction des modèles d'architecture, et souvent des modèles d'interaction intègrent une facette de description de l'architecture complète ou partielle des systèmes.

L'adaptation de l'outillage de conception des systèmes interactifs classiques à la conception des systèmes mixtes doit prendre en compte la complexité de l'interaction dans ces systèmes. En effet, les mécanismes d'interaction sont différents selon qu'on interagisse avec des entités physiques ou numériques, et les interactions directes entre les deux types d'entités peuvent être difficiles [KL04]. Certaines approches définissent alors des modèles et outils pensés pour ré-

pondre aux contraintes imposées par les SIM. Nous identifions trois axes majeurs selon lesquels se différencient ces outils : la richesse du modèle, le niveau d'abstraction et les outils d'opérationnalisation associés.

2.2.2.1 Richesse du modèle

La richesse d'un modèle de SIM est sa capacité à exprimer les concepts relatifs à ces systèmes, en d'autres termes son pouvoir déclaratif.

MCRpd [UI00] fait partie des approches qui adaptent certains outils des systèmes classiques au contexte des systèmes mixtes. Il s'inspire de MVC dont il reprend la décomposition générale et l'adapte aux interfaces tangibles. Le système est alors décrit en tant qu'un ensemble de composants (M) qui représentent la logique du système, de représentations physiques (Rp) qui représentent les objets physiques et jouent le rôle d'un composant de contrôle de MVC, et de représentations numériques (Rd) qui représentent les informations non tangibles. Ces composants permettent de représenter certains aspects statiques et dynamiques des interfaces tangibles. Le modèle ne permet par contre pas d'exprimer les objets physiques composés ou les échanges d'information entre objets physiques.

Destiné également aux interfaces tangibles, "Tokens And Constraints" (TAC) [SLCGJ04] permet de représenter les contraintes dans l'environnement physique. Les contraintes limitent les comportements des objets physiques auxquels elles sont associées. L'aspect dynamique de l'application est également représenté par des entités numériques qui peuvent être associées aux objets physiques.

Trevisan ([TNM08] et [TVM02]) propose un espace de modélisation centré sur les applications de chirurgie augmentée. Une particularité de ce domaine est l'importance du travail d'acquisition des données et des informations relatives à l'interaction effectué en amont de la situation d'interaction (Scanner, IRM, modèles anatomiques, etc.). Une autre particularité de la chirurgie augmentée est le besoin très important en termes de précision et de continuité de l'interaction, la continuité étant définie comme la capacité du système à promouvoir un passage naturel et fluide entre les entités physiques et numériques. De ce fait, l'environnement de modélisation offre un espace important à la représentation des contraintes spatio-temporelles. Les différentes facettes des systèmes sont représentées par plusieurs modèles. Un modèle général offre

une vue globale de l'interaction, complétée par un modèle d'interaction qui permet d'affiner cette description. Un troisième modèle permet de représenter en détail les objets manipulables par le système ou par l'utilisateur alors qu'un modèle de tâches exprime l'enchaînement de celles-ci au long de l'interaction.

L'utilisation de plusieurs modèles est une approche également adoptée par SSIML/AR [Vit06] inspiré de UML. Un modèle de tâches permet de représenter la succession des tâches et sous-tâches et de décrire les données sur lesquelles ces tâches opèrent. Un modèle de scène décrit les éléments physiques et numériques de l'environnement mixte, ainsi que des éléments "hybrides" où les composantes physique et numérique sont intimement liées. Un modèle d'interrelations permet également de définir les relations qui existent entre l'application et le modèle de scène, et notamment les relations de localisation et d'alignement.

D'autres démarches adoptent une approche de modélisation centrée sur l'utilisateur du système mixte, basée sur des modèles entièrement pensés pour ce type de systèmes. Ceci est par exemple le cas de IRVO [CD04] et de ASUR [DMBB07].

Le modèle ASUR décrit l'interaction de l'utilisateur dans le système mixte. Il aide dans l'analyse de l'intégration des entités physiques et numériques et dans l'exploration de l'espace de solutions. ASUR permet d'identifier les objets participants et les relations entre ces objets, ainsi que d'analyser la situation d'interaction d'un point de vue centré sur l'utilisateur. Cet utilisateur est identifié en tant que composant (U) du modèle, et est associé à des entités informatiques (S), réelles (R) et à des adaptateurs (A) qui traduisent les informations entre l'environnement physique et numérique (Ain) ou inversement (Aout). ASUR permet également de représenter les liens entre les éléments du modèle, tels les échanges d'information dans les environnements physique ou numérique, le couplage comportemental ou de représentation, les proximités entre les éléments physiques et les événements ponctuels tels le déclenchement d'une certaine activité. L'expression des caractéristiques des éléments, de leurs modes de modification (toucher, signal électrique, etc.), ou des moyens par lesquels ces modifications peuvent être détectées (camera, micro, etc.) est également supportée. Les relations peuvent être caractérisées par le type de médium utilisé pour le transfert de l'information (son, vibrations, audio, etc.) ou le moyen de coder les informations transmises. ASUR représente l'interaction par rapport à une seule tâche principale à la fois. Des travaux récents visent à l'intégrer dans un modèle de tâches [CDR07]

afin de décrire la succession des situations d'interaction. Une autre piste associe ASUR à un modèle d'implémentation ASUR-IL [DBG+06]. ASUR-IL identifie les composants et les relations logicielles nécessaires à l'implémentation de la tâche décrite par le diagramme ASUR. Il permet une caractérisation plus fine des adaptateurs et des entités numériques associées. L'échange de données est représenté par un flux d'informations entre les entités concernées, ce qui ne permet pas d'identifier les mécanismes précis de l'échange (synchrone, asynchrone, etc.).

IRVO étend la modélisation d'ASUR aux systèmes collaboratifs multi-utilisateurs. Il permet une plus fine description des contraintes spatiales, mais sa caractérisation de l'utilisateur demeure moins fine que celle offerte par ASUR.

La description des dispositifs et des composants logiciels de l'interaction est supportée par APRIL (Augmented Presentation and Interaction Language) [LS05]. APRIL permet de modéliser finement un certain nombre de périphériques d'entrée et de sortie et de techniques d'interaction, ainsi que la géométrie de la présentation mélangeant entités physiques et numériques. Cette modélisation permet à des logiciels de calculer les occlusions et les intersections. L'enchaînement des actions suit un modèle de défilement temporel, peu adapté pour la description des interactions non linéaires.

Les types d'informations représentées par les différents modèles en définissent le niveau d'abstraction. Ce niveau permet principalement de déduire la facilité de manipulation du modèle par les développeurs, ainsi que son adéquation à l'exploration intuitive de l'espace de solutions possibles.

2.2.2.2 Niveau d'abstraction

Le niveau d'abstraction d'un modèle est le niveau de proximité des entités et concepts qu'il exprime avec une description équivalente en langage naturel. APRIL par exemple permet d'identifier les types des données échangées (entier, réel, couleur, rotation, etc.). Il permet également de décomposer ses composants en sous-composants et fournit un ensemble de techniques d'interaction telles la localisation, la manipulation de boutons, la sélection par le toucher, etc. En ce sens, il s'apparente plus à un modèle impératif associé à une bibliothèque de composants qu'à un modèle de description haut niveau des systèmes. ASUR-IL est également destiné à représenter les éléments d'interaction à un bas niveau. Les composants sont ainsi divisés en des

sous-composants inspirés des éléments du modèle MVC. ASUR-IL permet de guider les choix de conception vers un ensemble ou une classe de périphériques. Reprenant également le modèle MVC, MCRpd permet de composer les applications sur ce modèle d'architecture. La caractérisation d'entités supplémentaires ne fait qu'adapter MVC au contexte des systèmes mixtes, mais n'offre pas un niveau d'abstraction plus élevé.

TAC propose une abstraction des détails techniques de l'implémentation logicielle, mais la multiplication des entités et des contraintes associées permet difficilement une analyse haut niveau de la situation d'interaction dans sa globalité. En revanche SSIML/AR sépare les différents aspects dans plusieurs modèles, ce qui permet à la fois des analyses à haut niveau dans les modèles de tâches ou d'interaction, ainsi qu'une modélisation fine des aspects du système dans les modèles de scène ou les diagrammes de classes de UML. Cependant, le lien entre les différents modèles et diagrammes nécessite une attention particulière et bénéficie de peu d'outils d'assistance. Ceci est également le cas du modèle proposé par Trevisan qui requiert une manipulation de plusieurs modèles en parallèle.

IRVO et ASUR se caractérisent tous les deux par une description haut niveau de l'interaction centrée sur l'utilisateur. Les caractéristiques des modèles et des composants sont facilement compréhensibles par les développeurs, qui bénéficient d'une vision haut niveau de l'interaction. Une vision de l'interaction dans sa globalité est également supportée dans ASUR par l'intégration des diagrammes au sein d'un arbre de tâches. Cette intégration permet un couplage direct entre les deux modèles, ce qui évite de les manipuler séparément.

L'intérêt principal de caractériser le niveau d'abstraction d'un modèle est d'identifier le niveau de complexité de sa manipulation par les développeurs. Cette caractérisation permet également de déduire la difficulté d'opérationnalisation d'un tel modèle.

2.2.2.3 Opérationnalisation

Cette dimension concerne la disponibilité des outils et processus nécessaires à la génération d'une implémentation du système à partir de son expression dans le modèle utilisé. Lorsque de tels outils n'existent pas, on ne peut qu'estimer une telle difficulté en fonction du niveau d'abstraction ou de la proximité du modèle d'autres paradigmes connus.En général, les modèles d'implémentation ou d'architecture sont les plus faciles à opérationnaliser, du fait de leur proxi-

mité avec les concepts bas niveau logiciels et matériels, tels les architectures, les périphériques ou les techniques d'interaction existantes.

Malgré l'apport du modèle proposé par Trevisan en termes de contraintes spatio-temporelles, son opérationnalisation demeure complexe et il n'existe à ce stade aucune plateforme logicielle pour son opérationnalisation. Cependant, l'approche adoptée qui mélange plusieurs diagrammes s'apparente à celle supportée par UML, ce qui laisse présager la possibilité d'adapter des outils de ce dernier au modèle des interfaces tangibles.

MCRpd adopte un modèle modifié de l'architecture MVC mais ne bénéficie pas non plus de plateforme d'opérationnalisation. En plus, le modèle MVC est rarement implémenté rigoureusement par les développeurs, et les frontières entre les éléments de MVC demeurent souvent poreuses [KL04], ce qui complique toute transformation directe entre MCRpd et une plateforme existante.

Certains modèles tels TAC, SSIML/AR ou APRIL développent des outils spécifiques pour opérationnaliser l'ensemble des modèles et notations utilisés. TAC bénéficie d'un prototype de boîte à outils offrant des fonctionnalités de base, telles la modélisation 3D ou la transformation d'une partie du diagramme TAC en un code Java3D. SSIML/AR a quant à lui bénéficié d'une attention particulière à sa facette implémentation. Après la conception initiale du système dans les multiples diagrammes de SSIML/AR, l'implémentation est répartie sur deux intervenants : les programmeurs et les développeurs de contenu 3D. La conception initiale est alors exportée en XMI [XMI05]. Les modèles adaptés de UML permettent alors de générer le code non fonctionnel, et le diagramme de tâche de générer une partie de la logique du système. Le modèle de scène est transformé en un format 3D (par exemple VRML). Les développeurs de contenu peuvent alors modifier et améliorer la scène chargée automatiquement. Les programmeurs complètent le code de l'application par les fonctionnalités spécifiques. Le passage de la partie conception à la partie implémentation est assuré par un cadre de développement offrant un certain nombre de services :

- un service de contrôle assure la gestion des entrées des périphériques (dispositifs divers, flux vidéo, etc.), et se charge de transmettre les données aux autres services.
- un service de localisation récupère le flux vidéo fourni par le système de contrôle, et selon la liste des objets à localiser, assure la détection de ces objets et de leur positions dans la

scène.

- un service de présentation utilise ces données pour assurer le rendu des éléments virtuels de l'interface.
- un service de gestion de tâches informe le système de présentation de l'état d'avancement de l'interaction, afin d'assurer un affichage compatible avec la tâche courante de l'utilisateur.

Chacun des services fournit des interfaces bien définies permettant une meilleure extensibilité de la plateforme. Le code généré hérite des classes déclarées par le cadre de développement, minimisant ainsi la quantité de codage manuel à effectuer.

APRIL profite de son lien étroit avec UML pour articuler le développement autour de différents acteurs : les experts du domaine, les rédacteurs du scénario, les créateurs de contenu, les développeurs de composants, et les intégrateurs de système responsables de l'assemblage du système entier. Ceci permet de définir un pipeline de développement en trois étapes :

- la première étape consiste à identifier le sujet de la présentation et à collecter le matériel de base (images, matériel 3D, etc.). Cette étape conduit à la définition des diagrammes UML/APRIL.
- ces diagrammes servent alors de base au développement du contenu et des comportements à intégrer. Si des composants compatibles existent, ils peuvent être réutilisés.
- le tout est enfin intégré afin de réaliser le système mixte. La description du matériel disponible conduit à l'implémentation du système.

Cette dernière phase consiste à traduire les concepts exprimés dans l'ensemble des diagrammes en un fichier XML qui peut être importé dans l'environnement d'exécution de Studierstube. Le format XML permet de définir les traducteurs du modèle sur la base de XSLT [W3C99]. APRIL évoque également un mode d'émulation de matériel faisant intervenir des fichiers XML modifiés. A notre connaissance, ce mode d'émulation n'a pas été implémenté.

Les modèles d'interaction descriptifs sont quant à eux les plus difficiles à opérationnaliser du fait de leur haut niveau d'abstraction. Ceci est notamment le cas de ASUR et IRVO où le passage d'une description haut niveau en langage proche du langage naturel à une implémentation matérielle et logicielle demeure difficile. Des mécanismes de transformations assistées sont alors mis en oeuvre par ces modèles pour traduire les modèles abstraits en des implémentations logicielles. Ceci est notamment le cas de ASUR qui permet de compléter sa description haut

niveau par celle de ASUR-IL des entités (A) et (S).

On remarque que, en dehors des particularités de l'une ou l'autre des approches à caractère descendant, la difficulté principale demeure le passage du modèle à une implémentation effective du système. Ceci est à l'opposé des approches logicielles ascendantes qui ont du mal à représenter les systèmes dans leur globalité. Il existe néanmoins des difficultés partagées par l'ensemble des approches ascendantes et descendantes. En effet, et indépendamment de la démarche de développement adoptée, l'utilisateur final peine en général à valider des spécifications fonctionnelles ou ergonomiques de son futur système sur la base de constructions qui ne sont pas exprimées dans un langage approprié [CKJ07]. Il demeure alors nécessaire de laisser aux utilisateurs le soin de tester l'interface et les fonctionnalités du système, afin d'obtenir un retour fiable. Ce retour est ensuite intégré dans la conception, menant à des cycles de développement itératifs centrés sur les utilisateurs (figure 2.10).

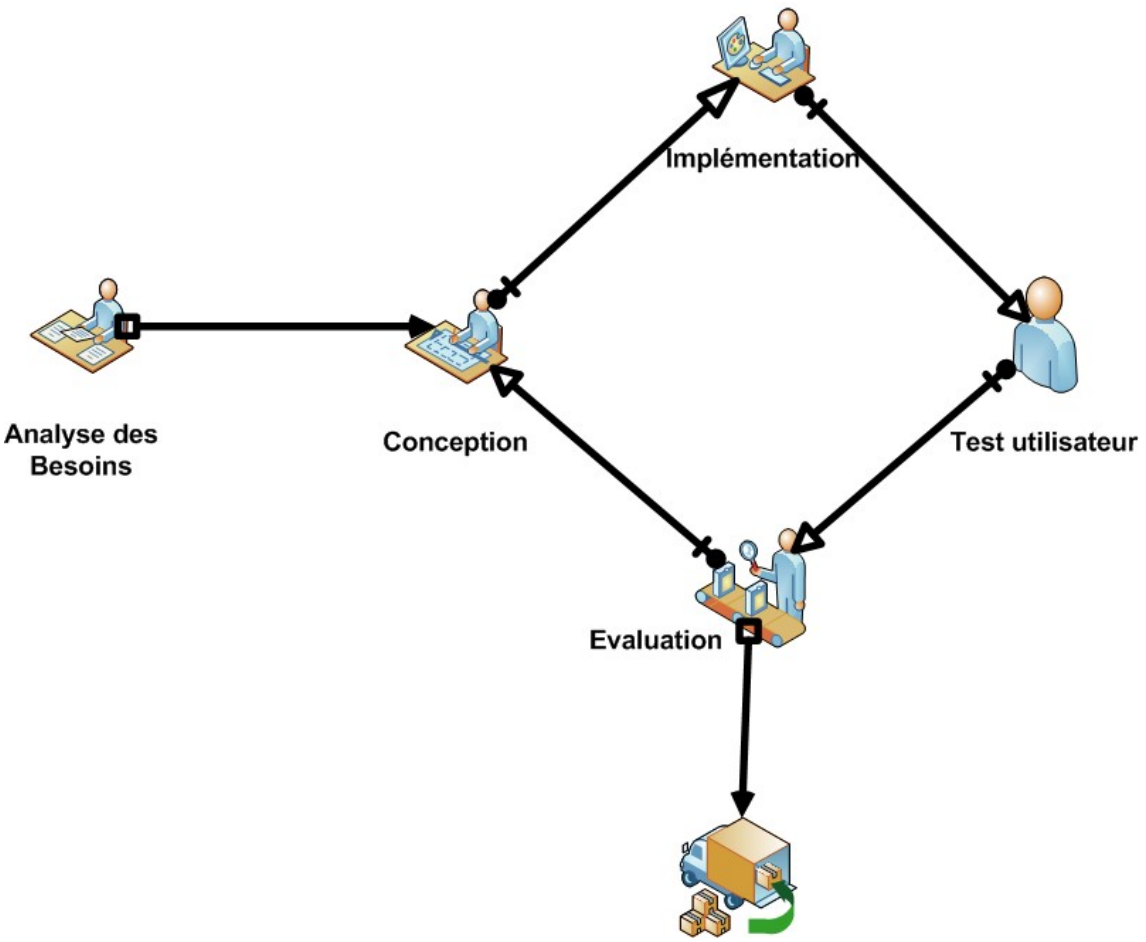

FIGURE 2.10 – Processus itératifs

Ces cycles posent un certain nombre de difficultés, bien connues dans les processus de développement itératifs des systèmes interactifs classiques. Dans le cas des systèmes classiques, le prototypage des systèmes ou de leurs interfaces est une solution possible.

2.3 Prototypage des systèmes interactifs

Développer un système selon un modèle itératif implique la création de différentes implémentations à des fins d'évaluation. Afin d'alléger ces itérations en termes de coûts et de temps de développement, les développeurs ont souvent recours au prototypage comme alternative à une implémentation complète des systèmes. Le prototypage est selon [Boa84] "une stratégie spécifique pour conduire l'identification des besoins qui sont extraits, présentés, et raffinés sur la base d'un modèle de fonctionnement de l'application". Un prototype est une représentation concrète de tout ou partie d'un système ; c'est un artefact tangible par opposition à une représentation abstraite [BLM03].

Plus concrètement, un prototype est une réalisation simplifiée du système qui doit permettre aux utilisateurs de fournir un retour d'information utile aux développeurs, pour faire évoluer la conception du système vers sa version finale. Il s'agit d'aboutir le plus rapidement à des spécifications précises, conformes aux besoins exprimés ou perçus comme tel par les utilisateurs [CKJ07]. Minimiser le coût total de la phase de prototypage tout en évitant de coder explicitement le prototype, demeure un enjeu majeur des méthodologies de développement itératives [SN03]. Une caractérisation du prototypage des systèmes interactifs classiques, ainsi que des tentatives d'extension de ce prototypage aux systèmes interactifs mixtes, se révèle alors d'une importance majeure pour mieux situer nos travaux.

2.3.1 Caractérisation du prototypage

Le prototypage ne se limite pas à l'interface des systèmes, et peut concerner l'ensemble des aspects logiciels ou, surtout dans le cas des systèmes mixtes, matériels. Néanmoins, à cause de l'importance que revêt l'évaluation des critères d'ergonomie et d'utilisabilité, le prototypage des systèmes interactifs se focalise souvent sur leurs interfaces. Selon [SBK97], un cycle de prototypage rapide de l'interface d'un système interactif se caractérise par trois propriétés :

- l'ouverture : à tout moment du cycle, de nouvelles spécifications peuvent être rajoutées.
- l'itération : le nombre d'itérations nécessaires avant d'aboutir au résultat final est à priori inconnu.
- l'incomplétude : un prototype reste par nature incomplet, dans la mesure où il ne peut pas dégager en une étape toutes les spécifications.

Cette dernière propriété est difficile à caractériser et différents niveaux d'incomplétude mènent à des prototypes représentant plus ou moins de facettes des systèmes en question. Ceci est communément appelé la fidélité du prototype. Les typologies présentées dans [Jes08] identifient en plus une caractérisation du prototypage selon la stratégie adoptée ou selon le positionnement par rapport au processus de conception.

2.3.1.1 Fidélité des prototypes

[MCP+06] définit la fidélité d'un prototype comme la distance entre le prototype réalisé et le système complet, mesurée selon cinq dimensions :

- le niveau de la représentation visuelle : cette dimension exprime la proximité entre l'aspect visuel du prototype et celui de l'interface finale. La définition du niveau de fidélité de la représentation est extensible à d'autres interfaces prenant en charge des facettes sensorielles telles les retours haptique ou auditif.
- la largeur des fonctionnalités : cette dimension concerne la proportion des comportements et fonctionnalités implémentées dans le prototype, par rapport à ceux définis dans la conception du système complet.
- la profondeur des fonctionnalités : cette dimension exprime la finesse de l'implémentation des fonctionnalités du prototypes.
- la richesse de l'interactivité : le prototype peut être statique ou dynamique, et supporter une partie ou la totalité des échanges suggérés par son interface. Alors que la largeur des fonctionnalités est essentielle à l'évaluation de l'utilité du système, la richesse de l'interaction est primordiale à l'évaluation de son utilisabilité.
- la richesse du modèle de données : cette dimension traduit la quantité de données qui peuvent être introduites ou manipulées dans le prototype, que ce soit par des entrées de l'utilisateur, ou via des sources de données informatiques ou électroniques.

2.3.1.2 Stratégies de prototypage

[Nie93] axe sa structuration du prototypage sur une division de l'architecture des systèmes interactifs en trois couches : l'interface, la couche d'abstraction de contrôle, et le noyau fonctionnel. Ceci conduit à une autre classification des approches de prototypage : verticales et horizontales. Le prototypage est dit horizontal s'il prend en charge un maximum des fonctionnalités de l'interface, indépendamment de la finesse de l'implémentation des couches inférieures. En contrepartie, le prototypage vertical cherche à valider le système via une implémentation assez fine d'une sous-partie de ses fonctionnalités. [CV05] définissent le prototypage diagonal comme le mélange entre les deux approches orthogonales verticale et horizontale. Dans le cadre du prototypage diagonal, certaines parties du système sont finement prototypées alors que d'autres demeurent plus ou moins superficielles et limitées au niveau de l'interface. Toujours selon [CV05], " le prototypage diagonal offre un support à des approches multi-fidélité, adaptées aux démarches incrémentales ou expansives" .

2.3.1.3 Positionnement par rapport à la conception

[Jes08] établit un positionnement du prototypage par rapport au processus de conception :

- d'un côté, le prototypage peut servir de plateforme d'exploration de l'espace de solution, et ainsi produire des options nouvelles. Ce mode de prototypage est dit expansif, puisqu'il aide à étendre l'espace de solutions initial.
- de l'autre, le prototypage peut servir, à travers les différentes phases de tests et de validations, à réduire le nombre de solutions possibles, et à faire converger la conception vers un choix final.

Dans le cas des systèmes mixtes, la gestion des aspects logiciels et matériels du prototypage demeure un vrai défi. En effet, il est difficile d'isoler les fonctionnalités et les interfaces, surtout quand les interfaces mélangent des entités numériques et physiques. L'extension du prototypage aux systèmes interactifs mixtes explore alors différentes pistes, dont la simulation.

2.3.2 Prototypage et systèmes interactifs mixtes

Indépendamment de la démarche adoptée (ascendante ou descendante), ou de la plateforme logicielle utilisée, les difficultés associées aux systèmes mixtes ne se limitent pas à la dimension conceptuelle et logicielle. En effet, les coûts et complexités associés aux acquisitions matérielles contribuent fortement à la longueur et au coût total des processus de développement des SIM. Le manque de standards complets et suffisants, tant au niveau des dispositifs qu'au niveau des logiciels et techniques d'interaction, conduit souvent à des itérations multiples et peu convergentes, associées à différents prototypes et configurations matérielles et logicielles. Certaines approches explorent l'utilisation de la simulation comme un moyen de simplifier le prototypage et la validation des systèmes et de leurs composantes.

La simulation consiste à utiliser des outils plus simples pour évaluer les facettes interactives et fonctionnelles des systèmes, et apparaît comme une solution intéressante. Ceci rejoint quelque peu l'approche par émulation de la plateforme matérielle proposée par APRIL, et restée sans réalisation concrète. Le terme "simulation" est également utilisé par DWARF dans le cadre de système SHEEP, pour signifier l'imitation de la logique applicative pour le contrôle des moutons virtuels. Cette "simulation applicative" peut être intéressante pour tester les protocoles et les algorithmes applicatifs d'un système. La simulation de la fonctionnalité d'un composant, que nous appelons "simulation fonctionnelle", est également exploitée par DART qui permet de mixer différents niveaux de fidélité d'implémentation de ses composants.

Afin de pouvoir évaluer les caractéristiques des systèmes sur la base des prototypes simulés, de tels prototypes doivent au moins fournir un niveau de fidélité suffisant, et selon les différents axes identifiés précédemment. La réalité virtuelle apparaît alors comme une technologie de simulation intéressante.

2.3.2.1 Intérêt de la réalité virtuelle pour le prototypage

La réalité virtuelle est souvent utilisée pour le prototypage des structures et infrastructures complexes. Son usage participe alors à la coordination du travail des acteurs concernés par les projets en cours de réalisation, favorisant ainsi l'innovation [Von88]. Dans le domaine de la construction, elle permet de réduire les risques et d'améliorer les processus de développement,

en fournissant une simulation dynamique des opérations. La RV permet d'explorer les différentes alternatives, et de minimiser le temps de conception via une assistance logicielle. Elle sert aussi dans d'autres domaines dans les phases d'apprentissage (simulateurs de vol, de conduite de trains, etc.) et pour tester l'utilisation des aéroports, les parcours, visualiser les modèles de bruit, le mouvement des utilisateurs, le trafic routier, la logistique (supermarchés et flux des marchandises, positionnement des produits et visibilité, éclairage etc.). En offrant un support de visualisation des choix de conception très tôt dans le cycle de développement, le coût des modifications et mises à jour est minimisé, ainsi que les risques opérationnels.

Le réalisme requis lors des phases de simulation dépend largement des buts recherchés. Alors que le positionnement des produits dans les rayons d'un supermarché nécessite une qualité visuelle moyenne et pas de simulation physiquement réaliste, les tests sur l'écoulement de l'air et le fonctionnement des moteurs de voitures de Formule 1, par exemple, mettent plus de contraintes sur le modèle physique. Il en ressort globalement trois contextes d'utilisation de la RV pour le prototypage [Les03] :

- au sein des projets de développement, afin de visualiser et partager les résultats des étapes de conception. Ces méthodes utilisent souvent des outils de CAO et des plans de travail 2D pour une conception détaillée, et privilégient la fonctionnalité des prototypes à leur qualité visuelle ou à leur interactivité.
- en tant que démonstrateurs reflétant l'expertise et le savoir-faire des développeurs ou de leurs entreprises, ou servant de support à la communication avec l'utilisateur / client final. La qualité visuelle des prototypes est alors privilégiée au détriment de la fidélité fonctionnelle. L'interactivité est souvent limitée à des actions de navigation simples.
- en tant que plateformes d'entraînement tels les simulateurs de vol ou de conduite. Ces systèmes privilégient à la fois les dimensions fonctionnelles, interactives et de représentation. Néanmoins, leur développement est très compliqué et coûteux, ce qui exclut leur utilisation dans un cycle de prototypage rapide.

2.3.2.2 Réalité virtuelle et systèmes interactifs mixtes

Un prototypage des SIM basé sur la réalité virtuelle offre trois atouts majeurs :

- une représentation tridimensionnelle de l'environnement : l'interaction n'étant plus limi-

tée, comme c'est le cas des systèmes interactifs classiques à un espace 2D ou 2D $\frac{1}{2}$, les prototypes doivent pouvoir représenter l'aspect tridimensionnel de l'espace d'interaction afin d'en évaluer les propriétés et caractéristiques. Tester l'interaction dans un espace tridimensionnel constitue un avantage important comparé aux approches papier-crayon. La réalité virtuelle permet également d'étendre les aspects testés au delà de l'unique facette visuelle.

– une interactivité accrue, supportant des espaces d'interaction multi-utilisateurs, multimodaux ou collaboratifs : la RV permet de placer l'utilisateur dans un environnement simulé interactif, et de le laisser évoluer " librement " selon différents scénarios qui peuvent émerger. Les entités de l'environnement peuvent répondre aux actions de l'utilisateur, simulant ainsi l'interface du système mixte. L'environnement virtuel permet également de représenter et reproduire les actions explicites ou les fonctionnalités implicites.

– des performances en temps réel, permettant d'intégrer dans un environnement assez représentatif de la situation réelle d'interaction, une largeur et une profondeur de fonctionnalités suffisantes. Le prototypage multi-fidélité ou diagonal demeure alors possible, offrant des possibilités d'évolution des prototypes vers les systèmes finaux.

Parmi les tentatives d'utiliser la RV pour le prototypage des systèmes mixtes, ASUR-SIM [VDG04] utilise Quake [IS96] pour la gestion du rendu, et UbiWise [BV03] pour la gestion de la simulation de dispositifs d'affichage en 2D. Cependant, l'utilisation des outils de réalité virtuelle existants pour le prototypage des systèmes mixtes pose un certain nombre de problèmes. En effet, la RV demeure une technologie difficile à maîtriser. Son intégration dans un cycle de développement de SIM risque alors d'accroître la complexité de ce dernier. Les plateformes de développement de la réalité virtuelle tels Bamboo [WZ98], Diverse [KSA+03] ou VRJuggler [BJH+01] permettent la création rapide d'environnements virtuels interactifs. De telles plateformes sont optimisées pour la création de solutions performantes, adaptées aux besoins des environnements purement virtuels. Le graphe de scène couramment utilisé pour structurer les objets virtuels ainsi que leurs comportements opère des traitements de filtrage et d'optimisation pour accélérer le rendu. Une telle structuration des modules des prototypes limite la liberté d'organisation de ces modules, ainsi que le niveau de fonctionnalités pouvant y être intégrées (parallélisme, comportements complexes, etc.) [Pon04].

L'état de l'art des approches et outils de développement des systèmes mixtes révèle certaines limitations des approches de développement existantes, des limitations que nous examinons dans le chapitre suivant. L'identification de ces faiblesses nous permet par la suite de définir une approche de développement plus adaptée aux contraintes imposées par les SIM.

3

Analyse de l'état de l'art

L'analyse du domaine des systèmes interactifs mixtes montre l'implication d'une grande variation d'expertises dans le processus de développement. Ceci est encore plus évident dans les domaines d'application qui s'éloignent des environnements purement informatiques, tels la médecine ou la maintenance industrielle. Afin d'accommoder ces variations d'expertises, [DMBB07] propose d'intégrer et structurer les différentes approches de conception dans des processus de co-conception qui comportent trois phases principales :

- une phase préliminaire d'analyses permet de dégager une proposition d'expérience interactive initiale.
- une seconde phase itérative optimise ensuite la proposition initiale, suivant un ensemble de dimensions telles que les techniques d'interaction, l'utilisabilité, la nature des expériences, leur mise en scène, les aspects socio-cognitifs qui y sont associés, la fiabilité des systèmes utilisés, etc.
- une troisième phase consiste à produire une expérience interactive pérennisée dans un contexte applicatif.

Chacune de ces phases fait appel à un ensemble de ressources théoriques existantes, et s'appuie fortement sur l'implémentation de solutions intermédiaires [BSD07]. De tels processus itératifs appliqués dans le contexte des systèmes interactifs mixtes soulèvent trois défis majeurs, que peinent à résoudre les plateformes actuelles de développement :

- la difficulté d'intégrer de façon méthodique les choix de conception issus des besoins des utilisateurs, dans les implémentations des systèmes ainsi que de maintenir la cohérence des

implémentations successives avec ces choix.

- l'accommodation des variations d'expertise impliquées dans le développement des systèmes mixtes.
- la longueur du temps de développement, due largement à la répétition des phases d'implémentation, et les coûts engendrés par l'intégration de dispositifs variés et de protocoles complexes.

3.1 Intégration des choix de conception dans des processus de développement performants

Cette intégration consiste, dans le cas des processus de co-conception itératifs, à générer des prototypes conformes aux choix de conception, et ce de façon méthodique. Cependant, l'état de l'art du domaine des SIM révèle le cloisonnement des différentes approches issues de l'IHM et celles issues de l'ingénierie logicielle :

- la première consiste à considérer ces systèmes comme un cas particulier de systèmes interactifs, et cherche à adapter et appliquer l'expertise issue de l'IHM et de l'ingénierie de ces systèmes aux systèmes mixtes. Ces approches augmentent les paradigmes et outils existants (MVC, SSIML, UML, etc.) ou définissent des modèles entièrement pensés pour prendre en compte les facettes supplémentaires des systèmes mixtes. L'IHM de l'interface est au centre de l'évaluation qui peut s'opérer de façon itérative centrée sur les utilisateurs. Une attention particulière est alors prêtée à la conception de l'interaction, ce qui aboutit le plus souvent à une démarche descendante comme celles décrites dans la section 2.2. Ces approches sont alors confrontées à la difficulté d'opérationnaliser les spécifications conceptuelles, du fait du peu de convergence de ces spécifications vers des services et modules existants. Ces approches conduisent souvent à des prototypes limités en fonctionnalités, explorant de nouvelles techniques et métaphores d'interaction ou de présentation de l'information (ASUR, IRVO, etc.).
- la deuxième approche considère les SIM comme un ensemble de fonctionnalités qu'il s'agit d'optimiser. Ces approches opèrent souvent à partir d'objectifs concrets et précis. La vision globale du système laisse la place à une optimisation par parties de ses modules

constituants. Les outils et techniques ainsi développés se retrouvent alors dans des bibliothèques ou des cadres de développement comme ceux décrits dans le chapitre 2, facilitant le développement rapide d'applications performantes. La problématique de ces approches ascendantes est la difficulté d'intégrer, de façon systématique, les besoins des utilisateurs et les choix de conception. Le problème n'est pas limité à la génération des premiers prototypes, mais couvre aussi la conduite d'un cycle itératif complet, en cohérence avec ces choix. En effet, même si les premiers prototypes respectent les choix conceptuels avec une conformité bien définie entre les modules fonctionnels et les spécifications conceptuelles, les étapes ultérieures du processus sont conduites selon un schéma ascendant, privilégiant les optimisations locales des sous-systèmes afin d'accélérer les itérations. Le maintien de la conformité des systèmes avec les choix conceptuels demeure alors entièrement à la charge des développeurs. Les systèmes développés sur la base de ces plateformes (DWARF, AMIRE, etc.) intègrent souvent mal les aspects liés à l'utilisateur, et demeurent peu adaptés à l'exploration systématique de nouveaux paradigmes d'interaction.

3.2 Accommodation des variations d'expertise

Le déploiement des SIM dans des contextes de plus en plus variés conduit à une hétérogénéité croissante des expertises mises en jeu dans le cadre du développement de ces systèmes. A cette hétérogénéité s'ajoute l'expertise requise pour la mise en place des technologies et techniques de la réalité mixte, telles la localisation, le calibrage, la gestion des interactions, le rendu, etc. Ces technologies nécessitent toujours des connaissances avancées dans des domaines précis qui s'éloignent du développement logiciel classique. Néanmoins, il s'avère presque toujours que les méthodes de développement avancées pour intégrer ces variations d'expertise, qu'elles soient ascendantes ou descendantes, sont limitées chacune à des catégories d'experts, et peu accessible aux autres acteurs du cycle de développement des SIM. Les plateformes qui tentent de lier les deux démarches (APRIL, SSYML/AR) se retrouvent encore plus restrictives pour l'un ou l'autre des acteurs du fait de la forte dépendance qui s'établit entre les modèles conceptuels et les plateformes d'implémentation.

3.3 Inadéquation des approches de prototypage classiques

Les approches de prototypage souvent utilisées pour réduire les coûts et la longueur des cycles de développement sont peu adaptées aux systèmes interactifs mixtes. Le prototypage basse fidélité consiste principalement en la création de maquettes statiques qui suggèrent l'aspect des interfaces des systèmes. Un tel prototypage ne fournit pas toujours une qualité d'interaction suffisante pour la prise en compte des aspects de mobilité ou d'intégration entre physique et numérique. En plus, l'espace d'interaction d'un système mixte étant essentiellement tridimensionnel, il nécessite des maquettes capables de reproduire un tel environnement. Les interactions implicites, telles le suivi du regard ou les gestes naturels sont tout aussi difficiles à reproduire en temps réel, et de façon non intrusive, sur des prototypes de basse fidélité.

Le prototypage haute fidélité consiste à développer et déployer des maquettes dynamiques, offrant un niveau d'interactivité élevé, ainsi qu'un rendu sensoriel (visuel, haptique, auditif, etc.) plausible. Les fonctionnalités implémentées doivent alors assurer le bon déroulement de l'interaction, ainsi que le test des aspects spécifiques aux SIM, tels le nombre et le placement des dispositifs. La génération de tels prototypes demeure souvent aussi complexe que l'implémentation des systèmes complets. Les configurations matérielles et logicielles du système sont également difficiles à évaluer séparément, ce qui nécessite des acquisitions et configurations multiples et coûteuses de matériel tout au long des phases itératives.

Les plateformes qui proposent des modes de prototypage rapide s'apparentent plutôt à des plateformes qui accélèrent le développement des systèmes via des services et architectures adaptées, plutôt qu'à des outils de génération rapide de prototypes légers et peu coûteux. Les approches de prototypage par simulation utilisant les plateformes de développement classiques de RV se heurtent souvent au manque de flexibilité de ces plateformes.

Le niveau d'expertise requis pour le développement de prototypes de qualité acceptable, ainsi que la difficulté d'étendre les outils de prototypage rapide de systèmes interactifs classiques aux SIM, amènent [Sán05] à suggérer que le prototypage rapide des systèmes mixtes reste une tâche quasi impossible .

Le tableau 3.1 établit un récapitulatif des différentes approches ainsi que de leurs limitations. On y distingue les deux catégories d'approches de développement et de prototypage de systèmes

interactifs mixtes : ascendantes et descendantes. Nous incluons aussi le prototypage par simulation sur la base de technologies de réalité virtuelle, même si ce mode de prototypage peut être associé à l'une ou l'autre des démarches. Les critères de comparaison sont la rapidité et le coût du processus de développement, l'adaptation aux variations d'expertise, ainsi que l'ancrage méthodique des prototypes et des implémentations dans les phases de conception. Le symbole $\times$ indique que le critère n'est pas pris en compte par l'approche. A l'inverse, le symbole $\surd$ indique que le critère s'applique à l'approche désignée.

	Ancrage dans la conception	Rapidité	Coût	Adaptation aux expertises
Ascendantes	$\times$Dépend des développeurs $\times$Difficile à maintenir lors de processus itératifs	$\surd$Réutilisabilité des modules $\surd$Réutilisabilité des architectures	$\times$Acquisitions matérielles $\times$Déploiement	$\surd$Certaines plateformes d'authoring $\times$Authoring difficle
Descendantes	$\surd$Par définition $\surd$Multiples modèles adaptés	$\times$Faible réutilisabilité $\times$Difficultés d'opérationnalisation	$\times$Divergence donc acquisitions $\times$Déploiement	$\surd$Modèles adaptés $\times$Besoin de méthodologies de co-conception
Simulation par RV classique	$\times$Dépend des développeurs	$\surd$Retardement de l'acquisition $\times$Non incrémental	$\surd$Déploiement assisté	$\surd$Maturité de la RV

FIGURE 3.1 – Récapitulatif

L'identification des besoins des processus de développement des SIM ainsi que l'étude des limitations des approches et méthodologies existantes, nous permettent alors de proposer une réponse que nous pensons mieux adaptée. La partie suivante détaille nos objectifs, et tente d'identifier les principales difficultés à surmonter par rapport à ces objectifs. Ces difficultés orientent les différents choix de conception et de développement de la plateforme SIMBA présentée par la suite.

Deuxième partie

SIMBA

1

Objectifs, difficultés et outils

Ce chapitre expose nos principaux objectifs, et identifie les principales difficultés associées à la réalisation de ces objectifs. Les outils utilisés pour la mise en oeuvre de notre approche sont également présentés, avec les principaux arguments justifiant du choix de ces outils.

1.1 Objectifs

Nous identifions le prototypage itératif des SIM comme la phase qui implique l'essentiel des investissements en termes de coûts et de temps de développement. Le domaine d'application auquel on s'intéresse particulièrement concerne les systèmes mixtes pour lesquels une évaluation interactive dans un environnement virtuel permet de situer les modifications et retouches à effectuer. Nos travaux proposent alors d'adapter les méthodologies de prototypage basées sur la réalité virtuelle afin de remédier aux limitations identifiées précédemment, et d'accélérer les processus de développement basés sur des méthodes de co-conception. Un prototypage par simulation permet de reproduire l'aspect dynamique du système, dans le but de fournir une expérience à l'utilisateur qui serait difficile ou coûteuse à reproduire sans le support de la réalité virtuelle. Le prototypage par simulation constitue de ce fait un prototypage de "moyenne fidélité". Il reprend aux approches de haute-fidélité l'expression d'aspects liés au positionnement et à la distribution des dispositifs dans l'espace, ainsi que certaines facettes de l'interaction homme-machine essentielles pour déterminer l'utilité et l'utilisabilité du système. Il permet également d'éviter l'acquisition de matériel coûteux très tôt dans le cycle de développement, et la manipulation de

configurations logicielles complexes.

Il s'agit néanmoins d'éviter de tomber dans des pièges classiques de la simulation par réalité virtuelle. Selon [Les03], "il s'agit de traiter l'usage de la réalité virtuelle non comme un remplacement de la réalité physique du monde, mais plutôt comme une réalité alternative. La réalité virtuelle est un prototype ayant une relation bien définie avec un environnement existant ou potentiel. La réalité virtuelle est alors utile pour négocier et valider les éléments conflictuels entre différentes versions ou conceptions de la réalité. Elle est alors un outil qui permet de faire le lien entre la représentation et la réalité sous-jacente. "

C'est justement ce lien que nous cherchons à exploiter en simulant l'interaction dans un environnement virtuel. La réalité virtuelle sert alors de médium pour la représentation de l'environnement et de l'interaction, ainsi que pour l'exploration des changements potentiels à introduire [Les03]. Le prototype n'est pas un but en soi mais un moyen pour améliorer le système final : " Il faut à tout prix éviter que le prototype ne devienne un sapin de Noël " [Sch99].

La mise en oeuvre de notre approche de prototypage par simulation, SIMBA, s'appuie sur une méthodologie de développement de SIM dirigée par les modèles. Le nom SIMBA "Simulation des Interactions Mixtes Basées ASUR" date d'une première version [AJD06] qui visait uniquement à simuler les interactions décrites dans des modèles ASUR. Contrairement à cette première tentative, l'approche présentée dans cette thèse ne se limite plus à l'unique simulation des SIM décrits dans les modèles ASUR. Notre objectif est alors de répondre aux limitations de l'état de l'art actuel, en construisant un support à :

- l'ancrage de l'implémentation des prototypes dans les choix de conception ;
- l'évolutivité des prototypes leur permettant de sortir de façon incrémentale de la simulation, tout en restant conformes aux choix de conception ;
- l'adaptation du prototypage aux différents domaines d'expertise des équipes de conception et de développement ;

1.1.1 Ancrage dans les choix de conception

Un tel ancrage du prototype lui impose et lui garantit une conformité avec les choix établis pendant les phases de conception. Dans un processus dirigé par les modèles, ces choix sont exprimés dans des modèles conceptuels qui peuvent être très abstraits. La vérification de l'interaction

de l'utilisateur avec le prototype permet de valider ces choix, et toute modification du modèle se reflète au niveau du prototype. Le transfert d'informations entre les modèles conceptuels et le prototype doit être systématique et pris en charge par des outils adaptés. Il s'agit par la suite d'éviter que les prototypes ainsi générés ne soient modifiés de façon à rompre la conformité avec les modèles conceptuels. Des outils d'assistance doivent alors accompagner le prototypage et empêcher toute modification qui nuirait à la conformité entre les prototypes et les modèles conceptuels.

1.1.2 Prototypage évolutif et incrémental

Un prototypage évolutif et incrémental permet à l'architecture du prototype ainsi qu'à certains de ses modules d'être récupérables pour l'implémentation ultime du système. Les méthodes de déploiement déjà testées sur les prototypes peuvent alors être réutilisées pour le système final. Cette caractéristique exploite la capacité de l'architecture à accommoder des modules de différents niveaux de fidélité, et à englober les différents modes de simulation existants :

- la simulation fonctionnelle, touchant à l'implémentation des composants individuels, et répliquant leurs interfaces et motifs d'entrées/sorties ;
- la simulation matérielle, consistant à émuler des dispositifs et périphériques d'interaction, ou même la plateforme matérielle de déploiement des modules numériques ;
- la simulation applicative qui consiste à répliquer la logique du fonctionnement d'un système dans sa globalité ;
- la simulation de l'environnement telle que proposée par SIMBA, et qui permet de compléter les apports des autres types de simulation par un retour sensoriel aux utilisateurs et aux concepteurs, portant sur la globalité de la situation interactive ;

L'hypothèse avancée pour simuler un module ou un aspect complexe d'un système mixte est qu'il existe des remplaçants pour l'entité concernée plus simples à utiliser, et dont l'utilisation n'affecterait pas ou très peu la perception du système par les utilisateurs. La simulation matérielle par exemple, dans le cadre d'un prototypage incrémental dans un environnement virtuel, consiste principalement à remplacer les dispositifs physiques complexes et coûteux par des composants physiques et numériques plus simples, mais offrant des capacités d'interaction suffisantes à l'utilisateur (périphériques virtuels, etc.). Le prototypage évolutif et incrémental

permet alors, à n'importe quel stade d'itération et une fois le choix du dispositif validé sur la base de la simulation, d'introduire le module physique et les modules logiciels associés à la place du module simulé, sans affecter les autres modules du prototype. La simulation fonctionnelle permet d'appliquer la même démarche à des composants numériques, en simplifiant la logique interne du composant sans affecter ses interfaces, ce qui permet une évolution incrémentale et fluide. Le prototypage s'intègre alors dans le processus de développement comme un passage intuitif entre la conception et l'implémentation, et ne reste plus une phase séparée et isolée du processus global.

1.1.3 Adaptation à différents domaines d'expertise

L'application des SIM dans des contextes variés (maintenance, divertissement, construction, tourisme, etc.) nécessite de fournir aux développeurs provenant de différents domaines, un espace de collaboration adapté. Accommoder le processus de développement à ces variations d'expertises passe premièrement par une adaptation des outils de conception pour l'exploration multidimensionnelle de l'espace de solutions. En dépit de la présence de tels outils, il demeure difficile de faire communiquer des concepteurs différents, uniquement sur la base de concepts et formalismes abstraits (modèles, notations, etc.). Pouvoir générer des prototypes rapidement et facilement à partir des multiples outils conceptuels permet aux différents acteurs de collaborer et de comparer leurs conceptions à moindre coût et effort.

1.2 Proposition

Une analyse des objectifs précédents et des différentes approches existantes, permet de dégager les grandes lignes de la méthodologie à adopter.

D'un côté, développer une plateforme de prototypage rapide par simulation implique forcément la disponibilité d'un support logiciel réutilisable en partie ou en intégralité, afin de minimiser le temps de développement des modules du prototype. En effet, un tel prototypage nécessite la disponibilité de services assurant la gestion des représentations des entités simulées, et la gestion de l'interaction des utilisateurs avec ces représentations. L'aspect incrémental du prototypage impose un certain nombre de contraintes à ces services, afin de ne pas gêner la sub-

stitution des modules simulés et non simulés. Un cadre de prototypage nous permet de fournir les services requis par les modules simulés, ainsi que les contraintes architecturales pour faciliter l'évolution des prototypes. D'un autre côté, le prototypage des systèmes interactifs s'avère encore plus utile dans le cadre d'une approche définissant clairement, et dès le départ, les objectifs à atteindre en termes d'utilité et d'utilisabilité. La pertinence d'une telle approche n'est plus à démontrer dans le cadre des systèmes interactifs en général, et à plus forte raison dans le cas des SIM où les recommandations générales efficaces demeurent limitées. Une telle approche se fonde sur des étapes de conception, et s'inscrit par conséquent dans une démarche à caractère descendant.

Allier les deux approches impose des choix et des compromis à tous les niveaux de la démarche. Il s'avère alors nécessaire d'identifier les difficultés et les compromis associés à nos objectifs, ainsi que l'étendue et le champ d'application du prototypage par simulation comme support à l'évaluation et la validation des choix de conception.

1.3 Identification des problèmes à résoudre par rapport aux objectifs

Définir une approche permettant à la fois de concevoir un système en prenant en compte les facteurs d'utilité et d'utilisabilité, et offrant un support au prototypage rapide qui favorise la ré-utilisabilité des entités et des architectures, implique un certain nombre de difficultés au niveau de la méthodologie de développement et de l'architecture de la plateforme. L'adoption d'un prototypage par simulation soulève également un certain nombre de défis supplémentaires par rapport à la démarche itérative et incrémentale que nous présentons dans les paragraphes suivants.

1.3.1 Intégration des choix de conception

Dans le cadre d'un processus de développement itératif et incrémental, l'ancrage des prototypes dans les phases de conception nécessite de compléter les choix conceptuels par des informations pertinentes à la génération des prototypes, sur la base de plateformes matérielles et logicielles différentes. Cependant, les modèles de conception les plus utiles à l'exploration des

solutions sont souvent abstraits, et fournissent peu ou pas de détails spécifiques aux plateformes d'implémentation. L'intégration des aspects liés au prototypage ou à l'implémentation au sein de ces modèles risque de les rendre illisibles, en plus de les lier à une plateforme matérielle ou logicielle spécifique. D'un autre côté, les modèles d'implémentation spécifiques aux plateformes demeurent trop complexes pour s'inscrire dans un cycle d'itérations rapides, et nécessitent des niveaux d'expertise élevés, empêchant leur manipulation par des non-experts. Il s'avère alors essentiel d'identifier les propriétés d'un espace intermédiaire entre les étapes de conception et d'implémentation qui permet de compléter la description du système, tout en respectant les choix architecturaux imposés par le prototypage par simulation.

1.3.2 Prototypage par simulation et cycle itératif

Afin de s'intégrer dans un cycle itératif, toute forme de prototypage doit répondre à des contraintes de coût et de complexité. Certes la simulation demeure en théorie moins onéreuse que le déploiement de systèmes dans l'environnement physique, du moment où ce coût n'est calculé que par rapport aux phases ponctuelles de génération des prototypes. Néanmoins, ce calcul ne prend en compte l'intégralité du processus de développement, où l'usage des technologies de la réalité virtuelle introduit des besoins nouveaux en termes d'expertise et de maîtrise des plateformes logicielles de la RV. L'accès aux services de simulation doit être à la portée de développeurs peu experts, en programmation 3D ou en animation par exemple.

1.3.3 Prototypage par simulation et processus incrémental

Le prototypage par simulation pose aussi un certain nombre de problèmes vis-à-vis de l'évolution incrémentale des prototypes. Les premiers prototypes étant simulés, leur évolution vers des systèmes dépourvus de tels modules implique une cohabitation à certains stades de cette évolution entre entités simulées et non simulées. Considérons l'exemple de la simulation d'un système de localisation d'opérateurs mobiles, utilisant un certain nombre de caméras réparties dans l'environnement. Une fois la solution validée par simulation, les développeurs doivent pouvoir tester le système en remplaçant certaines des caméras simulées par des cameras non simulées. Ceci aboutit à une configuration où caméras simulées et non simulées communiquent au sein d'un même système. L'évolution dans le sens inverse (remplacement d'une entité non simulée

par son homologue simulée) doit aussi permettre d'isoler les erreurs d'implémentation et/ou de déploiement (*Unit testing*). La plateforme doit permettre ce genre de cohabitation en assurant une prise en charge transparente de telles configurations “ hybrides ”.

1.3.4 Conformité du prototype avec les modèles conceptuels

La conformité des prototypes avec les modèles d'interaction est un des principes fondamentaux sur lesquels nous choisissons de fonder notre méthodologie de prototypage. Néanmoins, l'usage de la simulation nécessite la définition de services et schémas de communication supplémentaires pour assurer la représentation des entités simulées. Ces mécanismes et services n'étant pas définis dans les étapes de conception et n'ayant pas vocation à demeurer au-delà des étapes de prototypage, ils risquent de rendre difficile la garantie de la cohérence des prototypes avec les modèles conceptuels, en introduisant des chemins de communication et des interdépendances difficiles à identifier. Cependant, la réussite de l'intégration du prototypage rapide au sein d'une démarche cohérente dépend largement de la préservation de la conformité de ces prototypes avec les modèles conceptuels, unique lien entre les étapes à caractère descendant (conception de l'interaction) et celles fortement typées ascendantes (itérations rapides, implémentation, plateforme, etc.).

1.3.5 Intégration de services applicatifs

Notre approche ne consiste pas à proposer un cadre de développement offrant l'ensemble des services requis par les systèmes mixtes ou par leurs multiples domaines d'application. Cependant, le prototypage par simulation est d'autant plus utile qu'il peut s'appliquer à différentes catégories de systèmes mixtes. L'architecture de la plateforme doit intégrer et structurer les services dédiés au prototypage rapide par simulation, et permettre en parallèle l'intégration d'outils et de services applicatifs adaptés aux différents domaines et contextes d'utilisation des SIM. Les services propres au prototypage par simulation doivent être facilement supprimables dans les étapes finales de l'implémentation, afin de permettre aux prototypes d'évoluer vers les systèmes non-simulés.

1.3.6 Gestion de la distribution des utilisateurs

Les systèmes mixtes impliquent souvent des interactions multi-utilisateurs collaboratives. Ces interactions ne nécessitent pas toujours des plateformes logicielles distribuées, et disposent habituellement de plusieurs canaux de communication "intuitifs". Prenons l'exemple d'une tâche faisant intervenir différents opérateurs (figure 1.1). La communication de ces opérateurs n'utilise pas de support logiciel, et se fait via les canaux naturels de communication (voix, gestes, etc.). On considère aussi que l'augmentation de l'interaction des utilisateurs est individualisée, et que les éléments numériques de cette augmentation ne sont pas connectés entre eux (PDA personnels sans connexions réseaux par exemple). Simuler ce système nécessite de placer les utilisateurs dans un environnement virtuel interactif. Les échanges qui se faisaient de façon naturelle, nécessitent dans l'environnement virtuel des voies de communication logicielles. Un environnement non distribué limiterait en premier lieu le nombre d'utilisateurs dont l'interaction peut être simulée. Il est difficilement envisageable de tester une dizaine d'utilisateurs devant un même écran, même en utilisant des dispositifs coûteux tels les CAVE ou autres dispositifs de visualisation de grande taille [SPH+04]. L'absence de distribution limiterait aussi la distance maximale entre utilisateurs, même lorsque le nombre de ceux-ci demeure réduit. Des configurations logicielles plus simples tels les serveurs X distants [Jon94] permettent de dupliquer un même point de vue sur différents écrans distribués. Cependant, de telles configurations sont peu flexibles, et s'adaptent mal à un passage à l'échelle.

Ces configurations sont également peu adéquates au prototypage par simulation. En effet, les modes de distribution logicielle requis par les prototypes ne font pas partie des spécifications des systèmes mixtes en dehors des étapes de simulation. Le prototypage par simulation nécessiterait alors un effort de conception supplémentaire conséquent lors des phases d'intégration des choix conceptuels, mais aussi au long de l'évolution de ces prototypes vers les systèmes finaux.

1.3.7 Gestion de la distribution des dispositifs

Des contraintes liées à la distribution géographique du matériel et des dispositifs (entrées, localisation, etc.) peuvent aussi nécessiter une distribution des modules des prototypes. Considérons le cas d'utilisateurs suivis dans leurs environnements par une seule caméra, reliée à un

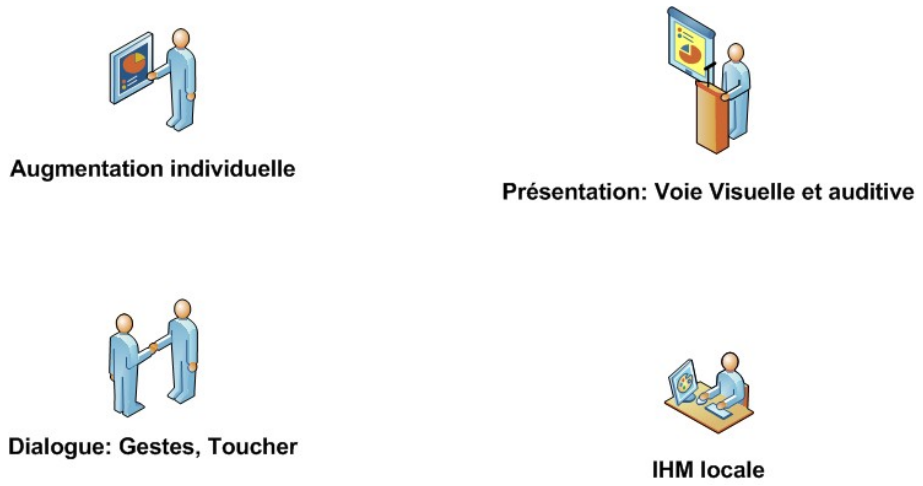

FIGURE 1.1 – Interactions individuelles

système informatique central via un réseau sans fil (figure 1.2). Cette architecture est à première vue distribuée, puisqu'elle nécessite des moyens d'échange d'information à distance. Néanmoins, lorsqu'il s'agit d'échanges simples (envoi de messages ou de valeurs numériques par exemple), l'interaction entre les entités concernées peut facilement être implémentée dans l'environnement mixte par des mécanismes simples (sockets, RPC, etc.). En revanche dans les prototypes, les modules qui simulent ces dispositifs peuvent être répartis sur un ou plusieurs hôtes indépendamment de leur répartition géographique réelle. La caméra doit également pouvoir assurer la gestion des interactions avec l'environnement et les utilisateurs, des interactions qui deviennent complexes dans un environnement simulé (les utilisateurs y sont probablement répartis). Dans le cadre d'un prototypage incrémental, certains des modules logiciels qui assurent la gestion des dispositifs simulés (module de filtrage associé à la caméra par exemple) peuvent être déployés directement sur les dispositifs réels. La communication entre ces modules et les parties des systèmes demeurant simulées nécessiteraient là encore un effort de (re) conception conséquent.

1.3.8 Gestion de la distribution des traitements et calculs

La simulation des interactions mixtes peut nécessiter, selon la fidélité des simulations et la complexité des systèmes, une puissance de calcul considérable. À la gestion des objets virtuels et des dispositifs d'interaction, s'ajoutent les besoins de l'application en terme de simulation fonctionnelle. Prenons l'exemple précédent avec une contrainte supplémentaire qui est de simuler les

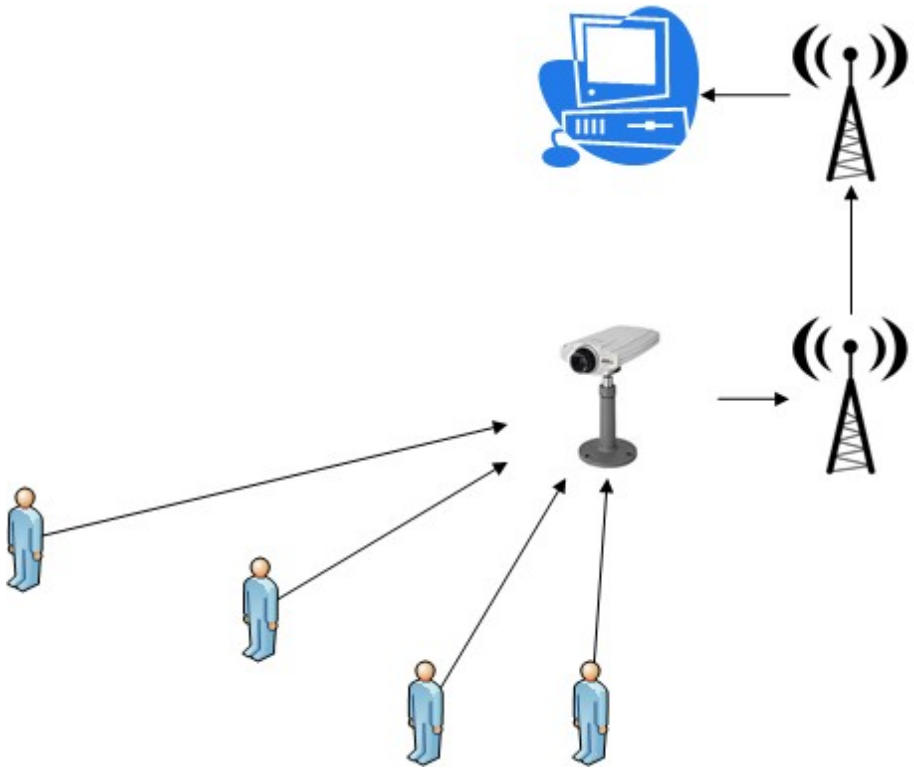

FIGURE 1.2 – Distribution des dispositifs

latences associées au réseau sans fil reliant la caméra au système informatique central. Ce type de simulation concerne un aspect fonctionnel qui demeure largement dépendant du domaine d'application du système interactif mixte. Les algorithmes en charge d'une telle simulation sont alors fournis par des experts du domaine, et intégrés à la caméra et au système informatique en tant que modules de transmission et de réception physiquement plausibles. L'intégration de ces modules engendrerait des surcharges de traitement conséquentes, qui viennent s'ajouter aux traitements des représentations virtuelles. Il s'agit alors de pouvoir garantir une interactivité en temps réel sur des machines ordinaires, puisque l'acquisition de matériel performant spécifiquement pour le prototypage ne ferait qu'augmenter les coûts et réduire l'attractivité de la démarche de prototypage par simulation.

Les chapitres 2 et 3 présentent les solutions proposées pour répondre à ces difficultés. Néanmoins, il demeure des limitations liées à l'utilisation même de la simulation de l'environnement d'interaction comme support au prototypage des systèmes interactifs mixtes, que nous tentons d'identifier dans le paragraphe suivant. L'identification de ces limitations permet alors, à dé-

faut de pouvoir à ce stade y apporter des réponses complètes, de mieux cerner l'étendue de l'application d'un prototypage par simulation.

1.4 Aspects traités et non traités par la simulation

Mettre un utilisateur assez tôt dans l'environnement d'un prototype simulé permet de tester et de valider certaines facettes de l'interaction. D'autres facettes demeurent néanmoins difficiles à analyser dans un contexte simulé et nécessitent une interaction située, principalement à cause des divergences importantes [DM96] (erreurs d'implémentations, erreurs intrinsèques, limitations de la technologie, etc.) entre environnements numériques et physiques (et par conséquent mixtes). Nous nous intéressons en particulier aux aspects suivants :

- l'intégrité procédurale de l'interaction dans le cadre de la tâche de l'utilisateur : il s'agit de valider l'adéquation des fonctionnalités et de la logique de l'application avec les besoins des utilisateurs par rapport aux tâches à réaliser. On peut par exemple envisager de vérifier le bon déroulement d'une procédure de maintenance assistée, ou la réponse d'un système à différentes situations ou actions des utilisateurs. Cette intégrité peut être testée dans un environnement simulé, du moment où les fonctionnalités concernées sont logicielles, ou en la présence de modules de simulation logiciels de bonne fidélité (simulation d'un réseau sans fil par exemple)
- la continuité physique-numérique : cette continuité se manifeste tant au niveau esthétique qu'au niveau cognitif. Il s'agit de vérifier si, d'un côté, les entités physiques et numériques sont bien fusionnées dans l'environnement, et d'un autre si le basculement entre une interaction avec des entités physiques et d'autres numériques se fait de façon fluide et continue. Le facteur d'échelle est intimement lié à l'évaluation de cette continuité. En effet, les environnements virtuels de simulation, même modélisés à l'échelle 1 : 1, sont souvent affichés à une échelle différente. L'affichage lui même sera généralement en 2D ou 2D1/2, ce qui risque aussi d'amplifier les erreurs d'évaluation. Cet aspect de l'interaction est difficilement validable dans un environnement simulé. La simulation peut néanmoins s'en approcher via la reproduction des interactions naturelles (sélection, manipulation) couplées à un niveau élevé de réalisme de la représentation des entités physiques simulées.

- l'ergonomie de l'interaction : l'intégrité ergonomique est normalement évaluée conjointement par les concepteurs et par des ergonomes. L'interaction avec le prototype doit être observable, et doit refléter les propriétés d'utilisabilité propres au système. Le positionnement des interfaces et le type d'affichage ou les modalités d'interaction constituent un bon exemple des aspects qu'il est nécessaire d'évaluer. Il serait aussi intéressant de valider les contraintes "humaines" telles le poids des dispositifs ou la fatigue due à la manipulation de certains objets. Le prototype simulé permet de tester et de valider certains aspects de l'interface du système, dans les limites imposées par le réalisme de l'environnement virtuel (poids des dispositifs, continuité, etc.). Les mouvements des utilisateurs demeurant artificiels (distorsions, retour d'effort limité, mouvement de l'environnement par rapport au corps), les observations et retours des utilisateurs doivent être adaptés aux contexte des systèmes mixtes simulés.
- les contraintes temporelles : il s'agit de vérifier que les mécanismes de synchronisation et de coopération entre éléments du système sont bien implémentés, et le cas échéant, identifier les besoins en termes de planification et d'organisation. Cependant, la notion de temps est fondamentalement différente entre un environnement réel et sa simulation virtuelle. Alors que le temps réel définit la capacité de traitement et de gestion des interactions du système vis à vis de son utilisateur, la dimension cognitive et perceptuelle du temps " virtuel " est plus complexe et difficile à évaluer [Les03]. Une attention particulière doit aussi être apportée à ces différences fondamentales.
- les contraintes spatiales des éléments : un prototype doit permettre d'optimiser le nombre et le placement (relations topologiques et directionnelles) des dispositifs (suivi, affichage, etc.), minimisant ainsi la durée et le coût des implémentations. La validation des contraintes spatiales doit être adaptée lorsqu'elle est mesurée dans un espace virtuel, la difficulté principale étant la mauvaise appréciation des distances absolues dans ce genre d'environnement [HF93].
- l'intégration sociale du système : il est essentiel, surtout pour des systèmes déployés dans des lieux publics ou à usage collectif, de valider certains aspects liés à la conformité avec les lois mais aussi avec les traditions et usages courants. Le suivi des utilisateurs ou l'augmentation des informations par des données " privées " sont toutes des variables essentielles à

maîtriser dans les SIM. La simulation offre un milieu idéal pour la validation de tels choix dans des environnements protégés et isolés.

Cette évaluation des aspects couverts par le prototypage par simulation demeure certes informelle. Une évaluation méthodique et formelle ultérieure devrait permettre une identification plus précise de ces aspects.

Afin de concrétiser l'approche de prototypage et son intérêt dans un processus de développement, une implémentation de SIMBA a été élaborée. Cette implémentation utilise un certain nombre d'outils qui serviront tout au long des chapitres suivants à illustrer les concepts et formalismes avancés. Le prochain paragraphe présente ces outils ainsi que leur apport à l'implémentation de SIMBA.

1.5 Outils de développement

Trois outils principaux ont été utiles à l'implémentation de SIMBA : un modèle de conception de l'interaction, un environnement de modélisation, et une plateforme d'implémentation des services.

1.5.1 Le modèle de conception

Parmi les différents modèles et notations existants pour la description et la spécification des systèmes mixtes, nous avons choisi de conduire l'étape de conception initiale avec GUIDE-ME, l'éditeur graphique de la notation ASUR. Ce dernier est conçu spécifiquement pour les SIM et décrit l'utilisateur en situation d'interaction avec le système par rapport à une tâche particulière. Cette description est de haut niveau, et utilise une sémantique facile et accessible. En particulier, nous nous intéressons à la séparation entre les entités physiques, numériques et les adaptateurs, puisqu'elle nous permet d'illustrer les particularités de la simulation des SIM. L'exploration méthodique et multi-critères des propriétés des interactions et de différentes configurations des entités du système permet également de conduire des cycles itératifs structurés, à un niveau d'abstraction élevé et tout en observant les effets des modifications sur les prototypes.

GUIDE-ME permet d'éditer les multiples caractéristiques des entités de la notation, d'en vérifier les contraintes, ainsi que d'exporter la description dans un format XML ou XMI. XML

permet d'intégrer la notation dans une approche dirigée par les modèles et assistée par des outils de transformation standards.

Nous avons également profité du fait que le modèle ASUR est développé au sein de notre laboratoire afin de mieux intégrer les besoins des concepteurs des SIM dans la conception et le développement de SIMBA. La démarche de prototypage et d'implémentation s'inscrit d'ailleurs parfaitement dans le processus développé par les auteurs de ASUR [BSD07]

1.5.2 Environnement de modélisation

GME [LMB01] est un environnement de modélisation graphique générique et programmable. Il définit un certain nombre d'artefacts et de concepts communs aux approches de conception logicielle, et offre une plateforme pour définir les artefacts et concepts spécifiques à différents domaines d'application sous forme de modèles et méta-modèles. GME nous permet de représenter les entités appartenant à des domaines d'application hétérogènes avec leurs attributs et relations, ainsi que les entités spécifiques aux services de simulation et d'interaction. GME supporte aussi la définition de règles et de contraintes associées aux différentes entités, ainsi qu'un cadre de transformations attachées au méta-modèle. L'espace de modélisation multi-aspects permet de séparer différentes facettes des systèmes mixtes. GME offre une interface plus facile à utiliser et plus légère à installer que d'autres outils de modélisation existants tels GMF [Pro08] utilisé par ASUR-IL.

1.5.3 Modèle d'implémentation

L'implémentation des prototypes nécessite une plateforme logicielle flexible et capable de fournir un certain nombre de services. Une infrastructure offrant ces services nous évite de devoir les développer et les intégrer avec les services de prototypage. Pour profiter également de services existants développés dans différents langages et afin d'en assurer la portabilité, l'infrastructure doit permettre une adaptation facile de ces services ainsi que leur interopérabilité. Cette interopérabilité s'accompagne certes d'un compromis au niveau des performances mais demeure essentielle à cause de l'hétérogénéité des plateformes logicielles et matérielles qui peuvent être impliquées dans le développement de systèmes mixtes (PDA, téléphones mobiles, etc.).

Au vu de ces contraintes, une plateforme de composants généralistes répond au mieux aux

besoins du prototypage incrémental. Ce choix n'est pas commun dans le domaine des SIM, où il est d'usage de développer ou d'utiliser des plateformes optimisées pour les domaines particuliers [Pon04]. Néanmoins, la croissance continue de la puissance de calcul, des capacités de la mémoire ainsi que de la bande passante des moyens de communication, laisse présager une inversion de cette tendance, en s'orientant plus vers des plateformes de plus en plus génériques, et à des niveaux d'abstraction de plus en plus élevés. DWARF par exemple utilise CORBA pour implémenter sa couche de distribution et interopérabilité, même si le modèle de composants greffé par dessus demeure spécifique. D'autres approches [NIH01] utilisent également CORBA. Wcomp utilise les composants .NET et envisage de migrer vers une implémentation basée sur les Web Services.

Nous avons, de notre côté, choisi les composants CCM [OMG06] pour l'implémentation des services de SIMBA. CCM fournit les services essentiels au développement d'applications complexes et prend en charge la gestion des modes de communication asynchrones, du cycle de vie des composants, et de la transparence de la distribution. CCM permet un développement des composants dans plusieurs langages de programmation, facilitant ainsi l'intégration de modules déjà développés.

La résolution des problèmes identifiés dans ce chapitre nous permet de proposer une méthodologie et une plateforme de prototypage incrémental des systèmes mixtes basé sur la simulation de l'interaction. Alors que la simulation peut concerner différentes facettes sensorielles de l'interaction (visuelle, auditive, haptique, etc.), nous nous limitons dans cette thèse à la reproduction de l'aspect visuel des systèmes prototypés. Le dernier chapitre de cette thèse étudie la possibilité d'extension du prototypage pour couvrir les autres facettes des SIM. Cette extension devrait permettre au prototypage par simulation de servir à la validation d'autres contraintes et choix de conception des systèmes mixtes.

SIMBA comporte deux volets principaux : une architecture de services apporte les fonctionnalités de simulation et d'interaction, et un cadre de modélisation permet de diriger un processus de développement basé sur les modèles. Le prochain chapitre aborde le premier de ces volets.

2

Architecture de SIMBA

Notre but avec SIMBA est de proposer une architecture identifiant clairement les services et concepts requis pour une démarche de prototypage par simulation. En parallèle, cette architecture doit pouvoir accommoder des services aussi diversifiés que les domaines d'application des SIM. La description détaillée de l'architecture d'un système est un domaine de recherche à part entière, abordé notamment dans [Alb03]. Le but de ce chapitre est néanmoins de proposer une architecture d'un point de vue focalisé sur la simulation, ce qui nous mène à adopter une définition et une spécification plus souples des entités, artefacts et termes utilisés. En particulier, aucune différentiation formelle entre les termes élément, module, composant n'est établie, ces termes désignant tous un constituant d'un système encapsulant des données et/ou des traitements. Une extension ou une spécialisation de l'architecture décrite dans ce chapitre pour intégrer d'autres concepts et principes du domaine de l'architecture des systèmes demeure néanmoins envisageable et relativement intuitive.

Nous définissons l'architecture des prototypes comme un couplage flexible entre les services dédiés au prototypage par simulation que nous introduisons dans ce chapitre, et les services faisant partie de la conception du système mixte, et qui constituent la facette fonctionnelle de ce dernier. Les services dédiés au prototypage par simulation assurent la gestion des représentations (graphiques mais potentiellement auditives et haptiques), ainsi que l'interactivité des prototypes par une gestion des entrées et des interactions. Les services de gestion des entrées et des interactions ne sont pas réservés aux étapes de prototypage, et peuvent être intégrés dans la conception des systèmes mixtes.

La définition et la conception des services de simulation et d'interaction répondent à un certain nombre de contraintes et s'inscrivent par conséquent dans une approche globale commune que nous présentons dans le paragraphe suivant. La présentation ultérieure de chacun des services raffine ces contraintes pour préciser les détails des approches spécifiques à chaque service.

2.1 Approche globale

Il s'agit en premier lieu d'identifier la structuration des services en fonction des besoins du prototypage, mais aussi en fonction des contraintes logicielles et matérielles liées à l'implémentation et au déploiement de ces services. Notre choix s'oriente vers une architecture de services distribuée orientée composants.

2.1.1 Approche orientée composants

L'architecture de SIMBA est orientée composants. Les avantages d'une telle approche ont été présentés dans le paragraphe 2.1.2.1, et notamment la séparation que font les composants entre leurs interfaces et leurs implémentations, ainsi que la définition explicite des dépendances. Ces deux caractéristiques revêtent une importance supplémentaire par rapport à une démarche de prototypage rapide basé sur la simulation :

- les dépendances explicites permettent d'identifier les fonctionnalités offertes par les services de simulation et d'interaction. Elles favorisent l'interchangeabilité entre les différents modules, ce qui permet de basculer rapidement entre différentes configurations matérielles et logicielles. Les dépendances explicites permettent également de faciliter l'intégration des modules développés séparément, ainsi que d'automatiser ou d'assister cette intégration, si les descriptions des modules peuvent être interprétées par des logiciels ;
- la séparation entre interfaces et implémentations permet d'allier des composants de différents niveaux de fidélité fonctionnelle (profondeur de l'implémentation), sans affecter les autres parties d'un système. Les services de la plateforme peuvent aussi évoluer sans affecter les prototypes qui en dépendent ;

2.1.2 Distribution transparente

La transparence de la distribution revêt une grande importance dans le cadre du prototypage rapide par simulation des systèmes interactifs mixtes, puisqu'elle nous permet de résoudre les contraintes liées à la distribution des utilisateurs, des traitements et des dispositifs identifiées dans le paragraphe 1.3.

En premier lieu, une transparence de la distribution permet d'éviter l'ajout d'étapes de conception spécifiques aux prototypes, lorsque la distribution des modules est propre aux étapes de prototypage par simulation. Quand cette distribution fait partie de la conception du système, la transparence permet de gérer indifféremment cette distribution en plus de celle impliquée par la simulation.

La transparence de la distribution permet également de répartir facilement les modules des prototypes sur différents hôtes, afin de mieux prendre en compte les contraintes liées aux surcharges de traitement et de synchronisation. L'accès aux services offerts par ces modules est alors indépendant de leur localisation sur les différents hôtes participant à la simulation.

Les services de simulation et d'interaction sont donc définis en tant que composants (ou ensemble de composants) distribués. Les caractéristiques supplémentaires de ces services sont détaillés dans les paragraphes suivants.

2.2 Services de simulation

Les avantages offerts par une approche orientée composants ne répondent que partiellement aux objectifs et contraintes identifiés précédemment, et notamment ceux liés au prototypage incrémental. Une structuration appropriée des services de simulation joue alors un rôle primordial dans la flexibilité de la méthodologie de prototypage. Elle facilite en particulier l'ancrage des prototypes dans les phases de conception ainsi que l'évolution de ces prototypes vers des systèmes non simulés. Ceci nécessite une isolation entre les aspects liés à la gestion des représentations graphiques (virtuelles) des entités simulées, et les facettes fonctionnelles des systèmes. Cette isolation est implémentée par les deux services de simulation : le gestionnaire des représentations et le composant horloge.

2.2.1 Séparation entre les représentations et les aspects fonctionnels

L'architecture adopte une séparation claire entre la gestion des représentations des entités participant à la simulation, et les autres aspects du système. L'environnement tridimensionnel de simulation est ainsi modélisé séparément (dans un éditeur d'environnements 3D par exemple), puis sauvegardé dans un format de fichiers standard. La conception de l'environnement de simulation s'effectue donc séparément de celle des autres aspects du système. Les représentations des entités participant à la simulation sont référencées par des identifiants uniques, spécifiés dans un format neutre (chaînes de caractères par exemple).

Avant de plonger dans les détails de l'architecture des services de simulation, il est utile de définir les termes que nous utilisons dans les paragraphes et chapitres suivants. Un composant fonctionnel est un module dont les propriétés (rôle, interfaces, etc.) sont définies dans la conception du système, indépendamment des étapes de prototypage par simulation. L'ensemble des composants fonctionnels structurés selon les schémas dictés par la conception constitue donc l'aspect fonctionnel du système. Cet aspect n'intègre aucune notion liée au mode de prototypage par simulation. Un composant simulé est un composant fonctionnel qui requiert, dans le cadre du prototypage par simulation, d'acquérir et de manipuler une représentation graphique (virtuelle). Le lien entre les représentations et les composants fonctionnels s'établit uniquement via les identifiants qui leur sont attribués. A noter que le terme de composant simulé (caméra simulée, borne sans fil simulée, etc.) utilisé en dehors du contexte de l'explication et de l'illustration de l'architecture des services de simulation, peut faire référence aux autres types de simulation introduits dans le paragraphe 1.1.2.

Le mode de gestion des représentations ainsi utilisé par SIMBA est à l'opposé de celui adopté par les approches classiques de développement de RV centrées autour de la représentation tridimensionnelle des entités virtuelles, et où les comportements viennent ajouter une dynamicité à ces représentations [Pon04]. En effet, dans le cadre d'un prototypage par simulation, les étapes de conception initiales définissent les aspects fonctionnels des systèmes, et les représentations graphiques ne viennent compléter ces aspects que lors des phases de prototypage. SIMBA place alors les composants fonctionnels au centre du prototype, structurés selon les choix définis par les concepteurs. Parmi ces composants, ceux qui requièrent des représentations graphiques (donc les

composants simulés) peuvent alors acquérir et accéder à ces représentations via les identifiants qui leur sont fournis et indépendamment de l'implémentation des mécanismes de rendu. Ces composants sont responsables du maintien de l'état de leurs représentations qui n'ont aucune fonctionnalité intégrée et ne font que refléter les états des composants fonctionnels.

Le transfert d'information entre les composants fonctionnels simulés et leurs représentations est unidirectionnel. Afin d'illustrer l'utilité d'une telle règle, prenons l'exemple d'un système où deux caméras assurent la localisation d'objets. Durant les phases de prototypage par simulation, les caméras et les objets gèrent leurs représentations graphiques respectives. Le modèle de conception, montré dans la figure 2.1 tel que défini dans ASUR, stipule que les positions des objets suivis sont obtenues par les caméras via un transfert d'information direct entre ces objets et les caméras qui transmettent ensuite ces données au système. La disparition de cette liaison en cours d'exécution du système (les objets sortent du champ de vision de la caméra ou la reproduction d'une situation de défaillance technique par exemple) doit interrompre le suivi de ces objets par les caméras. En parallèle, les services de simulation doivent continuer de recevoir les positions des objets afin de mettre à jour les représentations. Permettre aux caméras d'accéder aux informations détenues par les services de simulation signifie que les informations relatives aux objets suivis seraient accessibles indépendamment des schémas et spécifications dictées par le modèle conceptuel. En interdisant ce type d'échange, les services de simulation ne peuvent être utilisés pour contourner les limitations décrites dans les étapes de conception et ainsi détruire la cohérence entre conception et prototypes.

En séparant les aspects fonctionnels des systèmes des aspects liés à la gestion des représentations, SIMBA permet de supprimer ou de modifier les services de simulation sans affecter les facettes fonctionnelles des systèmes. Le passage incrémental entre des prototypes simulés et les systèmes finaux se retrouve alors facilité. Cette séparation est implémentée par les deux services de simulation : le gestionnaire graphique et le composant horloge.

2.2.2 Gestionnaire graphique

La gestion des représentations graphiques est centralisée dans un service isolé : le gestionnaire graphique (GM pour *Graphical Manager*). La mise à jour des représentations s'effectue via une interface de simulation exposée par le GM. L'interface de simulation permet un mode de

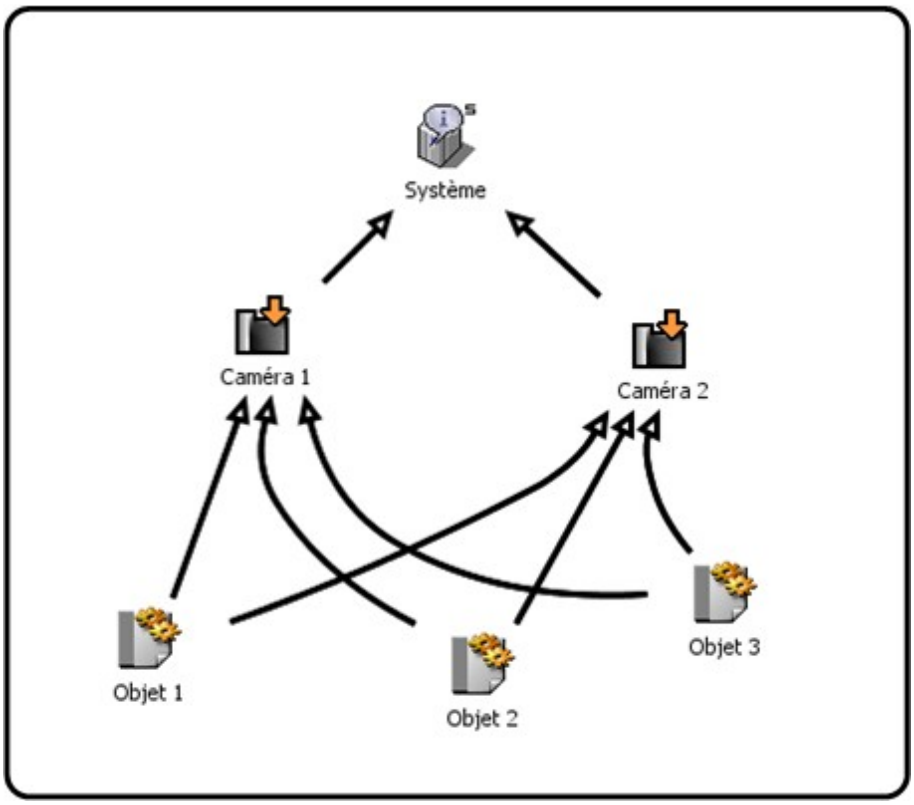

FIGURE 2.1 – Modèle ASUR d'un système de localisation par caméras

mise à jour asynchrone (événements) des représentations, sous condition que les composants qui détiennent les états de ces représentations exposent une interface compatible. Ces derniers n'ont pas besoin de connaître le fonctionnement interne du gestionnaire ni les mécanismes sous-jacents de la mise à jour qui s'opère alors via un simple envoi d'événements.

2.2.2.1 Types d'événements

L'interface de simulation accepte deux types d'événements : l'événement de mise à jour de position et l'événement de mise à jour du modèle graphique.

La mise à jour de position : lorsqu'un composant fonctionnel simulé souhaite modifier la position, l'orientation ou l'échelle de sa représentation, il instancie et émet un événement de mise à jour de position. Cet événement contient l'identifiant de l'objet graphique à manipuler qui permet au gestionnaire d'identifier la représentation concernée. L'événement contient également les nouvelles coordonnées absolues de la représentation (translation, rotation et échelle), calculées

par le composant avant l'émission de l'événement. Ces coordonnées sont alors appliquées à la représentation par le gestionnaire graphique.

Considérons l'exemple d'une simulation de conduite avec un seul véhicule piloté par un utilisateur (figure 2.2). Les fonctionnalités informatiques du système incluent la gestion des positions et des comportements des véhicules " intelligents " et la conversion des actions de l'utilisateur en des données de positionnement du véhicule piloté. Considérons ainsi une architecture du système où chaque véhicule est géré par un seul composant, qui calcule et transmet la position du véhicule aux autres parties concernées du système. Ce composant met à jour en parallèle et en fonction des paramètres internes de vitesse et de comportement de conduite, la position de sa représentation qu'il manipule grâce à l'identifiant qui lui est attribué. Cette mis à jour s'effectue alors via les événements de mise à jour de position, envoyés au gestionnaire graphique à travers l'interface compatible du composant reliée à celle du gestionnaire.

La modification du modèle : l'interface du gestionnaire supporte aussi la modification, pendant l'exécution de la simulation, de la représentation graphique. L'évènement de modification de modèle permet dans ces cas de remplacer une représentation existante par une autre, tout en lui attribuant le même identifiant. La modification de la représentation permet de combler certaines des limitations induites par le manque d'accès direct des composants du système à un niveau de détail fin de leurs représentations. En effet, tout élément dynamique (pouvant être manipulé) de l'environnement de simulation nécessite l'attribution d'un identifiant, unique lien entre les représentations détenues par le gestionnaire graphique et les composants qui détiennent les états de ces représentations. Afin de représenter les conséquences d'un accident dans l'exemple de la simulation de conduite (figure 2.3), il faut pouvoir manipuler les modèles 3D des véhicules à une échelle très fine, ce qui nécessite l'attribution d'un identifiant à chacune des sous-parties des véhicules affectées par l'accident. Ceci pose une contrainte assez complexe tant au niveau de la modélisation de l'environnement graphique qu'au niveau de la gestion des évènements de mise à jour par les composants du système et par le gestionnaire graphique. L'événement de changement de modèle permet de contourner cette limitation en remplaçant la représentation graphique affectée par une autre plus appropriée. Suite à l'accident calculé par les composants du système qui contrôlent les véhicules impliqués, ces composants émettent des évènements de mise

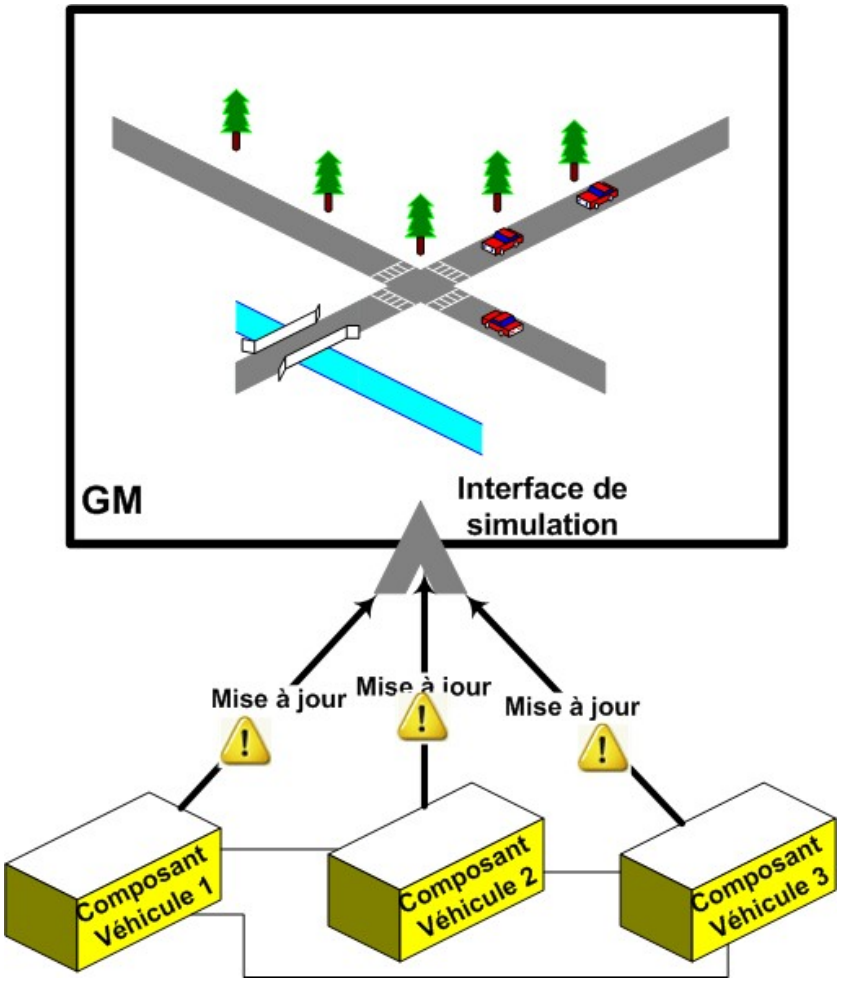

FIGURE 2.2 – Simulation de conduite

à jour des modèles de leurs représentations. Les nouvelles représentations graphiques sont alors chargées à partir de fichiers de modèles 3D, et viennent remplacer les représentations existantes.

L'événement de changement de modèle s'avère surtout indispensable lorsqu'il s'agit d'entités faisant partie de l'interface interactive simulée du système. En effet, ces éléments modifient fréquemment leurs représentations en réponse aux actions des utilisateurs. Il est utile de remarquer que, contrairement aux systèmes interactifs classiques adoptant une architecture MVC par exemple, la représentation graphique fournie par le GM n'existe que pendant les phases de prototypage par simulation. Centraliser les représentations au sein d'un seul composant facilite alors la gestion des dépendances spécifiques à la simulation.

Le mode de communication entre les composants fonctionnels et le gestionnaire graphique

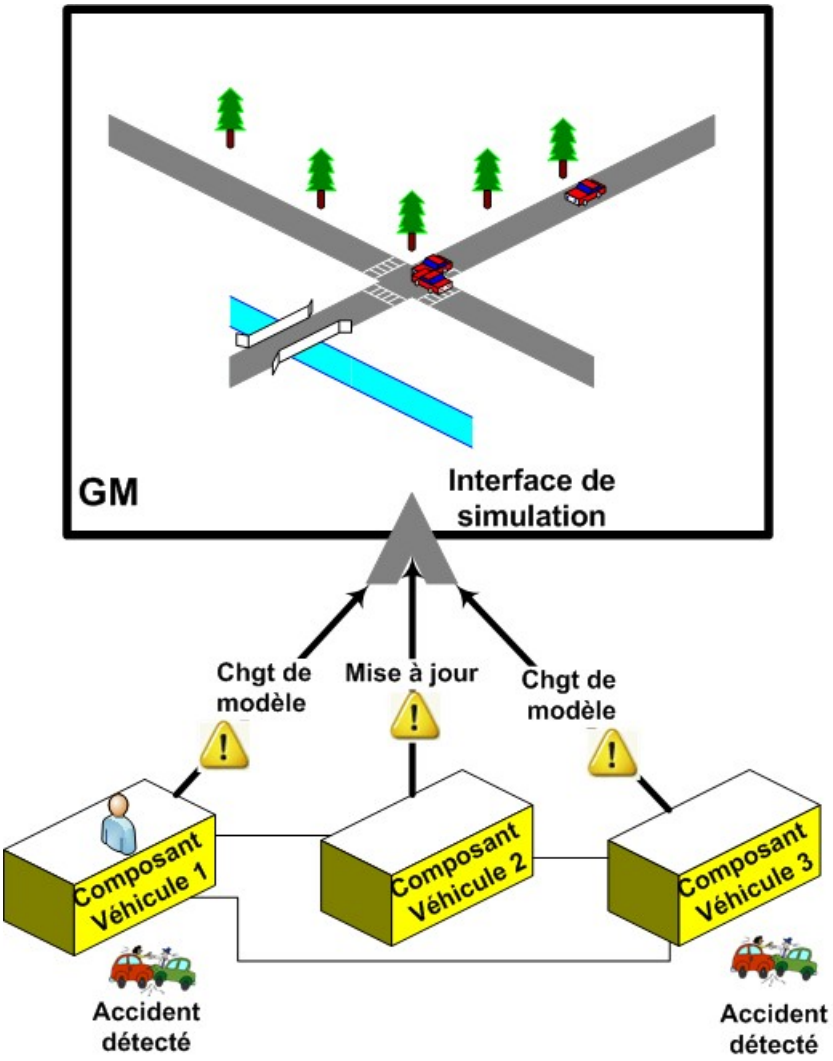

FIGURE 2.3 – Simulation d'un accident

nécessite une modélisation "hors-ligne" de l'environnement dans son intégralité, où toutes les représentations nécessaires sont connues avant le début de la simulation de l'interaction. Ce mode de modélisation "statique" de l'environnement peut s'avérer assez contraignant dans le cas des systèmes à grande échelle ou mobiles. Dans certains cas, cette modélisation est même impossible. Considérons le cas des systèmes mobiles et ubiquitaires, où les différents modules du système peuvent découvrir en cours d'exécution des services offerts par l'environnement. La découverte de services est une facette fonctionnelle du système simulé. Toutefois, si les services découverts sont offerts par des entités qui requièrent des représentations graphiques, et si la plateforme fonctionnelle permet de rajouter ces services pendant l'exécution du système et donc

de sa simulation, la modélisation statique ne permet pas une prise en compte et une intégration de ces nouvelles représentations dans l'environnement de la simulation.

Pour remédier à ce genre de situation, le gestionnaire graphique permet de rajouter des représentations de manière dynamique à l'environnement initial. Cette fonctionnalité utilise l'événement de mise à jour du modèle graphique, en fournissant un nouvel identifiant à la représentation rajoutée. Une fois ajoutée, cette représentation peut être manipulée de manière similaire aux représentations identifiées lors des phases de modélisation statique. Reprenons l'exemple de la

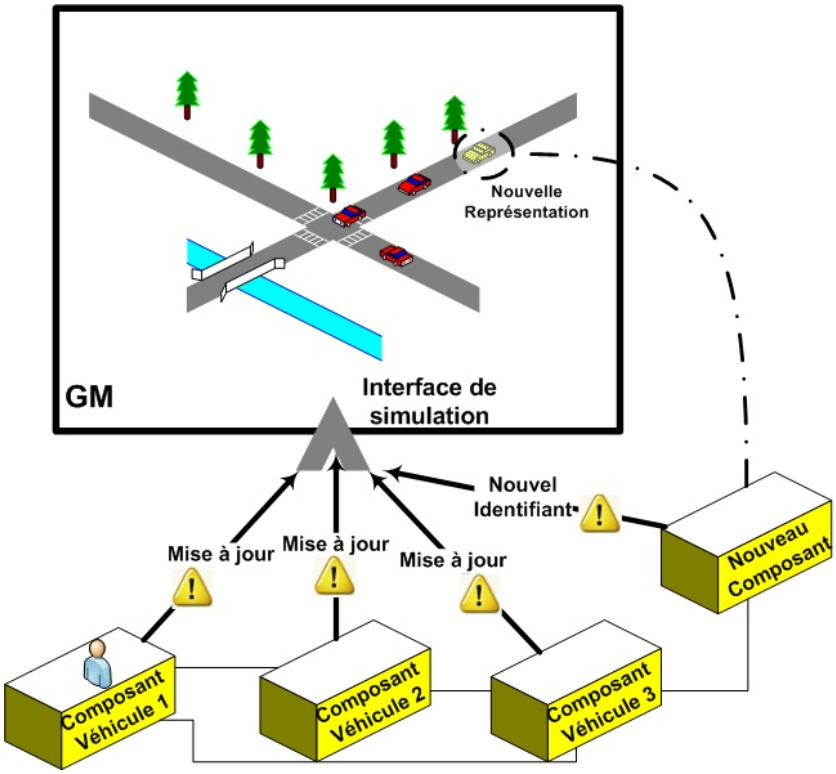

FIGURE 2.4 – Modélisation dynamique

simulation de conduite. En temps réel, les développeurs peuvent décider d'augmenter le trafic (figure 2.4) en ajoutant des dizaines de véhicules supplémentaires. Il est difficile d'imaginer le nombre de véhicules qui seront nécessaires dès le début, et ce nombre ne sera probablement connu qu'une fois le trafic visualisé via le GM. Ce nombre demeure aussi extrêmement variable selon le modèle de trafic choisi. Il serait éventuellement plus simple de définir des algorithmes de génération de véhicules lorsque le nombre de ceux-ci augmente. Une fois ajoutés au système, les composants fonctionnels qui gèrent le trafic supplémentaire se connectent au GM et demandent

le rajout des nouvelles représentations. Il demeure par contre à la charge du développeur de vérifier l'unicité des identifiants fournis puisque le GM ne peut être interrogé sur le bon déroulement des requêtes d'ajout.

Le GM expose en plus de l'interface de simulation, un certain nombre d'attributs qui permettent de le configurer. Parmi ces attributs l'environnement de l'interaction qu'il faut charger ou le point de vue principal que doit représenter le gestionnaire graphique. En effet, un gestionnaire graphique fournit toujours une vue sur l'environnement de simulation relative à la position d'un seul utilisateur, identifié comme principal et défini lors de l'initialisation du GM. L'utilisation de plusieurs instances de gestionnaires graphiques dans un même prototype est toutefois permise et permet entre autre de simuler des interactions multi-utilisateurs.

2.2.2.2 Simulation des interactions multi-utilisateurs

Afin de supporter la simulation d'interactions multi-utilisateurs, l'environnement de simulation doit pouvoir être représenté de différents points de vue. SIMBA permet alors de créer plusieurs instances du GM, chacune initialisée pour représenter le point de vue d'un utilisateur différent. L'environnement initial est alors dupliqué sur les différents gestionnaires, qui peuvent aussi être distribués. Les événements échangés via les interfaces de simulation utilisent un mode de publication qui permet de dupliquer les événements de mise à jour à plusieurs destinataires. Au stade actuel, aucune synchronisation directe entre les différents gestionnaires n'existe. L'intégrité de l'environnement distribué dépend donc de la plateforme d'implémentation sous-jacente ainsi que de sa qualité de service. À terme, différents moyens pourront être intégrés pour synchroniser cette représentation [ETJ07]. En utilisant un GM séparé, les concepteurs peuvent aussi suivre l'évolution de l'interaction d'un point de vue autre que celui de leurs utilisateurs.

Le GM ainsi défini offre aux composants d'un système un accès facile à la manipulation de leurs représentions graphiques. Cet accès est générique et ne nécessite pas de connaissance particulière des détails du rendu des représentations. Le mode de communication unidirectionnel basé sur les événements facilite l'application de la séparation entre les aspects fonctionnels et ceux liés à la simulation. D'autres fonctionnalités peuvent néanmoins être nécessaires pour améliorer la qualité de la simulation de l'interaction, notamment la gestion des animation, des collisions et des sélections graphiques.

2.2.2.3 Gestion des animations

Deux types d'entités peuvent exister dans l'environnement simulé géré par le gestionnaire graphique : les représentations graphiques de composants simulés qui sont manipulables par ces derniers via leurs identifiants, et les objets non-identifiés (sans identifiant connu) qui constituent des éléments statiques de l'environnement et ne peuvent être manipulés. Dans l'exemple de la simulation de conduite du paragraphe 2.2.1, il est difficile d'animer les roues des véhicules à moins d'attribuer un identifiant à chacune de ces roues et d'avoir un composant fonctionnel chargé de les piloter. Ceci pose deux types de problèmes :

- attribuer un identifiant à chacune des roues d'un véhicule afin de les animer risque de compliquer la modélisation de l'environnement.
- l'ajout de composants fonctionnels dépend uniquement de la conception du système et non pas des besoins éventuels des représentations.

L'animation par changements fréquents du modèle 3D du véhicule est quant à elle très coûteuse en termes de surcharge de traitement et d'implémentation des modules pilotant les représentations. L'interdiction d'interroger le GM ne permet pas de récupérer des références aux sous-objets (en l'occurrence les roues), d'autant plus que ceci exposerait les détails internes (primitives, noeuds, etc.) d'implémentation du GM, et nécessiterait des interfaces beaucoup plus évoluées que celles définies pour la gestion de la simulation. Dans les cas où une animation simple et indépendante des états des composants fonctionnels (par exemple l'animation des roues non liée à l'avancement du véhicule) est requise, le GM peut éventuellement fournir un support à l'animation. L'implémentation d'un tel support demeure néanmoins un choix laissé aux développeurs des services mais cette implémentation doit utiliser l'interface standard du gestionnaire. A titre d'exemple, l'implémentation du GM utilisée dans nos travaux permet la gestion des personnages animés définis dans le format de Delta3D. Quand ils sont identifiés, ces personnages peuvent être contrôlés comme toute autre représentation graphique par les composants fonctionnels associés. Ils acceptent néanmoins des paramètres supplémentaires (vitesse, accélération, etc.) qui permettent au GM d'en animer des sous-parties (par exemple les mouvements des pieds et des bras). Ces animations ne peuvent pas (et ne doivent pas) affecter l'état détenu par les composants fonctionnels puisqu'il est impossible d'interroger le gestionnaire graphique sur ces changements.

L'unidirectionnalité des échanges entre le GM et les composants fonctionnels empêche les entités participant à la simulation de récupérer les informations relatives aux entités virtuelles de l'environnement de simulation, que ce soit des représentations graphiques ou des objets statiques. Deux fonctionnalités intimement liés aux environnements virtuels (donc à la simulation) se retrouvent alors très limitées : la gestion des collisions et la sélection graphique.

2.2.2.4 Gestion des collisions

La gestion des collisions requiert un mode de communication entre les composants fonctionnels et le GM où ce dernier peut confirmer les mouvements des entités vis à vis des autres éléments de l'environnement, et par conséquent un retour d'information du GM vers les composants. Ce retour d'information doit inclure la position de la collision et potentiellement les identifiants des objets intersectés. En plus, un modèle de collisions qui serait offert par défaut par l'implémentation du GM peut demeurer inexact, approximatif ou radicalement différent de celui requis par la simulation fonctionnelle. La collision de deux véhicules par exemple dépend de l'interaction des composants fonctionnels qui les pilotent, et non pas des boîtes englobantes de leurs représentations qui peuvent être différentes selon la finesse de la modélisation 3D. Par conséquent, le gestionnaire graphique ne prend pas en charge la gestion des collisions. Les collisions doivent alors être gérées par les composants fonctionnels eux-mêmes ou par des moteurs de gestion de collision séparés. Les positions transmises au GM sont alors pré-validées vis à vis des collisions par les composants fonctionnels.

2.2.2.5 Gestion des sélections graphiques

L'intérêt de la sélection graphique dans le cadre d'un prototypage par simulation est qu'elle permet de reproduire dans le prototype simulé des gestes naturels liés à l'interaction de l'usager avec son environnement physique, tels le toucher ou la manipulation d'objets. Là aussi, le GM doit retourner les identifiants des objets sélectionnés ce qui est également interdit par le mode de communication unidirectionnel. Cependant, la gestion des sélections graphiques a une particularité par rapport à la gestion des collisions c'est qu'elle n'induit ou n'implique pas de modifications au positionnement des entités. Sélectionner un objet consiste à identifier l'entité intersectée et ceci indépendamment de toute manipulation ultérieure qui doit être gérée par les composants

fonctionnels eux mêmes. Dans le cas des systèmes n'implémentant pas de techniques de sélection graphique dans leurs versions non simulées, le GM offre un compromis entre le respect stricte de la règle d'unidirectionnalité des échanges et la flexibilité des mécanismes de simulation, et expose une interface spécifique à la sélection graphique. Cette interface supporte un mode de communication synchrone basé sur l'invocation de méthodes. L'interface de sélection accepte une coordonnée 2D et retourne l'identifiant de l'objet intersecté. Les détails de la sélection graphique dépendent de l'implémentation du GM, notre implémentation utilisant un lancer de rayon ayant pour origine la coordonnée fournie. La sélection graphique ne concerne que les éléments nommés de l'environnement et donc prévus pour être simulés. Cette interface ne peut retourner d'autres attributs tels la position des objets intersectés ce qui restreint son utilisation aux phases de prototypage par simulation. Une fois l'identifiant récupéré, le composant fonctionnel à l'origine de l'appel de sélection entre en contact avec le composant fonctionnel qui gère la représentation dont l'identifiant a été retourné. Cette communication ne peut avoir lieu que si elle est décrite par la conception du système et donc ne viole pas la règle de cohérence entre les prototypes et la conception de l'interaction.

La figure 2.5 montre un GM déployé dans le système de suivi d'objets par caméras décrit précédemment. Les composants des entités suivies transmettent leurs positions aux caméras selon les règles décrites dans la conception du système tels le champ de suivi et la fréquence de capture. Ceci permet par exemple de valider le nombre et le positionnement des caméras dans l'environnement. Les entités qui requièrent une représentation graphique notamment les caméras et les objets simulés, envoient leurs positions au gestionnaire graphique. Cette figure montre également un composant nommé horloge, que nous présentons dans le paragraphe suivant.

2.2.3 Horloge

Une fois les nouveaux attributs de position, rotation ou échelle appliqués aux représentations des composants fonctionnels, les gestionnaires graphiques ne déclenchent pas automatiquement les boucles de rendu chargées de mettre à jour l'affichage de l'environnement. Ceci permet de gérer les mises à jour indépendamment des fréquences d'envoi des événements par les composants fonctionnels et d'adapter ces mises à jour aux contraintes de performance des hôtes sur lesquels s'exécute le rendu de l'environnement de simulation. La gestion des boucles de rendu obéit en

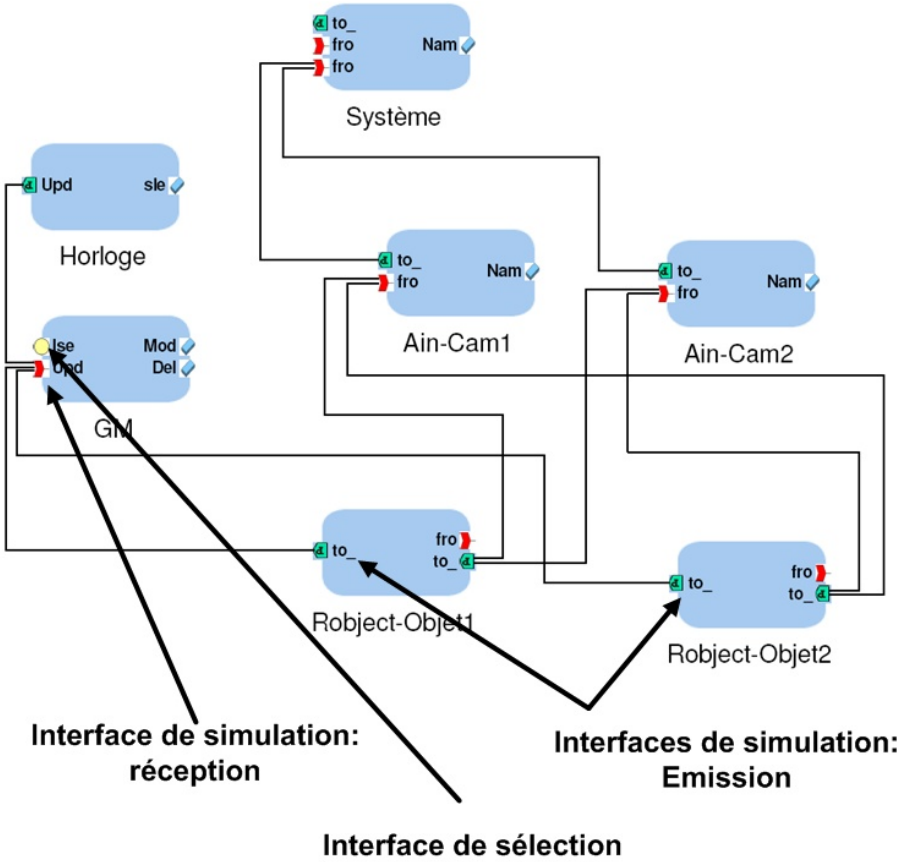

FIGURE 2.5 – Composant GM avec mise en évidence de l'interface de sélection

plus à deux autres contraintes :

- la possibilité de gérer différents gestionnaires graphiques distribués de manière transparente, uniforme ou spécifique à chaque gestionnaire ;
- la possibilité aux développeurs d'identifier et de contrôler de façon simple et intuitive les paramètres de cette gestion (fréquence, etc.) ;

Afin de répondre à ces contraintes, SIMBA utilise un composant séparé pour la gestion des mises à jour de l'affichage. Le composant horloge génère des évènements de mises à jour selon des règles définies via ses attributs. Un composant horloge unique peut servir à la mise à jour de plusieurs gestionnaires graphiques et plusieurs composants horloge peuvent être combinés pour créer une politique de mise à jour composée pour un seul ou plusieurs GM.

Les composants horloge émettent des événement compatibles avec l'interface de simulation du GM. L'utilisation d'une seule interface pour les mises à jour des représentations graphiques et de l'affichage permet de faciliter l'identification et la gestion des dépendances des services de

simulation et de mieux contrôler l'évolution des prototypes vers les versions finales non simulées. Cette évolution sera illustrée dans le chapitre 4.

A travers des mécanismes de communication simples, les composants d'un système peuvent facilement acquérir et contrôler des représentations graphiques intégrées dans un environnement simulé. Les prototypes simulés sont dynamiques mais demeurent peu interactifs en l'absence de support à l'interaction des utilisateurs. Cependant, les représentations des utilisateurs participants à la simulation, leurs avatars, sont un élément clé de l'évaluation des interactions mixtes sur la base de prototypes simulés. Cette interactivité est prise en charge par les services de gestion des interactions de SIMBA.

2.3 Services d'interaction

Les interactions entre un utilisateur et un système informatique ne sont pas uniquement limitées à des dispositifs simples et standards tels le clavier ou la souris. Les évolutions technologiques, logicielles et de communication favorisent le développement de périphériques et de techniques d'interaction avancés. Alors que nous nous intéressons particulièrement au prototypage rapide des systèmes interactifs mixtes, la gestion de ces périphériques et interactions doit être prise en charge et adaptée au contexte du prototypage incrémental basé sur la simulation.

2.3.1 Adaptation au prototypage incrémental par simulation

Pour permettre la création et la gestion de prototypes simulés interactifs, l'architecture de gestion des entrées doit répondre aux deux objectifs suivants :

- l'intégration des modules de gestion des entrées et des interactions dans des processus de prototypage rapide afin de mieux prendre en compte les changements de configurations matérielles et logicielles fréquents ;
- l'adaptation de la gestion des dispositifs à des contextes de distribution des utilisateurs, des dispositifs ou des calculs ;

Afin de répondre à ces objectifs, nous proposons une organisation des services d'interaction en plusieurs couches, offrant différents niveaux d'abstraction. Ces services exposent des interfaces génériques qui permettent une meilleure interchangeabilité des différentes configurations d'entrée

et des techniques d'interaction. Les services d'interaction uniformisent également la gestion de leur cycle de vie ainsi que l'accès aux données qu'ils fournissent. Différents modes d'interaction permettent d'adapter ces services aux contextes d'utilisation différents.

2.3.1.1 Organisation multi-couches

Nous organisons les services d'interaction en trois couches : la couche d'accès uniforme, la couche de capteurs logiques et la couche des métaphores (figure 2.6). Une telle organisation reprend globalement les principes des systèmes de gestion des entrées présentés dans l'état de l'art et nous permet de proposer des solutions logicielles adaptées à différentes configurations de dispositifs et de pilotes, ainsi qu'à différents contextes applicatifs.

La couche d'accès uniforme (CAU) constitue une strate qui permet de masquer les différences entre les multiples périphériques d'entrée et de suivi. Les composants exposent des interfaces génériques qui permettent aux couches supérieurs d'accéder aux données détenues par le composant de gestion de manière uniforme. Ces données ne subissent aucun traitement particulier dans la CAU. Les opérations nécessaires pour rendre ces données utilisables par les applications sont diverses, et s'étalent entre les traitements s'opérant sur les données de capteurs individuels et les opérations incluant différentes données de différents capteurs. Intégrer ces traitements dans la CAU conduirait à des composants surdimensionnés et peu efficaces. En plus, ces traitements étant indépendants du dispositif physique et liés aux types de données fournies, les intégrer dans la CAU serait inefficace et nuirait à la réutilisabilité. La couche de capteurs logiques (CCL) profite de l'uniformisation des données offerte par la CAU pour proposer un certain nombre de traitements génériques vis à vis des types de dispositifs particuliers. Les composants de la CCL constituent des filtres qui peuvent être enchaînés pour composer de nouvelles opérations, mais ne produisent qu'un seul type de données en sortie. Les compositions et opérations plus complexes sont alors réservées à la couche des métaphores (CM) qui permet d'adapter ces données à différentes techniques d'interaction. La CM propose alors un certain nombre de techniques d'interaction plus ou moins avancées. En séparant ces techniques des applications qui les utilisent, cette couche propose un ensemble de composants de haut niveau réutilisables. Une présentation plus détaillée des trois couches des services d'interaction est fournie dans les paragraphes 2.3.2, 2.3.3 et 2.3.4.

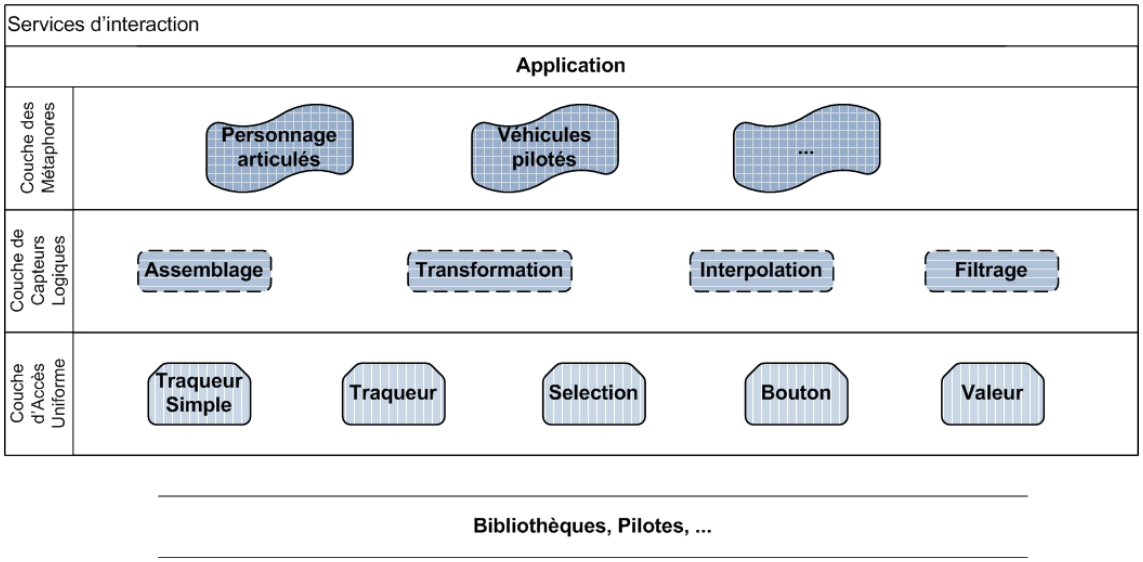

FIGURE 2.6 – Les trois couches de l'architecture des interactions

2.3.1.2 Interfaces génériques

Durant les phases initiales des processus de prototypage itératif, une gestion générique des entrées offre plus de flexibilité pour tester différentes configurations. Cependant, certains périphériques utilisés pour les interactions tridimensionnelles offrent des fonctionnalités qui leurs sont spécifiques et la prise en compte de ces spécificités permet d'optimiser l'intégration des périphériques dans le système final. Le choix entre une gestion générique et la prise en compte des spécificités des différents périphériques conduit souvent à un compromis entre la facilité d'intégration et la standardisation d'une part, et la largeur des fonctionnalités et l'optimisation de l'autre.

Les outils de gestion des entrées existants utilisent souvent les types de données fournies comme critère de classification (paragraphe 2.2.1.1). L'héritage de classes souvent utilisé par ces outils est peu adapté à l'évolution incrémentale et rapide des prototypes, le passage entre une gestion générique et spécifique nécessitant un effort de paramétrage ou de (re) codage conséquent. L'approche orientée composants permet de faire coexister dans un même composant des interfaces génériques et spécifiques ce qui facilite le basculement entre les deux modes de gestion.

Les composants capables de fournir un même type de données exposent les mêmes interfaces et sont considérés comme "fonctionnellement similaires" vis à vis de l'application. Ce terme limite la similarité à la fonctionnalité du composant affichée à travers ses interfaces (les types de

données qu'il est capable de fournir et les modes de présentation de ces données) et non pas la fonctionnalité réelle telle que définie par l'implémentation des interfaces. Les autres critères comme l'ergonomie ou l'adéquation à des tâches particulières ne dépendent pas non plus uniquement des interfaces exposées et par conséquent ne sont pas couverts par la similarité. Des périphériques aussi divers qu'une caméra de suivi vidéo, un traqueur magnétique ou une souris 3D deviennent interchangeables du moment où ils exposent les même interfaces requises par l'application. Dans le cadre d'un prototypage itératif, cette flexibilité permet d'accélérer les itérations lorsqu'il s'agit de tester et valider les techniques d'interaction ainsi que les configurations matérielles et logicielles associées. Les interfaces génériques permettent aussi d'implémenter le concept de périphériques virtuels [HK93]. La simulation d'un certain dispositif à l'aide d'un ou plusieurs autres composants peut alors s'opérer via des mécanismes standards d'assemblage de composants.

Faciliter l'interchangeabilité des composants/périphériques nécessite non seulement l'uniformisation des types de données fournies, mais également la gestion du cycle de vie de ces composants et les modes d'accès à leurs données.

2.3.1.3 Cycle de vie et accès aux données

Les services d'interaction uniformisent, en plus des types de données fournies, la gestion du cycle de vie et les modes de communication de leurs composants.

La gestion du cycle de vie concerne l'initialisation, le paramétrage du fonctionnement et la finalisation des différents composants. Nous proposons deux modes de contrôle de ces étapes du cycle de vie des composants : un contrôle via des méthodes offertes par des interfaces génériques et un contrôle automatique par l'environnement d'exécution selon des politiques paramétrables pour chaque composant. Le cycle de vie du composant est alors pris en charge par l'environnement d'exécution conformément aux politiques standards définies pour les composants logiciels, à moins que ce fonctionnement ne soit altéré par des appels aux interfaces de gestion du cycle de vie exposées.

Les modes d'accès aux données des composants de leur côté répondent à deux contraintes :

- les contraintes liées au contexte d'utilisation de ces données. Ces contraintes ont été identifiées dans l'état de l'art dans le paragraphe 2.1.3.1, telles les besoins des interfaces où

l'interaction est pilotée soit par le système soit par l'utilisateur lui-même ;

– les contraintes liées à la distribution et aux limitations des plateformes matérielles et logicielles, également abordées dans la description des principales difficultés associées au prototypage incrémental par simulation.

La simulation des interactions dans un environnement virtuel requiert beaucoup de flexibilité pour pouvoir représenter des interactions qui ne nécessiteraient pas de support logiciel dans l'environnement non simulé. La simulation fonctionnelle d'une liaison sans fil dans un prototype simulé peut par exemple nécessiter un mode de communication asynchrone. La distribution des composants nécessite quant à elle une prise en compte des contraintes de gigue et de latence qui peuvent influencer l'interaction. Si le système met à contribution plusieurs dispositifs d'entrées et de localisation, les surcharges de traitement et de bande passante peuvent aussi devenir contraignantes.

Les services d'interaction proposent alors deux variantes de chaque type d'interfaces : une basée sur l'invocation synchrone de méthodes et une basée sur un échange asynchrone par événements. Ces interfaces peuvent être utilisées en quatre modes différents : synchrone, synchrone partiel, asynchrone et asynchrone périodique. Les composants peuvent être configurés pour opérer dans l'un ou l'autre des modes individuellement, mais les combinaisons différentes de ces modes ne sont pas toujours pertinentes. Le paragraphe 2.3.5 revient en détail sur les caractéristiques de chacun de ces modes. Le prochain paragraphe détaille les composants de chacune des couches formant les services d'interaction, la CAU, la CCL et la CM.

2.3.2 Couche d'accès uniforme

Un composant de la CAU représente un et un seul dispositif matériel. Cette association facilite l'interchangeabilité des configurations matérielles en permettant d'identifier facilement les composants logiciels associés à ces configurations et de les remplacer. Un composant de la CAU possède un identifiant unique, fonction du périphérique matériel et des détails (version, bibliothèque, etc.) de l'implémentation du composant. Les composants de la CAU exposent une interface générique pour chaque capteur fourni par le dispositif physique. Cinq telles interfaces sont définies par la CAU, inspirées de la taxonomie de GKS3D présentée dans l'état de l'art au paragraphe 2.2.1.1 : les traqueurs simples, les traqueurs, les boutons, les sélections non graphiques

et les valeurs réelles. Ces interfaces associent à la donnée fournie par le capteur une empreinte temporelle et un identifiant du capteur par rapport au dispositif physique.

2.3.2.1 Les traqueurs simples

Ces interfaces fournissent des données de positionnement tridimensionnelles associées à une empreinte temporelle ainsi qu'à un identifiant du capteur relatif au périphérique. Concrètement, le positionnement est représenté par une matrice de transformation rigide qui inclut la translation, la rotation et la mise à l'échelle récupérées auprès du périphérique. Tout traitement supplémentaire de ces données est relégué aux couches supérieures. Une interface de traqueur simple permet de représenter tout capteur de périphérique fournissant une donnée de localisation.

2.3.2.2 Les traqueurs

Les traqueurs représentent les capteurs plus évolués qui fournissent, en plus des données d'un traqueur simple, les informations relatives à la vitesse et à l'accélération du capteur. Cette interface hérite de l'interface traqueur simple ce qui offre plus de flexibilité aux applications et couches supérieures qui peuvent l'utiliser en tant qu'interface traqueur simple.

2.3.2.3 Les boutons

Les interfaces de ce type représentent les capteurs à état binaire tels les boutons d'une souris ou d'un joystick. Tout comme pour les autres interfaces, aucun traitement spécifique n'est appliqué aux données. Ainsi, si la fonction nécessite une interprétation différente des données (appuyé/relâché ou on/off), cette information ne peut être déduite que dans les couche supérieures.

2.3.2.4 Les sélections

Les environnements virtuels identifient généralement le terme sélection comme équivalent aux techniques d'interaction qui permettent de choisir un élément graphique de la scène. Toutefois, l'interface de sélection définie dans la CAU reprend le terme original tel qu'utilisé par la taxonomie de GKS3D. Elle fournit des valeurs entières ayant une signification particulière pour les couches supérieures. L'intervalle de ces valeurs est limité et défini via des attributs du

composant. Les touches d'un clavier par exemple fournissent des valeurs numériques qui peuvent être limitées à l'ensemble des valeurs des caractères ASCII ou UTF, etc.

2.3.2.5 Les valeurs réelles

Cette interface représente les capteurs qui fournissent des valeurs numériques tels les capteurs de température ou d'altitude. Ces valeurs sont représentées par des nombres réels (virgule flottante).

2.3.3 Couche de capteurs logiques

Les composants de la CCL représentent les types de données plutôt que les dispositifs physiques. En utilisant les interfaces génériques des composants de la CAU, la CCL offre des composants de traitement réutilisables du moment où le dispositif sous-jacent expose l'interface compatible. A ce stade nous identifions quatre classes de services dans la CCL : assemblage, transformées géométriques, interpolations/extrapolations et filtrage. D'autres transformations possibles sont présentées dans [Kas98] par exemple.

2.3.3.1 Assemblage

Ces composants rassemblent des données simples et produisent en sortie un seul type de données. Les composants d'assemblage disposent d'un tampon interne qui contient les données récupérées en entrée associées à leurs signatures temporelles. Un composant d'assemblage de la CCL peut par exemple récupérer des données via une interface “ sélection ” d'un clavier. La suite de caractères est alors assemblée en une chaîne fournie en sortie par une interface du composant. Dans le cas où ces données proviennent de différents composants en entrée, les composants de la CCL ont pour tâche de synchroniser et arranger ces données dans le bon ordre. En effet, les latences dues au fonctionnement des périphériques, aux traitements ou à la distribution peuvent retarder la réception de ces informations. A ce stade, les composants d'assemblage ne peuvent gérer des interactions multimodales complexes. Des mécanismes plus avancés pour analyser les interactions multimodales sont introduits dans la littérature (par exemple dans [NC97]), et seront probablement ajoutés à la CCL.

2.3.3.2 Transformations géométriques

Les données fournies par les dispositifs nécessitent souvent un changement de référentiel ou une mise à l'échelle. Ce paramétrage permet d'adapter les données à différents contextes d'utilisation. Ces transformations peuvent être représentées par une seule matrice résultante applicable aux données récupérées. Il est aussi possible de composer de multiples transformations en chaîne (motif des filtres [GHJV95]) à partir de transformations plus simples.

Les composants de transformation offrent les même interfaces définies dans la CAU, ce qui permet aux applications d'accéder indifféremment aux données non traitées de la CAU ou transformées de la CCL ainsi que d'empiler plusieurs composants de cette dernière.

2.3.3.3 Interpolation/extrapolation

Ces composants permettent de prédire les valeurs à un instant " t " d'une certaine donnée, en se basant sur les données disponibles. Les données reçues sont alors gardées en mémoire sur une durée configurable, pour servir à cette prédiction. Un type de composants est requis par algorithme ou méthode d'interpolation ou d'extrapolation. Les données ainsi estimées et les données réelles arrivant en retard ne peuvent être corrélées qu'au niveau des couches supérieures, même si ces dernières sont utilisées en interne pour corriger la prédiction. Les composants de cette classe permettent la gestion des positions et des valeurs numériques. En sortie, ces composants offrent les mêmes interfaces définies par la CAU.

2.3.3.4 Filtrage

Les composants de filtrage modifient les données qui leur sont fournies. Le filtrage permet de trier et de lisser ces données afin d'en supprimer le bruit. Le filtrage ne concerne pas que les traqueurs et valeurs numériques. Les états d'un bouton peuvent être liés à l'état physique de ce dernier (enfoncé ou relâché), ou à une condition abstraite (activé ou éteint). Les composants de filtrage peuvent également utiliser les différents changements de l'état du bouton au niveau de la CAU pour itérer sur des valeurs entières (sélection).

Les composants de la CCL sont limités de façon à ne représenter qu'un seul type de données ce qui permet de définir un répertoire de fonctionnalités fines maximisant les performances et

la réutilisabilité de l'architecture. Les techniques plus évoluées et complexes sont réservées à la couche supérieure : la couche des métaphores.

2.3.4 Couche des métaphores

Dans un cycle de développement itératif, les techniques d'interaction sont susceptibles de changer avec les changements des configurations matérielles des dispositifs. Par conséquent, les métaphores d'interaction doivent pouvoir changer indépendamment de ces dispositifs. On peut par exemple envisager de garder une même métaphore tout en changeant les dispositifs en entrée ou de tester différentes métaphores tout en gardant les mêmes dispositifs.

Les composants de la CM récupèrent en entrée des données qui, contrairement aux données récupérées par les composants de la CCL, peuvent être hétérogènes. La synchronisation de ces données, utile pour les interactions multimodales, a lieu au niveau de la couche des métaphores. Afin de rester indépendants des applications, les composants de la couche des métaphores ne peuvent utiliser des informations relatives à l'environnement 3D ou à l'application elle-même. Ceci limite les types de métaphores possibles à celles ne nécessitant pas d'information de la couche applicative (sélections graphiques, etc). Les données en entrée doivent provenir des couches inférieures (dispositif, CAU et CCL) mais ne doivent pas être fournies par des couches supérieures. Ceci garantit aux composants de la CM une indépendance totale. Il demeure néanmoins impossible d'établir une liste exhaustive ou une taxonomie complète des métaphores d'interaction, ces métaphores étant largement variables en fonction des domaines et des contextes d'application. Nous présentons néanmoins trois métaphores afin d'illustrer l'utilisation et l'implémentation des composants de la CM.

2.3.4.1 Personnage simple articulé

Ce composant permet de contrôler un personnage articulé via des entrées simples (figure 2.7), comme c'est le cas dans les " jeux de tirs à la première personne " classiques. L'avatar est mobile et articulé mais les articulations sont animées automatiquement en fonction des paramètres internes du modèle tels la vitesse, la position et l'accélération. Les entrées servent uniquement à modifier la position globale de l'avatar virtuel.

Le composant récupère en entrée une donnée de positionnement du type " traqueur ". Seules

deux dimensions de cette donnée seront exploitées, la métaphore étant limitée aux déplacements planaires. En interne, cette métaphore contient les paramètres du modèle de l'avatar et notamment ceux de l'animation de ses articulation en fonction des changements de sa position. En sortie, le composant fournit une donnée de positionnement 2D de l'avatar ainsi que les angles (1D, 2D ou 3D) relatifs aux articulations. Ces derniers sont calculés à partir des paramètres internes de l'avatar ainsi que de ses changements de position. L'application récupère ces données et les applique au modèle graphique qu'elle utilise, ou dans le cas de l'architecture utilisant le GM, émet l'événement de mise à jour de position relatif au modèle 3D qu'elle contrôle.

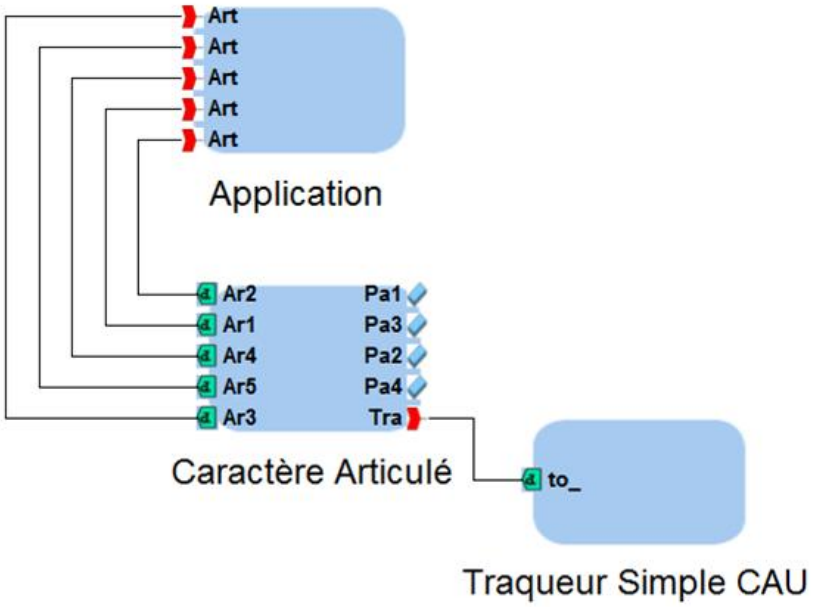

FIGURE 2.7 – Composant de la CM pour un personnage articulé simple

2.3.4.2 Personnage articulé

Ce composant permet de contrôler un personnage articulé via des entrées relatives à chacune des articulations du modèle (figure 2.8) telles celles obtenues par capture de mouvement. En entrée, le composant récupère une donnée de positionnement du type " traqueur " pour chaque articulation mobile en 2D ou 3D, et une donnée réelle pour chaque articulation mobile en une dimension. En interne, cette métaphore contient les paramètres du modèle de l'avatar, et notam-

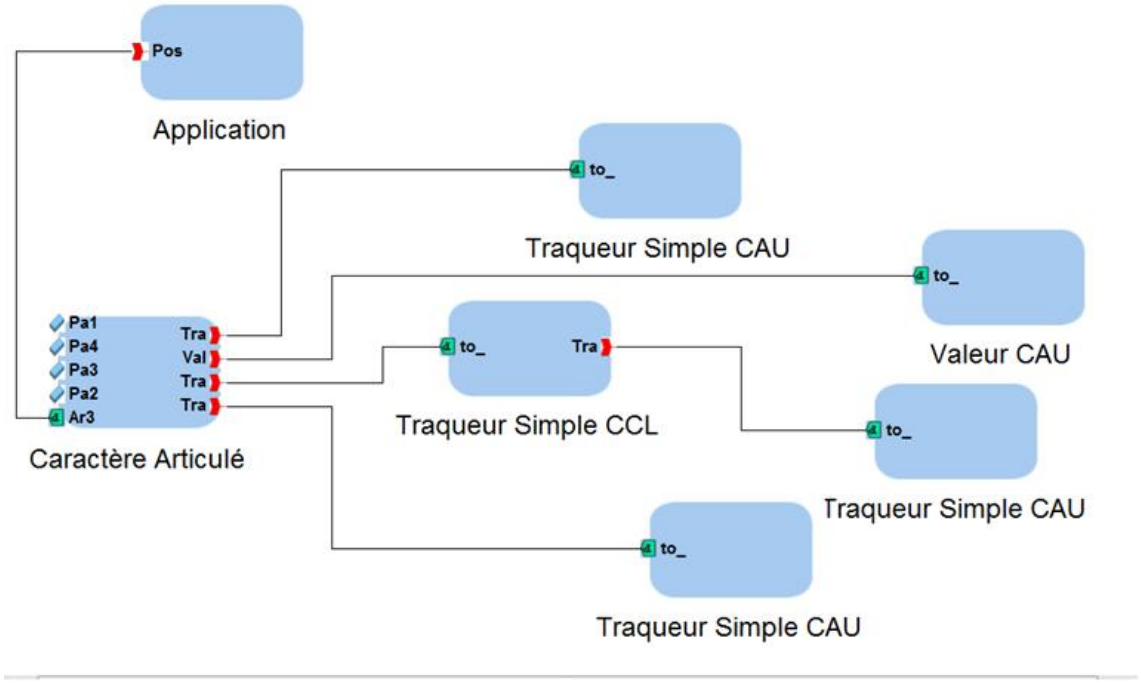

FIGURE 2.8 – Composant de la CM pour un personnage articulé complexe

ment ceux des correspondances entre les entrées et les mouvements à appliquer aux articulation. Elle contient aussi les paramètres de position globale de l'avatar en fonction des mouvements de ses articulations. En sortie, le composant fournit une donnée de positionnement de l'avatar.

2.3.4.3 Véhicule motorisé piloté

Ce composant permet de contrôler un véhicule motorisé via des entrées relatives aux positions des pédales, à la rotation du volant et aux changements de vitesses. Cette métaphore suppose la présence d'un dispositif (réel ou virtuel) capable de fournir de telles données. Un tel dispositif (réel) est montré dans la figure 2.9.

En entrée, le composant de cette métaphore récupère une donnée relative à la rotation du volant ainsi qu'une donnée pour chacun des angles des pédales (figure 2.10). Ces données sont de type " valeur réelle ". Il récupère aussi une donnée de " sélection " relative à la vitesse actuelle choisie avec le levier. En interne, cette métaphore contient les paramètres du modèle du véhicule et notamment ceux relatifs à son accélération, vitesse maximale, poids, etc. En sortie, le composant fournit une seule donnée de positionnement pour le véhicule calculée en fonction des entrées et du modèle interne.

FIGURE 2.9 – Exemple d'un dispositif physique de pilotage

Comme énoncé dans le paragraphe 2.3.1, les composants de la CAU, de la CCL et de la CM peuvent opérer dans différents modes que nous présentons dans le paragraphe suivant.

2.3.5 Modes de communication

Les composants exposent deux variantes de leurs interfaces génériques : une qui permet un mode de communication par invocation de méthodes et une qui permet un mode à base d'événements. L'accès aux données des composants peut alors s'opérer via quatre modes différents : le mode synchrone, synchrone partiel, asynchrone et asynchrone périodique.

2.3.5.1 Mode synchrone

Dans ce mode, les composants attendent des invocations provenant des couches supérieures. Ces appels sont bloquants et impliquent une mise à jour de l'état interne du composant en contactant les couches sous-jacentes (figure 2.11). Ce mode induit une forte latence, la somme des latences induites par chacune des couches logicielles superposées ainsi que l'accès au périphérique et au réseau. Néanmoins la surcharge de traitement et de communication est faible puisque les données envoyées sont limitées au minimum utile pour l'application. Ce mode est adapté à un fonctionnement impératif d'une application qui contrôle elle-même les entrées.

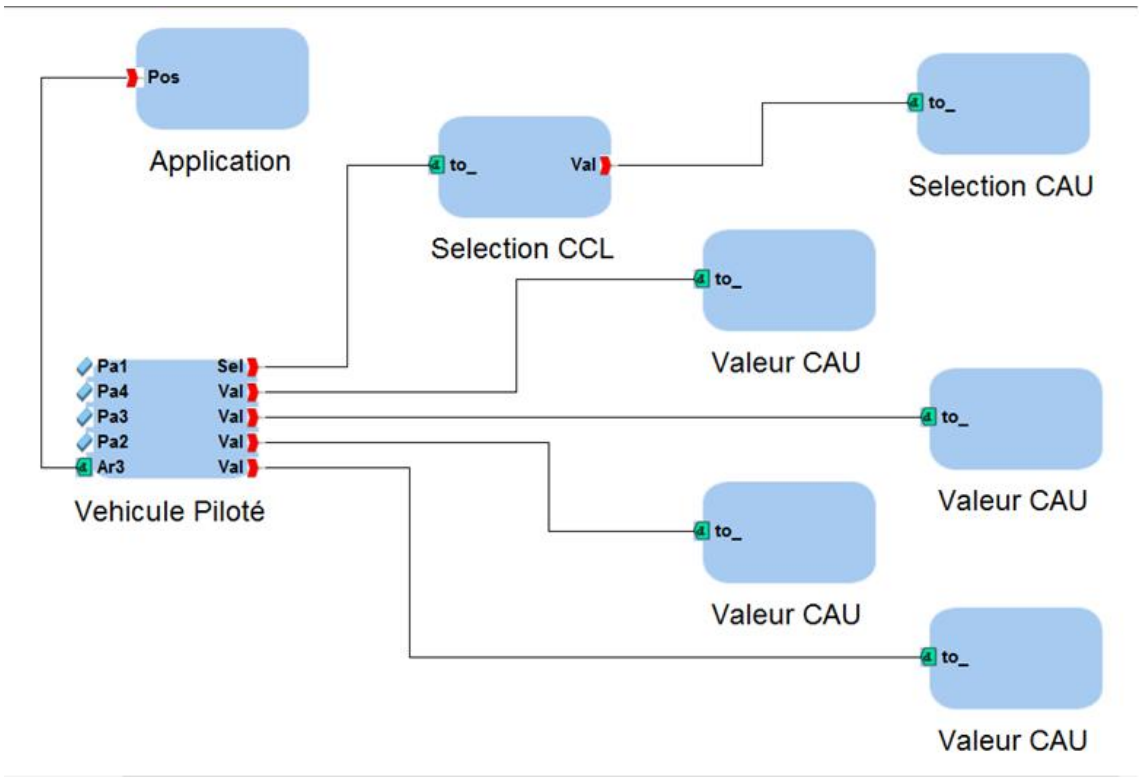

FIGURE 2.10 – Composant de la CM pour un véhicule piloté

2.3.5.2 Mode synchrone partiel

Dans ce mode, les composants attendent des invocations tout comme en mode synchrone. Néanmoins, ils renvoient immédiatement leur état interne sans le mettre à jour (figure 2.12). Cette mise à jour est alors assurée par des composants des couches inférieures opérant dans un des modes asynchrones ou bien par une boucle infinie du composant lui-même. Ce mode permet un compromis entre la surcharge et les latences, surtout lorsque les composants de différentes couches se retrouvent répartis sur différents hôtes.

2.3.5.3 Mode asynchrone

Ce mode offre un fonctionnement similaire à celui requis par des interfaces graphiques interactives. Les composants publient leurs états internes à chaque changement via leurs interfaces asynchrones. Le composant ne connaît pas les composants qui lui sont connectés. Tout comme pour le mode synchrone partiel, la mise à jour de l'état interne est assurée par un mécanisme séparé (figure 2.13). Le mode asynchrone garantit la réception la plus rapide des données par

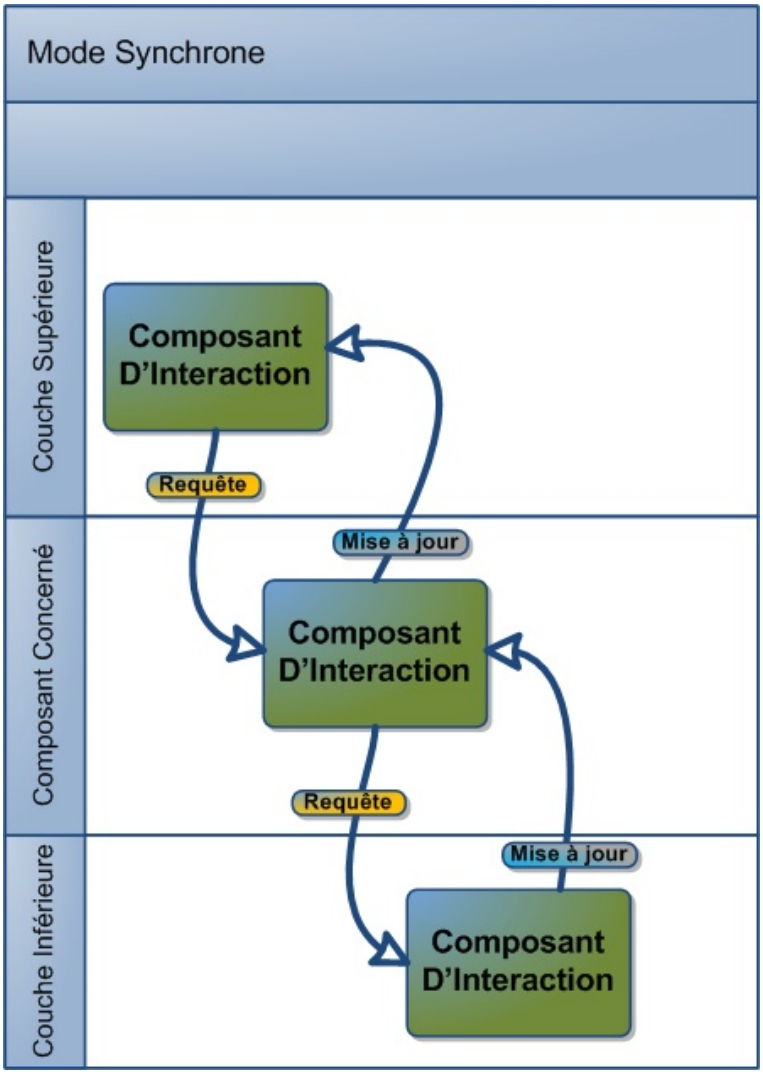

FIGURE 2.11 – Mode de communication synchrone

les couche supérieures mais nécessite une puissance de traitement et une bande passante plus importantes.

2.3.5.4 Mode asynchrone périodique

Ce mode permet d'émettre les données relatives à l'état interne du composant lorsque cet état change et à des intervalles réguliers (figure 2.14). Ce mode est proche du mode synchrone partiel dans son fonctionnement mais permet de s'adapter à des applications qui requièrent un mode à base d'événements. Le mode asynchrone périodique est adapté aux cas des systèmes de localisation où la position peut changer assez vite mais où une fréquence plus basse est souvent

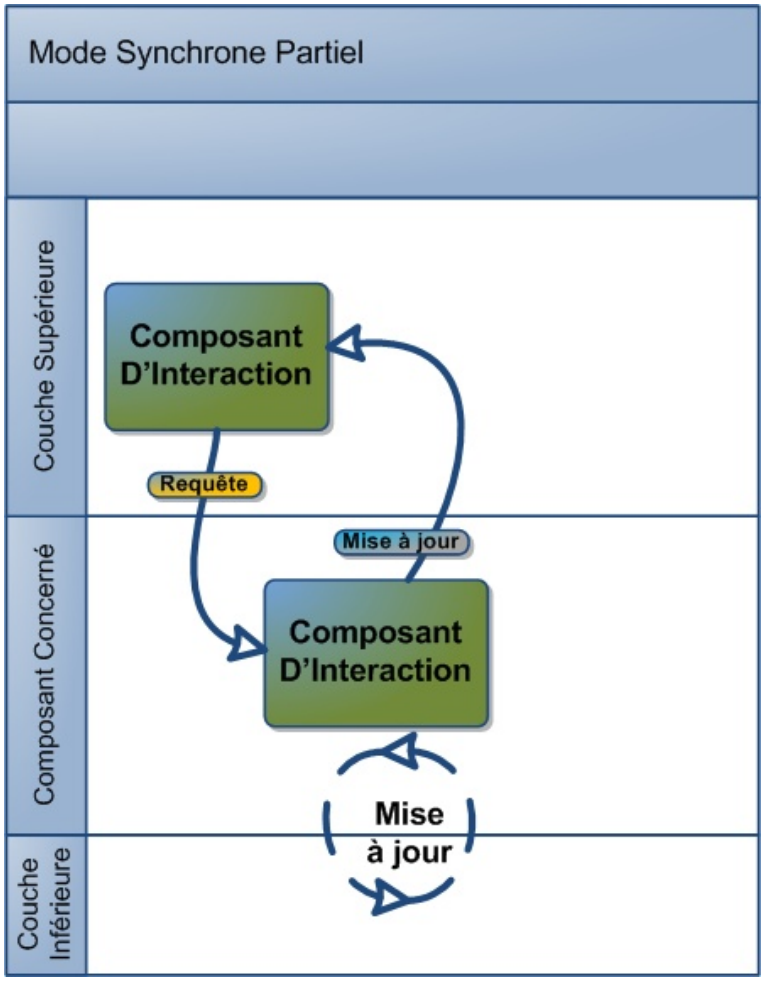

FIGURE 2.12 – Mode de communication synchrone partiel

suffisante pour garantir l'interactivité du système.

Pour offrir une architecture des interactions utile, la définition d'un cadre théorique n'est pas suffisante. Il s'agit aussi de définir et implémenter un répertoire de composants qui gèrent un large choix de dispositifs. L'intégration de pilotes et de logiciels existants permet de fournir un tel choix avec un minimum d'effort.

2.3.6 Intégration des pilotes et bibliothèques

Afin d'éviter de réinventer des solutions, la couche d'accès uniforme utilise des bibliothèques existantes telles VRPN pour la gestion des échanges bas niveau avec les périphériques ou ARToolkit pour la localisation à base de caméras. L'intégration de ces bibliothèques démontre aussi la facilité d'intégrer un code existant dans les composants de la CAU :

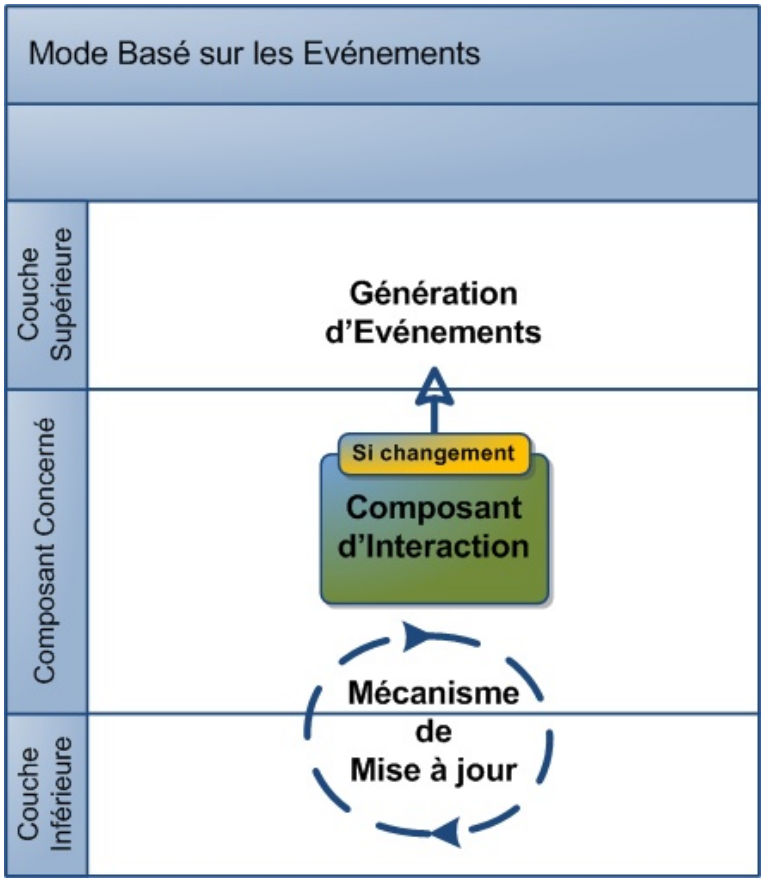

FIGURE 2.13 – Mode de communication asynchrone

2.3.6.1 VRPN

VRPN offre une architecture client-serveur distribuée. Cette capacité devient vite superflue vu que l'architecture des interactions gère la distribution de façon transparente. Pour intégrer VRPN, les composants de la CAU se greffent directement sur l'objet serveur de l'architecture client-connexion-serveur de VRPN, dépassant ainsi les objets client et connexion. VRPN sert alors comme une couche de pilotes avec une sémantique d'accès aux périphériques uniformisée.

2.3.6.2 ARToolkit

ARToolkit permet la gestion de la localisation à base de caméras assez facile à mettre en oeuvre. En sortie, la bibliothèque intègre des objets 3D dans le flux vidéo sortant à la place des motifs détectés. L'architecture de gestion des interactions intègre seulement la première partie du pipeline de ARToolkit. La gestion des sorties est alors reléguée à l'application.

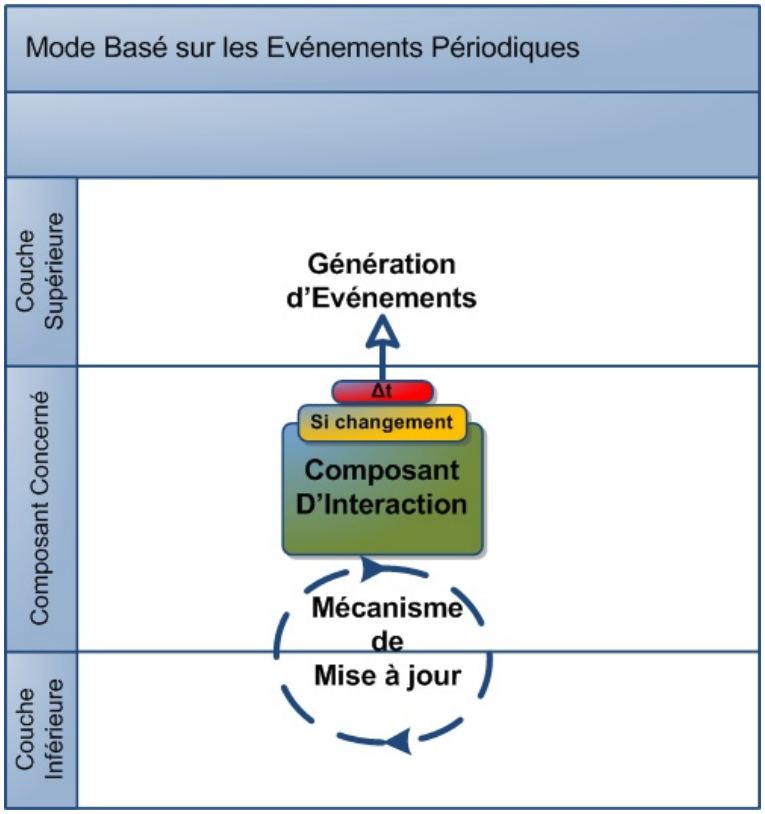

FIGURE 2.14 – Mode de communication asynchrone périodique

3

Approche dirigée par les modèles : méta-modèle, outils, et processus de développement

Une approche dirigée par les modèles nous permet d'identifier clairement les différentes étapes du processus de développement des systèmes, ainsi que les liens entre ces étapes. L'approche décrit un processus descendant qui consiste en l'opérationnalisation des modèles de conception. Cette opérationnalisation utilise un espace de modélisation qui permet de compléter les modèles de départ par les artefacts et informations utiles à l'implémentation. L'espace de modélisation fournit également les mécanismes nécessaires pour garantir la conformité des prototypes avec les choix de conception, supporter le développement incrémental et s'adapter à différents domaines d'application des SIM :

- conformité de la description avec les modèles conceptuels : l'approche dirigée par les modèles permet de construire les prototypes et implémentations des systèmes autour de l'architecture de services de SIMBA. En parallèle, elle permet de garantir le respect des spécifications définies par les concepteurs au delà des premiers prototypes générés et à travers les phases de prototypage successives. L'espace de modélisation prend ainsi en charge la validation des contraintes identifiées dans les modèles conceptuels.
- support au développement incrémental : l'architecture de SIMBA répond partiellement aux

contraintes imposées par le développement incrémental et notamment la séparation entre les composants fonctionnels et les représentations graphiques. Il demeure néanmoins difficile de différencier les entités simulées ou non-simulées ainsi que les interdépendances des composants des systèmes sans un support visuel qui explicite ces différences. En parallèle, un support logiciel à la modélisation permet de garantir des règles telles l'interdiction pour un gestionnaire graphique de fournir des informations aux composants fonctionnels, ainsi que de supporter l'automatisation de certaines des étapes de l'évolution incrémentales.

– adaptation à différents domaines d'application : le prototypage par simulation peut concerner des systèmes mixtes dans différents domaines applicatifs, que ce soit dans des contextes expérientiels ou orientés vers les tâches. L'approche dirigée par les modèles permet d'intégrer différents modèles conceptuels, ainsi que différentes plateformes d'implémentation et les services associés.

L'espace de modélisation de SIMBA se base sur un méta-modèle qui permet de décrire les prototypes des systèmes mixtes, et un ensemble des règles et outils qui permettent de guider l'évolution incrémentale de ces prototypes. L'espace de modélisation ainsi défini permet d'articuler un processus de développement dirigé par les modèles autour d'un certain nombre de rôles qui sont également présentés dans ce chapitre.

3.1 Le méta-modèle SIMBA

Le méta-modèle définit la structure (éléments et relations) des prototypes simulés. Les systèmes sont décrits avec SIMBA comme un ensemble de composants exposant chacun une ou plusieurs interfaces. Cette description reprend un certain nombre de concepts génériques communs aux modèles de composants et définit des spécialisations de ces concepts pour les adapter au prototypage par simulation. Le méta-modèle SIMBA déployé dans un environnement approprié offre ainsi un support visuel à la modélisation. Les prochains paragraphes présentent les éléments génériques du méta-modèle, les éléments représentant les services de simulation de SIMBA, les éléments représentant les services d'interaction, les composants miroirs qui facilitent l'évolution incrémentale des prototypes ainsi que les éléments qui permettent de conserver les choix de conception regroupés au sein d'un modèle annexe au modèle principal appelé modèle

miroir.

3.1.1 Éléments génériques du méta-modèle SIMBA

Ce sont les éléments de base du méta-modèle dont dérivent tous les autres éléments de description des prototypes et des systèmes. Ils permettent de représenter les divers artefacts des systèmes interactifs mixtes tels les services et composants fonctionnels et applicatifs. SIMBA identifie les éléments génériques suivants : les composants, les ports d'émission et de publication d'événements, les ports interface et réceptacle, les attributs et les connexions d'interface et d'événements.

3.1.1.1 Composant

Un composant représente les entités et services génériques. Il possède deux attributs dans le méta-modèle : son nom SIMBA lui servant d'identifiant, et le nom de l'archive du composant logiciel qui lui est associé.

3.1.1.2 Port d'émission d'événements

Lorsqu'il est exposé par un composant, un port d'émission d'événements permet à ce dernier d'émettre des événements vers d'autres ports compatibles. Les événement émis peuvent avoir un seul destinataire défini par la configuration de déploiement du système. Ce port est identifié par son nom SIMBA et par le type d'événements pouvant être émis.

3.1.1.3 Port de publication d'événements

Ce port est presque identique au port d'émission d'événements. Il opère néanmoins dans un mode de publication qui permet à un composant d'émettre les événements vers plusieurs destinataires à la fois selon un mode *multicast*.

3.1.1.4 Port de réception d'événements

Le port de réception permet au composant qui l'expose de s'abonner à des événements provenant des ports d'émission et de publication d'événements, tant que ces derniers sont du même type ou d'un type dérivé de celui défini dans l'attribut “ type ” du port de réception.

3.1.1.5 Port interface

Le port interface offre un mode de communication à base d'invocation de méthodes. L'interface comporte un certain nombre de méthodes pouvant être invoquées par d'autres composants du système. Une seule interface peut servir plusieurs composants et est identifiée par son nom en SIMBA ainsi que son type.

3.1.1.6 Port réceptacle

Un port réceptacle permet à son composant d'accéder aux méthodes offertes par un port interface. Les types des deux ports doivent être compatibles tels qu'identifiés par leurs attributs respectifs.

3.1.1.7 Attribut

Contrairement aux attributs des entités du méta-modèle (comme le nom et le type des composants) qui sont fixés en nombre et en type et qui servent à mieux qualifier les éléments du méta-modèle, cette entité est un éléments à part entière du méta-modèle SIMBA. Un tel attribut peut être ajouté ou supprimé du modèle du système et peut qualifier n'importe quel composant ou interface de ce dernier.

3.1.1.8 Connexion d'interface

Cette connexion permet de lier entre eux un port interface et un ou plusieurs ports réceptacle. Les composants ainsi connectés peuvent échanger des données via des appels de méthodes. Une connexion illustre la dépendance entre deux composants et ne peut s'établir qu'entre des ports compatibles.

3.1.1.9 Connexion d'événements

Cette connexion permet de lier deux interfaces à base d'événements. Une connexion entre deux interfaces représente l'abonnement du récepteur à l'événement généré par le port d'émission ou de publication. L'ensemble des connexions définit les dépendances des composants au sein d'un système.

Les éléments génériques sont représentés dans la figure 3.1. Ils servent de base aux autres éléments du méta-modèle, plus spécifiques à la méthodologie avancée par SIMBA.

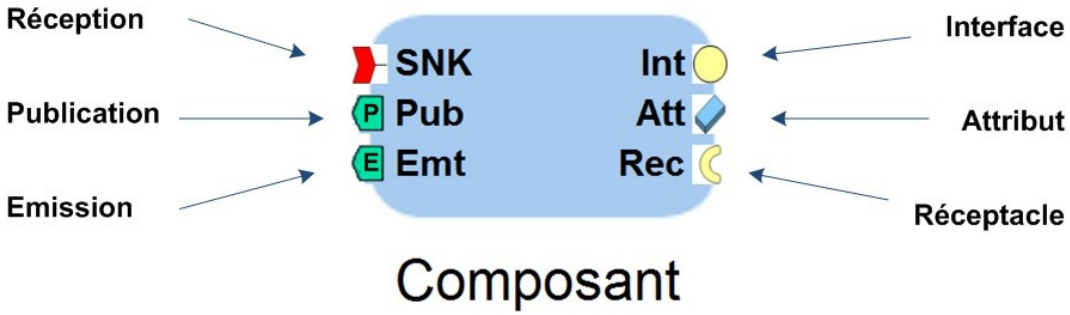

FIGURE 3.1 – Elements génériques du méta-modèle SIMBA

3.1.2 Éléments représentants les services de simulation

Ces éléments permettent au méta-modèle de représenter les services de simulation décrits dans le chapitre 2. Ces éléments dérivent des entités génériques du méta-modèle ce qui facilite leur intégration avec divers services applicatifs hétérogènes. La spécialisation permet d'identifier (visuellement) ces éléments par les développeurs des systèmes mixtes. Elle permet également de définir les règles et outils adaptés présentés dans le paragraphe 3.2. Nous identifions les éléments spécifiques suivants : le conteneur de simulation, le port de réception d'événements de simulation, le port de publication d'événements de simulation, l'interface de sélection, les composants simulés, la connexion de simulation, la connexion de sélection, le gestionnaire graphique et l'horloge.

3.1.2.1 Conteneur de simulation

Tout modèle de prototype défini dans l'espace de modélisation doit être déclaré dans un conteneur de simulation. Ce conteneur permet de séparer différents modèles de prototypes obtenus par exemple à différents stades du prototypage incrémental.

3.1.2.2 Port de réception d'événements de simulation

Lorsqu'il est exposé, ce port identifie un composant capable de recevoir les événements de gestion de l'aspect visuel de la simulation. Nous reviendrons dans le chapitre 5 sur des représen-

tations autres que visuelles à gérer dans le cadre de la simulation. Cette interface ne peut être exposée que par un gestionnaire graphique, une règle qui est détaillée ultérieurement.

3.1.2.3 Port de publication d'événements de simulation

L'émission des événements de gestion de l'aspect visuel de la simulation ne peut se faire que par un mode de publication. En effet, l'environnement de modélisation doit pouvoir gérer l'ajout ou le retrait de plusieurs GM de façon transparente et sans devoir modifier les interfaces des composants concernés. Un mode de publication offre une indépendance vis à vis du nombre de gestionnaires graphiques requis pour représenter l'environnement de simulation de différents points de vue.

3.1.2.4 Port interface de sélection

Ce port permet aux composants qui implémentent des techniques de sélection graphique d'obtenir les identifiants des objets graphiques simulés. Ces identifiants peuvent servir par la suite à contacter le composant fonctionnel qui gère la représentation graphique en question.

3.1.2.5 Composant simulé

Un composant simulé est un composant fonctionnel qui requiert une représentation graphique, et est capable d'en assurer la gestion via la publication des évènements de simulation. Ce composant expose un port de publication des événements de simulation et nécessite la définition d'un certain nombre d'attributs tels l'identifiant de l'objet 3D géré par le gestionnaire graphique.

3.1.2.6 Connexion de simulation

Cet élément représente le lien entre les ports de visualisation exposés par les composants simulés d'un côté et les gestionnaires de l'autre. L'implémentation d'un composant simulé peut toujours exposer et gérer une interface de simulation, mais seule une connexion au gestionnaire approprié lui permet de contrôler sa représentation.

3.1.2.7 Connexion de sélection

Cet élément représente le lien entre le réceptacle de sélection exposé par le composant fonctionnel et l'interface de sélection du gestionnaire. Les composants ainsi connectés ont accès aux méthodes de l'interface de sélection.

3.1.2.8 Gestionnaire graphique

Le gestionnaire graphique défini dans le méta-modèle représente le gestionnaire graphique de l'architecture de SIMBA. Il expose le port de réception d'événements de simulation et permet aux composants simulés d'accéder à la gestion de leur représentations graphiques. L'identification du GM dans le méta-modèle est essentielle puisqu'elle nous permet de définir les contraintes associées à ce composant et d'assurer le bon déroulement du processus de développement itératif et incrémental. Le GM peut également exposer l'interface de sélection.

3.1.2.9 Horloge

L'horloge représente également le composant du même nom défini par l'architecture de SIMBA. Une horloge expose un port de publication des événements de simulation et l'utilise pour publier des événements de mise à jour. Ces événements permettent au GM de rafraîchir le rendu des représentations. Tout comme le GM, le composant horloge est associé à un certain nombre de règles que nous exposerons plus tard.

3.1.3 Éléments relatifs aux services d'interaction

L'interaction constitue une facette des systèmes mixtes commune aux systèmes finaux ainsi qu'à leurs prototypes. En effet, les SIM nécessitent souvent des mécanismes de communication entre les entités numériques et physiques, tels les Ain de ASUR. Les prototypes simulés requièrent également des moyens pour contrôler les entités comme les avatars des utilisateurs par exemple.

Le méta-modèle SIMBA reprend les types d'interfaces de la couche d'accès uniforme (CAU) présentée dans le paragraphe 2.3.2 du chapitre précédent. Les composants modélisés peuvent alors exposer les interfaces de la CAU : les traqueurs simples, les traqueurs, les valeurs, les sélections et les boutons. Ces interfaces peuvent offrir un mode asynchrone ou synchrone. La prise en

charge par le méta-modèle des éléments relatifs à l'interaction permet d'étendre la modélisation à la facette interactive des systèmes mixtes. Cette prise en charge permet également d'intégrer l'interaction dans le processus de développement incrémental en offrant un support visuel à l'identification des composants d'interaction et des types de données requis ou fournis, ainsi qu'un support logiciel à la vérification de la compatibilité des interfaces qui permet d'identifier les composants d'interaction interchangeables.

3.1.4 Composants miroirs

La facilité du prototypage par simulation dépend largement de la facilité à attribuer à un composant fonctionnel logiciel une représentation graphique et de la facilité pour ce composant de contrôler cette représentation. Néanmoins lors de la conception des SIM, on est souvent amenés à intégrer des entités physiques n'ayant pas, en dehors du prototypage par simulation, de facettes logicielles. La simulation de ces entités leur attribue une telle facette (pour interagir avec les autres entités simulées) et peut conduire à des situations peu intuitives résultant du processus de développement incrémental. Les composants miroirs permettent d'identifier ces situations ainsi que de rationaliser les solutions.

Afin d'expliquer le rôle des composants miroirs, reprenons l'exemple du système mixte où des objets sont localisés par des caméras, présenté dans le paragraphe 2.2.1 du chapitre précédent. Ce paragraphe présente également le diagramme SIMBA du système entièrement simulé (nous reviendrons plus tard en détail sur les modalités de génération de ces diagrammes).

Les composants fonctionnels qui gèrent les objets suivis envoient les positions de ces objets et leurs identifiants aux caméras ce qui permet à ces dernières de les localiser dans l'environnement simulé. Dans le cadre de l'évolution incrémentale de ce prototype vers sa version finale, nous remplaçons dans un premier temps une seule des cameras simulées par une caméra réelle. Ceci mène au schéma SIMBA de la figure 3.2, avec une seule des cameras assurant la localisation d'objets réels via la bibliothèque ARTookit, la deuxième caméra demeurant simulée. Les objets physiques non simulés n'ont plus à ce stade de représentation logicielle et ne peuvent par conséquent être localisés par la caméra simulée. Une telle contrainte peut limiter la flexibilité des incréments de l'évolution des prototypes, contraignant ces évolutions à remplacer lors d'une même itération l'ensemble des éléments physiques simulées et les composants logiciels qui en

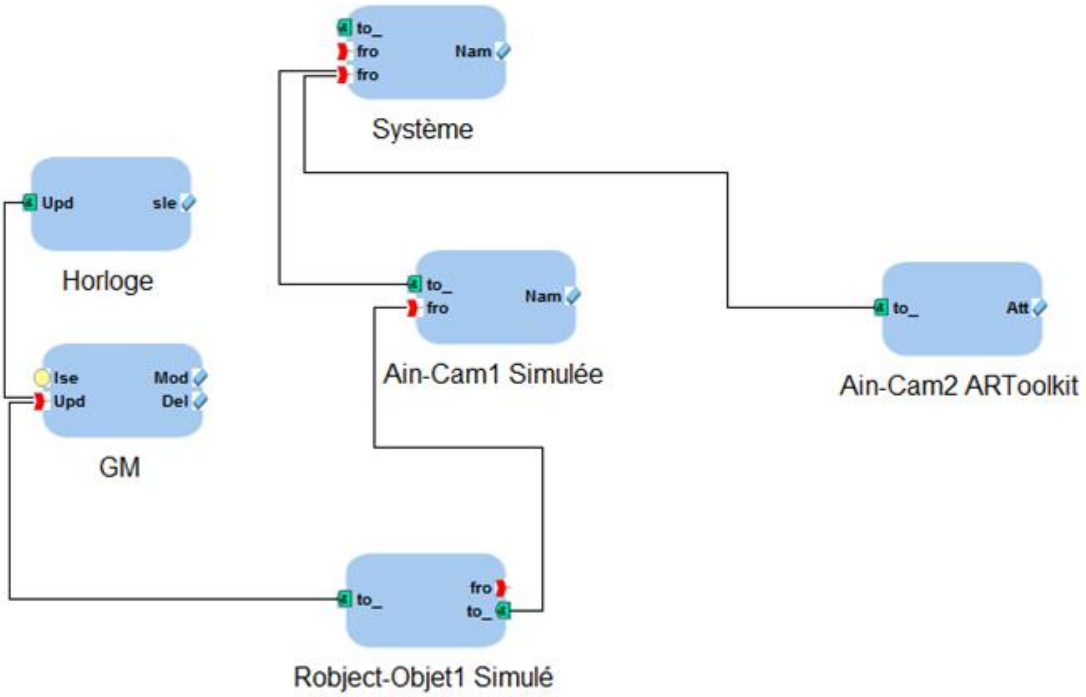

FIGURE 3.2 – Modèle SIMBA du système à un stade d'évolution intermédiaire

dépendent, par des éléments non simulés. Il paraît alors intéressant d'identifier au niveau du méta-modèle les données requises pour assurer le bon fonctionnement des entités (caméras, etc.) simulées, afin de faciliter la manipulation de ces données. Une fois identifiées, ces données (ici la position des objets physiques) sont représentées par des composants spécifiques pendant les phases intermédiaires du prototypage : les composants miroirs. Dans l'exemple des objets suivis par des caméras, un composant miroir récupère la position de l'objet physique qu'il remplace dans le modèle du prototype et permet à la caméra simulée de localiser l'objet physique non simulé (figure 3.3). Il faut noter toutefois que le méta-modèle ne définit pas dans son état actuel les modalités d'obtention des données associées aux composants miroirs. Les composants miroirs sont néanmoins régis par des règles assurant leur bonne intégration dans les phases de prototypage, et sur lesquelles nous reviendrons plus tard.

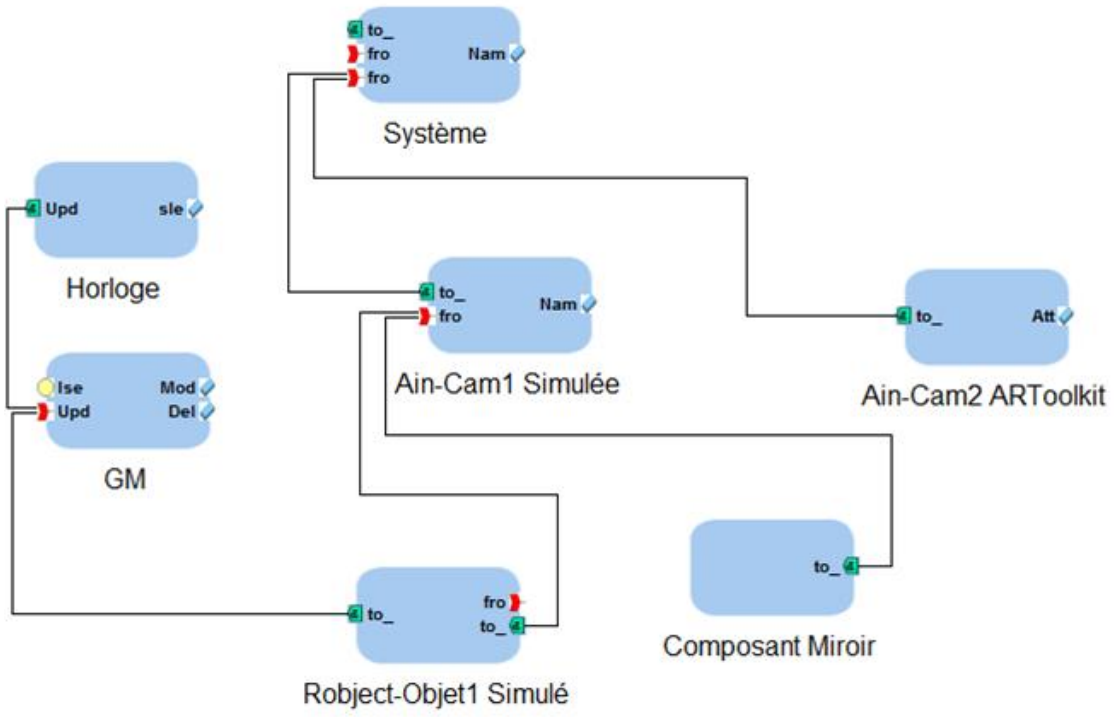

FIGURE 3.3 – Modèle SIMBA du système utilisant un composant miroir

3.1.5 Éléments de persistance des modèles conceptuels

La vérification de la conformité du modèle du prototype avec les modèles conceptuels de départ dépend largement de la capacité de SIMBA à conserver les dépendances et les attributs exprimés dans ces derniers. Dans certains cas (comme pour ASUR), les informations contenues dans le modèle conceptuel ne permettent pas d'exprimer les dépendances des composants dans SIMBA, à cause d'ambiguïtés ou d'absence de définition spécifique des types ou de la structure des composants et des interfaces. Le modèle initial généré dans l'espace de modélisation de SIMBA ne contient donc pas d'information concernant les dépendances des composants. Néanmoins, toute validation ultérieure de la cohérence entre les modèles des prototypes et les modèles de conception nécessite un accès à de telles informations. Le méta-modèle SIMBA définit donc un certain nombre d'éléments qui permettent de sauvegarder les dépendances exprimées dans les modèles conceptuels et qui ne peuvent être reproduites automatiquement dans le modèle du prototype. Ces éléments permettent de construire un second modèle, le modèle miroir (à ne pas

confondre avec le composant miroir). Le modèle miroir exprime, dans une terminologie proche du modèle SIMBA et en reprenant les éléments génériques de ce dernier (composants, interfaces, attributs et connexions), le(s) modèle(s) conceptuel(s) en entrée. Toute dépendance exprimée dans ces derniers est alors retranscrite sous forme d'interfaces et connexions correspondantes. Un tel modèle est montré dans la figure 3.4 tel que généré à partir de l'exemple du système développé dans le paragraphe précédent. Le modèle miroir est sauvegardé séparément du modèle principal des prototypes, afin de ne pas encombrer l'espace de modélisation principal. Avec GME, le modèle miroir utilise un aspect séparé de l'environnement de modélisation (figure 3.5). Le modèle miroir est primordial à la définition de certains outils et règles associés à SIMBA et

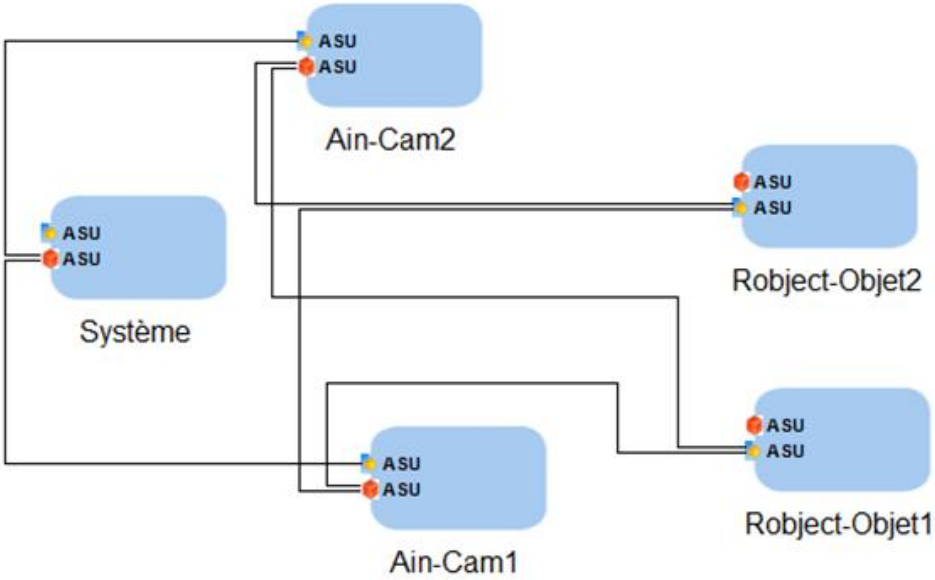

FIGURE 3.4 – Modèle miroir

que nous présentons dans le paragraphe suivant.

3.2 Règles et outils

Les règles et outils associés à SIMBA complètent la réponse apportée par le méta-modèle en assistant les différentes tâches et transitions du processus de développement. La combinaison des règles et des transformations de modèles permet une évolution cohérente et fluide des prototypes simulés vers les systèmes finaux.

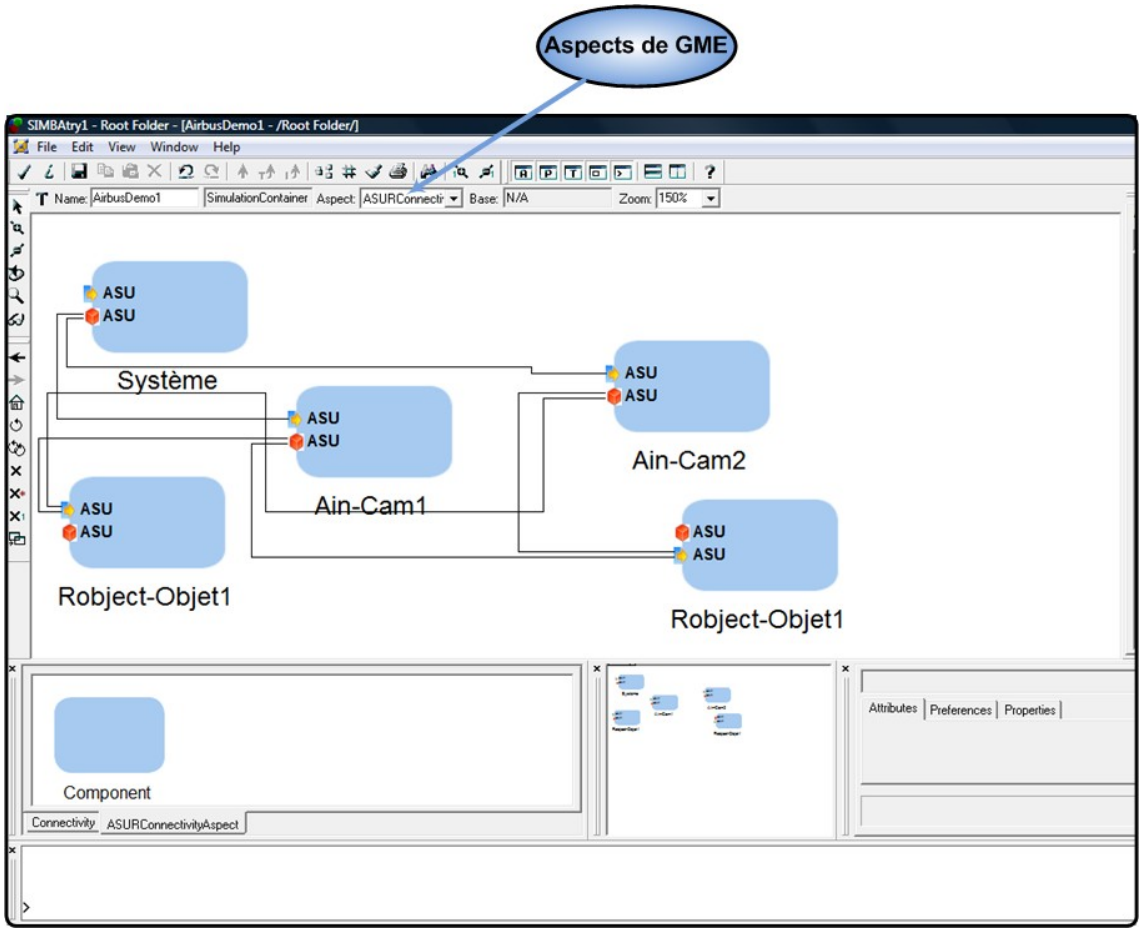

FIGURE 3.5 – Séparation des aspects dans GME

3.2.1 Règles

Les règles associées à SIMBA opèrent sur les éléments du méta-modèle afin de garantir le respect de certaines contraintes liées aux modèles de composants en général, ou à la méthodologie de prototypage incrémental par simulation. Ces règles maintiennent l'intégrité des modèles des prototypes en garantissant la cohérence des dépendances entre composants, et renforcent la séparation entre les services de SIMBA et les aspects fonctionnels des prototypes. Les règles sont indépendantes des modèles conceptuels ou des plateformes d'implémentation et font partie de la spécification du méta-modèle SIMBA.

3.2.1.1 Règle 1 : séparation des gestionnaires

Cette règle renforce la séparation entre aspects simulés et fonctionnels d'un prototype. Malgré le fait que le GM défini et implémenté dans le cadre de cette thèse rend difficile son utilisation en tant que partie fonctionnelle du prototype, d'autres implémentations peuvent éventuellement ne pas suivre les mêmes contraintes et rajouter par exemple des interfaces supplémentaires. Le problème se pose alors lors de l'automatisation de certaines phases du processus de développement comme par exemple supprimer, remplacer ou rajouter des gestionnaire graphiques. Il s'agit alors de pouvoir garantir que le GM, quelque soit l'implémentation particulière utilisée, ne joue pas de rôle dans le système autre que celui de la stricte représentation graphique. Cette règle interdit donc au gestionnaire graphique d'exposer d'autres interfaces que l'interface de simulation. Cette interface, rappelons-le, ne retourne pas de données dans le sens du GM vers les composants fonctionnels. La règle permet néanmoins aux GM d'exposer l'interface de sélection.

3.2.1.2 Règle 2 : séparation des horloges

La deuxième règle concerne le composant horloge et interdit à ce dernier de se connecter à un autre composant que le GM. Ceci évite son intégration en tant que composant de synchronisation dans le schéma fonctionnel du système et permet de le supprimer dans les phases finales de l'évolution au même titre que le GM, sans affecter les aspects fonctionnels du système.

3.2.1.3 Règle 3 : conformité des prototypes avec les choix conceptuels

A toute phase de l'évolution du prototype vers le système final, le modèle SIMBA du prototype doit rester conforme aux choix identifiés dans les modèles conceptuels. Cette vérification peut dépendre d'informations spécifiques au modèle conceptuel utilisé, et donc nécessiter des outils dépendants de ce dernier. Néanmoins, il est possible de définir certaines contraintes génériques telles la préservation des dépendances entre composants ou le nombre de composants exprimés dans les modèles.

Cette règle exploite les informations détenues par le modèle miroir pour comparer et valider les dépendances des composants. Dans les cas où des entités du modèle conceptuel n'auraient pas d'équivalent logiciel dans SIMBA (tels les objets physiques non simulés), cette règle signale

simplement les potentiels conflits ou incompatibilités, sans pour autant empêcher la poursuite du processus de développement.

3.2.1.4 Règle 4 : cohérence des services de simulation

Les services de SIMBA ne sont généralement pas décrits dans les modèles conceptuels puisqu'ils dépendent uniquement de la démarche de prototypage par simulation. Ces entités n'ont pas non plus vocation à subsister dans les phases d'implémentation finales. Ces services doivent alors être gérés par les outils de modélisation de façon transparente durant les phases de prototypage. Cette règle assure la bonne gestion des dépendances des services et artefacts de simulation selon quatre critères :

- la symétrie des connexions des composants simulés en cas de présence de plusieurs gestionnaires graphiques.
- la bonne instantiation des attributs de simulation.
- l'unicité des identifiants des représentations graphiques associés aux composants fonctionnels (hors modélisation dynamique de l'environnement).
- la détection d'un besoin éventuel en composants miroirs (quand celui-ci est détectable)

Cette règle peut être associée à des interpréteurs logiciels qui permettent de connecter automatiquement les interfaces de simulation, en cas d'instantiation de nouveaux gestionnaires graphiques par exemple.

3.2.1.5 Règle 5 : respect des règles génériques

Cette règle assure la conformité des modèles de systèmes vis à vis des contraintes générales liées aux composants, telles la compatibilité des interfaces connectées (émission ou publication et types des interfaces), l'instanciation des attributs, etc.

Malgré l'apport de ces règles en terme d'assistance à la modélisation et au prototypage, certaines optimisations et assistances requièrent une prise en compte des modèles de conception et des plateformes d'implémentation. Ces optimisations sont alors prises en charge par un certain nombre d'outils associés à SIMBA en tant qu'interpréteurs et transformations de modèles.

3.2.2 Transformations et interpréteurs de modèles

Ces outils permettent d'automatiser certaines phases du processus de développement entre les propositions conceptuelles initiales et le déploiement du système final sur sa plateforme logicielle et matérielle. Les transformations et interpréteurs dépendent à la fois de SIMBA et des modèles et platformes externes sur lesquels ils opèrent. Ils fluidifient et facilitent les différentes transitions et étapes du processus et minimisent le nombre d'erreurs dues à des mauvaises manipulations. Leur dépendance vis à vis des outils externes rend difficile leur présentation en dehors du contexte de ces outils. Néanmoins, la méthodologie demeure sensiblement identique et il est facile de l'adapter lors d'un changement de plateforme. Nous présentons donc trois outils : la Transformation ASUR vers SIMBA (TAS), l' Interpréteur de Descripteurs CCM (IDC) et le Générateur de l'Assemblage de Composants (GAC).

3.2.2.1 Transformation ASUR vers SIMBA

La TAS identifie dans ASUR les entités et relations du modèle et génère un modèle SIMBA correspondant. Puisque la caractérisation d'ASUR des types de données échangées ne permet pas d'identifier sans ambiguïté les composants logiciels et les interfaces fonctionnelles, le modèle SIMBA initialement généré ne contient que des composants vides et non connectés. La figure 3.6 montre un tel diagramme initial associé au système de localisation par caméras déjà présenté. Cependant, les informations sur les composants SIMBA ainsi que leurs dépendances sont sauvegardées dans le modèle miroir généré par la TAS. En cas de besoin, la TAS peut instancier des attributs supplémentaires dans le modèle miroir.

La supposition qu'un composant ASUR (ou d'un autre modèle de conception) peut être traduit directement en un composant logiciel est assez forte. En effet, un composant ASUR qui représente le système informatique central peut correspondre à un nombre relativement grand de composants logiciels. Certes la TAS étant conçue et implémentée conjointement avec les experts des modèles de conception, elle peut être adaptée afin de générer un nombre variable de composants pour chacune des entités ASUR, et selon les caractéristiques de ces dernières. Cependant, la perte d'un lien de "un à un" entre les entités des modèles conceptuels et celles de SIMBA rend difficile la vérification de la conformité de ce dernier avec les choix de conception,

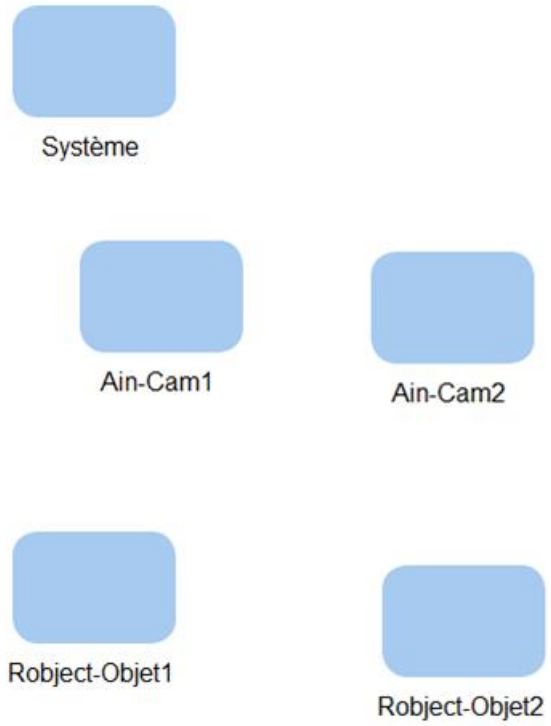

FIGURE 3.6 – Modèle vide initial

un pré-requis pour la bonne conduite des cycles itératifs incrémentaux. On peut néanmoins observer que, indépendamment du nombre ou de la complexité des composants logiciels générés à partir du modèle conceptuel, les propriétés de l'interaction ne doivent dépendre en principe que des entités définies dans ce dernier. En particulier, la multiplication dans les modèles d'implémentation du nombre de composants ainsi que de leurs interdépendances constituent un détail de l'implémentation qui n'influence pas les propriétés de l'interaction. La méthodologie adoptée par SIMBA qui consiste à enfouir les implémentations des composants derrière les interfaces de ces derniers reste applicable. Le composant système de notre exemple peut alors être conçu et assemblé séparément, puis intégré au sein du répertoire de composants de SIMBA en tant que composant unique. Les méthodologies basées sur les composants permettent la définition d'un tel composant " composé ", appelé alors "méta-composant". SIMBA n'a pas besoin de reconnaître un tel composant et n'a pas vocation à en représenter la structure interne.

3.2.2.2 Interpréteur de Descripteurs CCM

En fonction du modèle utilisé pour l'implémentation, les informations relatives aux composants (interfaces et attributs) doivent être importées dans le modèle SIMBA. Ceci permet de spécifier les types et caractéristiques logiciels des composants générés à partir des modèles conceptuels. Les informations importées doivent permettre d'associer de façon non ambiguë les composants d'implémentation logiciels aux composants SIMBA, sans modifier l'aspect générique du modèle SIMBA. Par conséquent, seuls les noms des types de composants, interfaces et attributs sont importés, et viennent compléter les composants générés à partir du modèle de conception. En l'absence d'ambiguïtés, l'IDC profite des informations contenues dans le modèle miroir pour connecter les interfaces correspondantes des composants dont les dépendances sont exprimées dans le modèle miroir. Si les composants choisis exposent l'interface de simulation, ils sont alors promus en composants simulés. Les ambiguïtés, lorsqu'elles existent, doivent être résolues manuellement par les développeurs qui continuent néanmoins de profiter des règles encadrant la modélisation des prototypes.

En l'absence d'un gestionnaire graphique, l'IDC instantie un GM et connecte les interfaces de simulation des composants simulés. Le lien entre le GM (qui est un élément du méta-modèle SIMBA) et son implémentation (donc le composant logiciel qui l'implémente) doit être spécifiée manuellement. Le développeur peut également rajouter d'autres gestionnaires pour représenter différents points de vue.

Dans le cas de la plateforme CCM (*CORBA Component Model*), l'IDC parse la description XML contenue dans les descripteurs des composants (*.CCD*). La notion de méta-composant utilisée pour enfouir la structure interne des composants complexes tient toujours puisqu'elle est supportée par CCM. L'IDC n'importe alors que les détails (interfaces et attributs) externes du méta composant.

3.2.2.3 Générateur de l'Assemblage de Composants

Une fois complété, le modèle SIMBA doit être exporté en un modèle spécifique à la plateforme d'implémentation. Le GAC opère une dernière passe de vérifications basées sur les règles définies précédemment (lorsque cette vérification n'est pas automatique), et établit le lien d'un côté entre

les noms des types de composants, interfaces et attributs importés par l'IDC, et de l'autre les implémentations des composants portant les mêmes noms. Le GAC génère ensuite les artefacts du prototype nécessaires à son déploiement.

Dans le cas de CCM, le modèle spécifique correspond à un assemblage de composants, une archive ZIP contenant la description XML (*.CAD*) de l'assemblage ainsi que les implémentations des composants individuels faisant partie du système. Cet assemblage peut être déployé par des outils de la plateforme comme la console Qedo présentée dans le chapitre 1 .

Les règles de SIMBA sont décrites, aussi souvent que leur complexité le permet, en OCL [OCL02], un langage de description de contraintes. Les outils sont enregistrés en tant qu'interpréteurs de GME, des composants logiciels ayant accès au modèle SIMBA via des interfaces définies par l'environnement de modélisation. Les implémentations de ces composants sont réalisées en C++. Ces outils articulent alors un processus de développement dirigé par les modèles. La partie suivante illustre ce processus ainsi que les rôles des différents acteurs impliqués.

3.3 Processus de développement

La méthodologie proposée définit un processus de développement dirigé par les modèles partant de la conception des SIM jusqu'au déploiement du système final (figure 3.7). Ce processus n'est pas entièrement automatisé et s'articule autour de trois types d'acteurs principaux : le concepteur de l'interaction, l'intégrateur du système et les développeurs de composants. Nous présentons et illustrons ces différents rôles dans les paragraphes suivants.

3.3.1 Les concepteurs de l'interaction

Ces concepteurs sont responsables de la définition du système et en particulier de la conception de l'interaction des utilisateurs (1). Ils s'appuient sur un modèle de l'interaction en coordination avec les experts du domaine et après une étape de recueil des besoins. Une fois les prototypes générés, ils supervisent l'interaction afin de valider les choix de conception (12). Dans un processus s'appuyant sur les outils utilisés dans cette thèse, le rôle du concepteur est d'établir le diagramme ASUR et de valider les choix de conception décrits dans ce diagramme. Cette étape est accomplie dans l'éditeur GUIDE-ME qui valide les contraintes et exporte le diagramme en

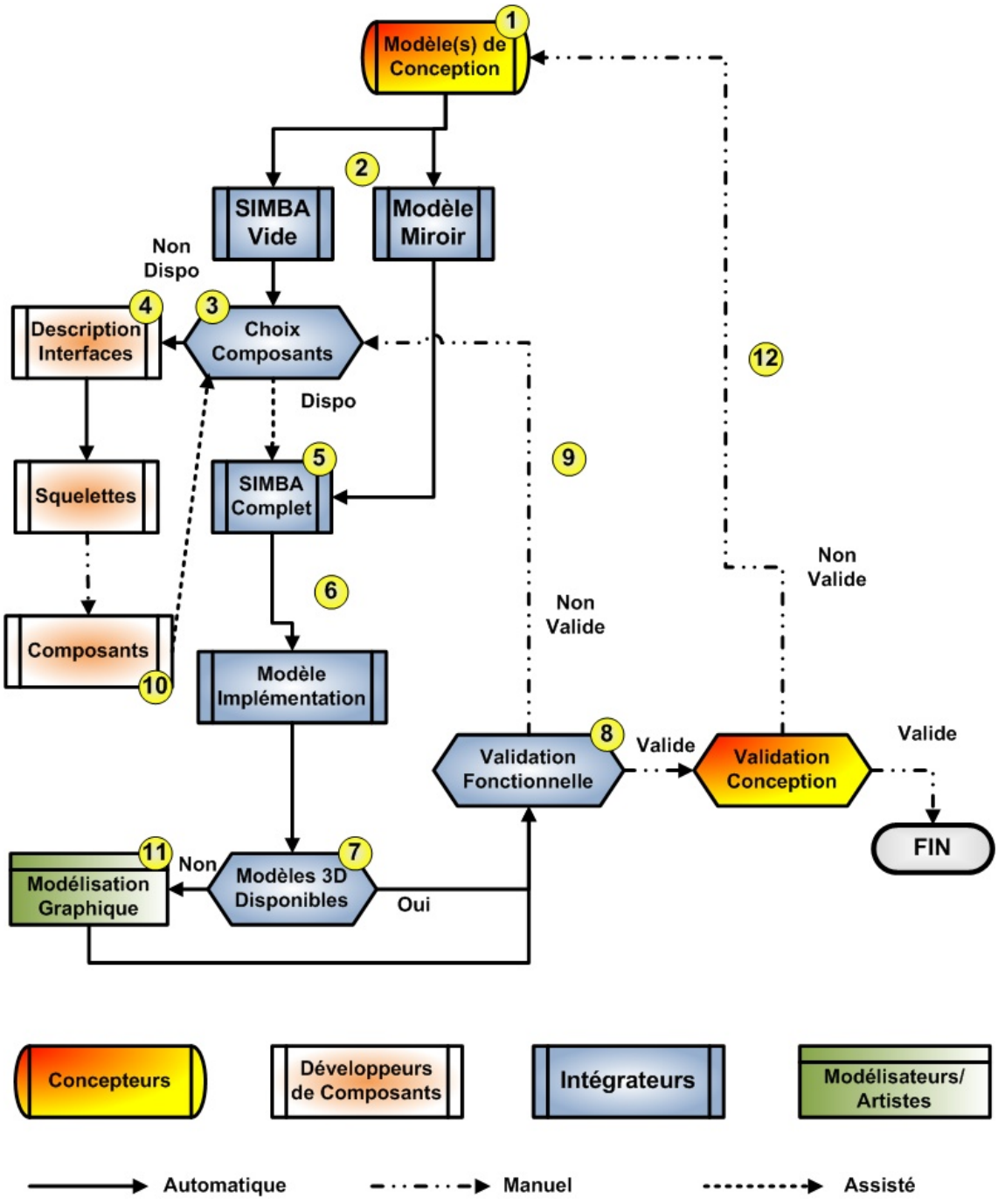

FIGURE 3.7 – Processus de développement

un format XML. La figure 3.8 montre un diagramme ASUR modélisé dans GUIDE-ME.

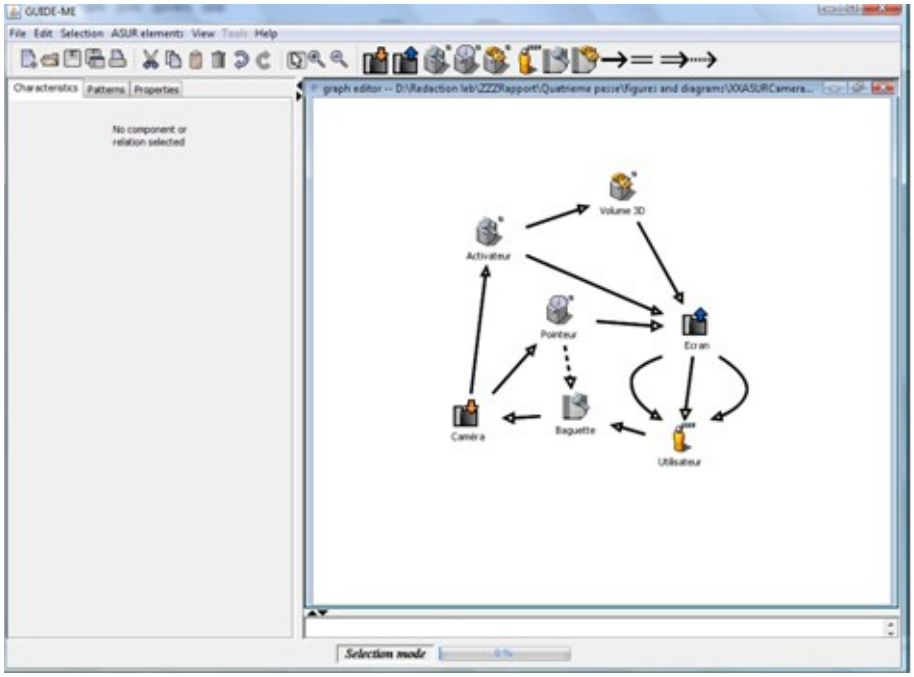

FIGURE 3.8 – Modélisation ASUR dans GUIDE-ME

Le concepteur peut alors lancer GME et définir un nouveau projet basé sur le méta modèle SIMBA. La TAS associée effectue la traduction du modèle ASUR en un modèle SIMBA. A ce stade, ce modèle ne contient que des composants vierges (figure 3.9). Cette description est plus tard affinée par les intégrateurs du système. Le schéma initial est cependant conservé dans le modèle miroir (figure 3.10) pour valider la conformité du diagramme final avec les choix de conception.

3.3.2 Intégrateurs des systèmes

Les intégrateurs du système sont responsables de conduire les étapes ultérieures du cycle de développement itératif. Ceci inclut notamment la mise en place des modèles des prototypes et des systèmes finaux via la traduction du modèle SIMBA en un modèle d'implémentation spécifique à la plateforme. Ils établissent le lien entre les entités du modèle de l'interaction et les composants matériels et logiciels nécessaires (3). Selon le nombre d'outils mis à leur disposition et la complexité du domaine d'application du système, ils peuvent faire appel à des experts supplémentaires (4) et (11) dans différents domaines tels la conception graphique ou le domaine

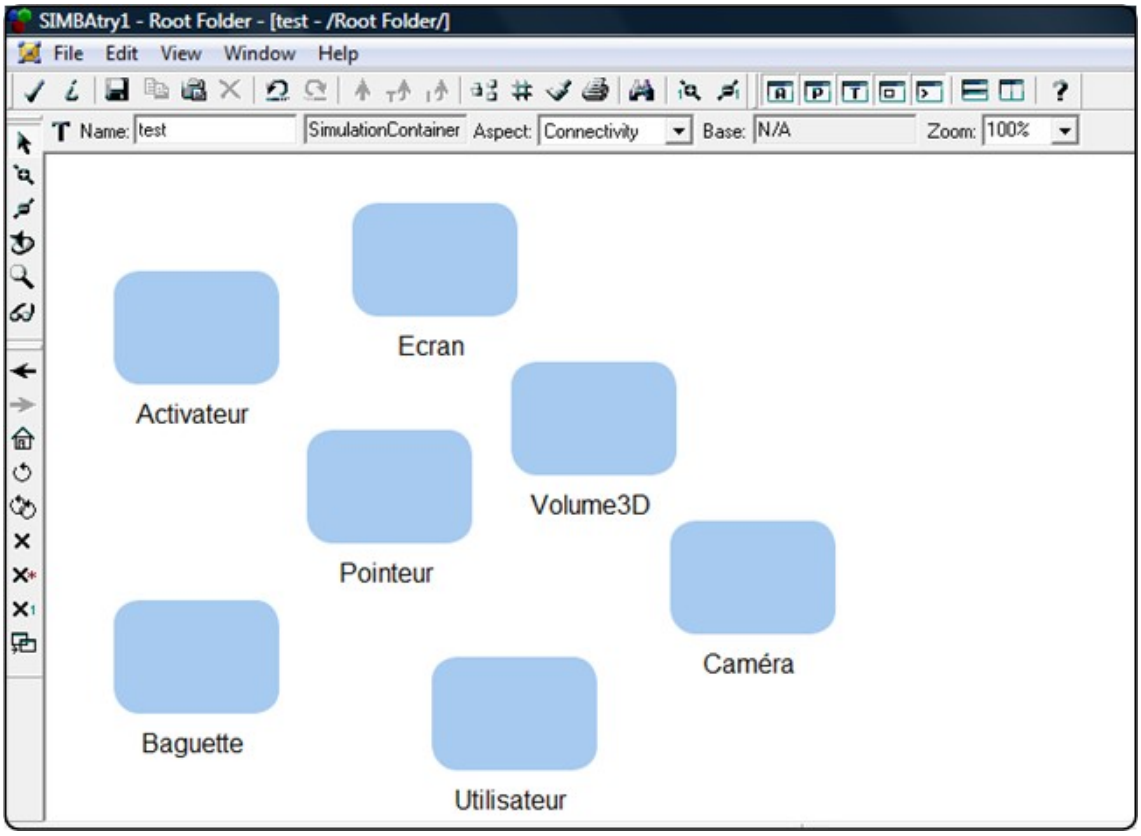

FIGURE 3.9 – Modèle vide initial

d'application du système.

Avec GME, l'intégrateur du système identifie les composants susceptibles de remplir les rôles identifiés sur le diagramme ASUR. L'IDC est alors utilisé pour importer une description abstraite des interfaces et attributs des composants, mettant à jour le modèle SIMBA. A ce stade, les composants exposent leurs interfaces et leurs attributs (figure 3.11). Si un des composants expose l'interface de simulation, l'interpréteur génère un gestionnaire graphique et connecte toutes les interfaces de simulation à l'interface du GM correspondante. En l'absence d'ambiguïtés, cet interpréteur génère aussi les connexions fonctionnelles correspondantes. L'intégrateur complète alors les connexions, initialise les attributs des différents composants et instancie, au besoin, des composants GM supplémentaires (5). Après validation des contraintes définies dans le méta modèle de SIMBA, le GAC génère l'assemblage de composants (6). Les composants, interfaces et attributs SIMBA sont traduits en leurs équivalents CCM. L'interpréteur génère aussi tous les artefacts nécessaires au déploiement du prototype via les outils standard de CCM. L'intégrateur

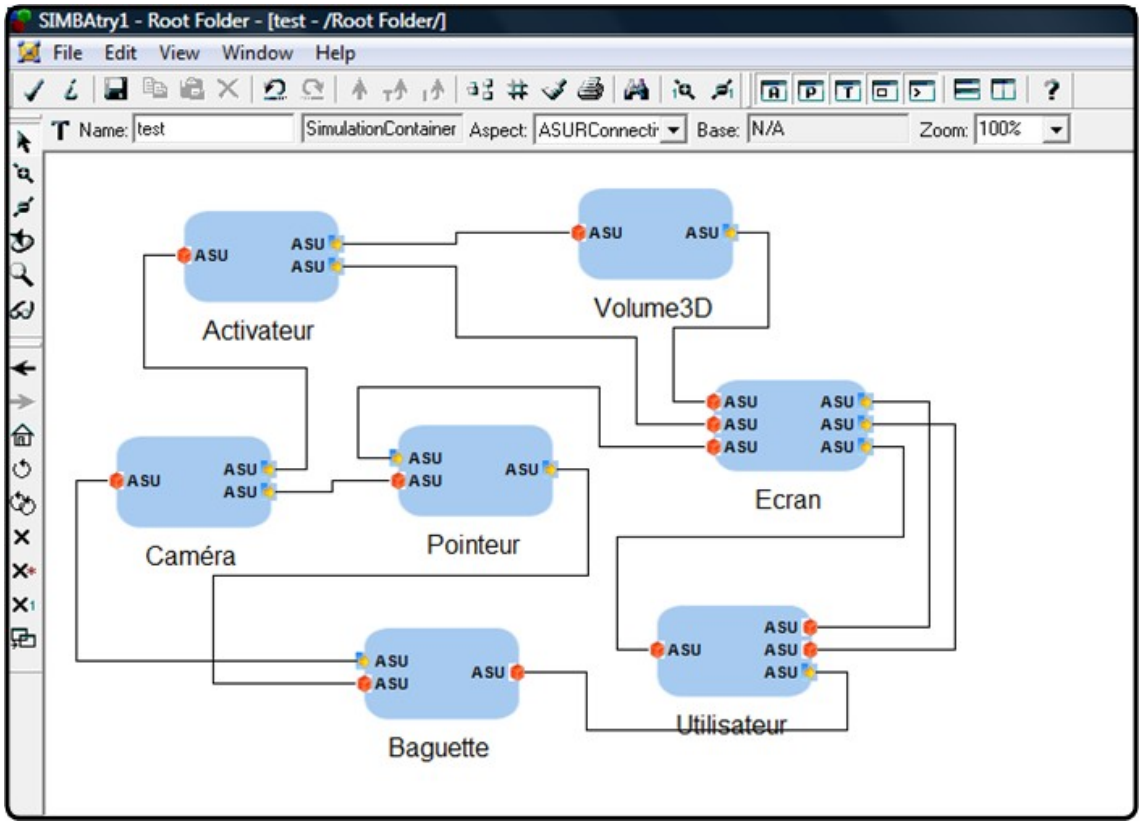

FIGURE 3.10 – Modèle miroir

conduit ensuite, au besoin, le cycle itératif. Le passage d'un prototype entièrement simulé à des prototypes mélangeant simulé et non simulé est intuitif. Il suffit alors de remplacer les composants logiciels de simulation par des composants non simulés, ou dans certains cas de supprimer uniquement la connexion de ces composants au GM.

3.3.3 Les développeurs de composants

Quand les composants requis pour répondre aux spécifications des modèles conceptuels ne sont pas disponibles, les développeurs de composants ont la tâche de coder les composants individuels pour répondre aux besoins fonctionnels de l'application, émanant de son domaine et de son contexte d'application. L'intégrateur du système fournit alors aux développeurs les détails de la fonctionnalité requise ainsi que les interfaces des composants à développer. Une fois développés, ces composants peuvent être ajoutés au répertoire de composants de la plateforme ou du projet pour servir dans des projets ultérieurs.

En reprenant l'implémentation de SIMBA basée sur les outils déjà décrits, les développeurs

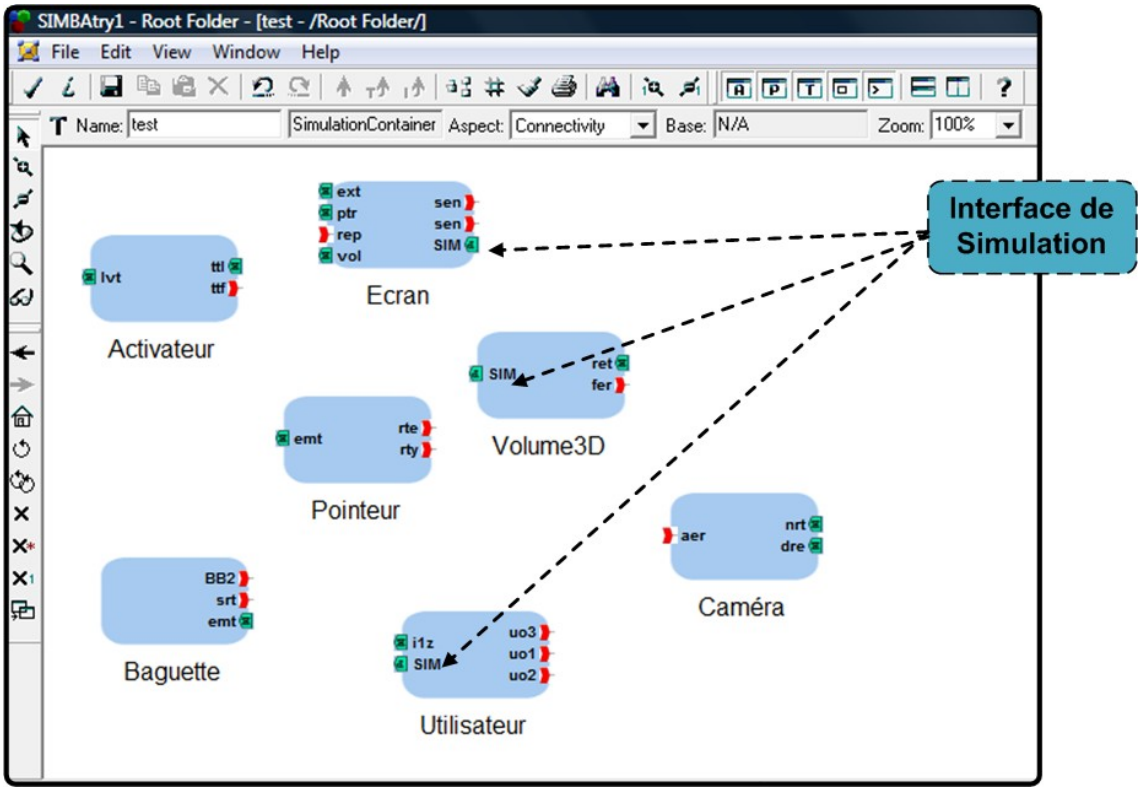

FIGURE 3.11 – Composants complétés, avant l'étape de connexion

bénéficient des outils fournis par les plateformes de développement de composants CCM. Ces plateformes se chargent de la génération des squelettes des composants ainsi que de leurs parties génériques (*glue code*). L'implémentation d'une fonctionnalité consiste alors à placer le code utile (applicatif, indépendant de CCM) à des emplacements réservés dans les fichiers d'implémentation générés automatiquement. La compilation est elle aussi prise en charge par ces plateformes, générant ainsi des composants distribués.

Les interfaces des composants d'interaction, horloge et gestionnaire graphique sont standardisés ce qui permet aux développeurs de les implémenter directement dans leurs composants. Cette standardisation permet de réduire l'effort nécessaire au développement de nouveaux modules.

3.3.4 Les acteurs divers

Jusqu'alors, certains aspects du développement ont été ignorés, notamment la création de contenu. En effet, la plateforme SIMBA ne fournit aucun utilitaire ou service spécifique à la création du contenu graphique des SIM. Cependant, la simulation nécessite la création de repré-

sentations graphiques pour les différents composants simulés, ainsi que de l'environnement de la simulation. Le développement de représentations graphiques complexes nécessite l'utilisation d'outils de création de contenu 3D plus évolués et parfois l'intervention d'experts en modélisation graphique.

Dans les cas d'étude présentés dans cette thèse, l'implémentation du GM utilise Delta3D. La création des environnements graphiques est alors assurée par STAGE, l'éditeur de scènes 3D de Delta3D. Son interface est intuitive (chargement et positionnement d'objets) et ne requiert pas d'expertise particulière.

Le prochain chapitre présente deux cas d'étude qui ont permis de mettre en application la méthodologie et les outils de SIMBA. Ces cas d'étude nous permettent par la suite de tirer un certain nombre de conclusions ainsi que de proposer des améliorations possibles sous forme de perspectives.

4

Applications

La méthodologie de SIMBA a été appliquée à deux cas d'étude, qui nous ont permis de mettre à l'épreuve le prototypage incrémental par simulation des systèmes interactifs mixtes dans deux contextes applicatifs distincts. Le prototypage du musée décrit dans [DBT08] applique la méthodologie de développement de SIMBA à l'opérationnalisation de la conception d'un système expérientiel. Le prototypage d'un système de maintenance aéronautique nomade [DAJ07] teste la plateforme de prototypage vis à vis des contraintes liées au contexte industriel de l'application et à l'intégration de services applicatifs. Néanmoins, l'objectif principal de la simulation de ce deuxième système est de démontrer la réutilisabilité et la flexibilité de configuration des composants de SIMBA pour la génération rapide de prototypes différents. Un troisième cas d'étude mettant en oeuvre la simulation dans le contexte d'interactions plus complexes est actuellement en cours de développement et sera abordé brièvement dans les perspectives de cette thèse.

4.1 Musée augmenté

Le musée augmenté constitue un cas d'étude qui permet d'illustrer les différentes étapes de la méthodologie de développement. Il s'agit d'assister un visiteur dans l'exploration des oeuvres d'art exposées dans un musée en lui fournissant un support d'information personnalisé. Le musée se compose de deux salles d'exposition. A l'entrée, les visiteurs peuvent s'équiper de matériel d'assistance classique (traducteurs, guides auditifs, plans, etc.). La grande diversité des oeuvres

d'art ainsi que celle des profils des visiteurs (langues, compétences, etc.) rend complexe tout support matériel personnalisé. En cas de visites de groupe ou de besoins en informations supplémentaires, le musée met à disposition de ses visiteurs des guides répartis dans l'espace. Il demeure parfois difficile, en cas d'affluence par exemple, de localiser et de contacter les guides disponibles.

4.1.1 Assistance

L'assistance consiste à proposer un système informatique intégré à l'environnement des visiteurs et couplé à des outils d'aide personnalisée. En entrant, les visiteurs sont équipés d'un PDA relié au système du musée via une liaison sans fil. Les visiteurs sont également équipés de motifs visuels assurant leur suivi par des caméras disposées dans les différentes salles d'exposition. Pour des raisons légales et d'usage, le suivi des visiteurs peut être désactivé par ces derniers à tout moment. Le suivi des visiteurs est également désactivé hors des salles principales et ce de façon automatique. Les utilisateurs peuvent choisir entre différents profils ajustables via des paramètres tels la langue ou le niveau d'expertise du visiteur. En fonction du profil et de la position du visiteur, le PDA affiche les détails relatifs aux oeuvres visualisées. Il permet aussi de localiser les guides, de vérifier leur disponibilité ainsi que de les contacter via le réseau du musée auquel sont connectés les PDA des visiteurs et des guides. La conception de ce système dans ASUR identifie plusieurs tâches principales. Deux de ces tâches, l'accès aux informations des oeuvres d'art en fonction de la localisation et l'entrée en contact avec les guides, sont représentées dans les figures 4.1 et 4.2.

Dans la figure 4.1, ASUR met en évidence la présence d'un utilisateur et d'un artefact physique (Rtool) représentant un guide. Ces deux entités physiques sont localisées (=>) par des adaptateurs (Ain) que les caractéristiques ASUR décrivent entre autre par leur médium (visuel) et leur technique de capture (CCD) : il peut donc s'agir de caméras dont la propriété de "localisation" dans ASUR permet de préciser l'emplacement dans deux salles distinctes du musée. Ces localisateurs modifient l'objet de la tâche (Sobject), i.e. le plan du musée et ses ressources, perceptibles sur un adaptateur (Aout) que les caractéristiques ASUR rendent compatibles avec un écran. Cet écran doit être tenu par l'utilisateur (=). Le modèle ASUR stipule enfin que cet écran doit être physiquement couplé (=) à un adaptateur (Ain) permettant à l'utilisateur de

modifier les paramètres d'affichage (Stool) des informations ajoutées au plan du musée.

Le diagramme de la figure 4.2 décrit un diagramme similaire avec en plus la possibilité de contacter les guides via les PDAs qu'ils ont à disposition. Les informations envoyées par les visiteurs via l'adaptateur en entrée couplé au PDA sont alors acheminées par le système informatique.

Le développement et le déploiement de ces configurations nécessite l'acquisition et le paramétrage des caméras de suivi, des systèmes de gestion de réseau ainsi que la mobilisation d'un espace et d'un nombre d'utilisateurs importants pour les phases de tests. La simulation offre alors une alternative intéressante.

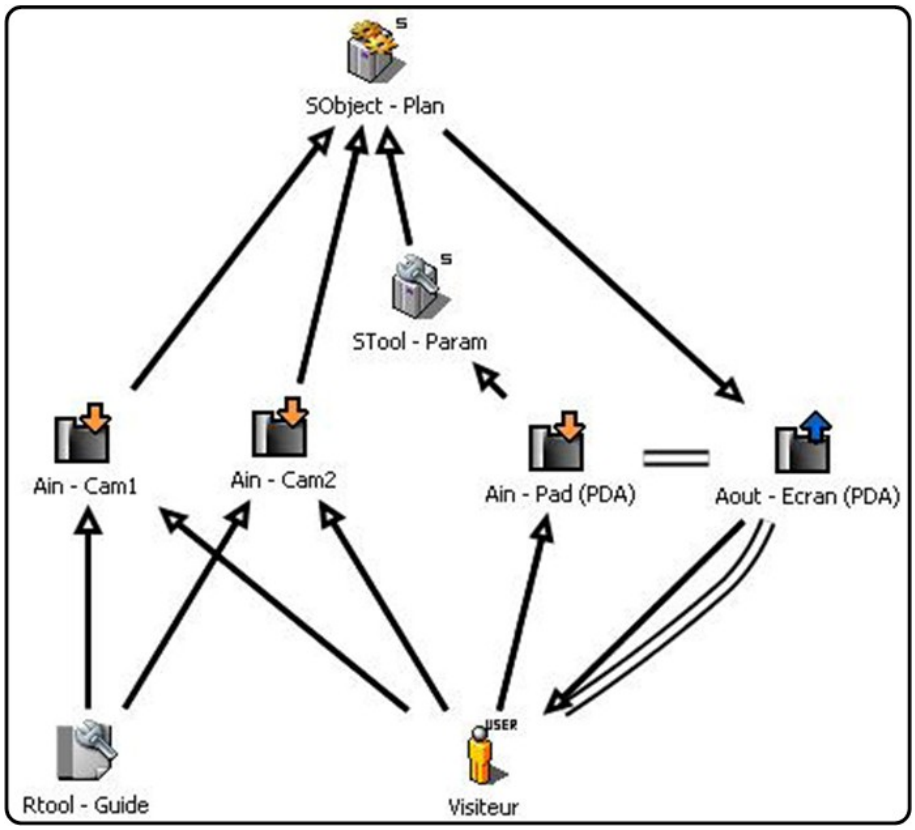

FIGURE 4.1 – Modèle ASUR par rapport à la tâche de localisation

4.1.2 Simulation

La simulation du système permet de valider certains des choix de conception comme le nombre de caméras nécessaires et leur positionnement dans l'environnement ainsi que la répar-

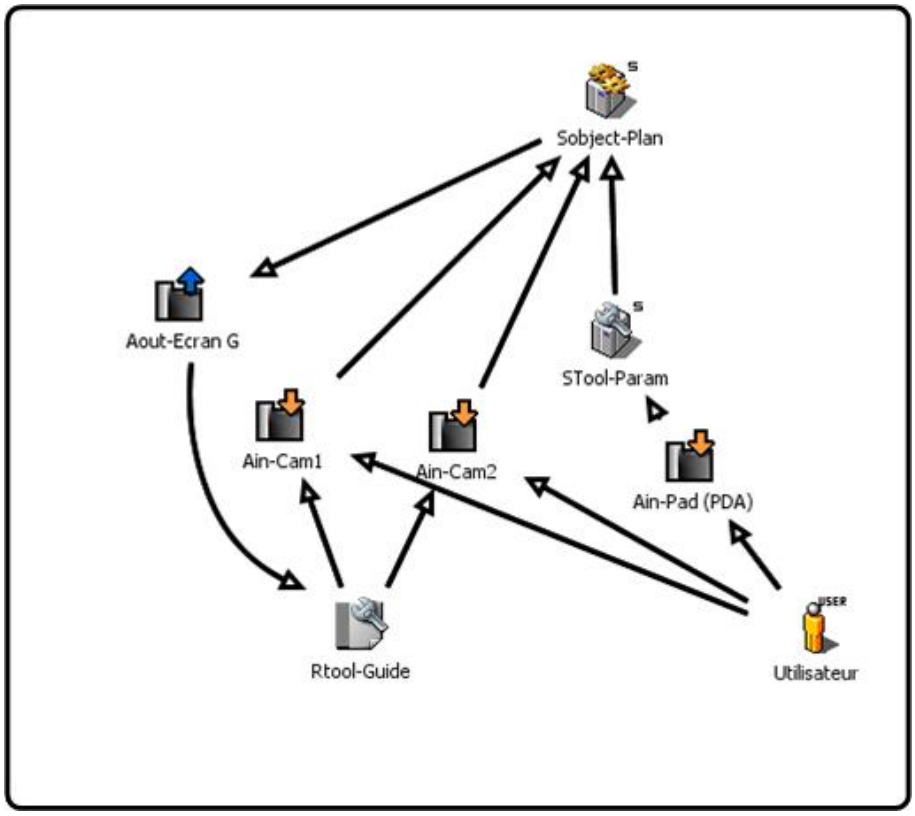

FIGURE 4.2 – Modèle ASUR par rapport à la tâche de contacter le guide

tition des oeuvres d'art et la distribution des guides. Les interfaces des PDAs sont développées séparément et demeurent logées dans des fenêtres séparées. Ainsi, l'environnement de simulation permet de tester les caractéristiques des interfaces en parallèle de celles de l'environnement simulé. La communication entre ces interfaces et les autres éléments du système est assurée par la plateforme de prototypage via les connexions décrites dans les modèles de conception.

4.1.3 Processus de développement

En respectant le processus de développement présenté dans le chapitre précédent, la première phase du prototypage consiste à générer le modèle SIMBA initial à partir des diagrammes ASUR. La version actuelle de SIMBA limitant la génération de prototypes à un seul modèle en entrée, nous effectuons une fusion entre les deux diagrammes ASUR ci-dessus. La transformation ASUR vers SIMBA (TAS) génère alors le modèle SIMBA initial (figure 4.3) ainsi que le modèle miroir qui conserve les entités et connexions du modèle ASUR pour une utilisation ultérieure. Ce modèle miroir est montré dans la figure 4.4.

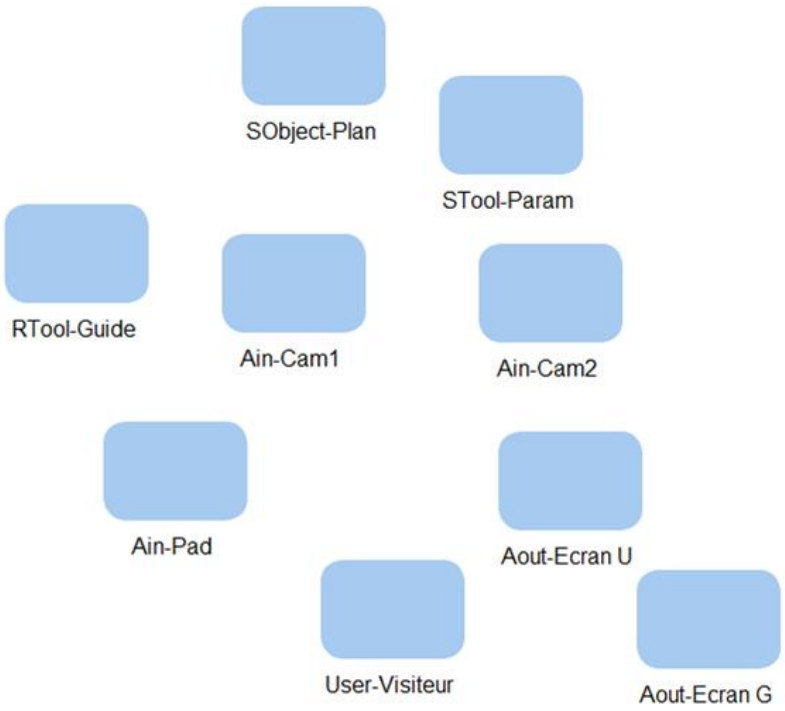

FIGURE 4.3 – Modèle vide initial

La deuxième étape consiste à compléter le modèle initial en associant à chaque composant vide un type de composant logiciel choisi parmi ceux disponibles dans le répertoire du système. Les composants non disponibles sont implémentés en utilisant les outils de la plateforme de composants :

- le composant visiteur est simple puisqu'il est piloté par l'utilisateur réel et ne fait que convertir la position résultante des actions de cet utilisateur en des événements compatibles avec les caméras pour la localisation, et avec l'interface de simulation pour contrôler l'avatar via le GM. Les entrées sont effectuées via une SpaceMouse [3DC02] gérée par un composant d'interaction de la CAU rajouté manuellement au modèle.
- le guide est un composant " intelligent " qui se déplace selon un algorithme pseudo-aléatoire jusqu'à ce qu'il soit contacté par un visiteur. Le guide transmet également sa position au GM et aux caméras.
- les caméras récupèrent les positions et les comparent à leurs champs de vision. En cas de

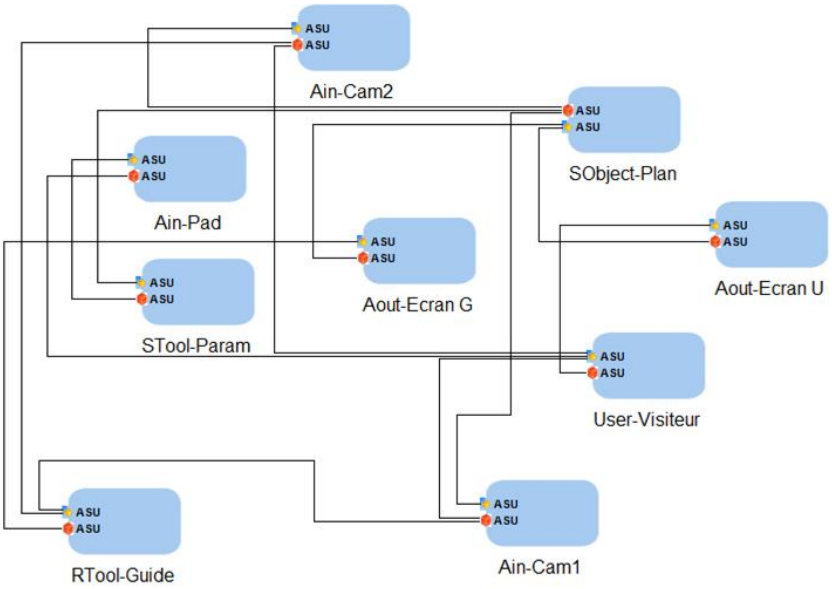

FIGURE 4.4 – Modèle Miroir généré par la TAS

correspondance, elles transmettent les identifiants et les positions au système informatique qui établit le lien entre ces positions et celles des oeuvres d'art. Il transmet ensuite les informations nécessaires aux composants de mises à jour des interfaces graphiques appropriées.

– les composants de gestion des interfaces graphiques utilisent FLTK [FLT08] pour la gestion des éléments de ces interfaces. En effet FLTK permet une mise à jour des interfaces à partir de processus parallèles, une propriété utile puisque cette mise à jour est commandée par la plateforme de composants (inversion de contrôle).

Une fois tous les composants nécessaires identifiés, l'interpréteur de descripteurs CCM (IDC) extrait les informations relatives à leurs interfaces et attributs à partir des archives correspondantes, et met à jour le modèle SIMBA. La figure 4.5 montre un modèle SIMBA du prototype simulé tel que complété par l'IDC en conformité avec le modèle miroir et finalisé par l'intégrateur du système. La dernière étape est prise en charge par le GAC qui vérifie le respect des règles de SIMBA et génère l'assemblage de composants.

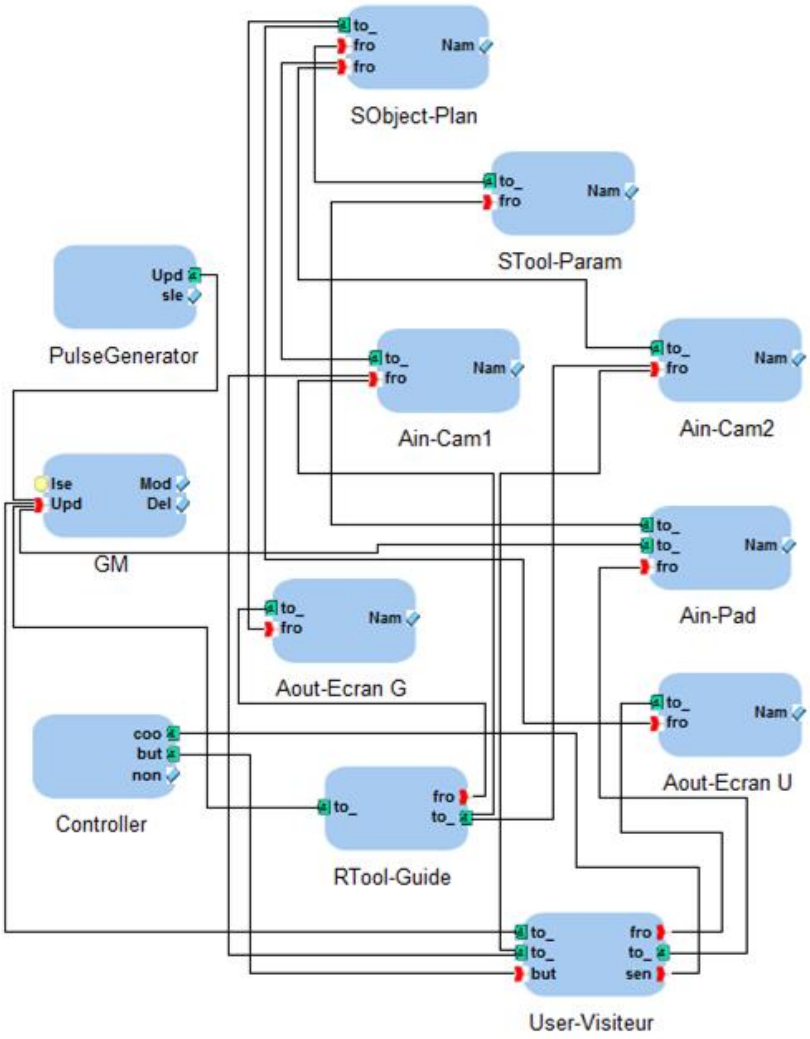

FIGURE 4.5 – Modèle SIMBA complet

4.1.4 Itérations

L'environnement ainsi généré permet de tester l'utilisateur en situation d'interaction avec le système. La modification des champs de vision des caméras, de leur nombre ou de leurs positions demeure facile dans l'environnement simulé. Une fois ces paramètres validés, l'évolution incrémentale permet d'aboutir au système final. Le modèle de la figure 4.6 montre le système à un stade d'évolution intermédiaire. Une des caméras simulées est remplacée par un composant de caméra ARToolkit. Le visiteur est à ce stade "extrait" de la simulation et porte un motif ARToolkit (figure 4.7). Un composant miroir permet à la caméra restée simulée d'assurer le suivi de l'utilisateur réel. Le champ de suivi de cette dernière est aussi modifié pour lui permettre de

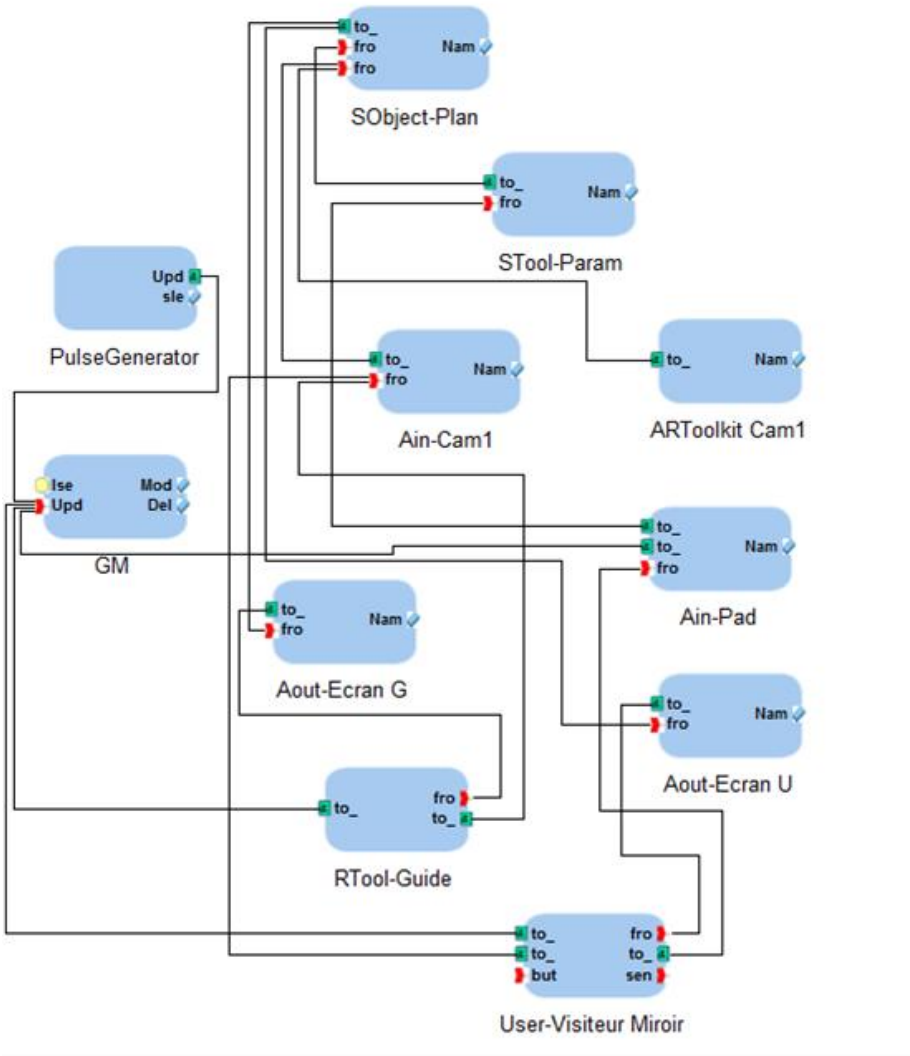

FIGURE 4.6 – Modèle SIMBA complet, avec une caméra non simulée

remplacer la caméra réelle dans le suivi du guide resté simulé. Cette solution quoique ad-hoc, est assez simple à mettre en oeuvre.

Le modèle du système obtenu au bout de l'évolution incrémentale (figure 4.8) n'a pas encore été déployé et testé dans un environnement réel. Les composants Aout de ce système contrôlent les interfaces graphiques sur de vrais PDAs. Le composant Ain-Pad contrôle un certain nombre de dispositifs d'entrée physiques. La figure 4.9 montre une capture d'écran du système décrit dans la figure 4.6 déployé sur une configuration constituée d'un seul hôte utilisant la version 0.9 de QedoCCM. L'interface du PDA est constituée par une fenêtre 2D implémentée en FLTK et un modèle 3D du PDA intégré dans l'environnement virtuel de simulation. Les informations

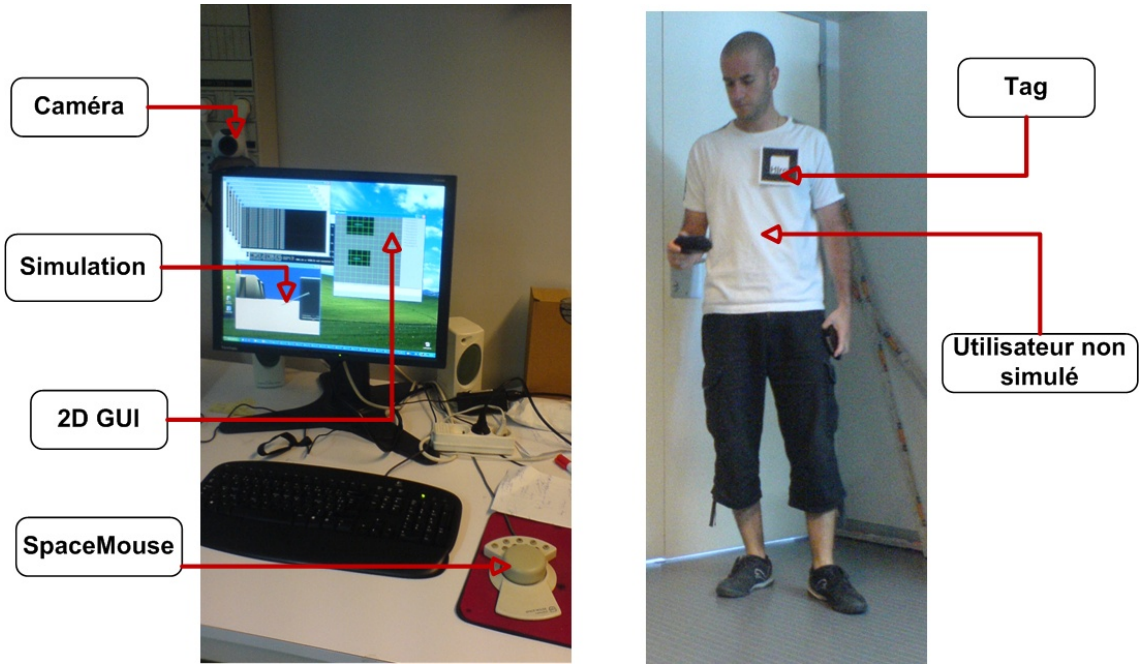

FIGURE 4.7 – Plateforme matérielle faisant tourner la simulation avec une caméra et un visiteur non simulés

relatives aux oeuvres d'art s'affichent dès que l'utilisateur localisé par les caméras s'en approche.

L'interface montre les positions des guides disponibles par rapport aux salles du musée.

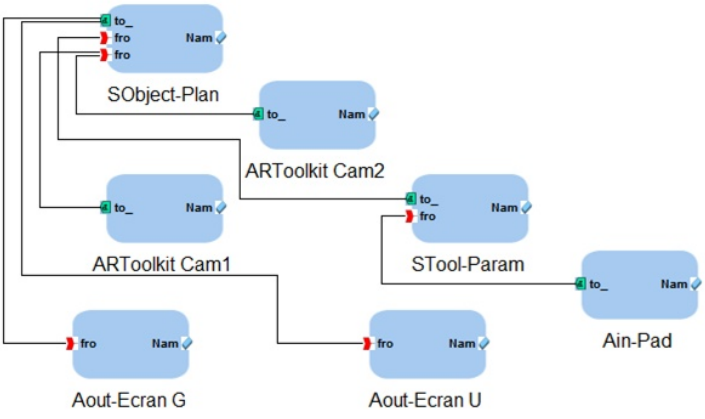

FIGURE 4.8 – Modèle SIMBA final sans simulation

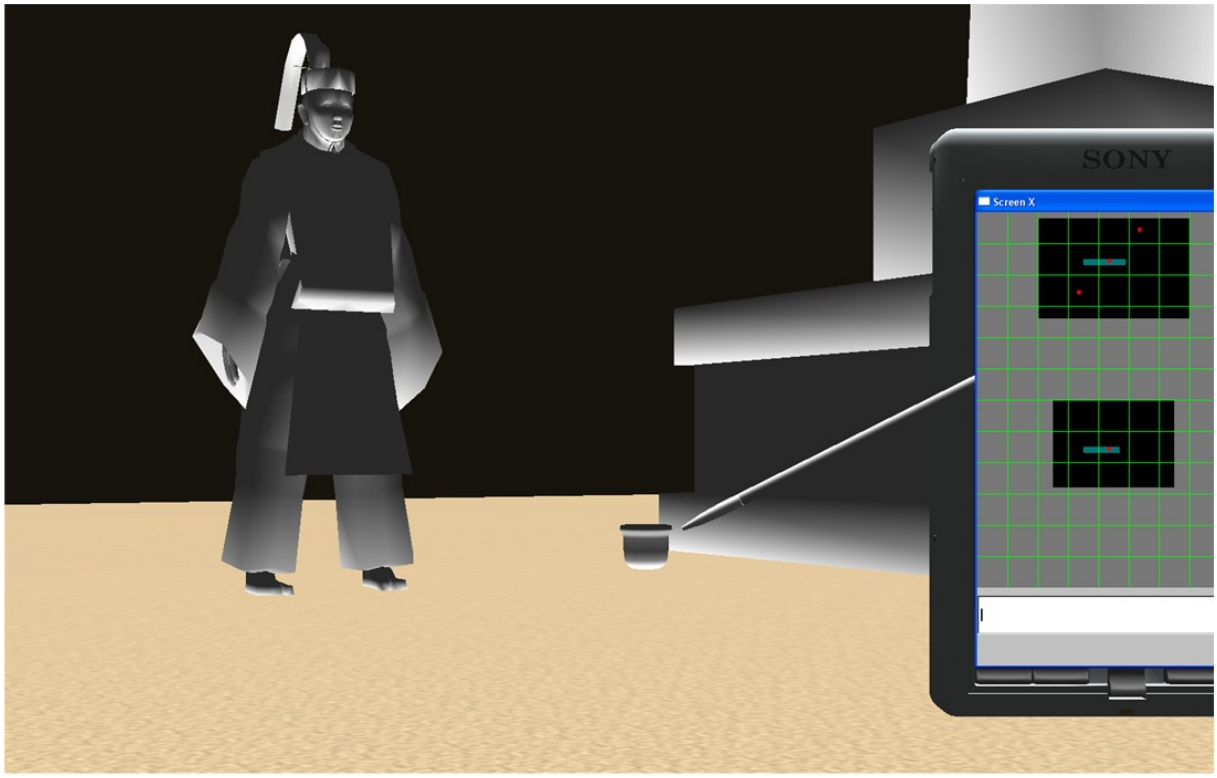

FIGURE 4.9 – Capture d'écran de la simulation à un stade d'évolution intermédiaire

4.2 Système d'assistance à la maintenance aéronautique nomade

Le prototypage par simulation dans ce contexte consiste à opérationnaliser une conception abstraite d'un système de maintenance nomade collaboratif, et à fournir un environnement de test et de validation des aspects liés à la sécurité et à l'intégration d'un tel système dans l'environnement de travail existant.

4.2.1 Contexte général

La disponibilité des avions dépend de la fiabilité et de l'optimisation des systèmes de maintenance utilisés par les opérateurs et les constructeurs. La réduction des délais de maintenance est conditionnée par des réglementations et des procédures nationales et internationales. La diversification de la flotte mondiale pose en plus un certain nombre de défis. D'un côté, les avions âgés requièrent une attention particulière quant à la détection des signes de vieillesse des moteurs et du fuselage. De l'autre, les technologies impliquées dans la construction des appareils récents posent des défis de formation et de documentation conséquents. Les activités de maintenance aéronautique sont naturellement collaboratives. L'opérateur nomade intervient rarement seul et travaille souvent au sein d'une équipe avec laquelle il doit collaborer et communiquer en permanence. Une équipe est composée de plusieurs opérateurs, techniciens et ingénieurs, dont un chef d'équipe. Plusieurs tâches doivent souvent être accomplies pendant la durée d'une rotation où l'équipe opère directement dans le voisinage des avions. De nombreux échanges d'information sont alors nécessaires même si les tâches ou sous-tâches sont allouées individuellement aux opérateurs. Cette communication permet d'échanger les expertises et de synchroniser l'avancement des opérations ainsi que de formuler des demandes d'assistance ou de validation.

4.2.2 Assistance

En vue des complexités inhérentes à la maintenance aéronautique, il demeure essentiel de spécifier un système d'information complet prenant en compte les paramètres de l'environnement et de l'organisation des tâches de maintenance. Le système proposé dans [DAJ07] factorise ces paramètres et se focalise sur l'accès aux informations des différents opérateurs.

Le système définit une source d'information comme toute entité capable de fournir des don-

nées pertinentes à l'accomplissement de la tâche de l'opérateur. Ceci inclut les expertises au sein des équipes, les expertises extérieures à l'équipe ainsi que les procédures et documentations diverses. L'accès à ces sources dépend à la fois de la position relative de l'opérateur par rapport à ces sources, mais aussi des moyens de communication disponibles. Cet accès se retrouve souvent limité par des moyens de communication non adaptés à l'environnement de travail et notamment l'absence de contact direct visuel (encombrement) ou auditif (bruit). Il s'agit alors de pouvoir apporter les informations nécessaires aux utilisateurs selon leurs besoins et leurs contraintes. Cette adéquation est assurée via trois mécanismes différents : le filtrage de l'information, la présentation de l'information et la dynamicité de l'interaction :

- le filtrage de l'information prend en compte les paramètres liés à l'avion, à la tâche à réaliser et à l'environnement global dans lequel se déroule cette tâche. Le filtrage identifie les opérateurs ainsi que leurs droits d'accès, leurs expertises, leurs positions au sein des équipes et surtout leurs positions par rapport aux objets des tâches. Les informations sont alors filtrées au niveau des systèmes avant leur transmission aux opérateurs.
- la présentation de l'information intervient après l'étape de filtrage et prend en compte les contraintes de l'environnement de la tâche ainsi que les aspects de sécurité et de fatigue.
- l'interaction doit être adaptée aux procédures et recommandations existantes afin d'améliorer l'acceptabilité des nouveaux moyens en tant qu'alternatives aux systèmes en place. Ceci permet de profiter de l'expérience acquise par les opérateurs et les entreprises. L'interaction doit également être adaptée aux contraintes physiques de l'environnement, et éviter toute modification perturbatrice de l'espace de travail.

En l'absence de standards décrivant un tel système interactif mixte dans sa globalité, la validation des choix de conception passe forcément par le biais de multiples cycles d'implémentation et de test. A l'échelle du système de maintenance concerné, ces multiples itérations s'avèrent très coûteuses. En effet, il demeure difficile de mobiliser un espace de travail adapté et des opérateurs de maintenance pendant toute la durée du développement. Les contraintes de sécurité font en sorte qu'il est aussi difficile de déployer un tel système avant toute pré-validation des paramètres de sécurité des opérations et des opérateurs. Simuler un tel système servirait à concrétiser certains choix de conception dans un contexte interactif et donc faciliter l'évaluation de certains des paramètres de l'interaction.

4.2.3 Simulation

La simulation fournit un environnement contrôlé et peu coûteux pour l'évaluation des facettes principales d'un tel système et notamment la localisation des opérateurs ou l'adaptation des canaux de communication et des configurations spatiales et logicielles. Elle permet surtout de conduire les tests sans monopoliser l'espace de travail des opérateurs. Le test des configurations matérielles difficiles à simuler ou le déploiement de celles-ci dans un environnement physique sont relégués aux étapes ultérieures du processus de développement.

Le système complet demeure, malgré les avantages apportés par la simulation très compliqué à détailler dans ce rapport. Afin d'illustrer la mise en oeuvre de la simulation et de démontrer la réutilisation des modules précédemment développés, nous allons présenter uniquement les phases de simulation liées à un scénario particulier.

Dans ce scénario, l'environnement consiste en un ensemble de hangars et un terrain à ciel ouvert. L'équipe de maintenance est constituée de trois opérateurs et d'un expert. L'interaction est observée du point de vue d'un seul opérateur responsable du remplacement d'une pièce défectueuse sur le réacteur d'un avion. La procédure standard se déroule en six étapes :

- accéder au compartiment moteur et localiser la pièce concernée ;
- enlever et remplacer la pièce défectueuse ;
- contacter un expert pour valider l'opération ;
- une fois l'étape validée, évacuer une zone de sécurité autour du réacteur ;
- lancer le réacteur sous la supervision de l'expert ;
- une fois terminé, notifier de la disponibilité de la zone en question ;

Ce scénario implique un certain nombre de contraintes au niveau du système. Il s'agit alors de pouvoir :

- localiser facilement le lieu d'intervention ;
- identifier un expert approprié pour la validation (droits, expertises et disponibilité) ;
- communiquer avec les autres opérateurs ;
- vérifier l'évacuation de la zone moteur dans un espace encombré et bruyant ;
- disposer d'un espace commun pour synchroniser et valider la tâche avec l'expert ;

4.2.4 Processus de développement

Contrairement au musée augmenté pour lequel on disposait d'une modélisation ASUR complète, le système de maintenance a bénéficié d'une conception et spécification informelles décrites dans [DAJ07]. L'utilisation de SIMBA a alors servi à la génération d'un environnement interactif destiné principalement à tester le positionnement des différents dispositifs dans l'environnement d'interaction ainsi qu'à fournir une visualisation du déroulement des différentes procédures de maintenance pour une validation par des experts de domaine (opérateurs, agents de sécurité, etc.). Afin de tirer profit des outils déjà développés, nous avons modélisé le scénario précédent dans ASUR. Le modèle résultant est alors exprimé dans la figure 4.10.

L'opérateur principal est identifié comme un utilisateur dans ASUR. Les autres opérateurs, y compris l'expert, sont mis en évidence comme des artefacts physiques (Rtool). Le modèle simplifié ne montre que l'utilisateur principal et un seul membre expert de l'équipe de maintenance. L'avion est l'objet de la tâche (Robject) et les opérateurs sont localisés dans les hangars de l'espace de maintenance par des adaptateurs (Ain) identifiés en tant que caméras par leurs propriétés ASUR. Le suivi en dehors des hangars est assuré par des adaptateurs (Ain) pouvant être assimilés à des récepteurs GPS. Le modèle simplifié ne montre qu'un seul GPS associé à l'utilisateur principal. Ces adaptateurs modifient l'objet de la tâche (Sinfo) qui regroupe le plan 2D de l'environnement et les ressources associées, sur l'adaptateur de sortie (Aout) qui n'est autre qu'un écran de PDA. Le regroupement physique entre les opérateurs et les PDAs est identifié par une liaison de proximité (=). Le PDA est aussi associé à un adaptateur d'entrée (Ain) qui permet à l'utilisateur de modifier l'interface et de communiquer avec le systèmes et les autres opérateurs. Ce dernier adaptateur peut être assimilé à des éléments d'entrée d'une interface graphique 2D.

Le modèle ASUR est ensuite utilisé pour générer le modèle SIMBA correspondant. Tout comme pour le système du musée augmenté, la TAS génère à partir du diagramme ASUR de la figure 4.10 le modèle SIMBA initialement vide (figure 4.11) et le modèle miroir (figure 4.12).

Afin de démontrer la flexibilité de SIMBA pour le prototypage de systèmes divers, nous avons choisi de réutiliser les composants déjà développés pour le musée augmenté :

– au composant visiteur est assigné le nom "opérateur" qui lui permet de contrôler le modèle

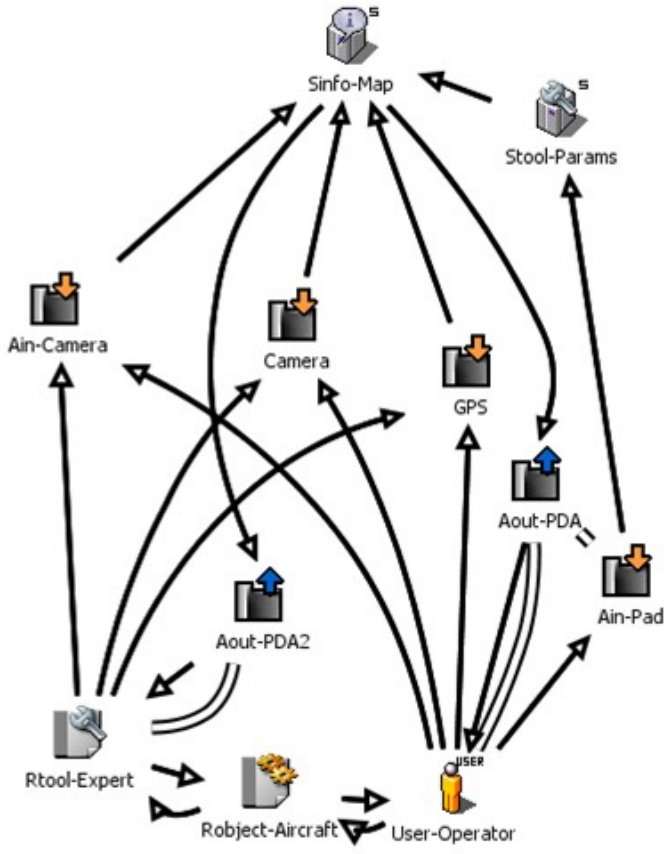

FIGURE 4.10 – Modèle ASUR partiel du système d'assistance

3D du même nom géré par le gestionnaire graphique.

- les composants "opérateur" et "expert" sont des copies du composant "guide" du musée modifiés pour contrôler leurs représentations. Ils sont également modifiés pour répondre à des requêtes provenant des autres membres de l'équipe ou envoyées par le système de contrôle en fonction de l'avancement des tâches.
- les composants "caméra" sont également maintenus, avec un changement du facteur d'échelle de leur zone de suivi pour s'adapter aux types de suivi qu'on souhaite simuler.
- les composants de gestion des interfaces graphiques maintiennent la même taille de cette dernière (adaptée à la taille du modèle 3D du PDA lui aussi réutilisé). L'interface affiche les hangars, les avions positionnés dans ces hangars et sur le terrain extérieur ainsi que la position des opérateurs et de l'expert tels que transmis par le système central. L'interface réagit également à l'avancement de la tâche de maintenance reflétant ainsi les différentes

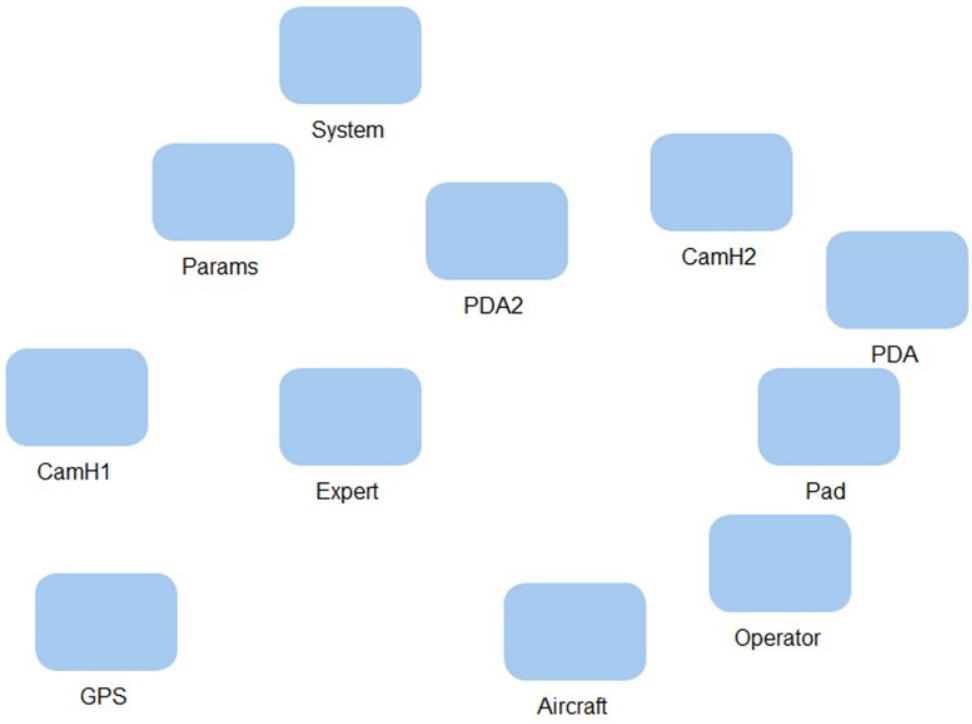

FIGURE 4.11 – Modèle vide généré par la TAS

étapes. La zone à évacuer pour réaliser la procédure de démarrage des réacteurs est affichée sur les PDAs de tous les membres concernés en superposition au plan de l'environnement.

– le composant "système" synchronise les différentes étapes, se charge du routage des différents échanges d'information et met à jour les interfaces 2D en fonction du protocole de maintenance prédéfini. En particulier, il évalue les positions relatives des opérateurs par rapport aux zones dangereuses ainsi que leurs profils pour mieux répondre aux demandes d'assistance émises par les autres membres de l'équipe.
– en l'absence de composant GPS déjà développé, nous avons choisi de le remplacer par une caméra ARToolkit (figure 4.14) associée à un changement du facteur d'échelle. Ceci montre la flexibilité de l'architecture et l'interchangeabilité entre entités réelles (ARToolkit) et simulées (GPS), ainsi que l'interchangeabilité des composants d'interaction (GPS et caméras de localisation fournissent le même type de données).

Le modèle ainsi complété par l'IDC est montré dans la figure 4.13. Les entités simulées sont

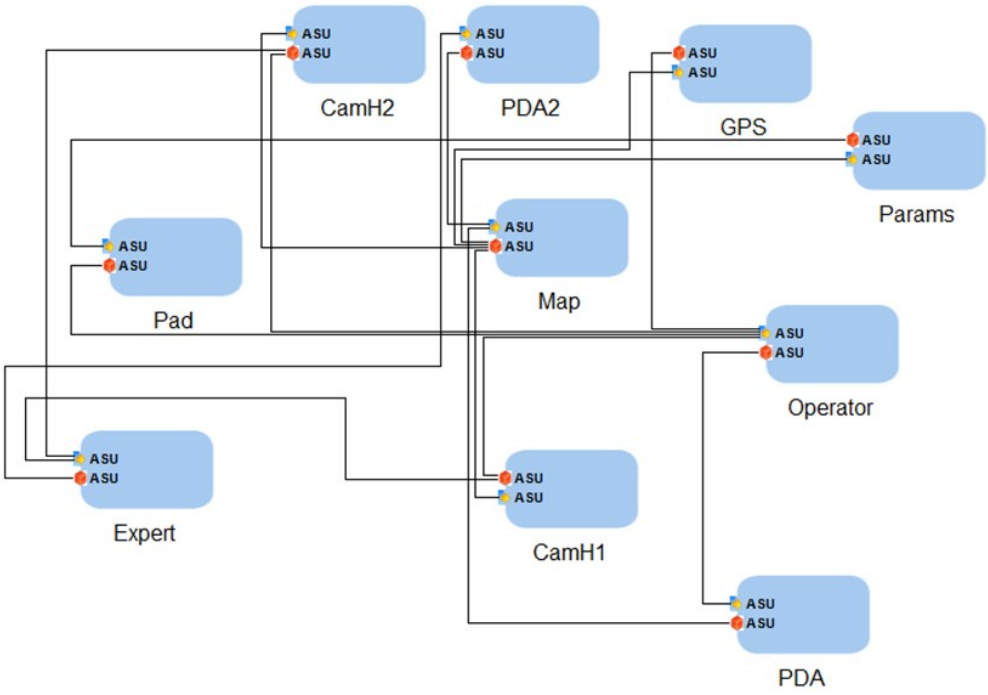

FIGURE 4.12 – Modèle miroir généré par la TAS

connectées au GM pour assurer leurs représentations. Le contrôleur est un composant de la couche d'accès uniforme présentée dans le chapitre 2 pilotant un périphérique SpaceMouse. Le contrôleur est ajouté manuellement au diagramme pour piloter l'avatar.

Le GAC génère alors une archive prête à être déployée par la plateforme Qedo. La figure 4.15 montre une capture d'écran de l'interaction à l'instant où l'opérateur principal s'apprête à démarrer le réacteur. La zone à évacuer est marquée dans un cercle blanc. Les autres opérateurs peuvent être localisés afin d'assurer la bonne évacuation de la zone. La figure 4.16 montre le modèle 3D du PDA sans l'interface graphique superposée alors que l'opérateur est à l'extérieur des hangars.

Ces deux systèmes ont servi à tester et valider certains aspects de la méthodologie de développement proposée par SIMBA. Dans les deux systèmes, l'environnement tridimensionnel a été modélisé dans STAGE, l'éditeur graphique compatible avec notre implémentation du GM. Les modèles des PDAs sont ajoutés de façon dynamique par leurs composants respectifs, et ce pendant le déroulement de l'interaction et sur demande de l'utilisateur. Le prochain chapitre

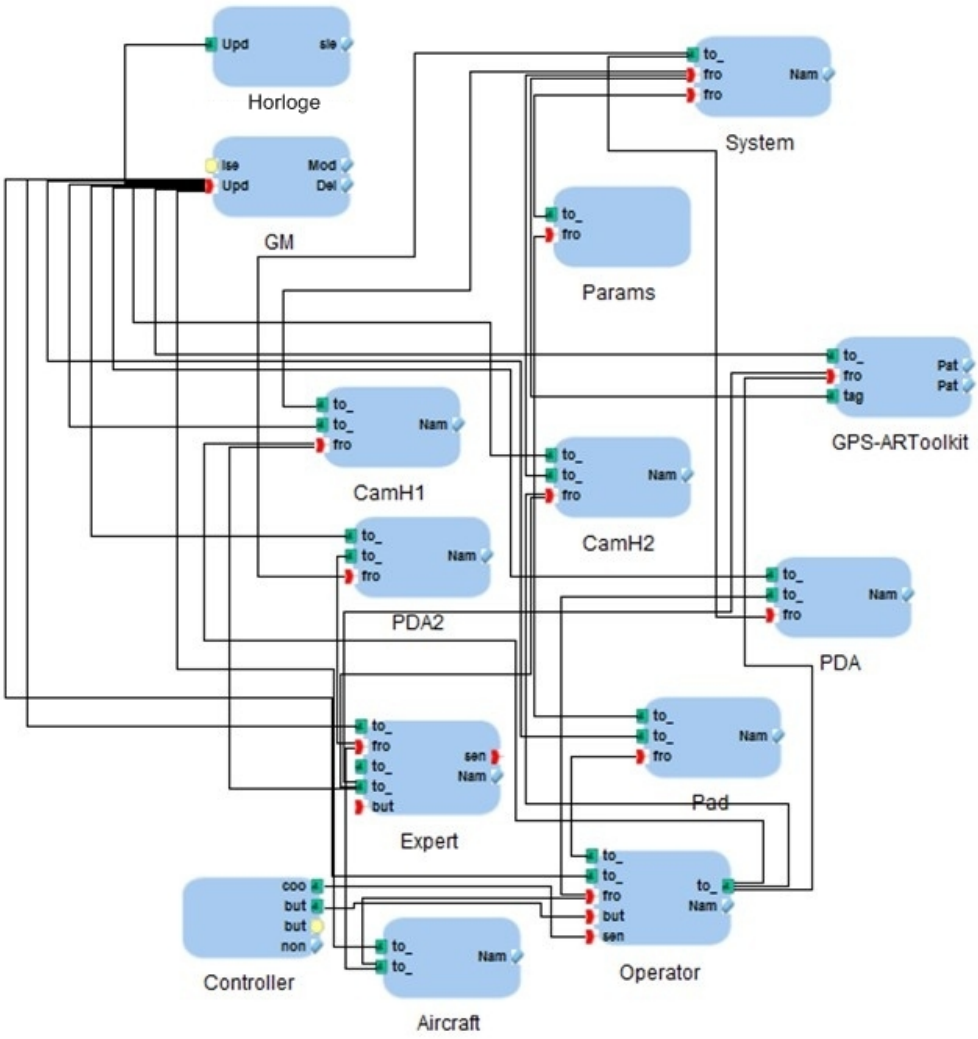

FIGURE 4.13 – Modèle SIMBA final, avec le GPS simulé via une caméra ARToolkit

présente en conclusion un récapitulatif des points principaux abordés dans cette thèse ainsi que les principales perspectives de SIMBA à court et moyen terme.

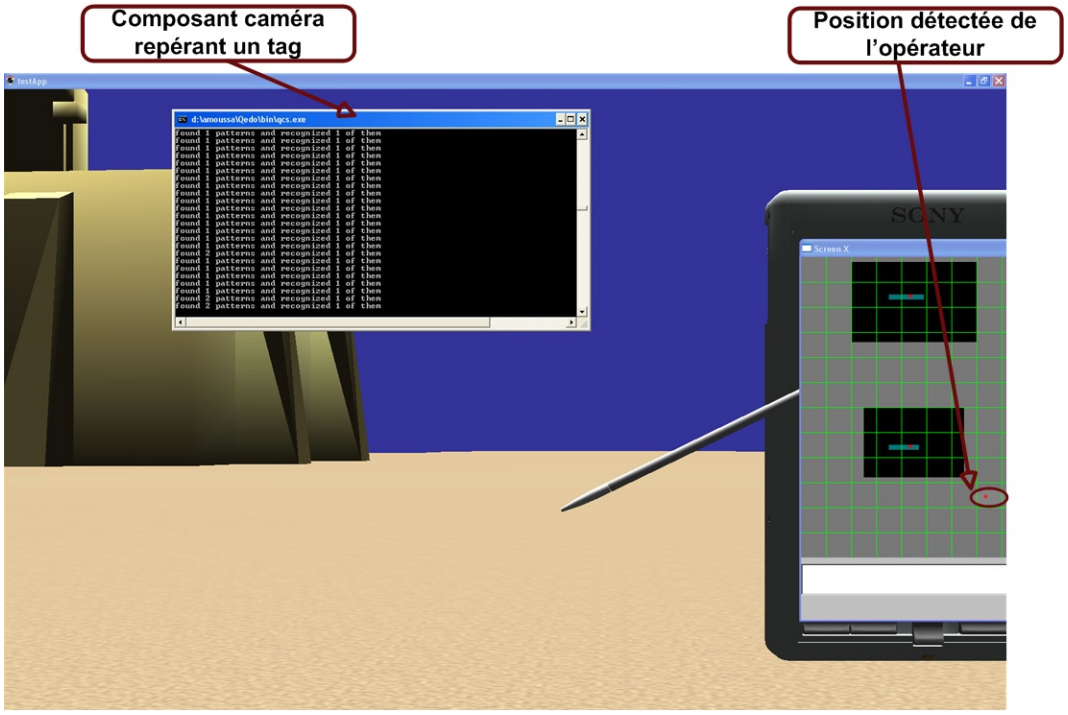

FIGURE 4.14 – Caméra remplaçant un composant GPS pour le suivi à l'extérieur

FIGURE 4.15 – Capture d'écran de l'interaction dans un des hangars

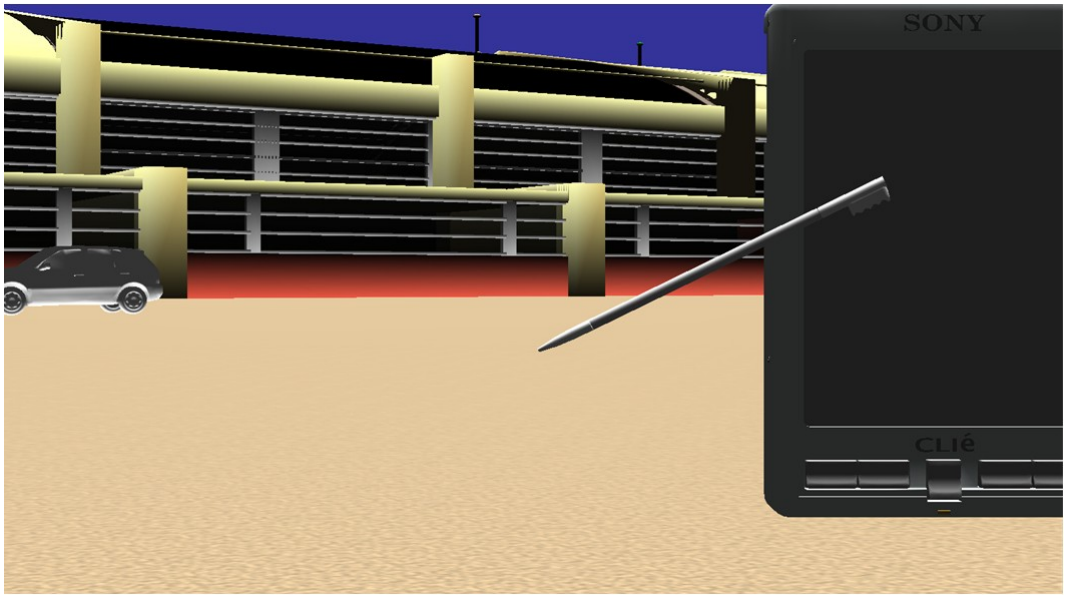

FIGURE 4.16 – Capture d'écran sur le terrain extérieur aux hangars

5

Conclusions et perspectives

Ce dernier chapitre reprend les principales caractéristiques de SIMBA et de l'approche de prototypage incrémental par simulation des systèmes mixtes. Il identifie également un certain nombre de perspectives qui permettront de pallier les limitations de SIMBA dans son état actuel ainsi que d'étendre le champ d'application du prototypage par simulation.

5.1 Récapitulatifs

Cette thèse a présenté une approche de prototypage de moyenne fidélité, SIMBA, pour accompagner les processus de conception ou de co-conception des systèmes interactifs mixtes. L'approche allie une démarche dirigée par les modèles et une plateforme de services orientée composants, et permet d'intégrer les rôles des différents acteurs au sein d'un seul et même processus de développement.

La simulation de l'interaction exploite la réalité virtuelle pour représenter l'environnement de l'interaction et permettre à l'utilisateur d'évoluer dans une réplique de l'environnement du système final où les entités statiques et interactives du système mixte sont représentées. L'environnement de simulation permet d'intégrer facilement les différents modules logiciels du système dans le prototype afin de tester leurs fonctionnalités.

L'architecture de SIMBA définit et structure des services pour la gestion des représentations graphiques mises en oeuvre dans le cadre du prototypage par simulation. La gestion des représentations graphiques mise sur la séparation entre les aspects liés à la simulation des pro-

totypes et ceux reflétant les choix de conception appelés aspects fonctionnels. Cette séparation isole les deux facettes des prototypes et offre un accès plus simple aux services de simulation. La gestion des représentations est assurée par un composant central : le gestionnaire graphique. Les mécanismes du gestionnaire sont isolés des aspects fonctionnels des systèmes, ce qui évite la plupart des limitations imposées par les moteurs de rendu actuels quant à l'intégration des comportements complexes ou des traitements parallèles. Le gestionnaire graphique offre une interface simple qui permet aux composants fonctionnels d'accéder à la mise à jour de leurs représentations via des types de données génériques (chaînes et matrices). L'application de ces mises à jour est alors gérée par l'implémentation du gestionnaire.

Les services de gestion des dispositifs d'entrée et des interactions offrent un support à l'interactivité des prototypes. La séparation appliquée par ces services entre leurs interfaces et les implémentations de ces interfaces ainsi que la standardisation des types de données fournies ou requises par les différents composants, permettent de définir et de développer des composants d'entrées et d'interaction interchangeables. Ces composants sont très utiles dans un contexte d'environnements simulés, puisqu'ils permettent de " piloter " les entités virtuelles telles l'avatar de l'utilisateur. Ils permettent aussi de gérer et de tester différentes configurations d'entrée et de localisation, ainsi que de remplacer des dispositifs complexes par d'autres dispositifs plus simples (périphériques virtuels). Les services d'interaction définissent également des couches de traitement et d'interaction à niveau d'abstraction plus élevé qui permettent d'étendre la portée des tests itératifs aux choix des techniques et métaphores d'interaction.

Le méta-modèle SIMBA, permet une modélisation des systèmes mixtes et de leurs prototypes en phase avec plusieurs approches de conception existantes (ASUR, IRVO, etc). Ceci facilite la description en SIMBA de systèmes décrits dans ces modèles. La description SIMBA est également en phase avec les modèles d'implémentation à base de composants ce qui permet d'exprimer les modèles des prototypes dans des modèles spécifiques à ces plateformes d'implémentation, ainsi que d'intégrer des services applicatifs divers sans qu'ils soient explicitement identifiés dans le méta-modèle. Le méta-modèle reconnaît également les éléments spécifiques à l'architecture des services SIMBA tels les services de simulation et d'interaction. Une identification claire de ces éléments dans le méta-modèle nous permet de définir des paradigmes et des outils spécifiques capables d'opérer sur ces artefacts et de les gérer tout au long des phases de prototypage itératif

et incrémental. Les éléments spécifiques dérivent dans SIMBA des composants et artefacts génériques. Par conséquent, il est facile de définir des transformations qui permettent de traduire le modèle SIMBA et ses entités en une description dans différents modèles d'implémentation, ainsi que d'intégrer dans les prototypes et les systèmes des services divers et hétérogènes.

SIMBA définit et intègre un certain nombre d'outils qui facilitent la mise en commun des compétences et expertises des différents acteurs du processus de développement des systèmes mixtes. Les interpréteurs et les transformations de modèles permettent de diriger le processus incrémental en assurant la génération rapide des prototypes simulés à partir des modèles de conception. Ils garantissent aussi, en complément des règles et contraintes associées au méta-modèle SIMBA, la conformité de ces prototypes et de leur évolution avec les conceptions initiales. Les situations de conflit peuvent être corrigées automatiquement ou, le cas échéant, détectées et signalées aux développeurs. Le processus dirigé par les modèles permet également de coupler SIMBA à différents modèles de conception.

En exploitant des résultats de conception en amont, SIMBA permet une génération rapide des simulations mais aussi une évolution incrémentale de celles-ci pour aboutir au système final. La séparation des aspects permet de définir et développer des composants simulés ou non simulés interchangeables. Le prototypage évolutif profite de cette capacité à commuter les composants entre leurs versions simulées (donc contrôlant une représentation graphique) et les versions susceptibles d'être déployées sur les plateformes finales. Le prototypage itératif devient incrémental avec des phases intermédiaires faisant apparaître des prototypes mélangeant entités simulées et non-simulées. La gestion de tels prototypes ainsi que de la distribution des modules est transparente, allégeant le processus de développement. Le couplage flexible entre les services applicatifs potentiels et les services de simulation permet de s'adapter facilement à différents domaines et contextes applicatifs et simplifie le changement de plateforme d'implémentation.

Les outils et services de SIMBA articulent un processus de développement autour de trois catégories principales d'acteurs : les concepteurs de l'interaction, les intégrateurs de systèmes et les développeurs de composants et de modules. La modélisation artistique de l'environnement simulé est totalement séparée de la conception (fonctionnelle) du système, tout en assurant l'interopérabilité de cette facette avec les différents services et modules des prototypes et des systèmes. Des outils de modélisation simples permettent aux acteurs du cycle de développement

des SIM, potentiellement sans expérience particulière dans le domaine de la modélisation et de la programmation 3D, de développer eux même les environnements de simulation. En cas de besoin, l'intégration d'experts graphiques au sein du processus de développement demeure facile puisque le cadre de collaboration de ces derniers avec les autres auteurs est bien défini.

Une implémentation de l'architecture et l'intégration du méta-modèle dans un environnement de modélisation flexible a permis de mettre à l'épreuve les hypothèses et choix avancés dans cette thèse. Les services de simulation et d'interaction sont implémentés en CORBA CCM. Cette implémentation offre un support standardisé à la transparence de la distribution et le choix des hôtes se fait lors du déploiement ce qui améliore la flexibilité de l'approche globale. Le méta-modèle ainsi que ses outils associés sont définis dans un environnement de modélisation générique, GME. GME offre un support visuel aux étapes de modélisation ainsi qu'un environnement logiciel pour la définition et l'implémentation des règles et outils articulant l'approche dirigée par les modèles. L'implémentation du gestionnaire graphique utilise le moteur de rendu Delta3D. Les mécanismes de gestion des représentations définis par SIMBA sont néanmoins indépendants de cette implémentation.

5.2 Limitations et perspectives

L'application de SIMBA à différents cas de prototypage de systèmes interactifs mixtes nous a permis de dégager un certain nombre d'améliorations possibles à apporter à la plateforme.

Tout d'abord, la plupart des approches de conception actuelles comptent sur différents modèles pour représenter les multiples facettes d'un système mixte. Les diagrammes ASUR par exemple ne décrivent que l'interaction par rapport à une seule tâche principale et la description du système dans sa globalité requiert plusieurs diagrammes. Des travaux en cours visent en particulier à lier ASUR à un modèle de tâches [CDR07] afin de contextualiser un ou plusieurs des modèles ASUR dans l'activité globale de l'utilisateur. Il s'agit alors de définir un cadre de transformation permettant non plus de générer le diagramme SIMBA à partir d'un seul mais de plusieurs modèles exprimés avec des paradigmes identiques ou différents. La flexibilité du processus de développement ainsi que son articulation autour de transformations bien définies permettent néanmoins de situer cette évolution de SIMBA au niveau de la transformation qui

génère le diagramme SIMBA initial. Il faut néanmoins prendre soin des processus qui dépendent de cette première transformation tels les processus de validations de la conformité des prototypes avec les choix de conception.

Une perspective à court terme consiste également à modifier le composant graphique afin de gérer d'autres types de représentations (auditives, haptiques). Cette gestion doit suivre le même principe que celui adopté pour la partie graphique et nécessite donc la définition d'interfaces et de modalités d'interaction compatibles. Les composants miroirs doivent également bénéficier d'une standardisation de leurs mécanismes après une étude détaillée de leur utilité selon les différents cas d'évolution des prototypes. La caractérisation ASUR des adaptateurs en entrée et en sortie nous aidera probablement à opérer cette standardisation, même dans les cas d'utilisation d'un autre modèle de conception en entrée.

Il s'agit également de procéder à une évaluation méthodique et systématique des apports de la simulation en terme de support à la validation des choix de conception. Ceci nécessite une identification des facettes des SIM que peuvent représenter les environnements simulés ainsi que l'adéquation de la généralisation des conclusions tirées lors de ces phases de prototypage aux systèmes réels non simulés.

Une limitation importante dans l'état actuel de SIMBA demeure l'absence de mécanismes spécifiques à la gestion des interfaces 2D pouvant faire partie des systèmes interactifs prototypés. Dans sa version actuelle, SIMBA considère la conception des interfaces 2D interactives ainsi que l'interaction avec ces interfaces comme un aspect que les développeurs doivent gérer au niveau de l'implémentation des composants. Les mécanismes de communication inter-composants fournis permettent alors d'intégrer la gestion de ces interfaces dans le système simulé (appuyer sur un bouton de l'interface modifie le comportement des caméras simulées par exemple). Cependant, il est clair que les interfaces 2D d'un système mixte constituent un élément essentiel de son interface, et contribuent fortement à son adéquation et adaptation aux attentes des utilisateurs. Il s'agit alors de pouvoir représenter ces interfaces dans l'espace tridimensionnel de la simulation. L'utilisateur doit ainsi pouvoir interagir avec ces interfaces au travers de leurs représentations dans l'environnement tridimensionnel. Cette intégration nécessitera probablement un mécanisme de texturage intuitif associé à une version améliorée de l'interface de sélection du gestionnaire graphique.

L'application de SIMBA dans le cadre d'un projet MIME [Dub07] est en cours. Le projet MIME consiste à tester différentes techniques de navigation dans un arbre généalogique des espèces. A terme, ce projet nous permettra de tester la méthodologie de prototypage incrémental par simulation sur un processus de développement complet, allant jusqu'au déploiement final du système dans un environnement non simulé.

Bibliographie

[3DC02] 3DConnexion. Spacemouse, 2002. http ://www.3dconnexion.com/.

[ABB+01] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6) :34–47, November/December 2001.

[AGG00] N. Aussenac-Gilles and D. Garcia. Exploitation pour la modélisation des connaissances causales détectées par COATIS . In *Ingénierie des connaissances : évolutions récentes et nouveaux défis* , Collection Technique et Scientifique des Télécommunications, pages 257–274. Eyrolles, Paris (F), 2000.

[AGLR96] N. Aussenac-Gilles, P. Laublet, and C. Reynaud. *Acquisition et ingénierie des connaissances : tendances actuelles.* Cepadues Editions, Toulouse, mai 1996.

[Agr06] M. Agrawala. Berkeley online courses, 2006. http ://vis.berkeley.edu/courses/cs160-fa06/WWW/lectures-WWW/mvceventdriven/walk004.html.

[AJD06] W. Abou Moussa, J. P. Jessel, and E. Dubois. Notational-based Prototyping of Mixed Interactions. In *International Symposium on Mixed and Augmented Reality (ISMAR), Santa Barbara, USA, 22/10/2006-25/10/2006*, http ://www.ieee.org/, 2006. IEEE.

[AL00] A. Am and F. Ledermann. *An Authoring Framework for Augmented Reality Presentations.* PhD thesis, 2000.

[Alb03] S. Albin. *The Art of Software Architecture : Design Methods and Techniques.* John Wiley and Sons, 2003.

[AMI04] AMIRE - authoring mixed reality, April 28 2004. cs1.ist.psu.edu.

[Azu97] R. Azuma. A survey of augmented reality. *Presence*, 6(4) :355–385, 1997.

[BCR06] P. Bazex, A. Canals, and J. C. Rault. *L'ingénierie dirigée par les modèles Processus de développement logiciel basé sur les modèles : Démarche et Outillage*, volume 77 of *Génie Logiciel*. Hermès, http ://www.editions-hermes.fr/, juin 2006.

[BJH+01] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira. VR Juggler : A Virtual Platform for Virtual Reality Application Development. In *VR '01 : Proceedings of the Virtual Reality 2001 Conference (VR'01)*, page 89, Washington, DC, USA, 2001. IEEE Computer Society.

[BKLP04] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev. *3D User Interfaces : Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.

[BLM03] M. Beaudouin-Lafon and W. Mackay. Prototyping tools and techniques. pages 1006–1031, 2003.

[BML05] S. Benford, C. Magerkurth, and P. Ljungstrand. Bridging the physical and digital in pervasive gaming. *Communications of the ACM*, 48(3) :54–57, 2005.

[BO04] D. Burns and R. Osfield. Open scene graph a : Introduction, b : Examples and applications. In *VR '04 : Proceedings of the IEEE Virtual Reality 2004*, page 265, Washington, DC, USA, 2004. IEEE Computer Society.

[Boa84] B. H. Boar. *Application prototyping : a requirements definition strategy for the 80s*. John Wiley & Sons, Inc., New York, NY, USA, 1984.

[BSD07] C. Bach, P. Salembier, and E. Dubois. Co-Conception centrée utilisateur de systèmes interactifs pour les Musées : application à la découverte de la cladistique. In *Congrès de la Société d'Ergonomie de Langue Française (SELF), Saint-Malo, 05/09/07-07/09/07*, pages 69–78, http ://www.octares.com/, septembre 2007. Octares Editions.

[Bux83] W. Buxton. Lexical and pragmatic considerations of input structures. *SIGGRAPH Computer Graphics*, 17(1) :31–37, 1983.

[BV03] J. J. Barton and V. Vijayaraghavan. UBIWISE, A simulator for ubiquitous computing systems design. Technical Report HPL-2003-93, Hewlett Packard Laboratories, May 07 2003.

[CD04] R. Chalon and B. T. David. Modélisation de l'interaction collaborative dans les systèmes de réalité mixte. In *IHM 2004 : Proceedings of the 16th conference on Association Francophone d'Interaction Homme-Machine*, pages 37–44, New York, NY, USA, 2004. ACM.

[CDR07] S. Charfi, E. Dubois, and R.Bastide. Articulating Interaction and Task Models for the Design of Advanced Interactive Systems. In *TAsk MOdels DIAgrams for UI design (TAMODIA), Toulouse, 07/11/2007-09/11/2007*, http ://www.springerlink.com/, 2007. Springer-Verlag.

[CKJ07] A. Coyette, S. Kieffer, and J.Vanderdonckt. Multi-fidelity prototyping of user interfaces. In *Proceedings of IFIP INTERACT'07 : Human-Computer Interaction*, volume 4662 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 2007.

[CM92] T. P. Caudell and D. W. Mizell. Augmented reality : An application of heads-up display technology to manual manufacturing processes. In *Proc. HICCS '92.*, 92.

[CTLR06] D. Cheung, J. Y. Tigli, S. Lavirotte, and M. Riveill. Wcomp : a multi-design approach for prototyping applications using heterogeneous resources. In *RSP '06 : Proceedings of the Seventeenth IEEE International Workshop on Rapid System Prototyping (RSP'06)*, pages 119–125, Washington, DC, USA, 2006. IEEE Computer Society.

[CV05] A. Coyette and J. Vanderdonckt. A sketching tool for designing anyuser, anyplatform, anywhere user interfaces. In *INTERACT*, volume 3585 of *Lecture Notes in Computer Science*, pages 550–564. Springer, 2005.

[CVV77] L. C. Caruthers, J. Van Den Bos, and A. Van Dam. GPGS, a device-independent general purpose graphic system for stand-alone and satellite graphics. *SIGGRAPH Computer Graphics*, 11(2) :112–119, July 1977.

[DAJ07] N. De Bonnefoy, W. Abou Moussa, and J. P. Jessel. Intuitive and adaptive mixed system to support nomadic maintenance collaborative activities. In *International*

Conference on Concurrent Enterprising - ICE, Antibes, 04/06/2007-06/06/2007, 2007.

[DBG+06] E. Dubois, C. Bach, G. Gauffre, S. Charfi, and P. Salembier. Instrumentation de focus-group dans la co-conception de systèmes mixtes. In *IHM*, volume 133 of *ACM International Conference Proceeding Series*, pages 163–166. ACM, 2006.

[DBT08] E. Dubois, C. Bach, and P. Truillet. Comparing Mixed Interactive Systems for Navigating 3D Environments in Museum. In *Design Specification and Verification of Interactive Systems (DSV-IS), Kingston, 16/07/08-18/07/08*, pages 0–0, http ://www.ifip.org, juillet 2008. IFIP.

[DG02] R. Dörner and P. Grimm. Building 3D applications with 3D components and 3D framework. *Workshop at the Web3D 2001 Conference*, February 2002.

[DGHP02] R. Dörner, C. Geiger, M. Haller, and V. Paelke. Authoring mixed reality - a component and framework-based approach. In *IWEC*, volume 240 of *IFIP Conference Proceedings*, pages 405–413. Kluwer, 2002.

[DM96] D. Drascic and Pa. Milgram. Perceptual issues in augmented reality, February 21 1996.

[DMBB07] E. Dubois, W. Abou Moussa, C. Bach, and N. De Bonnefoy. Modelling and Simulation of Mobile Mixed Systems. In *Handbook of Research on User Interface Design and Evaluation for Mobile Technology*. Idea Group, http ://www.idea-group.com/, 2007.

[Dub01] E. Dubois. *Chirurgie augmentée : un cas de réalité augmentée ; Conception et réalisation centrées sur l'utilisateur*. PhD thesis, July 06 2001.

[Dub07] E. Dubois. Mime, 2007. http ://www.irit.fr/ Emmanuel.Dubois/index.php ?2283,0.

[ETJ07] S. Elmerhebi, P. Torguet, and J. P. Jessel. Enhancing Distributed Virtual Environments Scalability using Effect Management with Inter-Server Multicast. In *Asia-Pacific Workshop on Visual Information Processing, Tainan, Taiwan, 15/12/2007-17/12/2007*, pages 213–218, http ://www.ncku.edu.tw, décembre 2007. National Cheng Kung University.

[FLT08] FLTK. Fast Light Toolkit, http ://www.fltk.org, 2008. http ://www.fltk.org.

[FMS93] S. Feiner, B. MacIntyre, and D. Seligmann. Knowledge-based augmented reality. *Communications of the ACM*, 36(7) :53–62, 1993.

[FS97] M. Fayad and D. C. Schmidt. Object-oriented application frameworks. *Communications of the ACM*, 40(10) :32–38, 1997.

[FSS97] C. Faisstnauer, D. Schmalstieg, and Z. Szalavári. Device-independent navigation and interaction in virtual environments. *VRAIS '98*, 1997.

[Fuh99] A. Fuhrmann. *Studierstube : A Collaborative Virtual Environment for Scientific Visualization*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 1999.

[FWC90] J. D. Foley, V. L. Wallace, and P. Chan. The human factors of computer graphics interaction techniques. pages 67–121, 1990.

[GHJV95] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns : Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1995.

[Hew86] W. T. Hewitt. Towards a graphics standard for 3d (gks-3d). pages 211–226, 1986.

[HF93] D. Henry and T. A. Furness. Spatial perception in virtual environments : Evaluating an architectural application. In *VR*, pages 33–40, 1993.

[HK93] T. He and A. E. Kaufman. Virtual input devices for 3d systems. In *VIS '93 : Proceedings of the 4th conference on Visualization '93*, pages 142–148, 1993.

[Hom08] Adobe Director Homepage. http ://www.adobe.com/fr/products/director, 2008.

[Hor96] B. Horn. On Xerox, Apple and progress, 1996. http ://www.folklore.org.

[HSFP99] G. Hesina, D. Schmalstieg, A. Furhmann, and W. Purgathofer. Distributed open inventor : a practical approach to distributed 3d graphics. In *VRST '99 : Proceedings of the ACM symposium on Virtual reality software and technology*, pages 74–81, New York, NY, USA, 1999. ACM.

[HV99] M. Henning and S. Vinoski. *Advanced CORBA programming with C++*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[IBJU+02] H. Ishii, E. Ben-Joseph, J. Underkoffler, L. Yeung, D. Chak, Z. Kanji, and B. Piper. Augmented urban planning workbench : Overlaying drawings, physical models and digital simulation. In *ISMAR '02 : Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, page 203, Washington, DC, USA, 2002. IEEE Computer Society.

[Inc08] Cunningham Inc. Hollywood principle, 2008. http ://c2.com/cgi/wiki ?HollywoodPrinciple.

[IS96] Id Software. Quake. *Computer game, http ://www.idsoftware.com/quake/*, 1996.

[Jes08] Y. Jestin. Prototypage et environnements de développement rapide, 2008. Cours dispensé au Master 2 IHM, ENAC, Toulouse.

[Jon94] P. C. Jones. Xvan : A true multiple screen X server. *The X Resource*, 9(1) :91–104, January 1994.

[Kas98] N. Kasabov. *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering.* MIT Press, 1998.

[KB99] H. Kato and M. Billinghurst. Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. In *Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99)*, San Francisco, USA, October 1999.

[Kil96] M. J. Kilgard. *OpenGL Programming for the X Window System.* Addison-Wesley, 1996. http ://reality.sgi.com/mjk/glut3/glut3.html.

[KL04] S. R. Klemmer and J. Lin. Papier-mache : toolkit support for tangible input. In *Proceedings of ACM CHI 2004 Conference on Human Factors in Computing Systems*, volume 1, pages 399–406, 2004.

[KSA+03] J. Kelso, S. G. Satterfield, L. E. Arsenault, P. M. Ketchan, and R. D. Kriz. DIVERSE : A Framework for Building Extensible and Reconfigurable Device-Independent Virtual Environments and Distributed Asynchronous Simulations. *Presence : Teleoperation and Virtual Environments*, 12(1) :19–36, 2003.

[Les03] S. Lessels. Virtual reality and the built environment by jennifer whyte. *Presence*, 12(5) :550, 2003.

[LMB01] A. Ledeczi., M. Maroti, and A. Bakay. The Generic Modeling Environment. *Workshop on Intelligent Signal Processing*, 2001.

[LS05] F. Ledermann and D. Schmalstieg. April a high-level framework for creating augmented reality presentations. In *VR '05 : Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality*, pages 187–194, Washington, DC, USA, 2005. IEEE Computer Society.

[MCP+06] M. McCurdy, C. Connors, G. Pyrzak, B. Kanefsky, and A. Vera. Breaking the fidelity barrier : an examination of our current characterization of prototypes and an example of a mixed-fidelity success. In *CHI '06 : Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1233–1242, New York, NY, USA, 2006. ACM.

[Men06] T. Mens. On the use of graph transformations for model refactoring. In *GTTSE*, volume 4143 of *Lecture Notes in Computer Science*, pages 219–257. Springer, 2006.

[MF96] B. MacIntyre and S. Feiner. Language-level support for exploratory programming of distributed virtual environments. In *UIST '96 : Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 83–94, New York, NY, USA, 1996. ACM.

[MF98] B. MacIntyre and S. Feiner. A distributed 3D graphics library. pages 361–370, New York, NY, USA, 1998. ACM.

[MFFM98] W. E. Mackay, A. L. Fayard, L. Frobert, and L. Médini. Reinventing the familiar : exploring an augmented reality design space for air traffic control. In *CHI '98 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 558–565, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[MGB+03] B. MacIntyre, M. Gandy, J. Bolter, S. Dow, and B. Hannigan. Dart : The designer's augmented reality toolkit. In *ISMAR '03 : Proceedings of the The 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality*, page 329, Washington, DC, USA, 2003. IEEE Computer Society.

[MK94] P. Milgram and F. Kishino. A taxonomy of mixed reality visual displays. *IEICE Transactions on Information Systems*, E77-D(12), December 1994.

[MSW+03] A. MacWilliams, C. Sandor, M. Wagner, M. Bauer, G. Klinker, and B. Bruegge. Herding Sheep : Live System Development for Distributed Augmented Reality. *ISMAR '03 : Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, page 123, 2003.

[Mun78] M. E. Mundel. *Motion and time Study : Improving productivity.* Prentice-Hall, 5. ed edition, 1978.

[NC97] L. Nigay and J. Coutaz. *Multifeature Systems : The CARE Properties and Their Impact on Software Design.* 1997.

[Nie93] J. Nielsen. *Usability Engineering.* Academic Press, 1993.

[NIH01] J. Newman, D. Ingram, and A. Hopper. Augmented reality in a wide area sentient environment. In *ISAR*, pages 77–86, 2001.

[OCL02] Object modeling with the OCL, the rationale behind the object constraint language, 2002.

[OMG06] OMG. Corba component model, 2006. http ://www.omg.org.

[PATM04] W. Piekarski, B. Avery, B. H. Thomas, and P. Malbezin. Integrated head and hand tracking for indoor and outdoor augmented reality. In *VR '04 : Proceedings of the IEEE Virtual Reality 2004*, page 11, Washington, DC, USA, 2004. IEEE Computer Society.

[Pon04] M. Ponder. *Component-Based Methodology and Development Framework for Virtual and Augmented Reality Systems.* PhD thesis, Lausanne, 2004.

[Pro08] Eclipse Project. Generic modeling framework, 2008. http ://www.eclipse.org/modeling/gmf/.

[PT01] W. Piekarski and B. Thomas. Tinmith-evo5 - an architecture for supporting mobile augmented reality environments, 2001.

[PTB+01] I. Poupyrev, D. Tan, M. Billinghurst, H. Kato, H. Regenbrecht, and N. Tetsutani. Tiles : A mixed reality authoring interface. In *Proceedings of IFIP INTERACT'01 : Human-Computer Interaction*, pages 334–341, 2001.

[RMTHS+01] II R. M. Taylor, T. C. Hudson, A. Seeger, H. Weber, J. Juliano, and A. T. Helser. Vrpn : a device-independent, network-transparent vr peripheral system. In *VRST '01 : Proceedings of the ACM symposium on Virtual reality software and technology*, pages 55–61, New York, NY, USA, 2001. ACM.

[Rog97] G. F. Rogers. *Framework-based software development in C++*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[RS01] G. Reitmayr and D. Schmalstieg. An open software architecture for virtual reality interaction. In *VRST '01 : Proceedings of the ACM symposium on Virtual reality software and technology*, pages 47–54, New York, NY, USA, 2001. ACM.

[SBK97] T. Sumner, N. Bonnardel, and B. H. Kallak. The cognitive ergonomics of knowledge-based design support systems. In *CHI '97 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 83–90, New York, NY, USA, 1997. ACM.

[Sch99] M. Schrage. Harvard Business School, 1999.

[SK05] C. Sandor and G. Klinker. A rapid prototyping software infrastructure for user interfaces in ubiquitous augmented reality. *Personal Ubiquitous Comput.*, 9(3) :169–185, 2005.

[SLCGJ04] O. Shaer, N. Leland, E. H. Calvillo-Gamez, and R. J. K. Jacob. The TAC Paradigm : Specifying Tangible User Interfaces. *Personal Ubiquitous Comput.*, 8(5) :359–369, 2004.

[SN03] C. Snyder and J. Nielsen. Paper prototyping : The fast and easy way to design and refuse user interfaces, 2003.

[Sán05] M.I. Sánchez-Segura. *Developing Future Interactive Systems*. Idea Group Publishing, London, UK, 2005.

[SPH+04] Colin Swindells, Barry A. Po, Ima Hajshirmohammadi, Brian Corrie, John C. Dill, Brian D. Fisher, and Kellogg S. Booth. Comparing cave, wall, and desktop displays for navigation and wayfinding in complex 3d models. In *CGI '04 : Proceedings of the Computer Graphics International*, pages 420–427, Washington, DC, USA, 2004. IEEE Computer Society.

[SR01] C. Sandor and T. Reicher. CUIML : A language for generating multimodal human-computer interfaces, July 13 2001.

[Sut68] I. E. Sutherland. A head-mounted three-dimensional display. In *AFIPS Conference Proceedings*, volume 33, pages 757–764, 1968.

[TNM08] D. G. Trevisan, L. P. Nedel, and B. Macq. Augmented vision for medical applications. In *SAC '08 : Proceedings of the 2008 ACM symposium on Applied computing*, pages 1415–1419, New York, NY, USA, 2008. ACM.

[TVM02] D. G. Trevisan, J. Vanderdonckt, and B. Macq. Analyzing interaction in augmented reality systems, 2002.

[TYT92] F. Teraoka, Y. Yokote, and M. Tokoro. *A network architecture providing host migration transparency*, pages 137–148. May 1992.

[UI00] B. Ullmer and H. Ishii. Emerging frameworks for tangible user interfaces. *IBM Syst. J.*, 39(3-4) :915–931, 2000.

[UTS+02] S. Uchiyama, K. Takemoto, K. Satoh, H. Yamamoto, and H. Tamura. Mr platform : A basic body on which mixed reality applications are built. In *ISMAR '02 : Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, page 246, Washington, DC, USA, 2002. IEEE Computer Society.

[VDG04] J. Viala, E. Dubois, and P. D. Gray. GUIDE-ME : Graphical User Interface for the Design of Mixed Interactive Environments Based on the ASUR Notation. In *UbiMob '04 : Proceedings of the 1st French-Speaking Conference on Mobility and Ubiquity Computing*, pages 74–77, New York, NY, USA, 2004. ACM.

[Vit06] A. Vitzthum. SSIML/AR : A visual language for the abstract specification of augmented reality user interfaces. In *3DUI '06 : Proceedings of the 3D User Interfaces (3DUI'06)*, pages 135–142, Washington, DC, USA, 2006. IEEE Computer Society.

[VKT05] E. E. Veas, K. Kiyokawa, and H. Takemura. Self-aware framework for adaptive augmented reality. In *ICAT '05 : Proceedings of the 2005 international conference on Augmented tele-existence*, pages 70–77, New York, NY, USA, 2005. ACM.

[VL06] S. Otmane et M. Mallem V. Le Ligeour. Architecture logicielle pour un système d'exploitation des environnements de réalité augmentée complexes. *ICTTA, 2nd IEEE International Conference on Information and Communication Technologies : From Theory to Applications*, 2006.

[Von88] E. Von Hippel. *The Sources of Innovation.* Oxford University Press, 1988.

[W3C99] W3C. XSL Transformations (XSLT), 1999. `http ://www.w3.org/TR/xslt`.

[Web08] ART Website. Advanced realtime tracking, 2008. http ://www.ar-tracking.de.

[Wei95] M. Weiser. The computer for the 21st century. pages 933–940, 1995.

[WQ05] A. Ju An Wang and K. Qian. *Component-oriented programming.* John Wiley and Sons, 2005.

[WZ98] K. Watsen and M. Zyda. Bamboo - a portable system for dynamically extensible, real-time, networked, virtual environments. In *VRAIS '98 : Proceedings of the Virtual Reality Annual International Symposium*, page 252, Washington, DC, USA, 1998. IEEE Computer Society.

[Xer07] Xerox. Xerox Palo Alto research center, 2007. http ://www.parc.com/.

[XMI05] XML metadata interchange (XMI), v2.0, 2005. http ://www.omg.org/cgi-bin/doc ?formal/05-05-01.