Jorge Luis PEREZ-MEDINA

Master 2 MATIS

UFR Informatique et Mathématiques appliquées

Rapport de projet de DEA 2005 – 2006

Spécifications techniques de la démarche Symphony dans le cadre des systèmes de réalité augmentée

ORGANISATION D'ACCUEIL

Laboratoire CLIPS – IMAG, Equipe IIHM
Laboratoire LSR – IMAG, Equipe SIGMA

RESPONSABLES DANS L'ORGANISATION D'ACCUEIL

Madame S. DUPUY-CHESSA
Madame D. RIEU

REMERCIEMENTS

La rédaction de ce rapport venant boucler cette belle historie que constitue cette première année de recherche, je tiens donc à utiliser le fait que cette page soit la première a être lue pour exprimer mes sincères remerciements a tous qui ont joue un rôle dans ma formation de DEA et la concrétisation de ce travail.

Je ne peux commencer sans évoquer mes responsables de Stage Sophie Dupuy-Chessa et Dominique Rieu. Merci Sophie de m'avoir supporté dans le DEA et pardon pour toutes les lectures et corrections pendant cette première année de travail.

Je souhaite aussi remercier mes professeurs de MATIS, puisque j'ai reçu des classes exceptionnelles.

Un grand merci à tous dans l'équipe IIHM, en particulier, à David Juras, Guillaume Godet-bar et Jullien Bouchet mes colocataires de bureau par l'appui reçu pendant toute cette année de travail.

De même manier je dois remercier mes nouveaux amis de MATIS, Wei, Rami, Marc, Charlotte, Thanh Thanh et Jacqueline, dans nos classes nous passons des moments inoubliables.

Je ne peux terminer sans remercier mes beaux-parents qui m'ont soutenu par leur patience et m'ont toujours encouragé.

En fin, ma petite amie Viviana, qui, malgré la distance, m'a porté de toutes ses forces et a beaucoup donné pour que je puisse terminer cette petite aventure.

TABLE DE MATIERES

TABLE DE FIGURES

LISTE DES TABLEAUX

Chapitre I. Introduction Générale : Contexte et Problématique

Introduction

Les systèmes informatiques sont aujourd'hui en pleine évolution entre l'ordinateur de bureau et les systèmes informatiques mobiles. L'hétérogénéité et la diversité du matériel a permis l'adoption d'une grande variété de dispositifs hautement spécialisés, en donnant naissance au développement de diverses tendances de systèmes interactifs. Plusieurs termes ont été introduits pour caractériser cette nouvelle tendance, cependant, nous utilisons le terme « Systèmes Mixtes » (SM) introduit dans [Dubois 01] pour nommer tout système interactif dont la caractéristique est de combiner des mondes réels et virtuels. Cette définition repose sur deux types d'augmentation par l'utilisateur : la Virtualité Augmentée (VA) et la Réalité Augmentée (RA). Dans le premier cas, l'interaction de l'utilisateur avec le monde numérique est enrichie par l'ajout d'outils et d'actions du monde physique. Un des exemples les plus connus est celui des MediaBlocks [ISHI 97], qui permettent de manipuler des séquences vidéo en agençant des cubes réels. On dit dans ce cas que l'exécution ou les actions de l'utilisateur sont augmentées. Quant au second, notre contexte d'étude, les systèmes de RA enrichissent l'interaction de l'utilisateur avec le monde physique au moyen de données et services offerts par le monde numérique. Typiquement, la plupart des systèmes de réalité augmentée utilisent des casques semi-transparents ou Head Mounted Display (HMD) pour surimposer, dans le champ de vision de l'utilisateur, des objets numériques sur le monde physique. On citera à titre d'exemple l'application Chirurgicaux Assistés par Ordinateur (GMCAO) comme Casper-V2 [DUBO 01]. Dans ce système le chirurgien peut visualiser la trajectoire idéale calculée par l'ordinateur alors qu'il opère. On dit dans ce cas que la perception de l'utilisateur est augmentée.

1.1. Problématique

Les systèmes de Réalité Augmentée suscitent aujourd'hui l'intérêt de domaines très divers, par exemple : ergonomie, interaction Homme-Machine, génie logiciel, etc. Leur conception devient alors un processus complexe visant à coordonner des nouveaux contextes, des sources d'information, des applications préexistantes et en évolution. Cette complexité rend nécessaire la définition d'un processus adapté qui minimise la difficulté de la conception et maximise la réutilisation et la sélection des technologies existantes.

Dans le contexte du développement des systèmes, la majorité des travaux de recherche a adopté une approche empirique, basée sur le développement de prototypes expérimentaux. La difficulté de conception des systèmes de réalité augmentée, en particulier, la sélection des dispositifs d'interaction diverses et de styles d'architecture logiciel rend nécessaire la formalisation de méthodes, modèles et techniques, proches du modèle de raisonnement du développeur.

Dans le cadre de notre mémoire la solution que nous proposons pour développer les systèmes de réalité augmentée est de proposer une démarche de développement, en particulier pour les parties techniques. Nos propositions se basent sur une démarche de développement de système d'information basée sur la réutilisation de composants.

Symphony est une méthode pour la conception de l'architecture logicielle qui intègre des règles de conception et des patrons de conception, Symphony dont l'objectif est de pallier les manques technologiques. En conséquence, Symphony se présente sous la forme d'un guide méthodologique offrant une solution basée sur l'utilisation de composants dès les phases amont du processus. Cette démarche est centrée sur des patrons, formalisée par un ensemble de patrons orientés vers le « processus » (c'est-à-dire de savoir-faire) combinés à des patrons orientés vers les « produits » (c'est-à-dire de savoir) réutilisables dans de nombreuses applications.

1.2. Objectifs

Dans le cadre de ce mémoire, notre objectif est donc de proposer des aides pour la conception des systèmes de Réalité Augmentée qui prennent en compte dans les besoins techniques les spécificités de la Réalité Augmentée. Dans ce but, nous avons suivi la démarche scientifique suivante :

- Etude de l'état de l'art sur les architectures logicielles pour les systèmes de réalité augmentée.
- Formalisation du processus d'analyse et de conception des architectures techniques basées sur la réutilisation des composants.

Finalement, le résultat de ce mémoire se présente sous la forme d'un catalogue de patrons en utilisant comme nomenclature le formalisme P-Sigma.

1.3. Structure du mémoire

Dans le **deuxième chapitre**, nous présenterons dans un premier temps les systèmes interactifs et leur classification. Dans cette partie, nous poursuivrons notre démarche en étudiant les systèmes de réalité augmentée et nous étudierons des aspects inhérents à ces techniques d'interaction.

Au **troisième chapitre**, nous aborderons les aspects techniques pour la réalisation des systèmes de RA à travers la conception logicielle et en particulier les architectures logicielles. Nous commencerons avec un état de l'art de la terminologie associée aux architectures, ensuite, nous étudierons les diverses architectures logicielles pour les systèmes interactifs.

Dans le **quatrième chapitre**, nous décrirons la méthode Symphony, ses caractéristiques principales et son cycle de vie, nous détaillerons sa branche technique. Ensuite, nous aborderons les concepts inhérents à la réutilisation des composants, puis aux patrons logiciels et leurs principales caractéristiques et

finalement nous présenterons le formalisme P-Sigma utilisé pour la représentation des systèmes de patrons.

Le **cinquième chapitre**, présente notre formalisation de la branche technique de la méthode Symphony pour les systèmes de Réalité Augmentée. Cette approche se concentre sur la présentation d'un système de patrons en utilisant le formalisme P-Sigma.

Enfin, dans le **sixième chapitre**, nous concluons cette étude en présentant une synthèse des contributions de l'approche proposée. Des perspectives figurent également dans cette dernière partie.

Chapitre II. Les Systèmes de Réalité Augmentée

L'évolution rapide des technologies informatiques et des dispositifs d'interaction (casques de visualisation, interfaces tangibles, gants tactiles et autres) ont permis la conception de systèmes qui permettent de fusionner des entités réelles et virtuelles. Considérons l'exemple d'un réfrigérateur augmenté. iHome [Whirlpool 01] est un réfrigérateur dans lequel sont intégrés un écran et un ordinateur, qui permettent aux utilisateurs d'accéder à Internet depuis le réfrigérateur pour acheter en ligne des produits. Chaque fois qu'un produit est retiré du réfrigérateur, l'ordinateur met à jour l'état des stocks.

Dans le domaine IHM différents termes on été introduits pour cette nouvelle tendance : Réalité Augmentée, Surfaces Augmentées, Interfaces Tangibles, Réalité Mixte, Utilisateur Augmenté, Environnement enrichi, etc. Tous ces termes décrivent des systèmes ayant un point commun, qui est la combinaison du monde réel et du monde virtuel.

L'interaction de l'utilisateur avec ces systèmes est modifiée par un supplément qui n'est pas présent dans une interaction classique entre l'utilisateur et un système informatique ou monde réel. Étant donné que notre intérêt est l'étude des systèmes de réalité augmentée, ce chapitre présente leurs principales caractéristiques et des aspects inhérents à leurs techniques d'interaction.

2.1. Le paradigme de la Réalité Augmentée

Dans la littérature, on note que le premier système de réalité augmentée a vu le jour avec les travaux de Sutherland, basé sur un casque suivi par un capteur de mouvement [Yvan 65][Yvan 68]. Son travail prétendait que l'utilisateur pouvait visualiser et naviguer autour d'éléments virtuels positionnés dans un espace réel.

Plus tard, le terme Réalité Augmentée a été introduit par [Caudel 92] et dans le domaine IHM par [Wellner 93]. La forte réduction des coûts des dispositifs matériels, l'évolution du graphisme et la disponibilité d'outils simples et évolués [Kato 99] ont permis l'étude et l'investigation de techniques d'interaction appropriées et des interfaces d'utilisateurs pour faciliter l'usage des systèmes de Réalité Augmentée.

La Réalité Augmentée peut être utilisée dans de nombreux domaines d'application. Durant ces dernières années, un certain nombre de prototypes et d'applications a été créés dans les domaines tels que l'informatique graphique, l'interaction homme-machine, la vision, les travaux autour du travail collaboratif, incluant les secteurs suivants : architecture et planification urbaine [Webster 96] [Thomas 99], design [Klaus 95][Kato 00], ingénierie et planification de production [Caudel 92][Ahlers 94][Friedrich 02], automobile [Klinker 02], robotique [Milgram 93], médecine et chirurgie [Taylor 92][Bainville 95][Cinquin 95][Grimson 96][Dubois 01], éducation, apprentissage et formation [Hedley 01][Kaufmann 03][Molineros 99], l'expression artistique [Cheok 02], jeu [Thomas 00][Murakami 01][Piekarski 02], loisir [Ohshima 98], etc. Actuellement, les applications se tournent vers la mobilité et le support multi-utilisateurs.

2.2. Caractéristiques de la RA

Autour de tout ce travail, nous pouvons constater que la RA est un paradigme d'interaction qui est né de la volonté de fusionner les capacités de traitement d'un ordinateur dans l'environnement réel de l'utilisateur. Une première approche pour cerner ce paradigme de RA consiste à le comparer avec celui de la réalité virtuelle (RV). La réalité virtuelle est définie comme un environnement interactif, générée par un ordinateur et dans lequel une personne est immergée [Aukstakalnis 92]. Cependant, l'immersion en RV est totale, à l'opposé de la RA qui ne nécessite pas obligatoirement la gestion par l'ordinateur d'une scène réaliste, puisque les données ajoutées au monde réel peuvent servir à comprendre celui-ci.

2.2.1. Système Mixte

Dans le domaine de l'IHM, l'objectif de la RA est de briser la frontière entre le monde réel, incluant l'utilisateur et son environnement, et le monde informatique. Une caractéristique fondamentale des systèmes de RA est que des éléments du monde réel interviennent dans l'interaction. L'existence de deux mondes (réel ou virtuel) implique que le but de la tâche, peut appartenir à l'un des deux. Dans ce sens, Milgram [Milgram 94] interprète la RA dans un continuum linéaire qui va du réel au virtuel. Il définit le terme de Réalité Mixte l'intervalle entre le réel et le virtuel.

De même manière, Dubois [Dubois 01] a proposé sa propre taxonomie basée sur l'objet de la tâche et le type d'augmentation[1]. Par rapport à l'objet de la tâche Dubois a défini deux continuités, La Réalité Augmentée et la Virtuel Augmentée. Dans son approche le point crucial réside dans l'identification de la tâche supportée par le système que l'on souhaite analyser ou concevoir.

La figure 1 nous permet de distinguer, les deux courants qui correspondent aux 2 extrêmes d'un continuum entre mondes réels et virtuels.

[1] Nous présenterons le type d'argumentation dans la section suivante

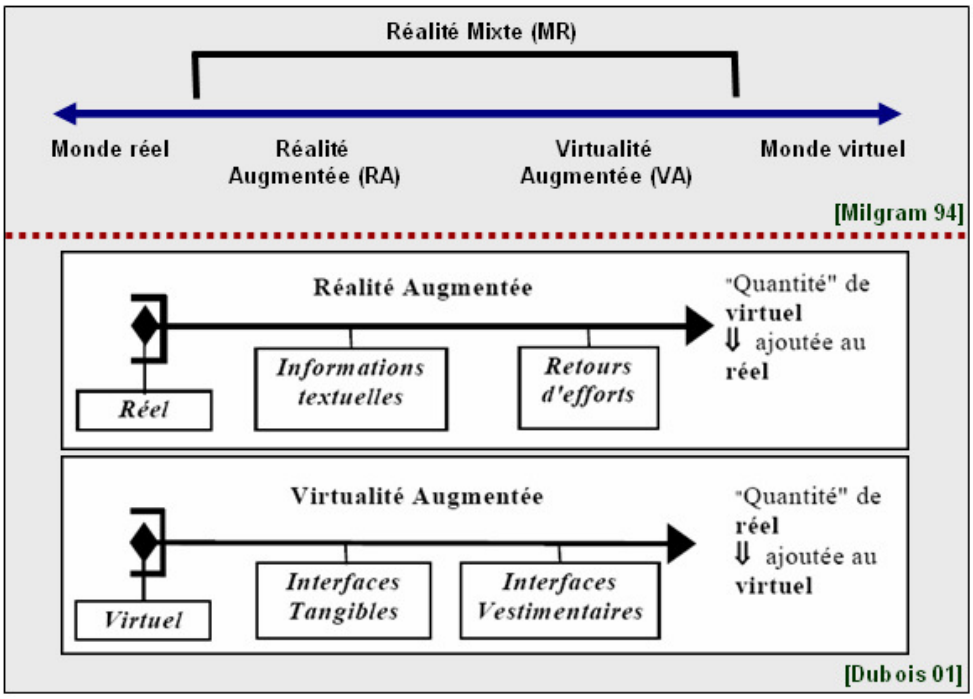


FIGURE 1 : CLASSIFICATION DES SYSTEMES MIXTES [Milgran 94] et [Dubois 01]

Les systèmes de **Virtualité Augmentée (VA)** : dans ces systèmes, l'interaction d'un utilisateur avec le monde numérique est enrichie par l'ajout d'outils et d'actions du monde physique. En VA, les systèmes peuvent être basés sur l'utilisation de briques physiques localisées par une caméra, ou de dispositifs dédiés localisés par des champs magnétiques. Nous pouvons citer le jeu Touch-Space [Cheok 02] comme un exemple de ce type de système. Cette application permet la manipulation d'entités numériques à travers d'objets physiques. L'interaction est basée sur l'instrumentation d'une surface (monde numérique) sur laquelle les utilisateurs se déplacent. Les liens sont réalisés par des marqueurs reconnus via une caméra. Les utilisateurs peuvent se déplacer d'un lieu à un autre en passant d'une case à une autre. On dit dans ce cas que l'exécution ou les actions de l'utilisateur sont augmentées. La figure 2 permet d'illustrer cet exemple.

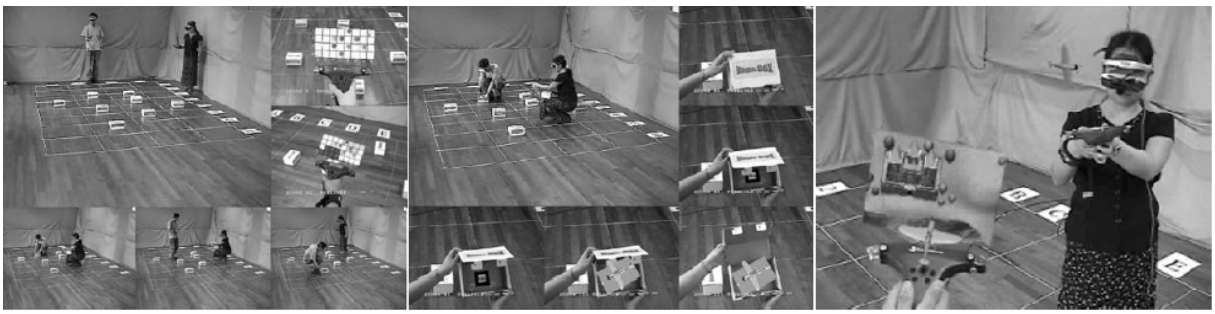


FIGURE 2 : EXEMPLE DE VIRTUALITE AUGMENTEE TOUCH-SPACE [Cheok 02]

Dans le domaine de l'IHM différentes manières ont été proposées afin d'interagir avec des documents numériques. Par exemple, **GaspDraw** [Ishii 97] propose le contrôle direct d'objets virtuels au moyen de « bricks ». Ces éléments peuvent être attachés aux objets virtuels pour servir de traducteurs dédiés, chacun en occupant son propre espace. De cette façon, les interfaces classiques de bureau sont remplacés par des interfaces plus adaptées à l'utilisateur.

Les systèmes de **Réalité Augmentée (RA)** : dans ces systèmes, l'interaction de l'utilisateur avec le monde physique est enrichie par l'ajout de données et de services offerts par le monde numérique (ordinateur), dans l'environnement naturel de l'utilisateur ; La RA permet l'augmentation de l'environnement physique de l'utilisateur par des informations numériques. Dans ces systèmes, les données peuvent êtres affichées dans un casque semi-transparent, projetées directement dans le monde physique, superposés à une vidéo du monde réel ou affichée sur un dispositif incrusté dans le monde physique. L'application Mémo développée par J. Bouche [Bouche 03] est un exemple de ce type de système. Elle permet à l'utilisateur de découvrir, lire, poser ou supprimer des mémos (post-its) virtuels directement dans l'environnement physique dans lequel il évolue, en utilisant une souris et/ou des commandes vocales. L'utilisateur voit au travers d'un casque semi-transparent le monde réel et des mémos numériques déposés précédemment à des endroits du monde réel. On dit dans ce cas que la perception de l'utilisateur est augmentée. La figure 3 permet d'illustrer cette application.

FIGURE 3 : APPLICATION DE REALITE AUGMENTEE MEMO [Bouche 03]

2.2.2. Type d'augmentation

Les deux continuums présentés par [Doubois 01], distinguent des systèmes qui augmentent soit l'interaction avec le monde réel grâce à l'ordinateur, soit l'interaction avec l'ordinateur en exploitant des éléments du monde réel. Cependant, l'identification du type d'augmentation permet facilement de comparer deux systèmes et d'envisager des versions différentes d'un système donné. Deux types d'augmentation sont identifiées : l'exécution augmentée et l'évaluation ou la perception augmentée (voir la figure 4).

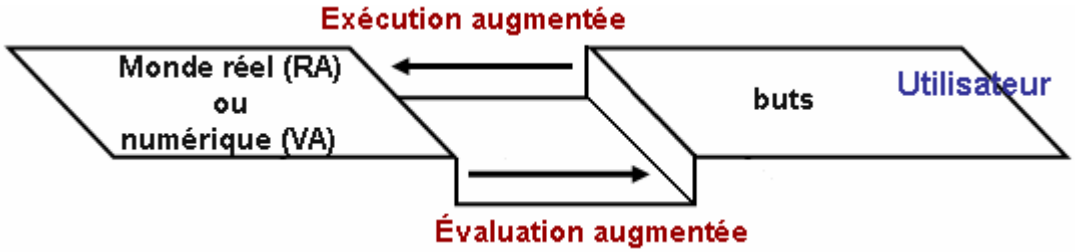


FIGURE 4 : TYPES D'AUGMENTATION [Doubois 01]

Dans **l'exécution augmentée**, le système augmente l'exécution pour permettre à l'utilisateur d'effectuer des tâches dans le monde réel d'une façon nouvelle ou par l'amélioration la qualité de réalisation de la tâche. Par exemple, **DigitalDesk** [Wellner

91] est un système qui permet à l'usager de digitaliser des documents du monde réel, puis de manipuler les images en utilisant des stylos réels. Cette action n'est pas directement possible dans le monde réel : la phase d'exécution est augmentée par l'ordinateur. Un autre exemple consiste à porter un « Active Badge » qui permet d'ouvrir une porte sans la toucher.

Dans le cas de **l'évaluation** ou **la perception augmentée**, un système de RA réduit le gouffre de l'évaluation par le biais de l'ajout de données qui ne sont pas perceptibles lors de la réalisation de la tâche sans le système. Le Musée Augmenté de Rekimoto [Rekimoto 95] est un exemple de ce type de perception. Dans ce cas, l'utilisateur peut visualiser des informations pouvant concerner l'auteur d'une œuvre d'art.

2.3. Conclusion

Nous avons présenté une classification des systèmes mixtes afin de préciser notre contexte d'étude des systèmes de réalité augmentée. L'étude de sa classification et des aires d'application, nous permet d'indiquer que le développement de tels systèmes diffère conformément aux dispositifs utilisés et au contexte d'interaction. C'est pourquoi, la conception logicielle d'un système de réalité augmentée doit s'appuyer sur un modèle d'architecture technique afin d'une part de favoriser sa réutilisation, flexibilité, maintenance et portabilité, et d'autre part de garantir l'utilisation maximale des technologies existantes.

Dans le chapitre suivant nous aborderons les aspects techniques pour la conception et la réalisation des systèmes de RA, nous étudierons les diverses architectures logicielles pour les systèmes interactifs applicables aux systèmes de RA.

Chapitre III. Concepts techniques pour les systèmes de Réalité Augmentée

3.1. Architecture logicielle

3.1.1. Définition

La complexité des systèmes interactifs, ont augmenté la nécessité et l'importance d'approches systématiques d'architecture logicielle. Le terme « architecture logicielle » a été proposé comme une discipline par [Perry 92], donnant naissance à la profession d'architecte de logiciel. Il est impossible de donner une définition unique d'architecture de logiciel, cependant, nous adopterons la définition utilisée par Booch, Rumbaugh et Jacobson dans [Booch 99] :

« …An architecture is the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural style that guides this organization - these elements and their interfaces, their collaborations, and their composition »

Comme complément à cette définition, nous citons l'IEEE [IEEE 1471] qui définit une architecture de logiciel de la manière suivante :

« …l'organisation fondamentale d'un système incarnée dans ses composants, les relations entre eux et l'environnement et les principes qui orientent sa conception et son évolution ».

Ces deux définitions tournent autour des termes suivants : Structure, comportement, collaborations et directives de composition. Avec la référence à la structure et au comportement nous pouvons affirmer qu'il est indispensable de penser en abstractions qui permettent formaliser des structures basé composants.

Dans ce sens, un composant est donc une unité d'abstraction, ce peut être un service, un module, une bibliothèque, un processus, une procédure, un objet, une application, etc. Sa conduite observable fait partie de l'architecture puisqu'elle détermine ses interactions possibles avec d'autres composants. Aussi, les composants sont des abstractions qui regroupent un certain nombre de fonctionnalités, par exemple : ils synthétisent les concepts et fonctions du domaine d'un système et traitent la gestion et le contrôle d'interaction de l'application. La décomposition fonctionnelle du système consiste à exprimer les besoins fonctionnels du système en des unités plus simples. Finalement, le résultat de cette activité est une structure appelée architecture conceptuelle.

Dans le cadre de ce mémoire, notre point d'intérêt est qu'une architecture conceptuelle d'un système logiciel ou une collection de systèmes consiste en des décisions importantes de conception à propos des structures logicielles et les interactions entre celles-ci. Les décisions de conception soutiennent un jeu désirable de qualités que le système devrait supporter (par exemple : modularité, portabilité, etc.). De la même manière, ces décisions fournissent une base conceptuelle pour le développement de système, son support, sa réutilisation, sa maintenance et son évolution.

Les intérêts s'expriment en termes d'exigences fonctionnelles, économiques et techniques. L'architecture y répond par l'intégration de plusieurs styles de développement informatique qu'elle adapte aux éléments logiciels d'un contexte existant. Un processus centré sur l'architecture impose le respect des décisions d'architecture à chaque étape de construction du modèle. L'architecture est donc la condition qui préside à l'intégrité d'un projet complexe, car elle permet la structure et la cohérence des points de vue.

L'organisation d'un modèle cohérent permet d'établir des règles précises de croissance, sans dépasser les capacités de compréhension humaine. L'architecte utilise la structure du modèle pour communiquer et contrôler les liens complexes entre ces parties. Le processus centré sur l'architecture permet donc de dégager des opportunités qui poursuivent les buts suivants :

- Les modèles correctement organisés offrent des moyens de réutilisation du logiciel à large échelle ;
- Leur découpage facilite l'élaboration de métriques, l'estimation et la répartition du travail entre équipes ;
- Les points de vue cohérents facilitent l'étude d'impact suivant les différents axes de changement qui sont l'évolution des fonctions, de la structure des entités, des techniques, des configurations d'exploitation, et des versions logicielles ;
- La documentation apportée par les modèles facilite les tests, l'intégration, et aide à identifier la source des erreurs.

3.1.2. Modèles d'architecture pour les systèmes interactifs

Les modèles d'architecture définissent l'organisation logicielle d'un système interactif permettant ainsi de disposer d'une meilleure modularité. Un principe commun à tous ces modèles est la séparation entre le noyau fonctionnel qui implémente les concepts propres à un domaine d'application particulier et l'interface qui présente ces concepts à l'utilisateur et qui lui permet de les manipuler [Nigay 94]. Ce principe très séduisant, puisque qu'il doit permettre en théorie de modifier l'interface sans affecter le noyau fonctionnel et inversement, est en réalité difficile à concrétiser pour la totalité de l'interface. En pratique et dans la majorité des cas, seule une partie de l'interface est réellement séparée du noyau fonctionnel.

Nous présentons ci-dessous les principaux modèles d'architecture logicielle pour les systèmes interactifs. Nous décrirons les modèles monolithiques : Seeheim, Arch. Ensuite, nous présenterons le modèle multi-agent Modèle Vue Contrôleur (MVC).

Puis, nous décrirons le modèle hybride PAC-Amodeuos et enfin le dernier est dédié au modèle AMF.

3.1.2.1. Le modèle Seeheim

Principes de base

Le modèle de Seeheim [Green 85] est le premier modèle global proposé permettant de modulariser l'interface d'une application. Seeheim considère qu'un système interactif est constitué de deux acteurs (l'application et l'utilisateur) qui dialoguent par l'intermédiaire de l'interface. Ce modèle se décompose en trois modules : la présentation, le contrôle du dialogue et l'interface du noyau fonctionnel. Ce modèle est illustré par la figure 5. Ce type de division est particulièrement adapté aux interfaces à base de menus et d'écrans de saisie, ou pour la construction de l'interface de contrôle des applications à interaction graphique.

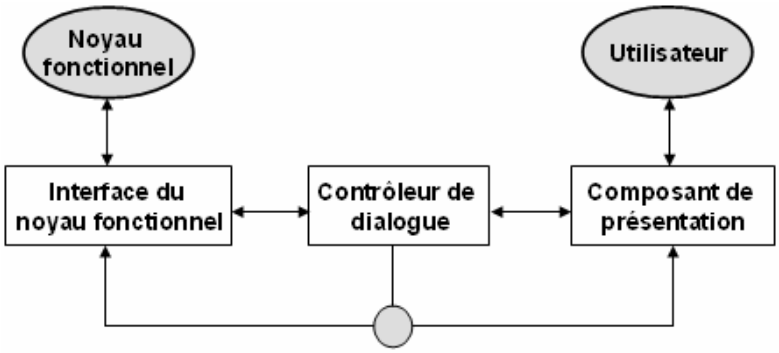


FIGURE 5 : MODELE SEEHEIM

Le composant de présentation est le composant qui est en contact direct avec les médias (On entend ici par média, dispositif physique ou périphérique) d'entrée et de sortie. Dans la perspective de l'utilisation de l'interface ce composant est chargé de présenter à l'écran les informations et d'interpréter les actions élémentaires de l'utilisateur [Baudel 95]. Il se charge d'une part de la production effective des sorties (affichage, production de sons, etc.) à partir des informations qu'il reçoit du contrôleur de dialogue, et d'autre part, de la réception et du traitement (d'un premier niveau) des événements d'entrée induits par les actions de l'utilisateur pour ensuite transmettre les informations résultantes au contrôleur de dialogue.

Le contrôleur de dialogue est responsable de la gestion des relations entre le noyau fonctionnel et le composant de présentation. Il est un médiateur qui contrôle la séquence des opérations au niveau des actions de l'utilisateur, des appels aux fonctions du domaine, et réalise un pont entre les objets concrets de l'interface (composant de présentation) et les concepts du domaine (noyau fonctionnel).

L'interface du noyau fonctionnel est la couche adaptative entre le système interactif et le noyau fonctionnel. Il contient en général une description des entités du domaine qui sont directement liées à l'interface ainsi qu'une description des procédures du noyau fonctionnel qui peuvent être invoquées par le contrôleur de dialogue.

Dans la figure 5, la relation qui lie le composant de présentation à l'interface du noyau fonctionnel montre en fait que la séparation totale entre l'interface et le noyau fonctionnel est difficile à atteindre. En théorie, toutes les informations en provenance du noyau fonctionnel et devant être présentées à l'utilisateur doivent être d'abord traitées par le contrôleur de dialogue avant d'être transmises au composant de présentation. En pratique, il s'avère que certaines informations ne font que transiter par le contrôleur de dialogue sans subir aucun traitement de la part de celui-ci. On constate alors une communication directe entre le composant de présentation et l'interface du noyau fonctionnel (voir même le noyau fonctionnel lui-même). Ceci est particulièrement vrai pour des opérations d'affichage complexe et/ou exigeant un temps de réponse minimal. Le contrôleur de dialogue se trouve alors court-circuité dans un souci d'efficacité.

Conclusion

Les avantages de cette approche ont été :

- D'ouvrir la voie vers la séparation entre interface et application, les problèmes de l'application et de l'interface peuvent être isolés et traités séparément;
- D'apporter un support à la conception d'architecture de systèmes en proposant un processus de conception descendant et en séparant clairement les niveaux d'abstraction de l'application.
- De faciliter l'évolution de l'application et de l'interface indépendamment. Cette séparation facilite le traitement de chaque partie séparée, la réutilisation et la portabilité du système, facilitant ainsi le développement rapide de prototypes.
- De permettre le partage des interfaces entre différentes applications.

Cependant, ce modèle d'architecture est inadapté à l'aspect dynamique, concurrent et parallèle de l'interaction homme-machine qui caractérise la manipulation des systèmes interactifs, Seeheim considère uniquement une architecture modulaire, sans prêter attention à la relation entre le modèle conceptuel et le modèle mental de l'utilisateur, c'est par cela que l'interface de l'utilisateur devrait être capable de représenter les objets de l'application de manière à ce que l'utilisateur comprenne la relation entre la représentation et le monde réel.

3.1.2.2. Le modèle ARCH

Principes de base

Comme le montre la figure 6, le modèle Arch [UIMS 92] s'appuie sur les composants fonctionnels du modèle Seeheim. Le principe véhiculé par ce modèle, est de séparer l'interface utilisateur du noyau fonctionnel (logique de l'application). En pratique, le noyau fonctionnel ne doit avoir aucune connaissance des fonctionnalités relevant de l'interface utilisateur pour faciliter une conception itérative de l'interface, ainsi que pour favoriser la réutilisation et la portabilité du logiciel.

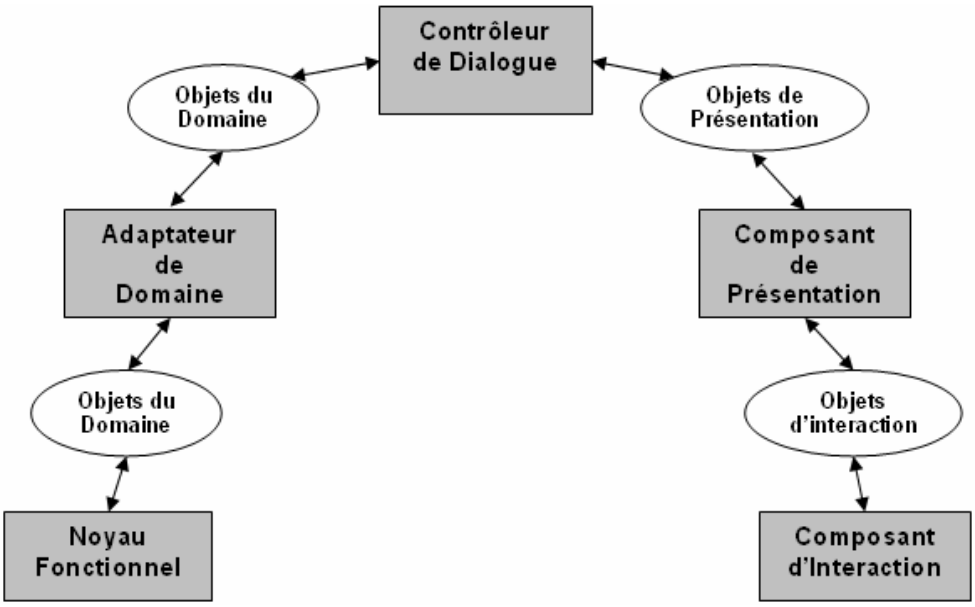


FIGURE 6 : MODELE DE REFERENCE ARCH

Les pieds de l'arche constituent les composants imposés par la réalité : le noyau fonctionnel et le composant d'interaction. **Le noyau fonctionnel** implémente les concepts propres du domaine de l'application et le **composant d'interaction** est en contact direct avec l'utilisateur et peut être mis en œuvre au moyen d'objets d'interaction (widgets) grâce à une boîte à outils.

L'adaptateur de domaine établit un pont entre le contrôleur de dialogue et le noyau fonctionnel. Ce composant permet d'ajuster les différences de modélisation des objets conceptuels entre les deux composants qui l'entourent. Par exemple, lorsqu'une erreur sémantique intervient au sein du noyau fonctionnel, il se charge de la communiquer au contrôleur de dialogue sous une forme adéquate.

Le contrôleur de dialogue est la pierre angulaire du modèle puisque ce composant a la charge de gérer l'enchaînement des tâches et assure le lien entre les composants de présentation et l'adaptateur de domaine.

Le composant présentation est un composant qui permet au contrôleur de dialogue de manipuler des objets d'interaction virtuels plutôt que les objets d'interaction d'une boite à outils réels (composant d'interaction). Il garantit l'indépendance du contrôleur de dialogue vis-à-vis du composant d'interaction qui recouvre la boîte à outils de la plate-forme d'accueil. Ce composant permet de définir une boîte à outils virtuelle qui permet d'assurer la propriété de portabilité. Ceci facilite le portage des applications d'un système graphique (ayant sa propre bibliothèque d'objets d'interaction) à un autre (par exemple du Macintosh à Windows, de Windows à X-Window etc.).

Les objets du domaine ils sont utilisés par le noyau fonctionnel et par l'adaptateur de domaine. On peut distinguer deux types d'objets du domaine : ceux qui n'ont aucun lien direct avec l'interface, et ceux qui sont reliés à l'interface. Le premier type d'objets est utilisé par le noyau fonctionnel pour mettre en oeuvre les concepts propres au domaine. Quant au second, il est utilisé par l'adaptateur de domaine pour mettre en relation ces concepts avec l'interface.

Les objets de présentation sont des objets virtuels d'interaction. Ils véhiculent un concept interactif sans en décrire les détails de mise en oeuvre. Par exemple pour représenter le champ Sexe d'une base de données, on utilisera un objet de présentation qui permet de réaliser un choix exclusif parmi un ensemble de deux valeurs.

Les objets d'interaction constituent la mise en oeuvre dans tous leurs détails des concepts interactifs. En général, il s'agira des objets d'interaction fournis par une boîte à outils particulière. Ils possèdent un style de présentation propre (look and feel) et sont liés de manière concrète à un ou plusieurs médias. Par exemple, à l'objet de présentation précédent correspondra un ensemble de deux boutons radios, manipulables par une souris. En général, à un objet de présentation pourra correspondre un ou plusieurs objets interactifs.

Dans le modèle Arch la satisfaction de la propriété de portabilité impose des contraintes au réalisateur. Elle exige souvent l'utilisation exclusive des services qui constituent le plus petit dénominateur commun entre les plates-formes d'accueil et interdit l'exploitation de toute la richesse d'une plat-forme donnée. Avec le composant présentation, il est cependant possible d'enrichir une boîte à outils suivant un mécanisme analogue à la délégation sémantique pour l'adaptateur du noyau fonctionnel. Finalement, Arch est un modèle de référence. Avec le méta modèle **Slinky** qui permet de faire varier l'importance relative de ses cinq composants, il peut être adapté aux contraintes d'un environnement particulier.

Le méta modèle Slinky

Le terme Slinky provient d'un jouet flexible (figure 7) qui une fois mis en mouvement voit son centre de gravité se déplacer suite à un changement dans la répartition de sa masse.

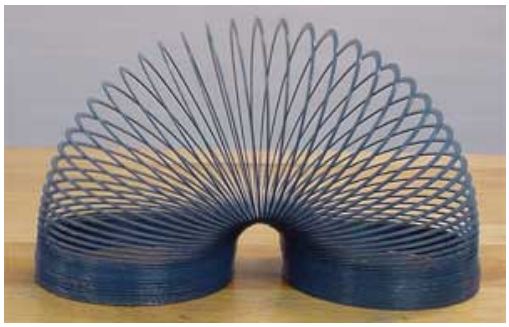


FIGURE 7 : LE JOUET SLINKY

Cette métaphore est utilisée dans le modèle Arch pour représenter la modification de la répartition des fonctionnalités à travers les différents composants du modèle, d'une instance à l'autre. Selon les objectifs visés lors de la conception et du développement du système interactif, on peut constater une migration des fonctionnalités d'un composant vers un autre [Bellik 95]. Par exemple, une fonctionnalité réalisée au sein du contrôleur de dialogue dans un système interactif particulier peut dans un autre système être réalisé au sein de l'adaptateur de domaine. L'importance, en terme de fonctionnalités, des différents composants du modèle Arch varie donc d'une instance à l'autre. Dans des cas limites, on peut même constater la disparition d'un

composant. Slinky constitue donc une belle métaphore, pour représenter cette migration des fonctionnalités entre les différents composants.

Conclusion

Le modèle Arch a repris le modèle de Seeheim et l'a affiné en éclatant le composant de présentation en deux composants : un composant de présentation qui définit des concepts d'interaction assez abstraits et un composant d'interaction qui implémente ces concepts de manière concrète. Cet éclatement dénote un souci d'instaurer une plus grande portabilité d'un environnement graphique particulier à un autre. Par ailleurs, l'utilisation de la métaphore du méta modèle Slinky correspond à une tentative de mieux représenter la réalité des systèmes interactifs actuels. En ce sens, on peut déplorer que le modèle Arch soit plus un modèle qui tente d'expliquer les architectures des systèmes interactifs actuels plutôt qu'un modèle réellement générateur de nouvelles architectures [Bellik 95].

3.1.2.3. Le modèle MVC

Principes de base

Le Modèle Vue Contrôleur (MVC) est une architecture multi-agents qui est apparu dans le cadre du langage de programmation SmallTalk dans les laboratoires de recherche Xerox PARC pour répondre aux difficultés posées par la conception d'applications fortement interactives [Krasner 88]. Il structure un système interactif en un ensemble d'agents. Comme le montre la figure 8, un agent MVC est constitué de trois facettes :

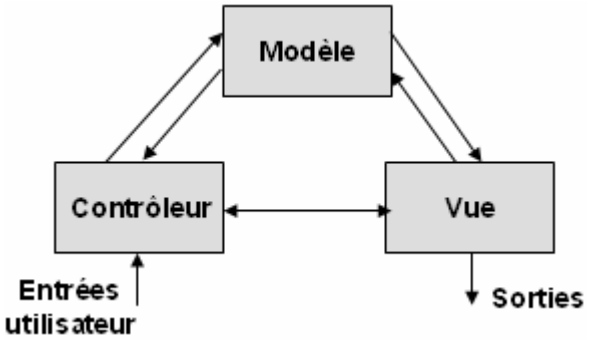


FIGURE 8 : MODELE MVC

Le modèle représente les concepts du domaine, c'est-à-dire le comportement de l'application. Il suppose la représentation des objets réels de forme totalement indépendante de sa représentation visuelle. Par exemple, il peut consister en un simple entier (modèle d'un compteur) ou être un objet, instance d'une classe beaucoup plus complexe requise par le système.

La vue correspond à la perception qu'a l'utilisateur du modèle. Elle offre une représentation en sortie au niveau de l'interface utilisateur (affichage, son, haptique, etc.). Les résultats renvoyés par le modèle sont dénués de toute présentation mais sont présentés par les vues. Plusieurs vues peuvent afficher les informations d'un

même modèle. Elle peut être conçue en n'importe quel langage de présentation et se charge simplement de la représentation visuelle du système.

Le contrôleur prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle. Il traite les entrées de l'utilisateur. Il peut envoyer des messages au modèle pour l'avertir d'une action de l'utilisateur ou directement à la vue pour lui demander de mettre à jour une sortie suite à un événement d'entrée. Il est important de noter que ce composant n'effectue aucun traitement, ne modifie rien sur les données, il analyse la requête du client et se contente d'appeler le modèle adéquat et de renvoyer la vue correspondant à la demande. Le contrôleur permet de gérer les actions de l'utilisateur sur une vue.

Conclusion

Parmi les caractéristiques de ce modèle, il se trouve que la cohérence du système est assurée par la communication des trois objets à travers d'un mécanisme de message. Cela apporte de la clarté à l'architecture qu'il impose. Chaque objet du système possède une vue et un contrôleur, ce modèle s'est avéré très commode pour la création d'interfaces graphiques. Les avantages du modèle résident dans la séparation de l'affichage, de la gestion des événements utilisateur et de la représentation interne des objets manipulés. Ceci simplifie la tâche du développeur qui tenterait d'effectuer une maintenance ou une amélioration sur le projet.

Cependant, l'aspect plus négatif de MVC est de grouper la gestion des événements des applications avec sa fonctionnalité, cela complique la réutilisation de l'application comme composants séparés.

3.1.2.4. L'agent PAC

Principes de base

Le modèle de Présentation Abstraction Contrôleur (PAC) [Coutaz 87] est un modèle multi-agent qui repose sur deux principes directeurs : le concept d'agents réactifs à facettes et l'organisation hiérarchique de ces agents[2]. Un agent PAC est constitué de trois facettes représentées par la figure 9.

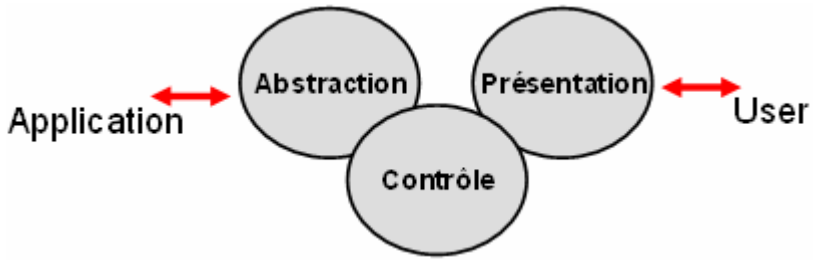


FIGURE 9 : UN EXEMPLE D'AGENT PAC

[2] *Un agent* [Feber 89] est une entité informatique, réelle ou abstraite, indépendante à durée de vie limitée. Un *agent réactif* est alors défini comme un agent dont le comportement est la conséquence de ses observations et de ses interactions avec l'utilisateur.

La présentation est chargée de la transmission (visuelle, sonore, etc.) des informations à l'utilisateur et de l'acquisition des entrées de l'utilisateur. La Présentation interprète les événements résultant des actions physiques que l'utilisateur applique sur l'agent via des dispositifs d'entrée, et génère des événements qui traduisent des actions physiques sur les dispositifs de sortie.

L'Abstraction représente la vue de l'application à travers de laquelle l'agent accède aux services sémantiques de l'application. Il est le coeur fonctionnel de l'agent; il définit la compétence de l'agent indépendamment des considérations de présentation.

Le Contrôle est l'unique élément qui se charge de la communication avec d'autres agents PAC et qui traite aussi le dialogue et la correspondance entre l'Abstraction (qui représente le noyau fonctionnel) et la Présentation d'un agent.

Conclusion

PAC permet un dialogue multi-fil (plusieurs dialogues en parallèle) grâce à la hiérarchie des agents PAC et à l'autonomie relative des agents dans l'interprétation des événements et dans la gestion de leur état. Cette décomposition hiérarchique permet d'accélérer les retours de l'application à l'utilisateur, indispensables pour la manipulation directe des systèmes interactifs. Cependant, la perception de l'interface globale est difficile pour le concepteur de nouvelles applications car elle est répartie entre les différents composants « Présentation » de chaque agent.

3.1.2.5. Le modèle PAC-Amodeus

Principes de base

Le modèle PAC-Amodeous [Nigay 94] est l'intégration du modèle multi-agent PAC au modèle fonctionnel Arch. Dans le modèle PAC-Amodeus, le Contrôleur de Dialogue d'Arch est organisé en une hiérarchie d'agents PAC, comme l'illustre la figure 10.

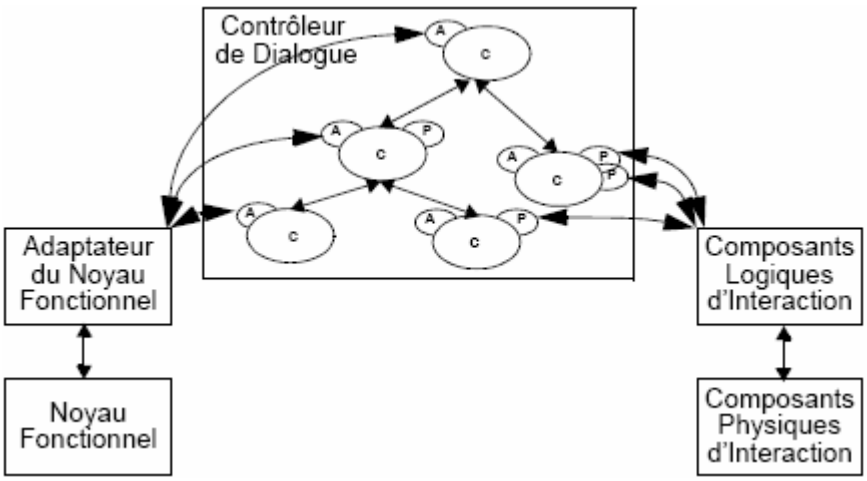


FIGURE 10 : MODELE PAC-AMODEUS

La facette Abstraction de chaque agent est en relation avec un objet du domaine (objet conceptuel) situé dans le Noyau Fonctionnel. De même, la facette Présentation de chaque agent communique avec le Composant Physique d'Interaction. Les messages échangés entre les facettes Contrôle permettent d'articuler l'activité de l'utilisateur et de réguler les interactions.

Dans PAC-Amodeus, la structuration en trois facettes d'un agent PAC n'est pas obligatoire. Ainsi un agent peut ne pas avoir de facette Présentation. Un agent peut aussi implémenter plusieurs facettes Présentation dans le cas de vues multiples. Un agent situé à un nœud de la hiérarchie n'implémentant que la facette Contrôle et/ou la facette Abstraction est un agent ciment : cet agent résulte d'une factorisation de compétences pour offrir un niveau supplémentaire d'abstraction et accroître la modularité du code. Finalement, PAC-Amodeus constitue un cadre conceptuel adapté à l'interaction multimodale en entrée [Nigay 94] et en sortie [Vernier 01].

Conclusion

Pac-Amodeus a pour objectif combiner les avantages du modèle Arch, qui intègre des aspects de génie logiciel comme la modifiabilité et la portabilité, et du modèle PAC, qui permet de structurer efficacement le contrôleur de dialogue jusque-là monolithique. L'adaptateur de domaine et la présentation de l'arch communiquent directement avec chaque agent PAc a travers son abstraction (pour le premier) et sa présentation (pour le second) en garantissant la performance des applications interactives.

3.1.2.6. Le modèle AMF

Principes de base

Le modèle Agent Multi-Facette (AMF) [Ouadou 94][Tarpin 99] est un modèle d'architecture multi-agents et multi-facettes qui permet de spécifier l'architecture logicielle d'un système interactif. Le modèle permet de concevoir des éléments réutilisables et peut être étendu et adapté aux besoins d'applications spécifiques. Ce modèle étend les modèles classiques type PAC et MVC en introduisant un nombre variable de facettes dans les agents afin de regrouper des thématiques fonctionnelles pertinentes et un formalisme graphique qui permet d'exprimer le contrôle d'exécution.

L'agent est le composant de base du modèle AMF. Chaque agent est constitué de facettes et d'administrateurs de contrôle. Il peut contenir d'autres agents et il peut y avoir plusieurs instances d'un même agent.

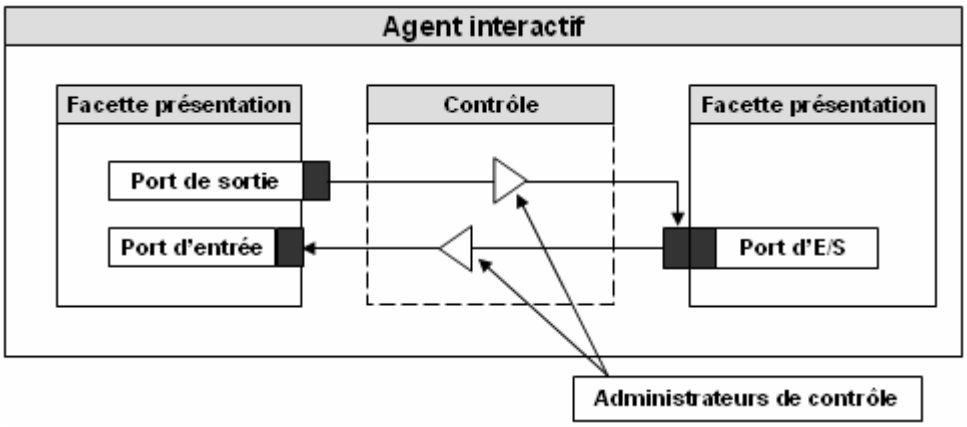


FIGURE 11 : COMPONENTS D'UN AGENT AMF

Le composant de contrôle de l'interaction est l'élément qui gère les communications entre les facettes des agents. Tout formalisme qui utilise ce composant doit être capable de traduire des relations de type : « *A veut le service ƒ de B* ». Cette relation se traduit par l'utilisation d'un administrateur simple entre un port source de la facette A et un port cible de la facette B. La fonction devient le démon du port d'entrée de la facette B. Par exemple, supposons que l'action de l'utilisateur sur un bouton doit conduire la facette Présentation (A) de l'agent considéré à invoquer le service sémantique associé (f) contenue dans la facette abstraction (B).

Pour représenter les concepts et les mécanismes de contrôle dans une architecture AMF, le modèle repose sur un formalisme de représentation des techniques de gestion du contrôle. Ce formalisme s'appuie sur des éléments situés à deux niveaux :

- Au niveau des facettes, les ports de communication définissent les services proposés et les services utilisés.
- Les facettes Contrôle sont des facettes particulières définies par les deux types d'éléments suivants : les administrateurs de ports et les administrateurs de contrôle de agents.

Les ports de communication constituent les interfaces des facettes. A ce titre ils expriment non seulement les services proposés par une facette, mais aussi les services indispensables au fonctionnement de la facette. Dans la figure 11, on distingue trois types de port : Les port d'entrée, de sortie et de E/S.

L'activation d'un port de sortie conduit à l'émission d'un message qui sera reçu par un port d'entrée, aussi à chaque port de communication est associé une fonction particulière appelée démon. On distingue trois situations :

- lorsque le port activé est un port d'entrée, ce démon a la fonction membre de la facette qui remplit effectivement le service.
- lorsque le port activé est un port de sortie, ce démon est généralement un démon par défaut qui ne fait aucun traitement particulier. Dans certains cas, un démon spécifique peut être utilisé par un port de sortie pour effectuer des prétraitements avant l'activation réelle du port.
- Lorsque le port activé est un port d'entrée/sortie, il se comporte tout d'abord comme un port d'entrée avant d'être considéré comme un port de sortie. Naturellement, un seul démon lui est associé.

Les facettes Contrôle sont des facettes que ne présentent pas de port de communication mais sont constituées par deux types d'administrateurs :

- Les administrateurs de ports stockent les références des ports de communication auxquels ils sont associés.
- Les administrateurs de contrôle des agents sont des éléments principaux des facettes Contrôle. Ils gèrent des liens à sens unique entre les ports de communication des facettes grâce aux administrateurs de ports. Un administrateur de contrôle joue trois rôles : un rôle de connexion, qui consiste à gérer les relations logiques pouvant exister entre les ports de communication qui lui sont attachés (source → cibles); un rôle de traduction qui consiste à transformer les valeurs des ports sources en valeurs compréhensibles par les ports cibles ; et un rôle comportemental qui traduit les règles d'activation des ports cibles.

Conclusion

AMF permet de concevoir des éléments réutilisables et peut être étendu et adapté aux besoins d'applications spécifiques. Le concept de « facettes » regrouper des thématiques fonctionnelles pertinentes et un formalisme graphique qui permet d'exprimer le contrôle d'exécution. Le composant de contrôle de l'interaction est l'élément qui gère les communications entre les facettes des agents. Les ports de communication constituent les interfaces des facettes. A ce titre ils expriment non seulement les services proposés par une facette, mais aussi les services indispensables au fonctionnement de la facette.

3.1.3. Synthèse

Une fois présenté les termes liés à la conception architecturale des systèmes interactifs, nous pouvons dire en général, que les modèles d'architecture permettent la décomposition d'un système interactifs dans des couches logicielles, correspondant à la Présentation, le Contrôleur de Dialogue et le Noyau Fonctionnel.

Le tableau 1 suivant résume de manière synthétique les principales caractéristiques abordées pour chacune des modèles d'architecture étudies dans le chapitre, auquel a été rajouté un certain nombre de critères de qualité du logiciel [MaCall 77][ISO 91] qui nous permettront par la suite établir un mécanisme de sélection.

	Seeheim	ARCH	MVC	PAC	PAC-Amodeus	AMF
Aspect plus significatif (Force)	Considère uniquement une architecture modulaire, sans prêter attention à la relation entre le modèle conceptuel et le modèle mental de l'utilisateur.	Proposent une décomposition plus fine de l'architecture du logiciel, cette architecture préconise 5 composants logiciels, le Contrôleur de Dialogue étant le composant principal.	Propose la séparation de l'affichage, de la gestion des événements utilisateur et de la représentation interne des objets manipules. Cela simplifie la tâche du développeur qui tenterait d'effectuer une maintenance ou une amélioration sur le projet.	Permet plusieurs dialogues en parallèle grâce la hiérarchie des agents PAC et à l'autonomie relative des agents dans l'interprétation des événements et dans la gestion de leur état.	Repose sur la décomposition logicielle d'Arch et affine le Contrôleur de Dialogue en termes d'agents logiciels PAC.	AMF étend les modèles PAC et MVC en introduisant un nombre variable de facettes dans les agents afin de regrouper des thématiques fonctionnelles pertinentes et un formalisme graphique qui permet d'exprimer le contrôle d'exécution.

Modalité	Architecture Monolithique	Architecture monolithique	Architecture Monolithique	Architecture basée agents	Architecture basée agents	Architecture basée agents
Complexité de communication	Bas niveau de complexité	Bas niveau de complexité	Bas niveau de complexité	Haut niveau de complexité	Haut niveau de complexité	Bas niveau de complexité
Complexité de conception	Bas niveau de complexité	Bas niveau de complexité	Bas niveau de complexité	Haut niveau de complexité	Haut niveau de complexité	Haut niveau de complexité
Usage de outils de conception	Non	Non	Non	Non	Non	Oui
Concurrence	Non	Non	Non	Oui	Oui	Oui
Facilite de maintenance	Plus facile	Plus facile	Plus facile	Plus difficile	Plus difficile	Plus difficile
Facilite de preuve	Oui	Oui	Oui	Non	Non	Non
Modularité de la solution (Niveau d'affinement / encapsulation)	Bas niveau de modularité	Haut niveau de modularité	Bas niveau de modularité	Haut niveau de modularité	Haut niveau de modularité	Haut niveau de modularité
Environnement distribuée	Non	Non	Non	Oui	Oui	Oui
Facilite pour évoluer le modèle	Non	Oui	Oui	Non	Non	Non
Portabilité	Non	Oui	Oui	Oui	Oui	Oui
Facilite de réutilisation	Niveau bas	Niveau haut	Pas dans touts les cas	Niveau moyen	Niveau haut	Niveau haut
Interopérabilité	Non	Oui	Oui	Non	Oui	Oui
Performance	Bas niveau	Bas niveau	Niveau moyen	Bas niveau	Haut niveau	Haut niveau

Tableau 1 : Caractéristiques des modèles d'architecture logicielle.

3.2. Solutions Logicielles existants

Si les architectures sont un cadre conceptuel nécessaire a une mise en œuvre propre des systèmes de RA, de nombreuses solutions logicielles ont aussi été proposées pour faciliter le travail des développeur.

3.2.1. Middleware (Intergiciel)

L'existence de nouvelles architectures, de nouveaux systèmes et de plateformes plus puissantes et à la fois plus économique, fait que beaucoup d'organisations migrent d'applications corporatives vers de nouvelles plateformes. Cependant, grâce aux échanges rapides des technologies la stratégie fréquemment utilisée inclut le concept de middleware dans le but de prévoir l'interopérabilité[3] des applications [Bernstein 96].

Comme la montre la figure 12, un middleware est un ensemble de services logiciels distribués existant entre l'application et le système d'exploitation.

[3] L'**interopérabilité** est le fait que plusieurs systèmes, qu'ils soient identiques ou radicalement différents, puissent communiquer sans ambiguïté et opérer ensemble.

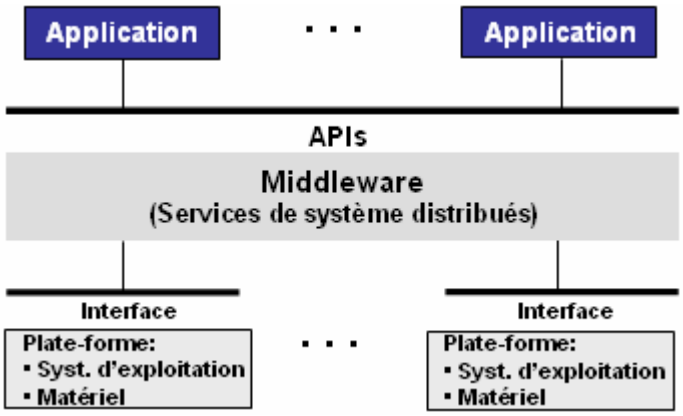


FIGURE 12 : MIDDLEWARE [Bernstein 96].

Ce schéma assure la portabilité et l'interopérabilité des applications dans les réseaux hétérogènes. Parmi ses caractéristiques les plus éminentes nous réalisons qu'il simplifie le processus de développement d'applications. Toutefois, en dépendant du niveau d'abstraction des applications logicielles, généralement, son adoption implique une perte d'efficacité de l'application ainsi qu'une dépendance de paquets spécifiques utilisés dans l'implémentation de cette solution.

3.2.2. Framework – Bibliothèque de classes

Les premières réflexions sur le concept de framework sont apparues pour établir sa différence à l'égard des bibliothèques de classes. Un framework est défini comme un ensemble de classes qui coopèrent et permettent des conceptions réutilisables dans des catégories spécifiques de logiciels [Johnson 88].

Ce concept est très proche de celui de bibliothèque de classes dans le sens où tous deux fournissent des entités codées (classes ou composants) réutilisables. Une bibliothèque de classes fournit un ensemble de classes qui pourront être réutilisées grâce notamment aux mécanismes d'héritage. Elles n'imposent pas de modèles particuliers à une application contrairement aux frameworks. En revanche, les frameworks définissent des relations et des interactions entre des instances de classes d'une application.

Par la suite, d'autres auteurs ont proposé des définitions complémentaires : un framework est un environnement de logiciels qui est conçu pour simplifier le développement d'application et la gestion de système pour un domaine d'application spécialisé [Bernstein 96] (voir la figure 13). En général, c'est un modèle réutilisable d'une portion d'un système logiciel existant et peut être raffiné par des développeurs.

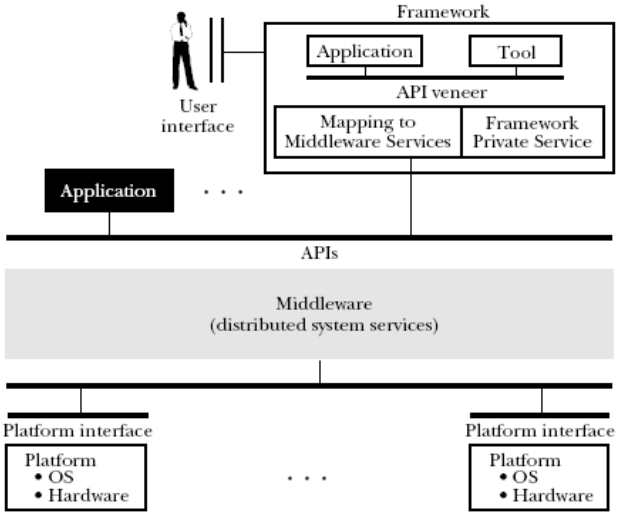


FIGURE 13 : FRAMEWORK [Bernstein 96].

Nous pouvons déduire de ces définitions, que l'utilisation d'un framework se fait par spécialisation des classes abstraites le constituant. Un framework impose ainsi une architecture particulière à une application et permet de définir la structure globale d'une application générique, les collaborations des classes, des objets ainsi que les flots de contrôle de l'application. Au moyen de ceux-ci nous pouvons représenter les mécanismes nécessaires à l'implémentation d'une couche logicielle. Par exemple : STRUTS constitue des frameworks qui réalisent la structure technique de la couche de présentation.

L'usage des frameworks permet le développement rapide et facile d'applications par spécification et - ou instanciation des éléments de sa structure générique. Il est possible de réutiliser le comportement et la structure fournie par défaut ou de les spécialiser en modifiant ou en ajoutant d'autres composants spécifiques à l'application cible. Chaque application réutilise l'analyse, la conception et l'implémentation du framework. Hormis la réduction du coût et l'amélioration de la qualité d'un logiciel, les frameworks ont d'autres avantages comme la modularité et l'extensibilité.

Par rapport à la modularité, la réutilisation et l'extensibilité, les frameworks favorisent la modularité par l'encapsulation des détails de l'implémentation avec des interfaces stables. Ces interfaces fournies par les frameworks mettent en valeur et améliorent la réutilisation par la définition de composants ou de structures génériques qui peuvent être utilisés lors de la création de nouvelles applications. La réutilisation par les frameworks se base sur les expériences des développeurs et des concepteurs dans un domaine particulier, pour éviter la recréation et la revalidation des solutions communes relatives à un même domaine et d'améliorer la productivité des développeurs, la qualité et les performances de développement. Finalement, les frameworks améliorent l'extensibilité en fournissant des méthodes explicites qui permettent aux applications l'extension d'interfaces stables. L'extensibilité des frameworks est essentielle car elle assure l'adaptation des applications développées aux attentes des utilisateurs.

3.3. Conclusion

Du point de vue de la conception architecturale les styles d'architecture, les middlewares et les frameworks permettent de formaliser et concrétiser les meilleures pratiques, les expériences et les directives utilisées dans des développements expérimentés, en offrant des solutions conceptuelles ou techniques qui cherchent à optimiser les décisions au début de la phase de développement de systèmes interactifs qui peut être adaptée aux systèmes de Réalité Augmentée.

Chapitre IV. La Démarche Symphony et les Patrons

Introduction

Comme nous l'avons indiqué dans des sections précédentes, il est nécessaire de réaliser le développement d'une application en suivant des méthodes afin de garantir les buts de l'équipe de projet. Dans notre cas particulier, proposer des aides pour faciliter la conception technique dans le cadre de systèmes de Réalité Augmentée, suggère de formaliser un processus d'analyse et de conception des architectures techniques basées sur la réutilisation des composants. Nous avons choisi la démarche Symphony qui se base sur la réutilisation de composants. Dans ce sens, la première partie de ce chapitre est consacrée à la description de la démarche Symphony, son cycle de vie, ses principales caractéristiques et son cycle de vie et spécialement, nous détaillerons sa branche technique.

La deuxième partie présente la technologie basée composants, le concept de patron de logiciel et ses caractéristiques basiques et finalement nous aborderons le formalisme P-Sigma utilisé pour la représentation des systèmes de patrons. Ces notions permettront de formaliser et instrumenter notre démarche pour la branche technique de Symphony dans la suite de ce document.

4.1. La démarche Symphony

Symphony [Hassine 05] est une méthode de développement logiciel élaborée et industrialisée par la société Umanis. Elle a pour but de spécifier les différentes phases d'un projet, de définir les tâches de chacun des intervenants, de contrôler les coûts, les délais et la qualité de l'application logicielle produite. Cette démarche se présente sous la forme d'un guide méthodologique offrant une solution basée sur l'utilisation de composants dès les phases amont du processus. Symphony s'appuie sur le langage unifié de modélisation UML et repose sur un certain nombre de pratiques de développement objet. Entre les caractéristiques principales de cette démarche nous pouvons citer :

- une démarche itérative,
- son processus est d'orienté l'utilisateur et piloté par les cas d'utilisation,
- une démarche orientée objets et composants métier,
- un modèle de développement de en Y [André 94][Larvet 94].

Symphony est adaptable selon la complexité des projets, leur temps et / ou leur coût de développement. Toutefois, elle comporte plusieurs points de passage obligés. Elle assure quatre fonctions principales :

- déterminer les activités et les responsabilités des différents acteurs du développement,
- spécifier les produits à développer,
- guider la tâche des concepteurs, des développeurs et la coordination de l'équipe de développement,

- offrir des critères pour le contrôle et l'évaluation des activités du projet et des produits.

4.1.1. Processus de développement en Y

Le développement d'une application avec Symphony doit prendre en considération deux aspects fondamentaux et complémentaires :

- Les fonctions attendues de l'application pour répondre aux besoins du métier de l'entreprise.
- Les contraintes opérationnelles de l'application tel que les contraintes matérielles et logicielles, la qualité de service en temps de réponse, la tolérance aux pannes, etc.

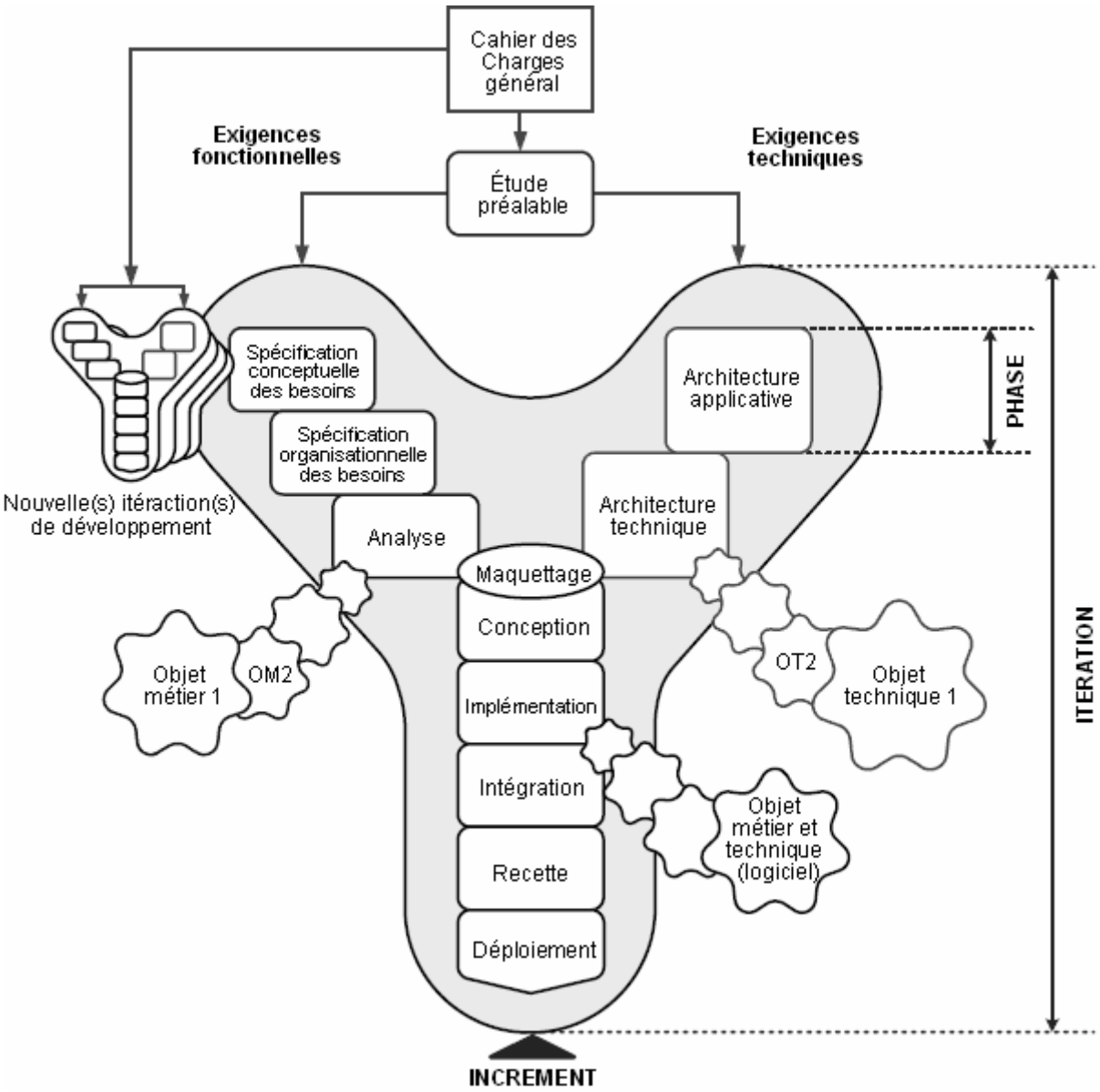


FIGURE 14 : CYCLE DE VIE EN Y DE LA DEMARCHE SYMPHONY

Symphony sépare l'étude des besoins fonctionnels de celle des besoins techniques et ceci dès le début du cycle de vie des applications. Les deux aspect fonctionnels et

techniques sont mis en évidence adoptant un modèle de cycle de vie en Y [André 94][Larvet 94]. Cela permet une meilleure analyse du problème et des risques, mais aussi une meilleure réutilisation de l'existant. Menées en parallèle, ces deux activités se déroulent néanmoins en étroite collaboration afin d'assurer la cohérence. La figure 14, présentée antérieurement, illustre le cycle en Y, dont nous pouvons observer trois branches : la branche fonctionnelle, la branche technique et la branche centrale.

La branche fonctionnelle (gauche) correspond à la tâche traditionnelle de modélisation du domaine et des besoins des utilisateurs. Les résultats de l'analyse ne dépendent d'aucune technologie particulière et se résument à un modèle d'objets métier (OM). La phase de spécification conceptuelle des besoins et la phase de spécification organisationnelle des besoins contribuent à une première identification des OM de haut niveau, et à un découpage ordonné (affectation de priorités) du processus de développement complet en itérations.

La branche technique (droite) a pour objectif le recensement des contraintes et des choix techniques nécessaires pour la conception du système tels que la sécurité, la montrée de charge, l'intégration à l'existant, etc. Ces éléments permettent, par la suite, l'élaboration des architectures applicatives et techniques de l'application. **L'architecture applicative** spécifie les composants techniques et les frameworks techniques à mettre en place pour supporter l'exécution des objets métier. **L'architecture technique** permet de décrire l'organisation matérielle et réseau du système de production. On y décrit la répartition des logiciels sur les machines physiques ainsi que les protocoles de communication des différents nœuds de l'architecture. La définition des différentes architectures est accompagnée par l'identification des règles de conception et des patrons dont l'objectif est de pallier les manques techniques.

La branche centrale est une intégration des branches fonctionnelle et technique. Elle permet de réaliser une conception intégrant le modèle d'analyse dans l'architecture applicative de manière à obtenir un modèle de conception traçant les composants du système à développer. Ce modèle de conception est par la suite traduit dans un langage de programmation en utilisant les outils de développement choisis dans la branche technique. Les tests unitaires des composants sont réalisés au fur et à mesure que leurs unités de code sont développées. La phase de recette permet de tester fonctionnellement l'application en suivant le cahier des charges préalablement rédigé. La phase de déploiement consiste enfin à mettre en production l'application testée techniquement et fonctionnellement.

Un tel modèle de cycle de vie apporte une réponse aux contraintes de changement continuel imposées aux systèmes d'information (selon les axes fonctionnels et technique) et permet ainsi une meilleure maîtrise des évolutions pour ces systèmes.

4.1.2. Phases, activités et produits de la démarche Symphony

Le processus Symphony est constitué de plusieurs phases : Spécification conceptuelle des besoins, analyse, architecture applicative, etc. Les phases sont structurées en activités. Ces activités concourent à la production d'un ensemble

d'entités appelées **ProduitSymphony** ou « artefact » qui décrivent les différentes facettes du système final en fonction de celles fournies en entrée. Un produitSymphony est d'un type donné, il peut s'agir d'un document texte, d'un diagramme UML, d'un exécutable, etc. Une activité Symphony est réalisée par un ou plusieurs **ActeursSymphony**, chacun pouvant être le responsable de la réalisation d'un certain nombre de produits du système.

La démarche se décompose en onze phases indiquées dans la figure 14. Cette décomposition inclut une phase préliminaire d'étude préalable du système d'information à développer. Chaque projet démarre après la spécification du cahier des charges général (CDCG). Ce dernier décrit les exigences métier du système à travers les processus métier qu'elles mettent en œuvre généralement décrits en langage naturel.

Le tableau suivant permet de visualiser les activités de chacune des phases de la branche droite (et leurs étapes respectives). Dans la section suivante nous détaillerons l'architecture technique, l'intérêt de notre mémoire.

Phase/Etapes de la branche technique	Description et Artefacts
Architecture applicative	Spécifie les composants techniques et le framework technique à mettre en place pour supporter l'exécution des composant métier spécifies et détailles pendant la phase d'analyse. Dans cette phase, sont aussi identifiés et définis les patrons et les règles de conception utilisés dans la phase de conception. **Artefacts** : Dossier d'architecture applicative, Cartographie de composants techniques, Description des composants techniques, Normes.
Architecture technique	Permet de décrire l'architecture matérielle et réseau du système de production. On y décrit la répartition des logiciels sur les machines physiques ainsi que les protocoles de communication des différents nœuds de l'architecture. **Artefacts** : Dossier d'architecture technique.
Branche commune	
Maquettage	Son objectif est la mise en place de la cinématique de l'application, de la charte graphique, etc. La maquette résultat doit pouvoir refléter les besoins des clients et donc doit être testée et avalisée par ces derniers. Elle doit aussi pouvoir mettre en avant les difficultés majeures pour les développeurs, ce qui leur permettra de recentrer leur effort. **Artefacts** : Maquette (suite d'écrans montrant la navigation à travers l'application ainsi que son utilisation).
Conception	Décrit la transformation du modèle métier vers le modèle de conception par l'intégration des aspects techniques de l'application définis pendant la phase d'architecture applicative. **Artefacts** : Modèle de conception, Modèle de tables, Diagramme de classe de distribution, Diagrammes de séquences des objets métier (entité ou processus) enrichi par les choix techniques, DCT (Dossier de conception technique).
Développement	A pour objectif la traduction du modèle de conception dans un langage de programmation en utilisant les outils de développement définis ultérieurement. Cette phase inclut les tests unitaires des composants. **Artefacts** : Environnement de développement et un planning détaille par développeur, Source de développements (codes sources des différents objets métier), Test unitaires et l'Application logicielle.
Intégration	Permet de « packager l'application », de tester techniquement l'application et de s'assurer de l'intégration globale des composants du système.

	Artefacts : Application testée techniquement.
Recette	A pour objectif de tester fonctionnellement l'application en suivant le cahier de recette préalablement rédigé. **Artefacts** : Planning des tests, Cahier de recette, fiches de test, Macros de tests, Suivi des résultats des tests, liste des Fiches de Non Conformité.
Déploiement	Consiste à mettre en pilote ou en production l'application testée techniquement et fonctionnellement. **Artefacts** : Dossier d'intégration, Application de production Déployée.

Tableau 2 : PHASES ET ACTIVITES DE LA METHODE SYMPHONY

4.1.3. La branche technique de la démarche Symphony

Comme nous l'avons précédemment remarquée, la branche technique a pour objectif le recensement des contraintes et des choix techniques nécessaires pour la conception du système logiciel. Elle comporte les phases d'architecture applicative et d'architecture technique.

4.1.3.1. L'architecture applicative

L'architecture applicative a pour objectif de spécifier les composants techniques, voir le framework, à mettre en place pour supporter l'exécution des objets métier. Cette phase est réalisée en parallèle avec les exigences fonctionnelles. Elle permet d'anticiper les développements des composants génériques (ne contenant aucun savoir métier). Elle représente la spécification des composants techniques, de leur interaction et de toute spécialisation d'un composant technique.

Les activités de cette phase sont les suivantes (la figure 15 montre l'enchaînement des activités de cette phase) :

- Décrire les différentes couches applicatives,
- Recenser et cartographier les composants techniques,
- Décrire les composants et classes techniques,
- Définir les règles de conception et de transformation.

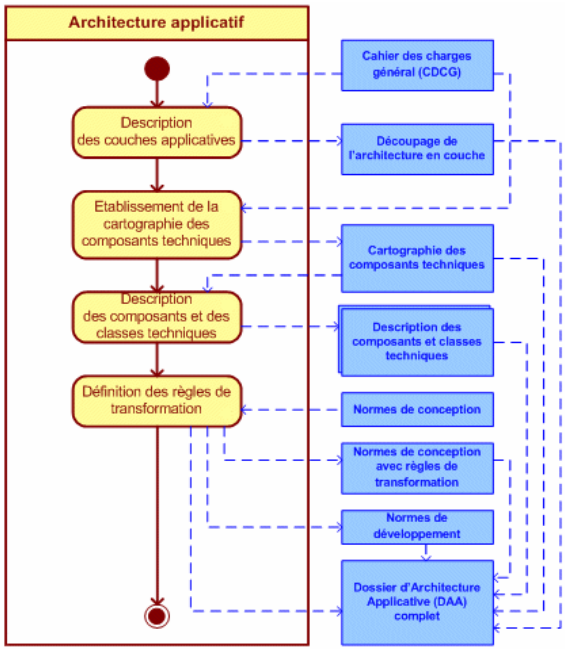


FIGURE 15 : ACTIVITES DE LA PHASE D'ARCHITECTURE APPLICATIVE

Description des couches applicatives

Le découpage architectural est produit par l'application de patrons de conception (tels que façade [Gamma 95], Layers [Bushmann 96] par exemple) pour spécifier l'ensemble des couches applicatives de l'architecture logicielle. L'application des patrons layer produit, par exemple, une architecture multicouches (figure 16) répondant à des objectifs qualitatifs de productivité, de maintenance et de réutilisation grâce à la décomposition du système en sous systèmes. Chaque couche peut être modélisée, développée et testée individuellement. Cette architecture sépare la logique de présentation, la logique applicative, la logique transactionnelle et de persistance (la mise à jour des bases de données).

La relation de dépendance entre les différentes couches signifie qu'un composant de la couche inférieure ne peut pas accéder à un composant de la couche supérieure, cette notion correspond au patron de conception « layer » et permet :

- D'attribuer aux équipes de réalisation des responsabilités sur le développement de chaque couche,
- De masquer les détails d'implantation d'une couche, ce qui permet de ne pas être affecté par son évolution à venir.

NB : Il est important de signaler que Symphony préconise une architecture multicouches pour les systèmes classiques. Cependant, le choix d'un style d'architecture nous permettra de spécialiser la couche de présentation couche et d'adapter cette activité pour la conception des systèmes de RA.

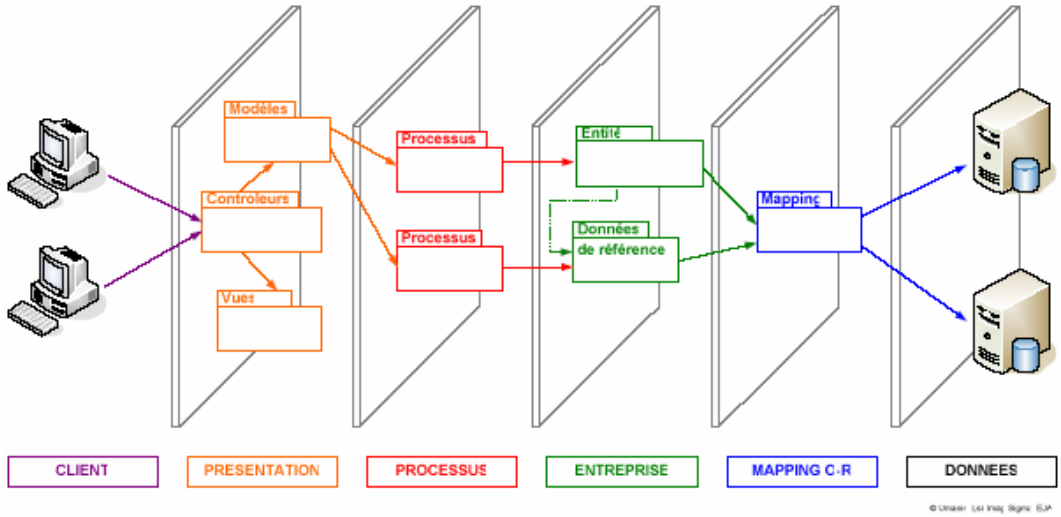


FIGURE 16 : ARCHITECTURE MULTICOUCHES PRECONISEE PAR SYMPHONY

- **La couche Client** est relative aux « composants » (navigateur Internet par exemple) permettant aux utilisateurs d'utiliser l'application.
- **La couche Présentation** s'assure de la définition des composants permettant de gérer la présentation et l'interaction avec les couches inférieures. Dans le cadre de applications J2EE, elle est souvent basée sur le modèle MVC en utilisant les Servlets, JSP et les Javabeans. Ce patron est généralement pris en charge par le framework STRUST.
- **La couche Processus** est réalisée par un ensemble d'objets métier processus encapsulant la logique applicative comme le processus de gestion des dossiers par exemple.
- **La couche Entreprise** représente l'ensemble des composants métier entités et données de référence. Cette couche représente les entités qui seront persistantes, donc projetées dans la base de données. Elles définissent les règles de gestion et les méthodes spécifiques au composant.
- **La couche Projection Objet-Relationnel** permet de transformer les objets métier Entité et Données de référence en requêtes SQL. Cette couche peut être prise en charge par un framework. Il est souhaitable dans ce cas de respecter au mieux les standards (par exemple JDO pour les applications J2EE).
- **La couche Données** est généralement relative aux SGBD et aux différentes bases de données dans lesquelles sont sérialisés les objets métier de la couche Entreprise (via l'utilisation de la couche Projection Objet-Relationnel).

Etablissement de la cartographie des composants techniques

Une fois déterminé le cadre architectural, l'objectif est de définir la cartographie des composants techniques qui seront mis en place dans l'application, et de décrire l'ensemble des interactions entre les composants de l'application et l'éventuel framework de l'entreprise. Dans cette activité, l'architecte doit définir les technologies à mettre en place en convergence avec les besoins fonctionnels.

Dans le cadre des applications n-tiers, Symphony recense plusieurs catégories de composants nécessaires pour le bon fonctionnement et la maintenance de ces applications. Les catégories présentées ci-dessous sont souvent présentes dans les applications de gestion :

- Composants ou framework de présentation : APACHE STRUTS [Struts 00], JavaServer Faces [JSF 03] par exemple,
- Composants de mapping objet relationnel : LiDO [LiDO 06], CASTOR [CASTOR 06], Hibernate [Hibernate 06] par exemple,
- Composants d'authentification : JAAS [JAAS 06] par exemple,
- Composants de notification : JavaMaiL [JavaMail 06], NetSize [NetSize 06] par exemple,
- Composants d'éditions : FOP [FOP 06] par exemple,
- Composants de recherche : Lucène [Lucène 06] par exemple.

Cette cartographie est souvent décrite par des schémas spécifiques du fait d'un manque dans la notation des diagrammes de composants d'UML. La figure 17 illustre un exemple de cartographie des composants techniques.

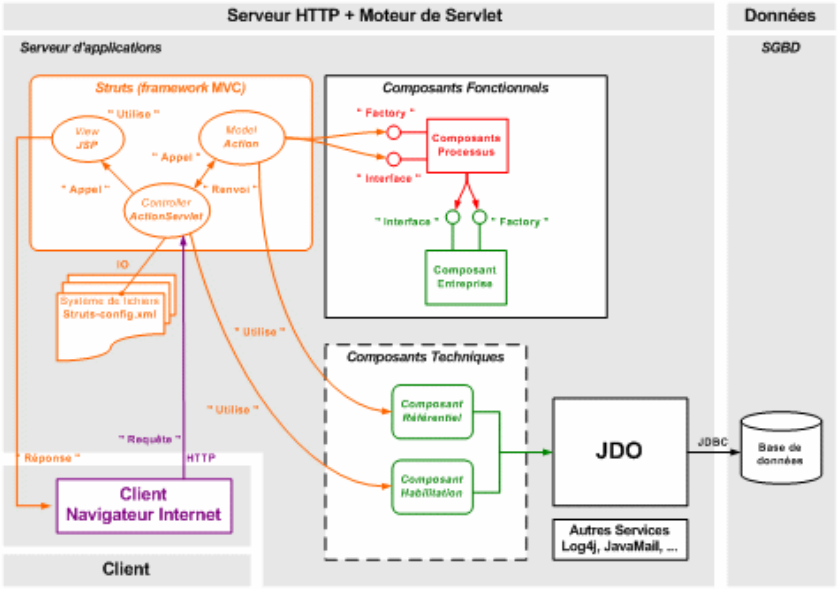


FIGURE 17 : CARTOGRAPHIE DES COMPOSANTS TECHNIQUES

Description des composants et des classes techniques

Cette étape consiste à détailler les composants techniques, leurs interfaces en particulier, les classes spécialisables, si nécessaire leur fonctionnement interne notamment pour les frameworks techniques (par exemple STRUTS). Certains composants nécessitent un enrobage pour faciliter leur utilisation dans le cadre des développements. Par exemple, l'utilisation du composant FOP[4] pour les éditions nécessite la définition de façades pour faciliter l'appel de la commande d'édition (comprenant par exemple, la recherche automatique de la feuille de style, les facilités de génération de XML, …). La conception de ces classes d'enrobage se fera dans la phase de conception. Nous retrouvons des situations similaires pour l'utilisation des composants d'accès aux bases de données (accès à une connexion, exécution

[4] FOP (Formatting Objects Processor) est un API Opensource du Project Apache XML, de génération et de mise en forme de documents aux formats PDF, PCL, PS, SVG, XML, Print, Awt, MIF et TXT. http://xmlgraphics.apache.org/fop/

d'une requête, etc.). Même l'utilisation d'un outil de projection nécessite un enrobage certes très simple, mais utile pour éviter la redondance de code.
Dans la réalisation de l'architecture applicative, l'architecte applicatif doit définir le cadre d'utilisation du composant (d'origine ou spécialisé) dans l'application. Cette description peut se faire par des exemples concrets d'utilisation. Dans le cas d'un composant complexe (ou framework), une documentation doit être annexée au dossier d'architecture applicative (ex : normes d'utilisation de STRUTS). L'exemple de la figure 18 illustre une description de composant technique.

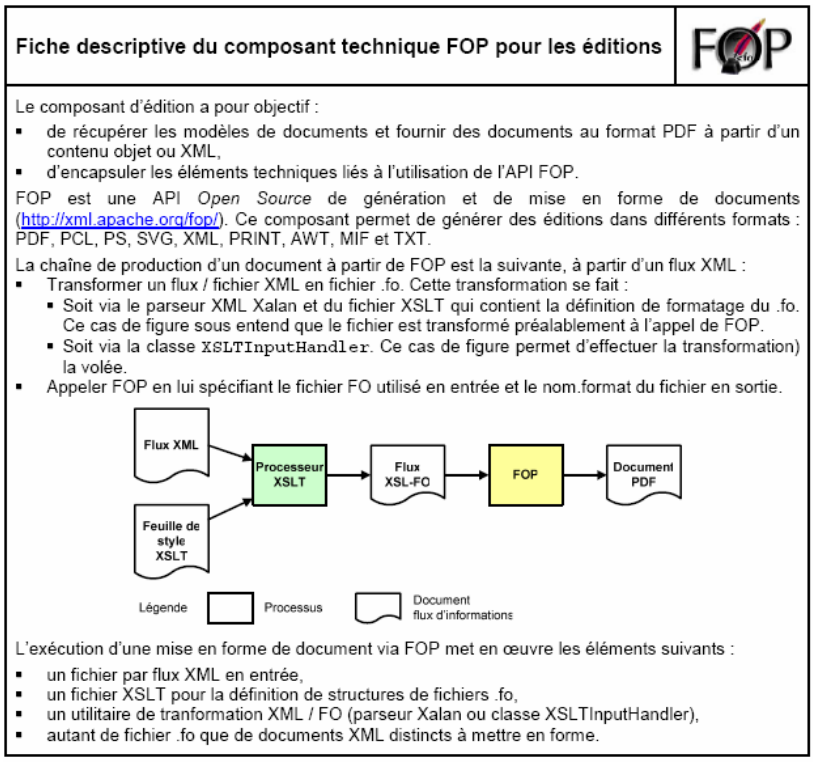


FIGURE 18 : DESCRIPTION D'UN COMPOSANT TECHNIQUE

Définition des règles de transformation

Les activités de transformation consistent à spécifier et/ou à identifier les règles de conception (patrons et règles de transformation) applicables dans la conception de l'application. Ces règles doivent permettre de :

- Transformer le modèle d'analyse en modèle de conception,
- Générer les interfaces des composants distribués (par exemple les IDL),
- Générer les classes des composants (inutile si la génération est réalisée par un outil),
- Générer le schéma relationnel,
- Traduire les modèles d'états.

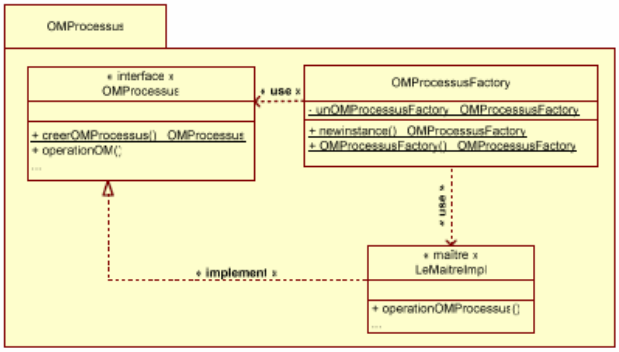
Cette activité permet également de spécifier les patrons à utiliser (généraux et spécifiques à la plate-forme) notablement dans les phases de conception et d'architecture technique (décrite ci-après). La figure 19 illustre un exemple de règle de conception et de transformation.

Règle de transformation : conception d'objets métier Symphony en classe *JavaBeans*

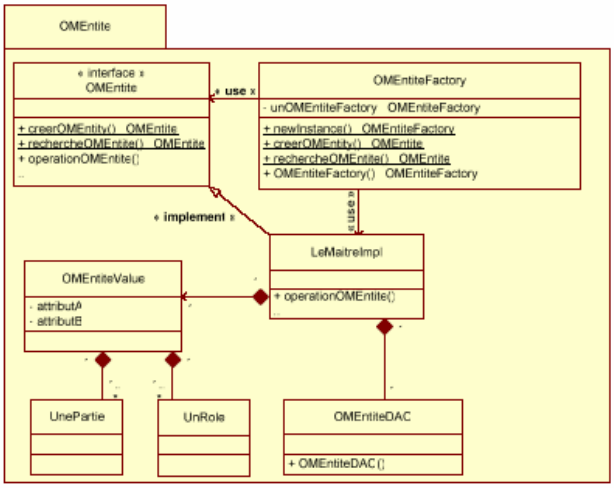
Problématique : cette règle de transformation permet d'implémenter les objets métier Symphony sous la forme de classes *JavaBean*.

Pré-requis : disposer du modèle de conception des objets métier.

Principe de transformation pour les objets métier Processus : les objets métier processus sont implantés sous forme d'un paquetage Java contenant une classe Interface, une classe implémentant l'interface et une classe fabrique (*Factory*). La classe Interface précise les différentes méthodes représentant les étapes du processus implanté.



Principe de transformation pour les objets métier Entité et Données de référence
Les composants Entité et Données de référence sont implantés sous la forme d'un paquetage Java contenant une interface, une classe implémentant cette interface, une *factory* (classe de fabrication d'instances), les classes Partie et les classes Rôle. Ce paquetage doit définir également une classe DAO (*Data Access Object*) regroupant les méthodes d'accès à la base de données.



Les classes Rôle et Partie sont définies comme des classes Java. La classe Rôle définit les attributs dérivés comme des attributs standards. La synchronisation des attributs du rôle avec ceux de l'objet métier requis peut se faire soit en partageant les tables relationnelles (solution couramment utilisée) soit à l'aide de mécanismes de synchronisation comme les déclencheurs (*triggers*).

Pour les classes rôles, deux situations se présentent :

- Si la classe rôle définit uniquement des attributs d'identification du composant destinataire et que la relation « maître - rôle » est mono-valuée, ces attributs pourront être définis dans la classe d'implémentation du composant avec une méthode de recherche du composant destinataire.
- Si la relation « maître – rôle » est multi-valuée, ou que la classe rôle définit des attributs propres (non dérivés) ou que la classe maître définit plusieurs branches rôles, alors la classe rôle sera définie par une classe dépendante du composant.

Remarque générale
Une interface ne peut pas être instanciée directement. C'est l'objet qui implémente l'interface qui est instancié. Ainsi avec la « philosophie composant », il n'est possible de manipuler des interfaces qu'à la condition d'avoir instancié l'objet Maître. L'instanciation des objets maîtres est donc pris en charge par les classes *Factory*. Il existe une classe *Factory* par composant. L'objet maître est instancié soit lorsque l'application doit créer un nouvel objet maître, soit lorsque l'application en récupère un à partir d'un enregistrement de la base (cas des objet métier Entité et Données de référence).

FIGURE 19 : REGLE DE CONCEPTION/TRANSFORMATION

4.1.3.2. L'architecture technique

L'objectif de cette phase est de décrire l'environnement de production, l'architecture réseau et matérielle. Cette phase sert également à spécifier les contraintes d'exploitation. Cette phase est constituée d'une seule activité : la définition de l'architecture technique (figure 20).

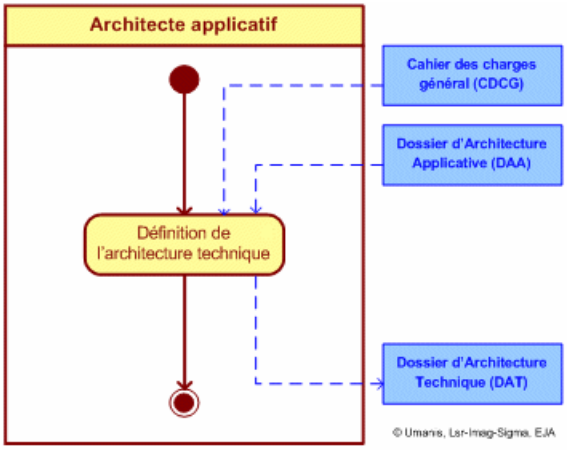


FIGURE 20 : ENCHAINEMENT DES ACTIVITES DE LA PHASE D'ARCHITECTURE TEECHNIQUE

L'activité de définition de l'architecture technique consiste à :

- Décrire l'architecture existante et les contraintes d'utilisation et d'exploitation.
- Définir l'architecture technique de production en répondant aux besoins suivants :
  - Définir les nœuds de l'architecture distribuée,
  - Définir la nature et la cardinalité des connexions des nœuds (spécification des protocoles de communication, exemple : FTP, EAI, SOAP, http, etc.).
  - Associer les logiciels et les packages techniques aux nœuds. Positionner les serveurs d'applications, serveurs Web, moteurs de base de données sur les nœuds.
- Définir les autres environnements (de recette, d'intégration, etc.).
- Définir l'environnement de développement.
- Définir les règles d'exploitation.
- Définir la politique d'archivage.
- Définir la politique de supervision.
- Définir la volumétrie des données.
- Définir les batchs d'alimentation (ordonnancement).

4.2. Les Patrons

4.2.1. Concepts

Le terme patron n'est pas spécifique à un domaine donné. Il est couramment utilisé dans de divers domaines mais aussi en informatique. L'origine et l'inspiration pour la recherche sur les patrons ont été introduites dans les années 1970 par Christopher Alexander qui discute le rôle des patrons dans l'architecture urbaine [Alexander 77]. Dans le domaine de l'informatique, les premiers patrons ont été présentés par Beck

[Beck 87]. Cependant, les travaux de la « bande des 4 » [Gamma 95] ont produit le document fondateur centré sur l'ingénierie des systèmes. Depuis, de nombreux travaux ont porté sur l'identification de patrons dans des domaines d'application variés et sur les différentes phases d'un processus de développement.

Dans la littérature, on trouve plusieurs définitions de patrons. [Coad 95] définit un patron comme une abstraction d'un groupe de classes réutilisables plusieurs fois dans un développement orienté objet. Selon [Schmidt 99], un patron offre une solution à un problème dans un contexte donné. Ils peuvent être vus comme des micro-architectures spécifiant des interactions abstraites entre des entités collaborant pour résoudre un problème dans un domaine particulier.

Comme nous l'avons indiqué un patron est souvent présenté comme une solution consensuelle à un problème qui se répète fréquemment dans un contexte et domaine particulier. Dans cette phrase, chaque mot a un sens. Contexte réfère à un ensemble de situations récurrentes dans lesquelles les patrons sont appliqués. Le Problème définit un but à atteindre dans ce contexte. Enfin, la Solution se réfère à un modèle que l'on peut appliquer pour résoudre le problème.

4.2.2. Utilité et caractéristiques des patrons

Les patrons apportent des solutions efficaces à des problèmes d'analyse et de conception permettant ainsi aux développeurs de créer des conceptions réutilisables. De même, entre les caractéristiques basiques, avantages et types d'usages nous pouvons mentionner les suivantes :

- une aide à l'analyse et à la conception des systèmes,
- une validation de la conception de modèles existants,
- une validation de la spécification des systèmes,
- une aide à l'implantation,
- une documentation des systèmes,
- un moyen de formaliser des démarches de développement.

Trois intérêts essentiels peuvent être soulignés :

- **un moyen de transfert des connaissances**, l'idée du patron se base sur la capitalisation d'un savoir-faire qui offre ainsi un gain de temps et d'efficacité en réduisant largement les tâtonnements fastidieux nécessaires à l'élaboration d'un logiciel de qualité.
- **Une élévation des facultés d'abstraction**, les patrons aident les concepteurs à mieux structurer la représentation d'un environnement (par exemple un framework), guident la perception du monde réel et permettent d'en obtenir une description à un niveau d'abstraction élevé. Ils capturent les propriétés essentielles d'une architecture en occultant les détails non pertinents au niveau d'abstraction considéré.
- **Un enrichissement du vocabulaire**, les patrons fournissent une souplesse de vocabulaire commun aux concepteurs, qui peuvent faire référence à des abstractions d'un niveau plus élevé que celles qu'ils utilisent habituellement.

4.2.3. Classification des patrons

Il existe deux grandes catégories de patrons selon le **type de connaissance** exprimée :

- Les patrons de **Processus** [Ambler 98][Gzara 00a], décrivent un savoir-faire sous forme d'actions et - ou des tâches à suivre pour le développement des logiciels,
- Les patrons **Produit** offrent une prescription que l'on adopte ou pas pour construire tout ou partie des spécifications ou de l'implantation d'un système d'information.

Par rapport à la méthode Symphony, les patrons processus correspondent avec les patrons de phase ou activité. Les patrons produit décrivent les résultats crées à l'intérieur du développement d'une activité Symphony.

4.3. Les systèmes de patrons

Un système de patrons est une collection de patrons formant un vocabulaire qui permet de comprendre et de communiquer les idées [Alexander 77]. Si un patron constitue une solution à un problème récurrent dans un contexte donné, alors un système de patrons est une collection de solutions qui coopèrent pour résoudre un problème complexe selon une solution ordonnée pour aboutir à un but prédéfini.

Un système de patrons bien élaboré doit permettre aux concepteurs d'avoir la liberté d'exprimer leurs problèmes et de concevoir eux-mêmes les solutions qui répondent à leurs besoins particuliers dans le contexte où les patrons peuvent être appliqués. Plusieurs systèmes de patrons ont été proposés, nous pouvons nommer spécialement le formalisme P-Sigma utilisé pour formaliser la démarche Symphony.

4.4. Le formalisme P-Sigma

Un formalisme correspond à une structure adoptée par le concepteur de patrons pour représenter des patrons. Cette structure est conduite par la définition d'un patron en prenant notamment en compte le triplet informatif « problème, contexte, solution ». Les formalismes de représentation actuels sont dédiés soit à l'expression des patrons Produit en mettant l'accent sur la représentation des solutions modèles (exemple : formalisme de E. Gamma [Gamma 95] ou de P. Coad [Coad 95]), soit à l'expression des patrons Processus en mettant l'accent sur la représentation des solutions démarches (exemple : formalisme de S.W. Ambler [Amb 98]). Cependant, le formalisme P-Sigma permet de représenter à la fois les patrons produits et processus.

Le formalisme P-Sigma [Conte 01][Tastet 04] a été conçu en partant de la volonté de définir un formalisme commun au niveau de l'équipe Sigma du laboratoire LSR. Parmi ses buts nous pouvons mentionner les suivants :

- Uniformiser la représentation des patrons produit et processus,
- Améliorer la formalisation de l'interface de sélection des patrons,
- Proposer une organisation des systèmes de patrons.

4.4.1. Structure générale du formalisme P-Sigma

Le formalisme P-Sigma est constitué de trois parties : Interface, Réalisation et Relation. La partie « **Interface** » contient tous les éléments qui permettent la sélection d'un patron. La partie « **Réalisation** » exprime la solution d'un patron en terme de solution modèle ou de solution démarche. La partie « **Relation** » permet d'organiser les relations entre patrons, donc d'organiser le catalogue de patrons.

Chaque partie regroupe un certain nombre de rubriques (figure 21). Chaque rubrique peut contenir un nombre variant de champs de nature différente (texte, diagramme UML, expression logique de mots-clés, etc.). Les rubriques se divisent en deux groupes : les optionnelles et les obligatoires (Identifiant, Classification, Contexte et Problème).

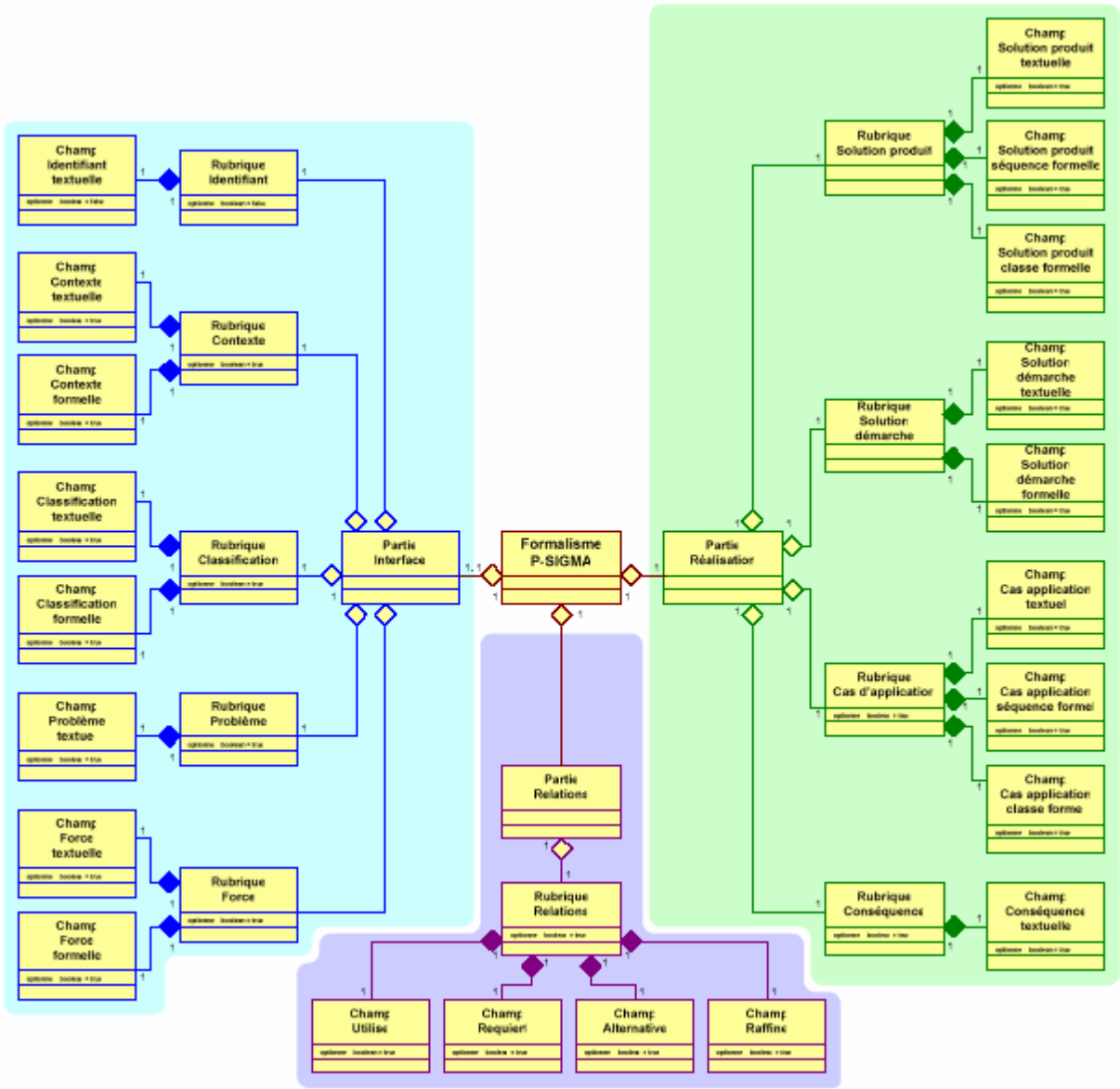


FIGURE 21 : FORMALISME P-SIGMA [Jausseran 05].

La structure générale du formalisme groupe dans chaque partie un certain nombre de rubriques, lesquelles proposent en général une spécification textuelle et une spécification formelle. Les sections suivantes montrent les rubriques du formalisme P-Sigma. Las dernières extensions réalisées par [Jausseran 05] figurent en italique.

4.4.1.1. La partie Interface

L'interface d'un patron est composée de cinq rubriques servant à la sélection d'un patron (Tableau 3).

Rubrique	Définition et Champs et exemple
Identifiant (obligatoire)	Défini le couple problème / solution a partir duquel le patron pourra être référencer. Cette rubrique permettra par la suite aux autres patrons du langage de le référencer. **Champ(s)** : texte court.
Classification (obligatoire)	Définit la fonction du patron par un ensemble de mots-clés du domaine (termes du domaine). Donne intuitivement la classification du domaine. **Champ(s)** : Un champ textuel, éventuellement un champ formel sous la forme d'une expression logique utilisant les mots clés du domaine.
Contexte (obligatoire)	Décrit le ou les pré-conditions pour l'application du patron. Peut être obtenu en appliquant la solution modèle d'un ou de plusieurs patrons ; les noms des patrons correspondants constituent alors le champ formel. **Champ(s)** : Un champ textuel, éventuellement un champ formel : {ensemble de patrons}
Participant(s) (extension P-Sigma)	Décrit le rôle des acteurs des participants dans le processus de réalisation de la solution Produit. **Champ(s)** : Un champ textuel
Problème (obligatoire)	Définit le problème résolu par le patron. **Champ(s)** : Un champ textuel
Force (obligatoire)	Définit les apports induits par l'application du patron. **Champ(s)** : Un champ textuel, éventuellement un champ formel : expression logique des critères de qualité associés à une technologie

Tableau 3 : PARTIE INTERFACE D'UN PATRON AVEC P-SIGMA.

4.4.1.2. La partie Réalisation

La partie d'un patron comprend quatre rubriques présentées dans le tableau suivant :

Rubrique	Définition et Champs et exemple
Solution Démarche (Obligatoire pour un patron Démarche)	Indique la solution du problème en terme de processus à suivre. **Champ(s)** : Un champ textuel, éventuellement un champ formel de type diagramme UML ou autre.
Solution Produit (Obligatoire pour un patron Produit)	Décrit la solution en terme des produits attendus après l'application du patron. Cette rubrique n'est pas uniquement destinée aux patrons Produit. Il est parfaitement possible de décrire la structure des produits résultants de chaque activité (patron Processus). **Champ(s)** : Un champ textuel, un champ de type diagramme de classes (structure de la solution), éventuellement un champ de type {diagramme de séquence} pour préciser les responsabilités et les interactions entre les différentes classes et finalement {des diagrammes

	informels} représentation graphique, aucune langage de modélisation particulier.
Cas d'application	Décrit des exemples d'imitation de la Solution Produit ou résultant de la solution Démarche. Rubrique optionnelle mais fortement conseillée pour faciliter la compréhension de la solution du patron. **Champ(s)** : Un champ textuel, un champ de type diagramme de classes, éventuellement un champ de type : {diagramme de séquence}
Conséquence d'application	Présente les limites et les bénéfices de l'application de la solution. Peut inclure un nouvel ensemble de problèmes faisant apparaître la nécessite d'appliquer de nouveaux patrons. **Champ(s)** : Un champ textuel
Copyright(s) (extension P-Sigma)	Cette rubrique permet de préciser les droits d'auteur (copyrights) pour chaque patron. Cette rubrique permet également de situer les origines de chaque patron. **Champ(s)** : Un champ textuel

Tableau 4 : PARTIE REALISATION D'UN PATRON AVEC P-SIGMA.

4.4.1.3. La partie Relation

La partie Relation est composée de quatre rubriques correspondantes aux quatre types de relations possibles entre les patrons. La signification de chacune de ces relations est exprimée dans le Tableau 5 et s'appuie principalement sur les rubriques de la partie Interface.

Rubrique	Définition et Champs et exemple
Utilise	Si un patron P1 utilise un patron P2, alors : • La solution démarche de P1 doit être exprimée en utilisant le patron P2. • La classification de P2 peut être enrichie par rapport a celle de P1 : de nouveaux mot clés sont éventuellement ajoutes dans la classification de P2. • Le contexte de P2 peut être enrichi par rapport a celui de P1. **Champ(s)** : Utilise : ensemble de patrons.
Raffine	Si un patron P1 raffine un patron P2, alors : • Le problème de P1 doit être une spécification de celui de P2. • La classification de P1 peut être enrichie par rapport à celle de P2. • La force de P1 peut être enrichie par rapport à celle de P2. • Le contexte de P1 peut être enrichie par rapport à celui de P2. **Champ(s)** : Raffine : ensemble de patrons.
Requiert	Si un patron P1 requiert un patron P2, alors : • L'application de P2 doit être un pré-requis a l'application de P1. • P2 doit apparaître dans le contexte de P1. Dans l'exemple ci-dessous : pour exécuter P1, il faut avoir obligatoirement exécuté P2. **Champ(s)** : Requiert : ensemble de patrons.
Alternative	Un patron P1 est une alternative d'un patron P2 si les deux patrons se différencient par leur force qui justifie deux solutions différentes au même problème : • P1 et P2 ont la même classification, le même contexte et le même problème. • Seule la rubrique « force » des deux patrons est différente. **Champ(s)** : Alternative : ensemble de patrons.

Tableau 5 : PARTIE RELATION DE UN PATRON AVEC P-SIGMA.

4.4.2. Le formalisme P-Sigma adaptée aux Systèmes de Réalité Augmentée

Dans la section précédente, nous présentons le formalisme P-Sigma, pour uniformiser la représentation de patrons. Nous utiliserons le nomenclature utilisée par Juras [Juras 06] qui a proposé une extension du formalisme P-Sigma par les systèmes de réalité Augmentée (Tableau 6). Il est important d'indiquer que Juras se concentre sur l'étude de la branche fonctionnelle de la démarche Symphony pour les systèmes de réalité augmentée.

Le formalisme proposé par Juras consiste de règles et critères pour faciliter la compréhension des patrons. Entre las modifications plus signifies on se trouve que la rubrique **« Copyright »** de la partie réalisation est substituée par une rubrique **« Document(s) »**. Ce change permet d'indiquer dans un patron de phase l'ensemble de tous les documents de la méthode, alors que pour une activité, les documents élaborés seulement a niveau de l'activité.

Partie du Formalisme	Rubrique	Description du formalisme proposé par Juras dans le cadre des systèmes de RA
Interface	Participant(s)	Décrit le rôle des acteurs des participants dans le processus de réalisation de la solution Produit. **Critères :** • Si le patron correspond à une phase de la démarche nous indiquerons la spécialité de chaque acteur sans détails. • Si le patron correspond à une activité nous utiliserons la couple : {Spécialité, Responsabilité}.
	Classification	Représente l'ensemble de mots clés associées au domaine de développement (voir annexe 3). Pour la représentation formelle de cette rubrique, nous pouvons établir la hiérarchie suivante : {« Niveau », « Type », « Nom »}. **Critères :** • Le Niveau est déterminé par l'étape du cycle de vie de la démarche qui est décrite : Phase ou Activité. • Le Type correspond au type de patron qui se présente : Processus, Produit. • Le Nom correspond à l'identifiant du patron et à la liste de termes à laquelle le patron est assigné.
	Contexte	Décrit le ou les pré-conditions pour l'application du patron. Peut être obtenu en appliquant la solution modèle d'un ou de plusieurs patrons ; les noms des patrons correspondants constituent alors le champ formel. **Critères :** Nous utiliserons le terme « générale » pour décrire le contexte textuel et assignerons un qualificatif « REQUIS » [REQ] ou « OPTIONNEL » [OPT] pour indiquer le niveau de compromis des artéfacts requis.
Réalisation	Document(s)	La rubrique à Document(s) que substitue la rubrique Copyright(s), représente dans une phase l'ensemble de tous les documents, alors que pour une activité, les documents élaborés seulement à niveau de l'activité. **Critères :** Champ textuel : texte long.

Tableau 6 : NOMENCLATURE UTILISEE PAR LES SYSTEMES DE REALITE AUGMENTEE.

4.5. Conclusion

Nous avons présenté une approche pour représenter notre méthodologie. Notre projet consiste à structurer et formaliser les spécifications techniques de la démarche Symphony dans le cadre des systèmes de réalité augmentée. Dans notre contexte,

les patrons ont en effet de bonnes propriétés concernant l'appropriation et l'acquisition des connaissances. La démarche Symphony est décrite par des patrons combinant des solutions qui se basent sur l'application de patrons d'architecture et de conception. La formalisation sous forme de patrons peut augmenter de manière conséquente la capacité à maximiser la réutilisation de ces composants.

Chapitre V. Un Système de Patrons pour la Branche Technique en Systèmes de Réalité Augmentée

Introduction

Ce chapitre est consacré à la présentation de notre étude d'architectures logicielles basées sur un système de patrons, dédié à l'ingénierie des Systèmes Mixtes et centrée sur les Systèmes de Réalité Augmentée. La formalisation des architectures techniques dans le cadre des Systèmes de RA, a pour l'objectif de mettre en place un guide de conception méthodologique abordant les aspects suivants :

- Proposer au concepteur d'un processus pour la conception technique.
- Capitaliser et réutiliser des styles d'architecture adaptes aux systèmes de RA.
- Proposer des patrons produits en support au processus pour les systèmes de RA.

Nous nous appuyons, dans la suite de ce chapitre, sur un exemple de gestion d'« Etat des Lieux » (EDL). Nos travaux se basent sur une vision générique du métier des agences immobilières proposée par David Juras [Juras 06]. Nous commençons par présenter brièvement cette étude de cas, afin d'obtenir une compréhension qui permet de commencer avec notre démarche. Dans un second temps, nous présenterons des changes de la branche droite de la méthode Symphony, ces modifications consistent en la création et reformulation des activités de la phase de architecture applicative. Cette nouvelle structure nous servira de base pour élaborer notre démarche et pour construire notre système de patrons qui sera présenté dans le annexe 4 de ce document.

5.1. Etude de cas : « Gestion des EDL »

L'étude de cas concerne dans le domaine d'activité immobilière, particulièrement dans l'activité d'« Etat Des Lieux » (EDL). On entend habituellement par Système de Gestion des EDL (SGEDL), un système destiné à faciliter la gestion de l'ensemble des informations sur les états des lieux des biens immobiliers et administratifs d'une agence immobilière.

5.1.1. Application cible

Dans le cadre d'un EDL, l'objectif est de proposer un noyau fonctionnel et un paradigme d'interaction adaptés pour réaliser un état des lieux immobilier pendant lequel l'utilisateur pourrait identifier, localiser et annoter des dommages dans un logement, effectuer des comparaisons rapides avec l'état des lieux précédent, etc.

Typiquement, ces processus sont gérés manuellement avec un crayon, un papier avec des fiches d'état des lieux ou bien avec des enregistrements vocaux sur dictaphone retranscrits sur feuille par la suite. Cela nécessite une double saisie afin

d'intégrer, éventuellement, les informations recueillies dans les systèmes d'informations immobiliers existants. Cependant, l'apparition de dispositifs mobiles comme les PDA a permis une évolution de ces pratiques et de ces usages. L'EDL est saisi en remplissant des formulaires sur un PDA qui est ensuite synchronisé avec une base de données. La saisie informatique de l'EDL par les agents immobiliers, les experts ou les huissiers peut désormais être effectuée directement sur le terrain et la synchronisation des données avec des systèmes d'information plus vastes est simplifiée est transparente pour l'utilisateur.

Par rapport à la RA, un extrait du scénario du processus « Réaliser un EDL » en langue naturelle est fourni dans le tableau suivant :

(…) L'Expert EDL rentre dans une pièce, et sa position est indiquée sur le plan numérisé du logement consultable sur son support d'affichage portable. L'ancien EDL de la pièce, chargé automatiquement à son entrée dans la pièce, n'affiche sur le support de vision (HMD) aucun ancien dommage à vérifier. Il parcourt la pièce et remarque une tache sombre sur le papier peint d'un des murs. Il décrit le dommage observé à l'aide d'une reconnaissance vocale et / ou en effectuant une saisie sur son support tactile d'affichage et de saisie, lui associe une position spatiale, une forme et une dimension en se servant de la direction de son regard, de la commande vocale et éventuellement du support tactile (multimodalité) **(…)**

Tableau 7 : Extrait du scénario du processus « Réaliser un EDL ».

5.2. Architecture technique pour les systèmes de réalité augmentée

Nous décrivons dans cette partie les règles de bonne pratique préconisées par Symphony et leurs adaptations pour construire une architecture applicative et technique à base de composants vers les systèmes de réalité augmentée. Nous illustrons cette architecture en utilisant le cas d'étude « Etat des lieux » introduit dans la section précédente.

5.2.1. Architecture applicative

À partir de l'étude préalable et menée en parallèle des phases de spécification conceptuelle et organisationnelle des besoins et d'analyse, la phase d'architecture applicative est consacrée à la réalisation du Dossier d'Architecture Applicative (DAA) [5]. Comme nous l'avons présenté, ce dossier est composé des parties suivantes :

- description des couches applicatives (DCA),
- cartographie des composants techniques,
- description des composants et classes techniques,
- définition des règles de conception et de transformation.

De manière générale, au début de cette phase l'architecte applicatif réalise le découpage architectural, cette activité a comme but de spécifier l'ensemble des couches applicatives de l'architecture logicielle. La nécessité de développer un

[5] Dans le cas d'«Etat des lieux » nous avons présenté seulement les documents: cahier de charges général et l'étude préalable. Cependant, pour plus d'informations, se reporter au travail de Juras [Juras 06].

système en réalité augmentée sous forme partiel ou total, induit certaines modifications du processus de réalisation du Dossier d'Architecture Applicative (DAA)[6]. Ce nouvelle dossier est compose des parties suivantes :

- Dossier d'architecture en couches (DCA),
- **dossier des dispositifs d'interaction (DDI),**
- **spécifications détaille des dispositifs d'interaction (SDDI),**
- cartographie des composants techniques,
- description des composants et classes techniques,
- définition des règles de conception et de transformation.

Par rapport au processus nous avons ajouté trois activités pour formaliser la sélection des dispositifs d'interaction et puis des styles d'architecture logicielle. Le diagramme d'activités suivant (que nous expliquerons ensuite) résume l'évolution du processus de réalisation de l'architecture applicative.

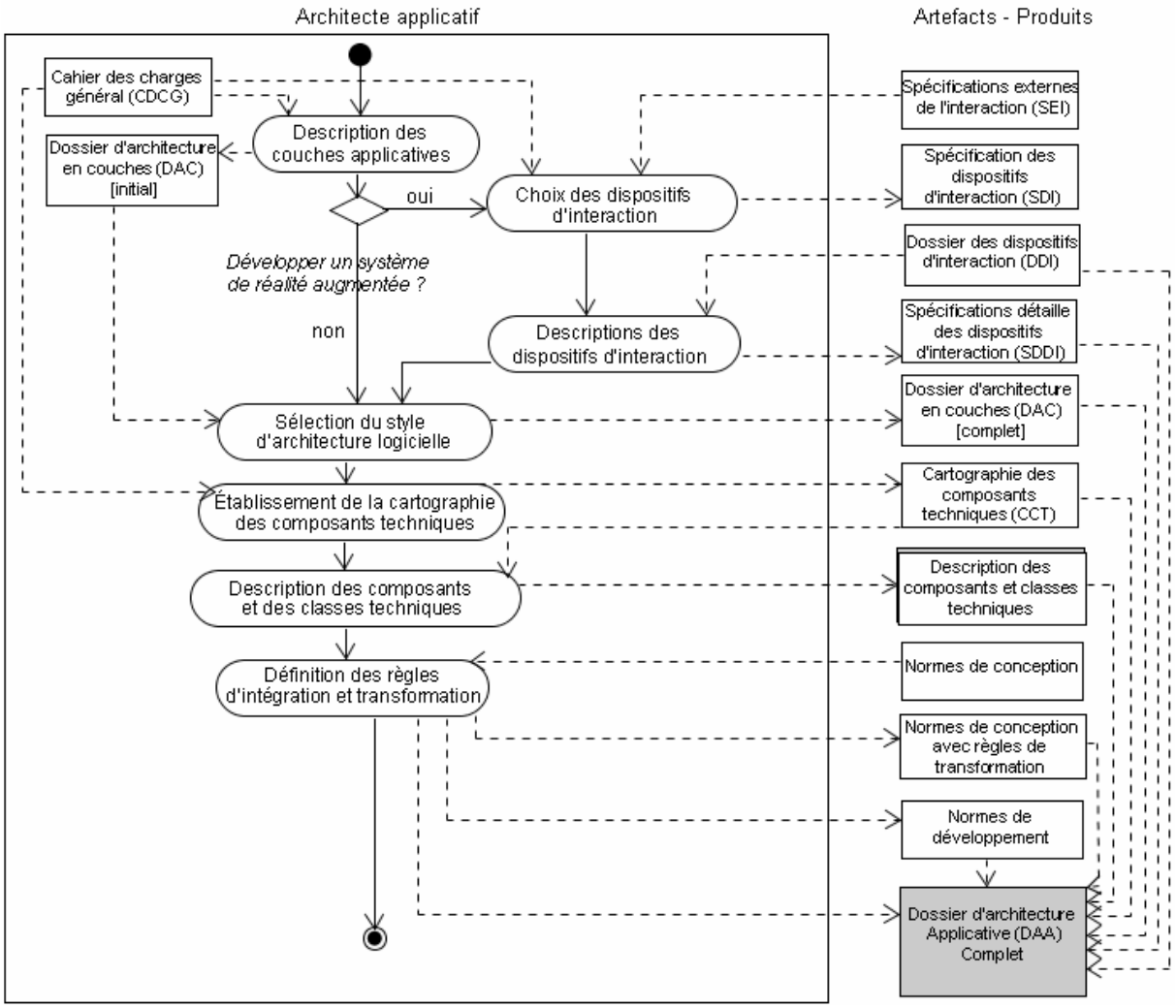
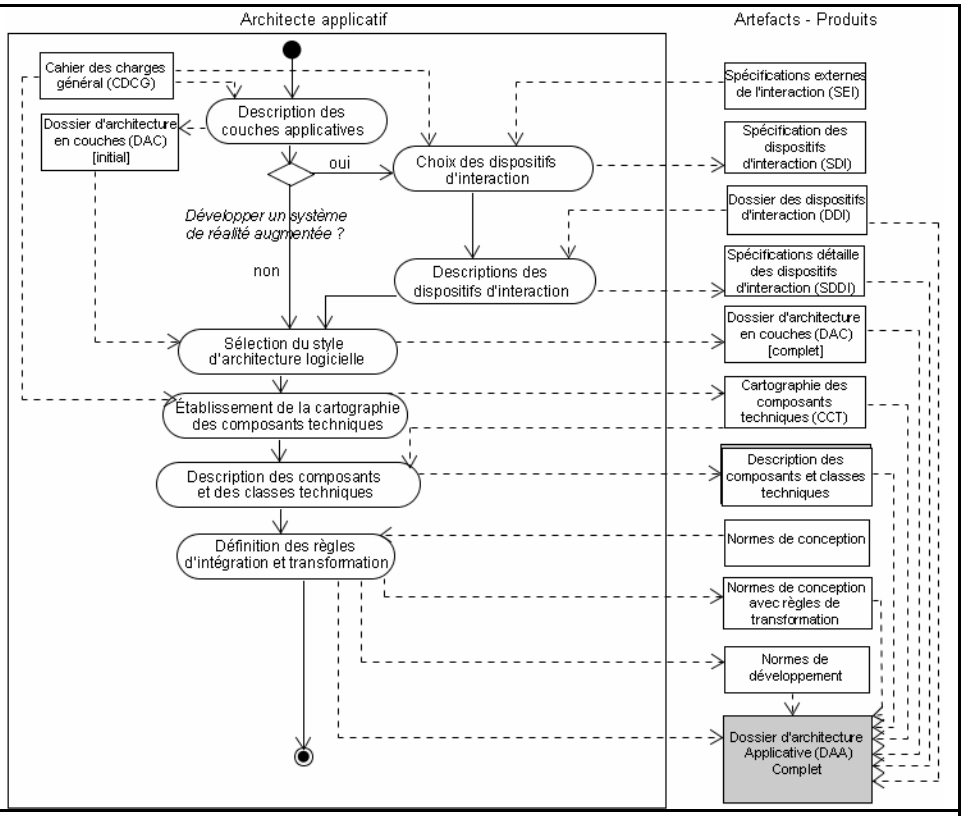


FIGURE 22 : ACTIVITES DE LA PHASE D'ARCHITECTURE APPLICATIVE ÉXTENDU POUR LES SYSTEMES DE REALITE AUGMENTEE

[6] Il est nécessaire de mettre en considération que la sûreté de développer un système de réalité augmentée est connue quand l'étude des spécifications organisationnelles et interactionnelles des besoins correspondant à la branche fonctionnelle sont réalisés.

Nous présentons le patron d'architecture applicative. Ce patron constitue une aide à la conception et l'analyse structurale et comportementale de l'architecture d'un système interactif. Il doit constituer la base sur laquelle s'appuient la méthode et les outils de conception et de développement.

Patron : Phase d'architecture applicative	
Auteur (s)	J.Pérez
Domaine	Domaine de développement
Technologie	Technologie de développement
P-System	Système de Patrons pour les systèmes de Réalité Augmentée
Formalisme	P-SIGMA + étendu pour les systèmes de Réalité Augmentée
Identifiant	
Phase d'architecture applicative	
Participant (s)	
• « Architecte applicatif »	
Classification	
Terme (s) de domaine	{Phase ^ Processus ^ Architecture applicative ^ Composant technique ^ Conception}
Contexte	
Artefact(s) : { [REQ] Cahier Des Charges Général (CDCG) }	
Problème	
L'objectif de cette phase est de spécifier et d'analyser les composants techniques et ses supports logiciels pour un système de réalité augmentée à partir du cahier des charges général.	
Force (s)	
Force (s)	Dossier d'architecture applicative (DAA) : • Description des buts métier dans un contexte architectural, basés en couches applicatives, • Choix et description des dispositifs d'interaction, • Établissement de la cartographie des composants techniques, • Définition de règles de développement, intégration et transformation des composants techniques.
Technologie (s)	{ Réutilisation ^ Composants ^ Orienté objet }
Solution démarche	

Démarche formelle	
Démarche textuelle	Cette phase est réalisée en parallèle avec les exigences fonctionnelles, son but est de spécifier et de décrire sous une forme détaillée les composantes techniques à partir de la distribution des couches applicatives et de la sélection d'une architecture de logiciel. Cette phase décrit également les règles d'intégration et de transformation des composants techniques détaillés. Ce patron s'appuie sur les activités suivantes : • **Description des couches applicatives** : consiste en la décomposition et distribution des buts métier en plusieurs composants. • **Choix des dispositifs d'interaction** : consiste en la sélection des dispositifs à partir de cahier de charges général et l'étude de l'activité réelle de l'utilisateur. • **Description des dispositifs d'interaction** : consiste en détailler des spécifications techniques de chacun des dispositifs d'interaction qui a été choisi. • **Sélection du style d'architecture logicielle** : consiste à appliquer une architecture logicielle en fonction de la distribution de couches applicatives. • **Établissement de la cartographie des composants techniques** : consiste en la sélection des composantes et frameworks techniques sur la base des spécifications des composantes de l'activité antérieure. • **Description des composants et des classes techniques** : consiste à détailler les composants et les frameworks techniques, leurs interfaces et leurs classes spécialisables. • **Définition des règles d'intégration et transformation** : consiste à spécifier et / ou identifier les règles de conception et de transformation applicables dans la conception de l'application (par ex.

	la transformation du modèle d'analyse en modèle de conception).
Document (s)	{ Dossier d'architecture Applicative (DAA) }
Utilise	
{ Description des couches applicatives, Choix des dispositifs d'interaction, Description des dispositifs d'interaction. Sélection du style d'architecture logicielle, Établissement de la cartographie des composants techniques, Description des composants et des classes techniques, Définition des règles d'intégration et transformation}	
Requiert	
{ Phase d'étude préalable }	

Les sections suivantes détaillent les différentes activités utilisées par ce patron.

5.2.2. Description des couches applicatives

Lors du découpage architectural il s'agit de définir un schéma d'organisation en sous systèmes en termes de composants. L'organisation fixe les directives générales du développement, et les structures qu'elle induit aident au maintien de l'intégrité du modèle. Nous devons dans un premier temp prend comme dans Symphony une architecture multicouches. Cette activité sépare la logique de présentation, la logique applicative, la logique transactionnelle et de persistance (mise à jour des bases de données). Cependant, dans le cas des systèmes de RA, on doit choisir un style d'architecture. Cela sera fait par la suite. Avant il convient de déterminer les dispositifs d'interaction.

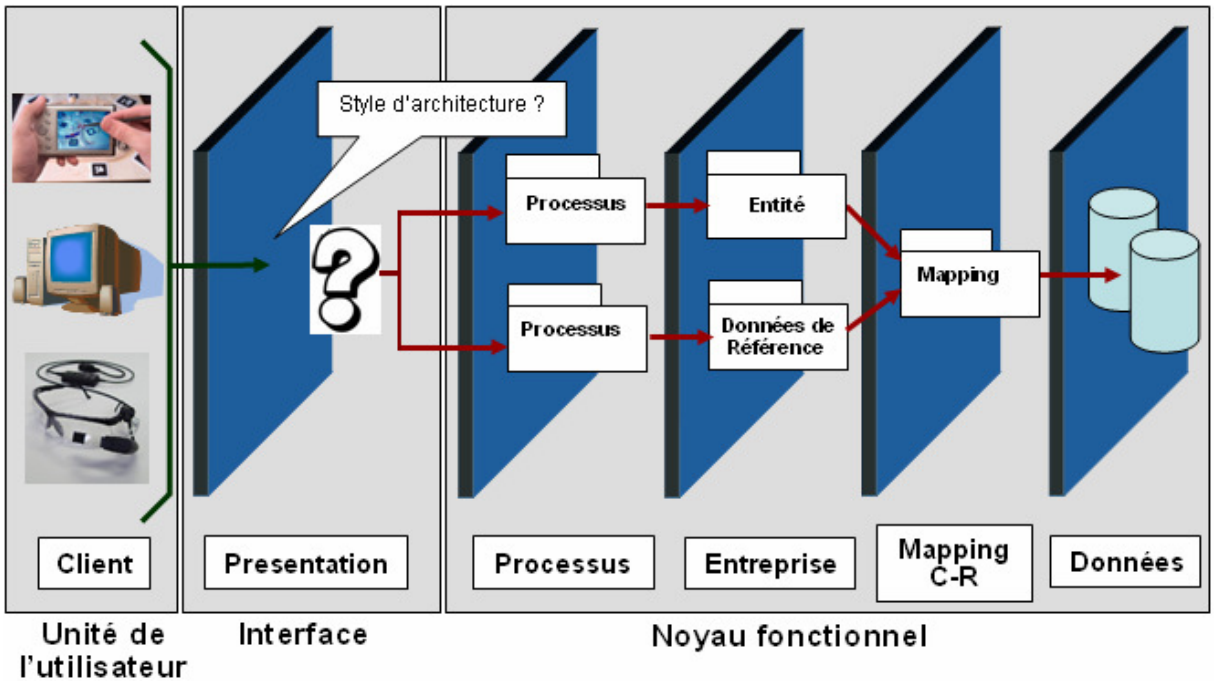


FIGURE 23 : DOSSIER D'ARCHITECTURE EN COUCHES [INITIAL]

5.2.3. Choix des dispositifs d'interaction

Une fois identifiée la nécessite de développer un système de RA dans la branche fonctionnelle, l'objectif est de créer un Dossier des Dispositifs d'Interaction (DDI). Les

dispositifs d'interaction sont très nombreux et il est souvent difficile pour un concepteur d'application de choisir tel ou tel dispositif. Cependant, nous considérons que la sélection doit être effectuée sur la base des besoins d'interaction à partir des documents : Cahier des charges général (CDCG) et spécification des dispositifs l'interaction (SDI) et pour les caractéristiques propres du dispositif requis.

Dans le cadre de l'EDL, l'étude de la interaction a déterminé les besoins suivants :

Type de dispositifs requis
1. Support vision.
2. Surface tactile.
3. Capteur de localisation et d'orientation.
4. Microphone.
5. Ecouteur.

Tableau 8 : SPECIFICATIONS DES DISPOSITIFS D'INTERACTION

Notre méthode pour faciliter la sélection des dispositifs, s'appuie sur l'usage de la notation QOC[7] [MacLean 91], qui permet de faciliter, de classifier et de documenter les décisions de conception.

Pour faciliter notre choix on doit considère les types de besoins interactifs suivants :

- Vision
- Affichage tactile
- Capteur de localisation et d'orientation
- Commandes vocales

Vision

Par rapport au besoins de vision on doit considère les critères de choix suivants :

Critère	Description
Accessibilité	Désigne le caractère possible de la liberté de déplacement dans l'espace, d'utilisation de dispositifs, et de sa compréhension.
Diversité du signal	L'image générée par l'ordinateur est affichée sur un écran vidéo qui peut être de différentes technologies. Les affichages à tubes cathodiques sont aujourd'hui dépassés (car trop lourd et trop gourmand en énergie) et ce sont surtout les afficheurs à cristaux liquides (LCD) qui sont utilisés. Généralement, sur les écrans l'image arrive de manière non entrelacée, elle est dite "progressive". La perception réelle d'une image entrelacée est équivalente à 60 % du nombre de lignes affichées à l'écran.
Consommation d'énergie	Un niveau bas de consommation d'énergie étend l'usage du dispositif.
Mobilité du dispositif	Ce critère facilite l'usage et l'accès des solutions de réalité augmentée, en offrant la plus grande satisfaction à l'usager.
Poids	Représente une mesure maximale dans des termes de poids des dispositifs de visualisation
Espace de visualisation	Nous devons considérer maximiser la portée et l'angle de vision qui peut être obtenue par un dispositif avec le but de donner les bonnes informations sans surcharger l'interface du l'utilisateur.
Taille	Représente une mesure maximale dans des termes de résolution utilisée par les

[7] QOC: Questions, Options et Critères, est une notation semi formelle proposée par MacLean. Cette notation donne lieu à une représentation sous forme de diagramme. Pour plus d'informations voire annexe 2.

	dispositifs de visualisation. (Par exemple : résolution 480 x 640, 65 536 couleurs (VGA)). Ce paramètre nous permettra de mesurer la capacité du dispositif d'affichage à conserver les dimensions et proportions du monde réel.
Prix	Le prix d'un dispositif évidemment peut influer sa sélection. Nous adopterons ce critère comme une valeur minimale.
Résolution	Consiste dans le nombre de points que peut représenter le monitor pour écran, en horizontal et vertical
Qualité de vision	Consiste de critères tells comme : définition de couleurs, capacité alphanumérique de l'écran, clarté de l'image (luminosité), latence ou retard de la signal. La différence de luminosité entre les scènes réelles et virtuelles qui font que l'une des deux scènes est mal perçue par l'utilisateur car trop sombre. Ceci est particulièrement sensible avec les casques semi-transparents.

Tableau 9 : CRITERES DE CHOIX DES BESOINS DE VISION

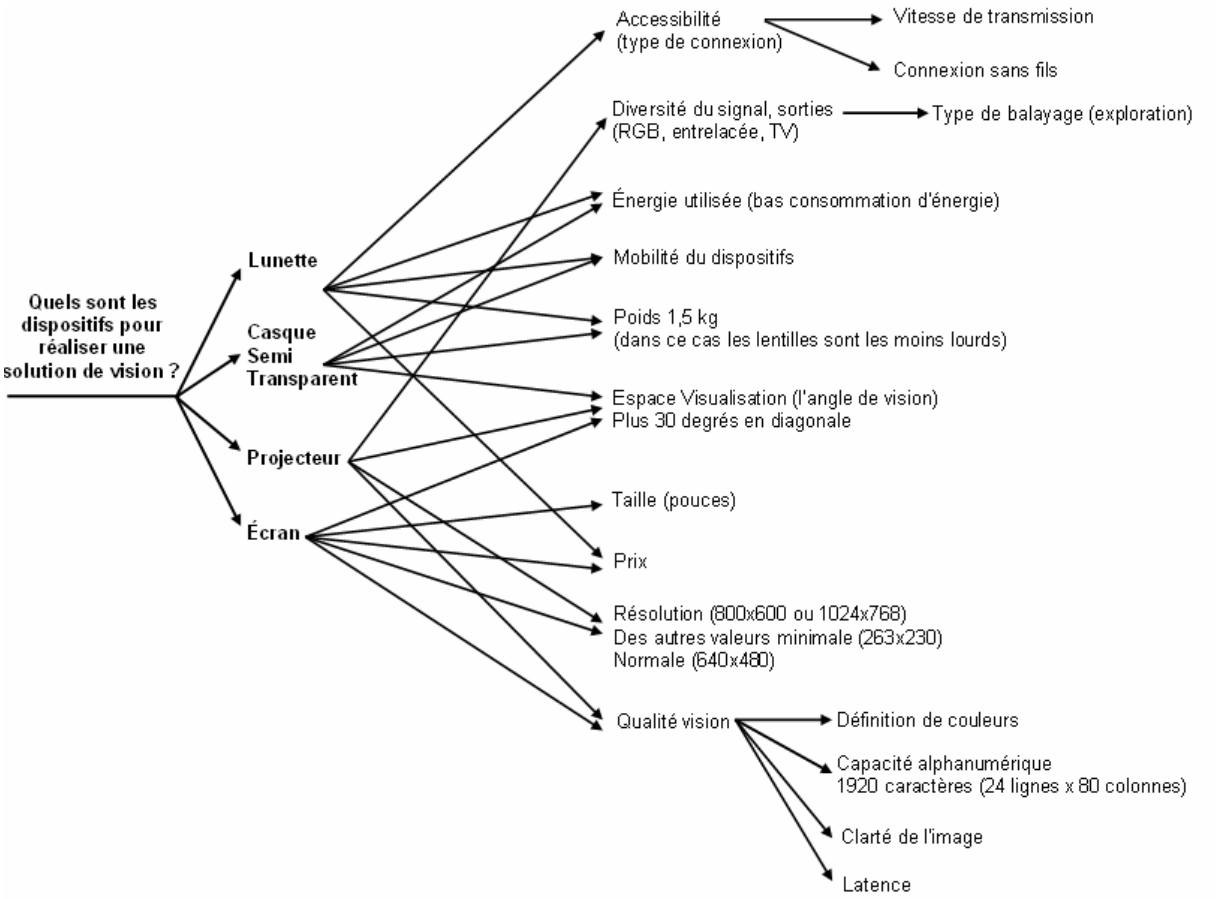


FIGURE 24 : CRITERES DE SELECTION DES DISPOSITIFS DE VISION

Dans cadre de l'EDL, nous avons choisi un **casque semi transparent**, ce type de dispositif est de consommation d'énergie bas, garantit la mobilité de l'utilisateur et offre un ample espace de visualisation

Affichage tactile

Par rapport au besoins d'affichage tactile on doit considère les critères de choix suivants :

Critère	Description
Accessibilité	Désigne le caractère possible de la liberté de déplacement dans l'espace, d'utilisation de dispositifs, et de sa compréhension.

Résolution	Consiste dans le nombre de points que peut représenter le monitor pour écran, en horizontal et vertical
Prix	Le prix d'un dispositif évidemment peut influer sa sélection. Nous adopterons ce critère comme une valeur minimale.
Poids	Représente une mesure maximale dans des termes de poids des dispositifs de visualisation
Consommation d'énergie	Un niveau bas de consommation d'énergie étend l'usage du dispositif.
Taille	Représente une mesure maximale dans des termes de résolution utilisée par les dispositifs de visualisation. (Par exemple : résolution 480 x 640, 65 536 couleurs (VGA)). Ce paramètre nous permettra de mesurer la capacité du dispositif d'affichage à conserver les dimensions et proportions du monde réel.
Diversité du signal	Ce critère considère les caractéristiques du la signal. Généralement, sur les écrans l'image arrive de manière non entrelacée, elle est dite "progressive". La perception réelle d'une image entrelacée est équivalent à 60 % du nombre de lignes affichées à l'écran.
Mobilité du dispositif	Ce critère facilite l'usage et l'accès des solutions de réalité augmentée, en offrant la plus grande satisfaction à l'usager.
Type d'interaction	Manipulation directe/indirecte
Capacité de stockage	Ce critère consiste en capacité du dispositif de stocker des donnes d'une application.
Disponibilité de Systèmes d'exploitation	Ce critère contemple la sélection d'un dispositif, en tenant en compte du type de système d'exploitation supporté.

Tableau 10 : CRITERES DE CHOIX DES BESOINS D'AFFICHAGE TACTILE

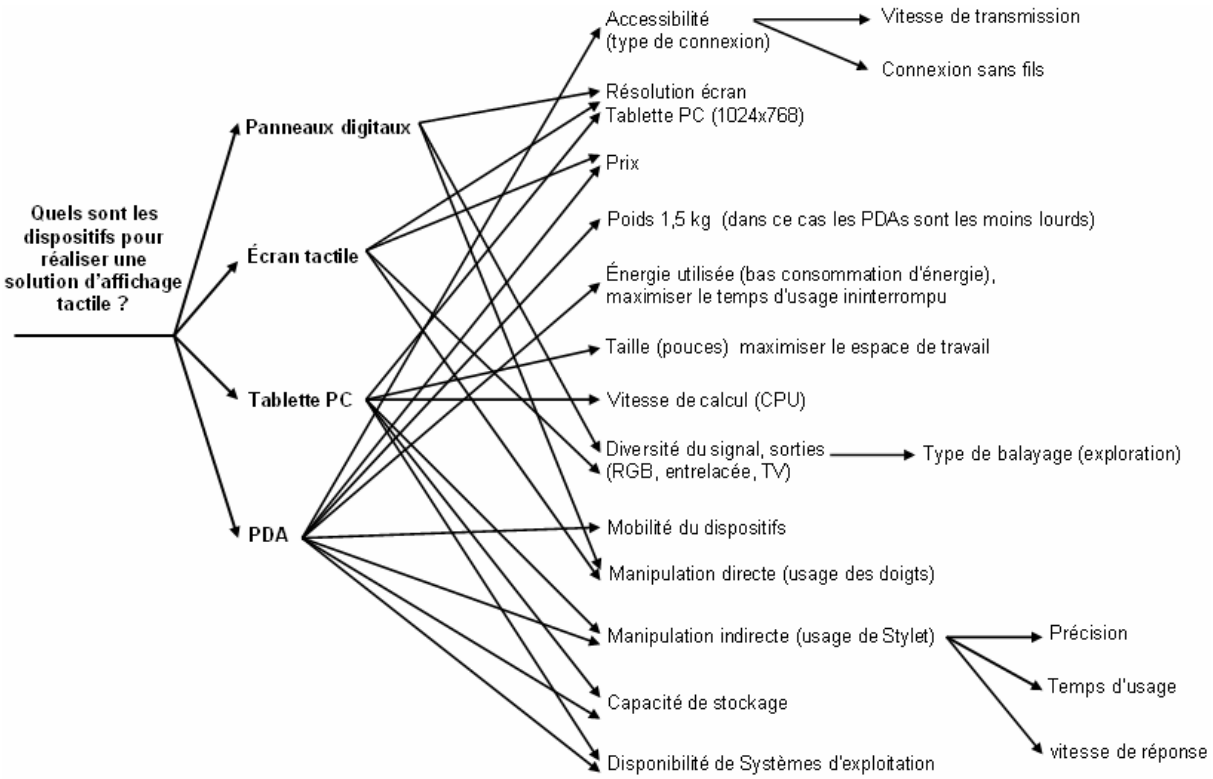


FIGURE 25 : CRITERES DE SELECTION DES DISPOSITIFS D'AFFICHAGE

Dans cadre de l'EDL, nous avons choisi un dispositif **PDA**. ce type de dispositif permet l'accessibilité du système en utilisant une connexion Wi-Fi, aussi, garantit la manipulation vers un système de menus en utilisant un stylet, est de consommation d'énergie bas, bon marché, garantit la mobilité de l'utilisateur et offre des facilites

pour le stockage des données.

Capteur de localisation et d'orientation

Par rapport au besoins de localisation et d'orientation on doit considère les critères de choix suivants :

Critère	Description
Accessibilité	Désigne le caractère possible de la liberté de déplacement dans l'espace, d'utilisation de dispositifs, et de sa compréhension.
Diversité du signal	L'image générée par l'ordinateur est affichée sur un écran vidéo qui peut être de différentes technologies. Les affichages à tubes cathodiques sont aujourd'hui dépassés (car trop lourd et trop gourmand en énergie) et ce sont surtout les afficheurs à cristaux liquides (LCD) qui sont utilisés.
Consommation d'énergie	Un niveau bas de consommation d'énergie étend l'usage du dispositif.
Mobilité du dispositif	Ce critère facilite l'usage et l'accès des solutions de réalité augmentée, en offrant la plus grande satisfaction à l'usager.
Poids	Représente une mesure maximale dans des termes de poids des dispositifs de visualisation
Espace de visualisation	Nous devons considérer maximiser la portée et l'angle de vision qui peut être obtenue par un dispositif avec le but de donner les bonnes informations sans surcharger l'interface du l'utilisateur.
Taille	Représente une mesure maximale dans des termes de résolution utilisée par les dispositifs de visualisation. (Par exemple : résolution 480 x 640, 65 536 couleurs (VGA)). Ce paramètre nous permettra de mesurer la capacité du dispositif d'affichage à conserver les dimensions et proportions du monde réel.
Prix	Le prix d'un dispositif évidemment peut influer sa sélection. Nous adopterons ce critère comme une valeur minimale.
Résolution	Consiste dans le nombre de points que peut représenter le monitor pour écran, en horizontal et vertical
Qualité de vision	Consiste de critères tells comme : définition de couleurs, capacité alphanumérique de l'écran, clarté de l'image (luminosité), latence ou retard de la signal. La différence de luminosité entre les scènes réelles et virtuelles qui font que l'une des deux scènes est mal perçue par l'utilisateur car trop sombre. Ceci est particulièrement sensible avec les casques semi-transparents.

Tableau 11 : CRITERES DE CHOIX DES BESOINS DE LOCALISATION ET D'ORIENTATION

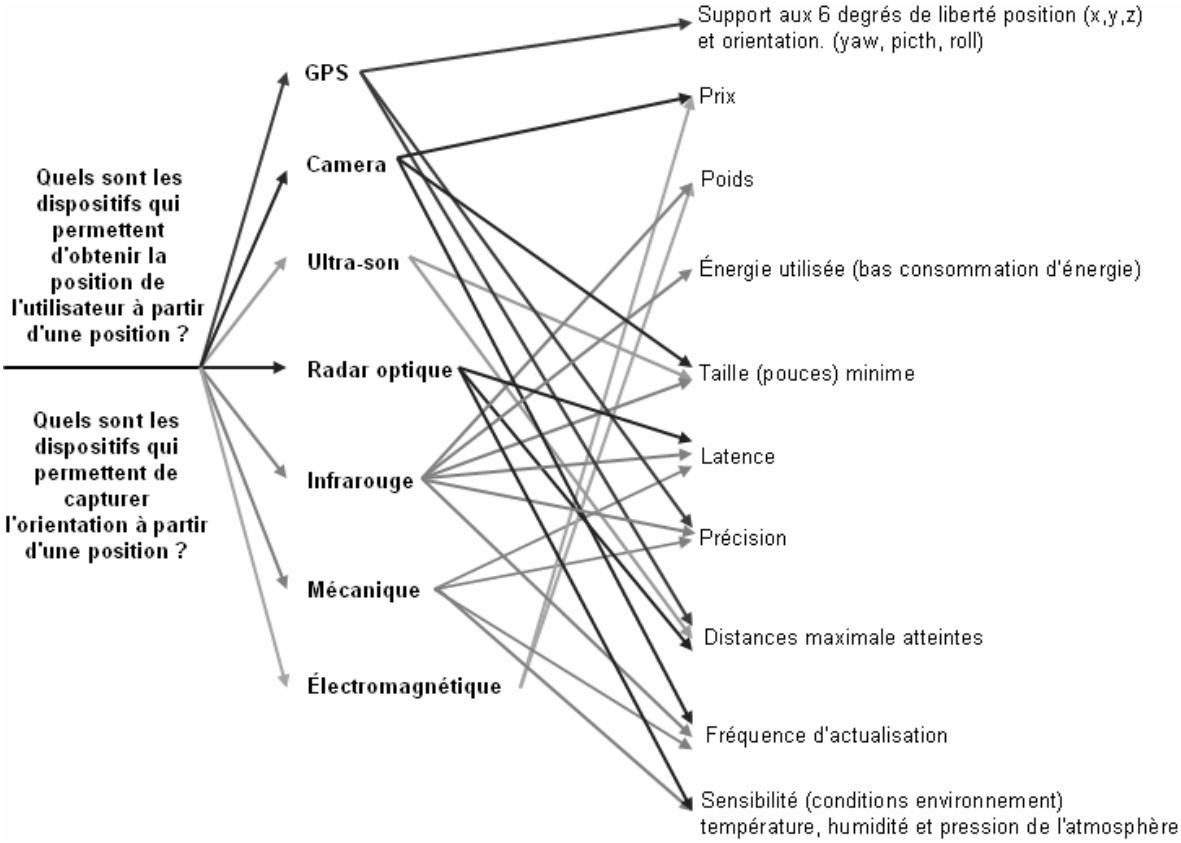


FIGURE 26 : CRITERES DE SELECTION DES DISPOSITIFS DE LOCALISATION ET D'ORIENTATION

Dans cadre de l'EDL Nous avons choisi une capteur de orientation de type **Ultra son** ce capteur à cause de sa taille, sa précision et sur la base de ses caractéristiques (les spécifications seront présentées dans sections suivants).

Par rapport au capteur de localisation, nous avons sélectionné la technique **Magicien d'Oz** par être une solution effective et à un très bas prix. Cependant, une solution basée sur un capteur de localisation pouvait utiliser des critères que nous proposons.

Commandes vocales

Par rapport au besoins de commandes vocales on doit considère les critères de choix suivants :

Critère	Description
Directivité	Détermine dans quel direction capte mieux (d'une forme plus efficiente) le son un microphone, c'est-à-dire, indique la sensibilité du microphone à différentes directions.
Sensibilité	Consiste en la efficience du microphone. En dépendant du type de micro nous aurons une plus grande ou mineur sensibilité.
Fidélité	La fidélité indique la variation de sensibilité respect à la fréquence. De même, la fidélité vient définie comme la propre réponse dans une fréquence du microphone, puisque le son capté par un micro ne va jamais être exactement égal au réel. Il y aura des fréquences qui ont été atténuées, alors que les autres auront été augmentées.
Bruit de fond	C'est la tension ou marque qui nous remet le microphone sans qu'aucun son

	n'existe en tombant sur lui. Le bruit de fond doit être comme maximum autour de 60 dB. Pour qu'un microphone soit idoine le bruit magnétique doit être mineur de 15 dB et le champ magnétique doit être mineur de 10 dB.
Prix	Le prix d'un dispositif évidemment peut influer sa sélection. Nous adopterons ce critère comme une valeur minimale.
Type d'usage	Ce critère nous permettra de réaliser une sélection d'un dispositif dans le cas d'être requis pour l'usage individuel ou pour l'usage collectif.

Tableau 12 : CRITERES DE CHOIX DES BESOINS DE COMMANDES VOCALES

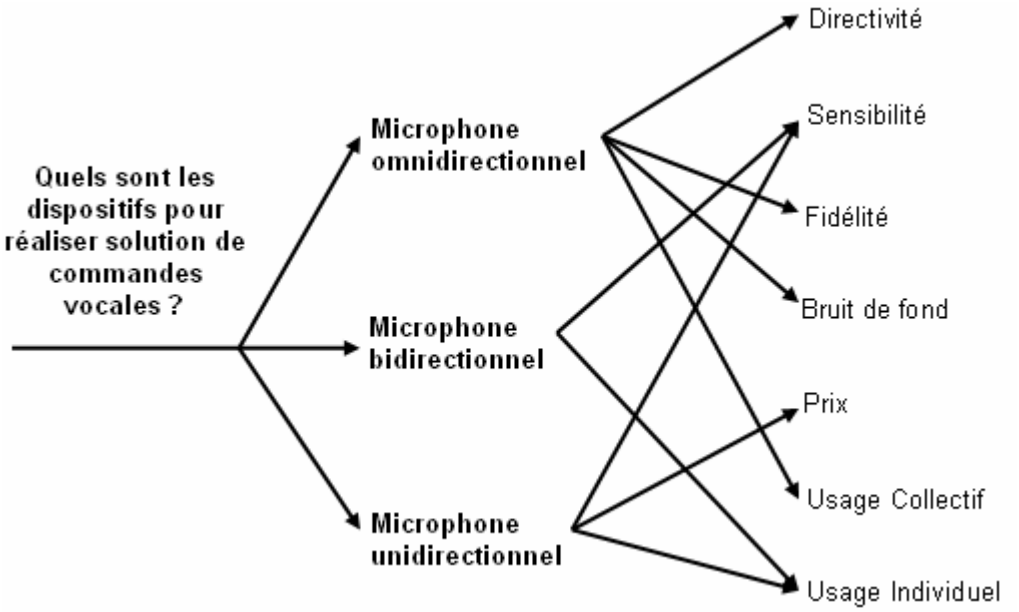


FIGURE 27 : CRITERES DE SELECTION DES DISPOSITIFS DE SONS

Dans notre cas d'expérimentation, nous avons choisi un **microphone unidirectionnel**. Ce projet requiert un microphone bon marché et d'un usage individuel.

Dans le cadre de l'EDL, la sélection des dispositifs d'interaction se résume par un diagramme de déploiement, puis, dans la section suivant nous présenterons pour chaque dispositif ses spécifications.

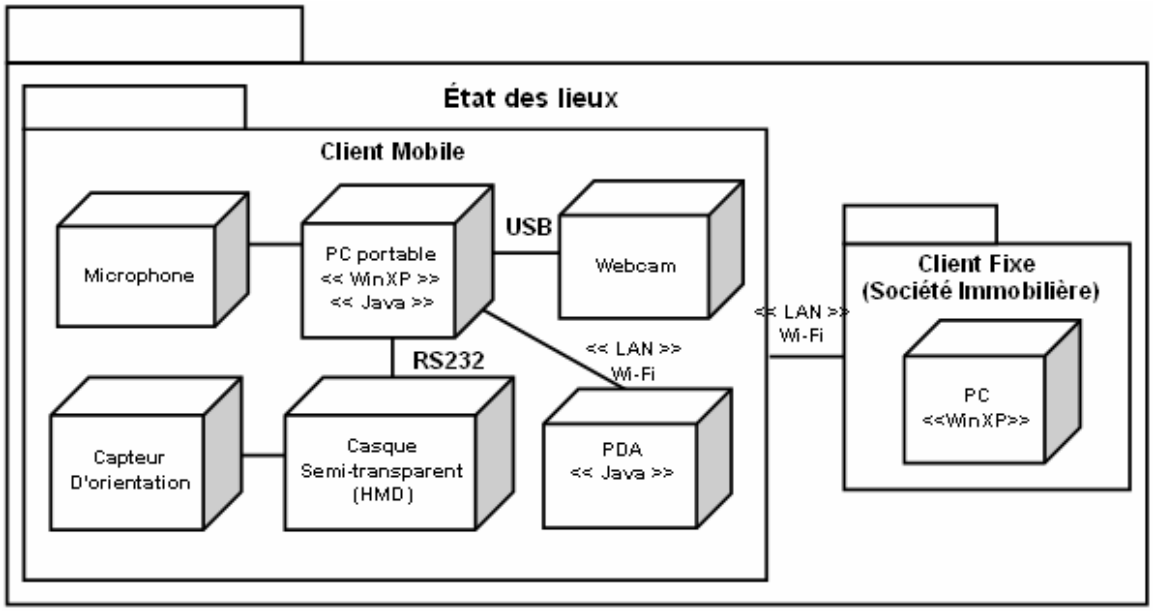


FIGURE 28 : SELECTION DES DISPOSITIFS D'INTERACTION

5.2.4. Description des dispositifs d'interaction

Cette activité consiste en détailler des spécifications techniques de chacune des dispositifs d'interaction qui a été choisir dans l'activité précédente.

	Dispositif	Spécifications techniques
Unité mobile	PC Portable	Modèle **SONY VAIO PCG-F809K**, processeur : Pentium III - 850MHz, RAM : 128Mo, Système d'exploitation : Windows XP, Disque dur : 30Go, rotation à 4200 tours/min, 2 ports PCMCIA : 1er port : carte Ethernet 3COM 10/100Mbps, 2nd port : carte WiFi U.S. Robotics 802.11b - 11 Mbps. Entrée audio (microphone), sortie VGA, 2 ports USB, 1 port COM/Serial, dimensions : 32.4 x 26.6 x 5.4 cm, Poids : 3.2 Kg.
	Casque semi-transparent (HMD)	Modèle **SONY PC Glasstron PLM-S700E** (écran à cristaux liquides 1.5 mégapixels à transparence réglable, résolution maximale : 800 x 600 à 85Hz, Entrée VGA classique, RGB et S-Video pour assurer le transfert des données vidéo. Oreillettes pour des notifications sonores et/ou des messages vocaux à l'utilisateur).
	Capteur d'orientation	Modèle **InterSense InterTrax** (Capte les 3 angles de liberté (X, Y et Z), Intervalles : X : +/- 80°, Y : +/- 180°, Z : +/- 90°, Latence : 4ms, Surface à scratch pour la fixation, Ports : USB, COM/Serial, Poids : 39 grammes).
	Capteur de localisation	En vue des coûts élevés d'un capteur de localisation l'on peut facilement le remplacer par un « Magicien d'Oz ».
	Microphone	Microphone générique avec deux sorties d'audio. Il est fixé au casque et relié à l'ordinateur portable, assure la capture de la voix de l'utilisateur et rend ainsi possible le contrôle de l'application par commandes vocales.
	PDA	Modèle **Dell AXIM X30**, utilisant Windows Mobile 2003. Il est doté du Standard Wi-Fi (802.11b) et d'un écran tactile avec stylet de résolution : 240 x 320 pixels en 16 bits. C'est cette dernière caractéristique et sa petite taille qui justifient le choix d'un tel dispositif. : les étapes demandant de positionner et/ou de sélectionner des objets (des dommages) demandent une précision que la commande vocale ne peut assurer. Le PDA transmet à l'ordinateur portable les déplacements réalisés par l'utilisateur.

	Webcam	Modèle **Philips ToUcam PRO PCVC740K** ou similaire (Capteur CCD 1.2 mégapixels, enregistrement jusqu'à 60 images/seconde, résolution maximale en capture vidéo : 640 x 480 et en capture photo : 1280 x 960 en 24bits).
Poste fixe	Ordinateur fixe	Ses spécifications contemplent la technologie Wi-Fi (802.11b) 11 Mbps. Ce dispositif supportera la plupart de l'application et sera relié au PDA par liaison Wi-Fi.

Tableau 13 : DESCRIPTION DES DISPOSITIFS D'INTERACTION

5.2.5. Sélection du style d'architecture logicielle

Un modèle d'architecture définit un guide pour l'organisation du logiciel et permet de décomposer le système en sous-systèmes. De la même manière que par les dispositifs, notre méthode pour la sélection d'une architecture logicielle s'appuie sur l'usage de la notation QOC.

Par rapport à la sélection des styles d'architecture on doit considère les critères de choix suivants :

Critère	Description
Complexité de communication	ce critère consiste de règles de communication proposées par les styles d'architecture qui facilitent ou pas la communications parmi les composants techniques.
Type d'architecture	ce critère permet de classifier les architectures en monolithiques et basée agents.
Complexité de conception	consiste en la difficulté de analyser, concevoir et développer une application en utilisant un style d'architecture.
Environnement distribue	ce critère consiste dans le degré qui offre une architecture logicielle pour permettre des développements basés sur des environnements distribues. Les styles d'architectures basées agents garantissent l'usage d'applications dans des environnements distribues.
Usage des outils de conception	ce critère considère qu'une architecture logicielle est fortement éligible, si elle dispose des outils de conception.
Concurrence	ce critère est liée au l'exécution des processus en parallèle. Dans le cas des styles d'architectures les agents peuvent garantir ce principe.
Facilité de maintenance	ce critère consiste en l'effort nécessaire pour trouver et pour corriger une erreur dans une application qui est base sur un style d'architecture particulier.
Modularité	consiste en les principes de découpage/affinement des structures (composants) d'une architecture logiciel.
Facilité pour évoluer le modèle	ce critère consiste dans la complexité pour évoluer un style d'architecture logicielle.
Portabilité	ce critère est liée au la facilite pour adapter, intégrer ou remplacer des composants de une architecture logicielle.
Facilité de réutilisation	Ce critère mesure le degré dans lequel une style d'architecture logiciel peut être utilisé dans une autre architecture.
Interopérabilité	est le fait que plusieurs architectures, qu'ils soient identiques ou radicalement différents, puissent communiquer sans ambiguïté et opérer ensemble.
Performance	Ce critère mesure le degré d'efficience et le temps de réponse d'une application qui est basée sur une architecture logiciel particulier.

La figure 29 permet d'illustrer les différentes options qui permettent de réaliser la

sélection d'un style d'architecture.

FIGURE 29 : CRITERES DE SELECTION DES ARCHITECTURES LOGICIELLES

Par rapport à notre cas d'expérimentation, nous avons décidé d'adopter le style d'architecture ARCH conçu pour les applications interactives. Nous avons choisi Arch parce qu'il est une architecture avec bas complexité de conception et de communications des composants. Aussi que, Il est de maintenance et preuve facile. Le découpage en 5 niveaux de cette architecture permet la modularité de l'application et par conséquence la réutilisation des composants. La sélection d'un ensemble d'architectures se résume dans le Dossier d'Architecture en Couches (DAC).

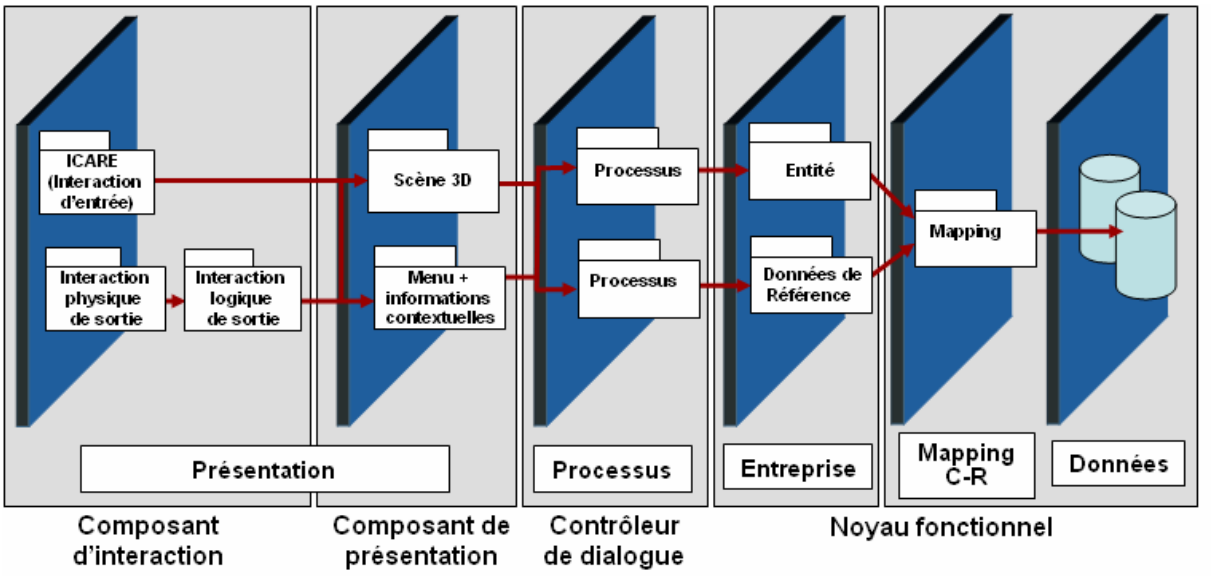


FIGURE 30 : DOSSIER D'ARCHITECTURE EN COUCHES [COMPLET]

NB : Les caractéristiques d'interaction décrites dans la suite de ce chapitre permettent d'identifier que la conduite de l'interaction d'entrée est multimodale (Le PDA est utilise comme surface tangible mais aussi il devra reconnaître des commandes vocaux).

5.2.6. Établissement de la cartographie des composants techniques

Dans cette phase nous devons décrire l'ensemble des interactions entre les composants de l'application et éventuellement le framework de l'entreprise. Dans cette activité, l'architecte doit d'organiser les technologies à mettre en place en convergence avec les besoins fonctionnels.

Généralement, le résultat de cette activité est décrit par des schémas spécifiques du fait d'un manque dans la notation des diagrammes de composants d'UML. La figure suivante permet d'illustrer la cartographie des composants techniques pour notre cas d'expérimentation. Une brève description de certains choix est présenté.

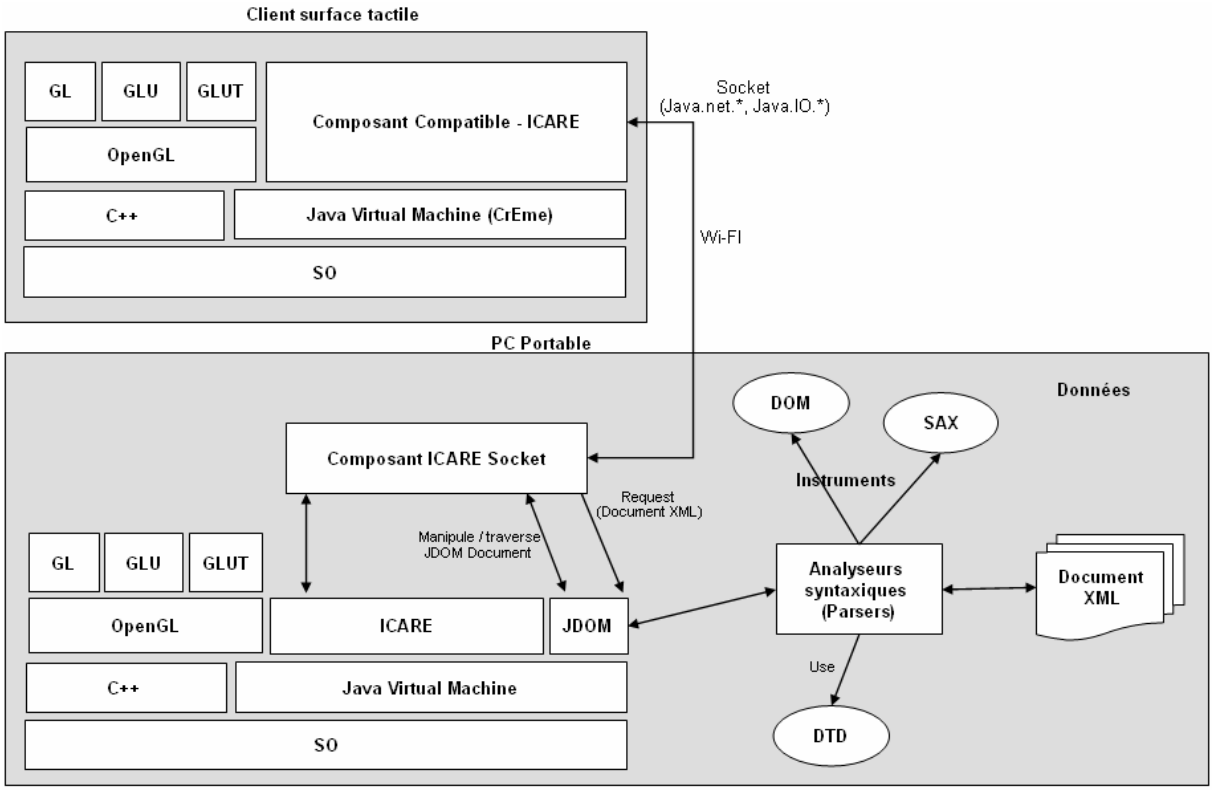


FIGURE 31 : CARTOGRAPHIE DES COMPOSANTS TECHNIQUES (CCT)

Dans la figure nous pouvons visualiser l'usage de la plate-forme Java, pour la conception des composants techniques. Nous avons choisi cette plate-forme pour l'existence de plusieurs bibliothèques et outils essentiels écris sur la plate-forme Java.

Choix de la bibliothèque pour les objets 3D

Comme nous l'avons indiqué précédemment, le projet nécessite de réaliser une scène en trois dimensions du logement. Dans ce sens, nous avons choisi la bibliothèque JoGL. Il est une interface Java à OpenGL, célèbre bibliothèque graphique libre, codée en C++ et utilisée par nombre de jeux et applications professionnelles. De ce fait, la rapidité d'exécution est supérieure. Par analogie, on utilisera le terme OpenGL ou même GL pour désigner JoGL.

JoGL est fourni avec plusieurs autres bibliothèques, de nature plus haut niveau, comme GLU et GLUT, qui sont également des portages Java des bibliothèques de mêmes noms écrites en C++. Elles regroupent des méthodes plus concrètes et plus simples, accélérant le développement. On y trouve par exemple, une gestion basique de la caméra, une gestion avancée des textures et la création de formes géométriques de base.

La bibliothèque GLUT propose la création de tores, ce qui convient à la concept de marqueur encerclant les dommages. Elle propose également l'affichage basique de texte à l'intérieur de la scène. La bibliothèque GLU, elle, fût utilisée notamment pour ses opérations portant sur la gestion de la caméra et ses réglages.

Choix du mode de stockage

Le format XML a été choisi pour sauvegarder les données de l'application. JDOM est une implémentation spécifique à Java. Elle s'appuie au choix du programmeur soit sur DOM, soit sur SAX, technologies que permettent de manipuler des fichiers XML.

5.2.7. Description des composants et des classes techniques

Comme nous l'avons présenté précédemment, cette activité consiste à détailler les composants techniques, leurs interfaces en particulier, les classes spécialisables, si nécessaire leur fonctionnement interne notamment pour les frameworks techniques. L'architecte applicatif doit également définir le cadre d'utilisation du composant (d'origine ou spécialisé) dans l'application. Cette description peut se faire par des exemples concrets d'utilisation.

La description suivante illustre une exemple d'un composant technique :

Description des composants ICARE (Interaction Complémentarité Assignation Redondance Equivalence) pour la d'interfaces multimodal[8]

ICARE aide à la construction d'applications interactives multimodales en proposant des composants réutilisables. Il utilise une approche à composants (JavaBeans) qui facilite la mise en œuvre de la combinaison de modalités. De plus, il propose un éditeur graphique qui à partir d'un schéma ICARE génère le code correspondant.

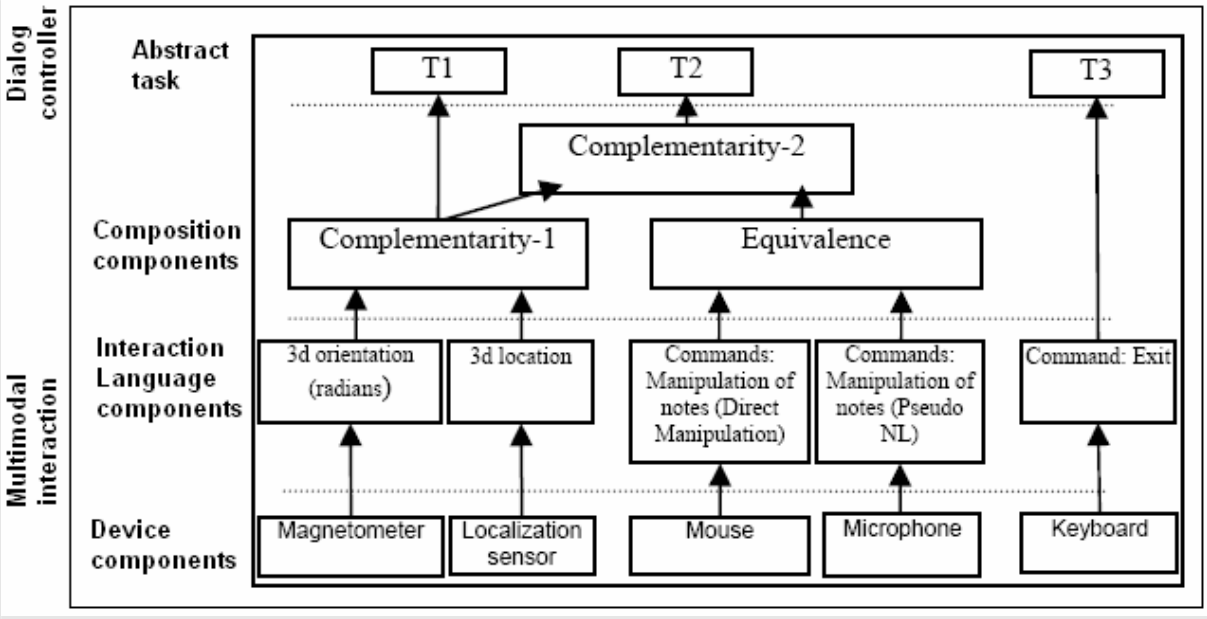
Les composants d'ICARE sont indépendants des modalités et peuvent donc être réutilise dans différentes applications. Ils remplissent les fonctions d'interaction physique et logique en entrée/sortie en fournissant au système via le contrôleur de dialogue une expression symbolique de la commande. Dans le cas des systèmes de réalité augmentée ce composant permet de utiliser diverses types d'interaction, telles comme : Commandes vocales, binaire (manipulation directe), orientation et localisation. Cela, permet de dire que dans las applications basées ICARE la perception de l'utilisateur est augmentée.

On distingue deux types de composants ICARE : les composants élémentaires et des composition.

Composant élémentaires : Les composant élémentaires sont l'association des composants Dispositifs et Systèmes Représentationnels qui permet la multi modalité. Le composant Utilisateur représente les propriétés de l'utilisateur qui peuvent concerner l'identité, les coordonnées ainsi que les données biométriques et sociales qui le caractérisent [Bouchet 04]. Le composant Environnement permet d'enregistrer des données sur l'environnement physique du contexte d'utilisation (température, niveau sonore, etc.).

Composants de composition : On distingue trois sortes de composants de composition : Le composant Complémentarité traduit la propriété ergonomique de Complémentarité et représente la composition de plusieurs dispositifs physiques (niveau physique) ou de plusieurs systèmes représentationnels (niveau logique). Le composant Redondance traduit la propriété ergonomique de redondance, c'est-à-dire la possibilité d'utiliser simultanément deux modalités différentes pour interagir avec le système. Le composant Equivalence/Redondance traduit les propriétés ergonomiques d'équivalence et de redondance, c'est-à-dire la possibilité d'utiliser deux modalités différentes, de manière simultanée ou non, pour interagir avec le système. Finalement, les opérations de fusion de deux modalités se font au niveau de ces trois composants.

[8] Un système est multimodal s'il dispose d'au moins deux modalités pour le traitement des données de entrée ou sortie.



NB : Structure ICARE.

Tableau 14 : DESCRIPTION DU COMPOSANT ICARE

5.2.8. Définition des règles d'intégration et transformation

Une des dernières activités de la phase de architecture applicative de la méthode Symphony consiste à spécifier et / ou identifier les règles de conception et de transformations applicables dans la conception de l'application. Nous n'avons actuellement pas identifie de transformations spécifique a la RA.

Chapitre VI. Contributions et Perspectives

6.1. Contribution

Dans le cadre de ce travail, nous avons étendu la démarche Symphony pour prendre en compte les problèmes techniques lies a la conception des systèmes de réalité augmenté. Cette approche est à retenir pour, non seulement, apprendre toute méthode de développement mais aussi, pour transmettre les connaissances ainsi formalisées. Il s'agit ici de regrouper des savoir-faire et des connaissances appliqués à l'ingénierie des Systèmes Réalité Augmentée.

En résumé, les principales contributions de notre travail sont :
1. Des patrons processus pour documenter et capitaliser des connaissances relatives à la démarche de conception des spécifications techniques pour les systèmes de Réalité Augmentée. Nous obtenons ainsi un guide méthodologique définissant une démarche propre à l'ingénierie des systèmes de réalité augmentée.
2. Des critères de choix pour les architectures logicielles et les dispositifs d'interaction.
3. Une contribution à une meilleure description des architectures logicielles en utilisant patrons produit.

6.2. Perspectives

Ce travail a permis la mise en place d'un cadre conceptuel pour l'ingénierie des systèmes de réalité augmentée. Il s'agit d'un système de patrons, formulant des modèles et une démarche techniques de la méthode Symphony pour la conception des systèmes de réalité augmentée. Ce travail peut se prolonger vers plusieurs perspectives que nous présentons par la suite.

Le système de patrons proposé représente en majorité des patrons de processus. Certains de ces patrons pourraient en plus être complets pour préciser la formalisation et la manière de documenter les composants et frameworks techniques. De plus, notre guide méthodologique ne couvre actuellement pas toute la branche droite de Symphony. Une perspective à long terme consiste à définir une méthode complète (de l'étude préalable jusqu'aux tests) de développement des systèmes de réalité augmentée.

Quant aux patrons produit décrivant les architectures logicielles, ils pourraient être avantageusement enrichis au niveau de leur rubrique Solution Produit afin de formaliser, sous forme de diagramme de classes, la structure des artefacts à produire. Des patrons produit pourraient aussi être conçus pour toute solution logicielle (framework, toolkit ou composant) existant pour les systèmes de réalité augmentée. Cela permettrait aux concepteurs de connaître toutes les solutions techniques existantes afin de choisir la plus appropriée à leur solution interactive.

Le système de patrons pour la RA a été utilisé et mis à jour lors de l'expérimentation

de la démarche Symphony étendue sur un système d'Etat de lieux. Dans l'état, ce système de patrons peut être utilisé globalement ou partiellement. Néanmoins le développement du cas d'étude de l'état des lieux nous a montré l'intérêt d'effectuer des expérimentations pour valider nos propositions. Nous suggérons donc de continuer les expérimentations avec quelques systèmes de Réalité Augmentée.

Finalement, nous considérons important l'intégration de notre système de patrons avec d'autres travaux en cours, en particulier, nous suggérons l'étude de ces patrons de réalité augmentée dans des contextes interactifs où l'informatique est ubiquitaire et les dispositifs fortement distribués.

Chapitre VII. Références

A

[Ahlers 94] K. Ahlers, D. Breen, C. Crampton, E. Rose, M. Tuceryan, R. Whitaker, and D. Greer. An augmented vision system for industrial applications. In *Telemanipulator and Telepresence Techonolgies*, vol. 2351, pages 345–359, Boston, 1994.

[Alexander 77] Alexander C., Ishikawa S., Silverstein M., Jacobson M., Fiksdahl-King I., Angel S., A Pattern Language, Oxford University Press, New York, 1997.

[Ambler 98] Ambler S. Process Patterns: building Large Scale Systems using Object Technology, Cambridge University Press, December 1998.

[André 94] André J. Moving From Merise to Schlaer and Mellor, vol. 3, 1994.

[Azuma 01] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6), pages 34–47, November/December 2001.

B

[Baudel 95] Baudel J. Aspects Morphologiques de l'Interaction Humain Ordinateur : Etude de Modèles d'Interaction Gestuels. Thèse de Doctorat en Informatique soutenue le 15 décembre 1995 à l'Université de Paris Sud.

[Bainville 95] Bainville, E., Chaffanjon, P., Cinquin, P. (1995), Computer generated visual assistance to asurgical operation: the retroperitoneoscopy, *Computers in Biology and Medicine,* 25(2)*,* pages 165-171, 1995.

[Bauer 01] Bauer M., Bruegge B., Klinker G., MacWilliams A., Riß S., Sandor C., Wagner M. Design of a Component-Based Augmented Reality Framework. Proceedings of The Second IEEE and ACM International Symposium on Augmented Reality (ISAR 2001), pages 45-54, October 2001.

[Beck 87] Beck k. Using Pattern Languages for Object-Oriented Programs, OOPSLA-87 workshop on the Specification and Design for Object-Oriented Programming. Technical Report Nro. CR-87-43, September 1987.

[Bellik 95] Bellik Yacine, Interfaces multimodales: concepts, modeles et architectures. These de Doctorat en Informatique soutenue le 30 mai 1995 à l'Université de Paris-XI.

[Bernstein 96] Bernstein P. Middleware: A Model for Distributed Services. Communications of the ACM Nro. 39, pages 86-97, February 1996.

[Booch 99] Booch G., Rumbaugh J., Jacobson I., The Unified Modeling Language User Guide, Addison-Wesley, 1999.

[Bouchet 04] Bouchet, J. & Nigay, L. ICARE: A Component-Based Approach for the Design and Development of Multimodal Interfaces, pages 1325-1328, April 2004.

[Broll 05] Broll W., Lindt I., Ohlenburg J., Herbst I., Wittkämper M., and Novotny T. An Infrastructure for Realizing Custom-Tailored Augmented Reality User Interfaces. IEEE Transactions on Visualization and Computer Graphics. Volume 11, Issue 6 (November 2005) Pages: 722 – 733.

[Buschmann 96] Buschmann F., Meunier R., Rohnert H., Sommerlad P., and Stal M., Pattern-Oriented Software Architecture - A System of Patterns. John Wiley and Sons, 1996.

C

[CASTOR 06] Site officiel, http://www.castor.org

[Caudel 92] Caudel T. and Mizell D. Augmented reality : an application of heads-up display technology to manual manufacturing processes.1992.

[Cheok 02] A. Cheok, W. Weihua, X. Yang, S. Prince, F. Wan, M. Billinghurst, and H. Kato. Interactive theatre experience in embodied and wearable mixed reality space, IEEE and ACM, pages 59–68, 2002.

[Cinquin 95] Cinquin, P., Bainville, E., Barbe, C., Bittar, E., Bouchard, V., Bricault, I., Champleboux, G., Chenin, M., Chevalier, L., Delnondedieu, Y., Desbat, L., Dessene, V., Hamadeh, A., Henry, D., Laieb, N., Lavallée, S., Lefebvre, J.M., Leitner, F., Menguy, Y., Padieu, F., Péria, O., Poyet, A., Promayon, M., Rouault, S., Sautot, P., Troccaz, J., Vassal, P. Computer Assisted Medical Interventions, IEEE Engineering in Medicine and Biology, pages 254-263, May/June 1995.

[Coad 95] Coad P., Object Models : Strategies, Patterns and Applications, Yourdon Press Computing Series, 1995.

[Conte 01] Conte A., Fredj M., Giraudin J., Rieu D., P-Sigma : un formalisme pour une représentation unifiée de patrons, Inforsid'01, Martigny, Juin 2001.

[Coutaz 87] Coutaz J., PAC: an Implementation Model for Dialog Design, Proceedings of Interact'87, H-J. Bullinger, B. Shackel ed., North Holland, Stuttgart, pages 431-436, September 1987.

D

[Dubois 99] Dubois, E., Nigay, L., Troccaz, J., Chavanon, O., Carrat, L.: Classification Space for Augmented Surgery, an Augmented Reality Case Study. Proceedings of Interact'99, pages 353-359,1999.

[Dubois 01] Dubois, E.: Chirurgie Augmentée : un Cas de Réalité Augmentée ; Conception et Réalisation Centrées sur l'Utilisateur. PhD Thesis University of Grenoble I, France pages 275, 2001.

F

[FOP 06] Site officiel, http://xmlgraphics.apache.org/fop

[Fowler 97] Fowler M. Analysis Patterns – Reusable Object Models, Addison-Wesley, 1997.

[Friedrich 02] Friedrich W. Arvika, augmented reality for development, production and service. International Symposium on Mixed and Augmented Reality (ISMAR), 2002.

G

[Gamma 95] Gamma E., Helm R., Johnson R., Vlissides J., Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.

[Grimson 96] Grimson W.E.L., Lozano-Perez T., Wells W.M., Ettinger G.J., White S.J., and Kikinis R. An automatic registration method for frameless stereotaxy, image guided surgery, and enhanced reality visualization. Transactions on Medical Imaging, 1996.

[Green 85] Seeheim 1985.

[Gzara 00] Gzara L., Les patterns pour l'ingénierie des Systèmes d'Information Produit, Thèse de Doctorat, spécialité Génie Industriel, Institut National Polytechnique de Grenoble, décembre 2000.

H

[HASS 05] Hassine, I., Spécification et formalisation des démarches de développement à base de composants métier: la démarche Symphony, Thèse de Doctorat, Institut National Polytechnique de Grenoble, décembre 2005.

[Hibernate 06] Site officiel, http://www.hibernate.org

I

[Ishii 97] Ishii H. and Ullmer B. Tangible bits : Towards seamless interfaces between people, bits and atoms. Computer Human Interaction (CHI), pages 234–241, March 1997.

[ISO 91] International Standard ISO/IEC 9126. Information technology –Software product evaluation- Quality characteristics and guidelines for their use. Geneva, Switzerland : International Organization for Standarization.

J

[JAAS 06] Site officiel, http://java.sun.com/products/jaas

[Jausseran 05] Jausseran E., Démarche Symphony étendue, formalisation et expérimentation sur un Système d'Information Hospitalier, Mémoire d'Ingénieur C.N.A.M. spécialité Informatique, Grenoble, Juillet 2005.

[JavaMail 06] Site officiel, http://java.sun.com/products/javamail

[Johnson 97] Johnson R., Frameworks = Patterns + Components, Communications of the ACM, Vol. 40, No.10, October 1997.

[Juras 06] Juras, D., Rieu, D., Dupuy-Chessa, S., Front A., Conception collaborative pour les Systèmes Mixtes. Congrès INFORSID'06, pages 33-48, May 2006 2006.

K

[Kaufmann 03] Hannes Kaufmann, Dieter Schmalstieg. Mathematics and geometry education with collaborative augmented reality. Computers and Graphics, 2003.

[Kato 99] Kato H., and Bilinghurst M. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. International Workshop on Augmented Reality, San Francisco, USA, October 1999.

[Kato 00] Kato H., Billinghurst M., Poupyrev I., Imamoto K., and Tachibana K.. Virtual object manipulation on a table-top ar environment. International Symposium on Augmented Reality (ISAR), 2000.

[Klinker 02] Klinker G., Dutoit A., Bauer M., and autres. A presentation system for product design. International Symposium on Aumgented and Mixed Reality (ISMAR), 2002.

[Klaus 95] Klaus H. Ahlers, Andr´e Kramer, David E. Breen, Pierre-Yves Chevalier, Chris Crampton, Eric Rose, Mihran Tuceryan, Ross T. Whitaker, and Douglas Greer. Distributed augmented reality for collaborative design applications. Computer Graphics Forum, pages 3–14, 1995.

[Krasner 88] Krasner, G., Pope, S., A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80, Journal of Object Oriented Programming, pages 26-49, 1988.

L

[Larvet 94] Larvet P., Analyse des systèmes : de l'approche fonctionnelle a l'approche objet, InterEditions, 1994.

[LiDO 06] Site officiel, http://www.xcalia.com

[Lucène 06] Site officiel, http://lucene.apache.org/java/docs/

M

[McCall 77] McCall J., Richards P., and Walters G., Factors in Software Quality, General Electric Company, RADC-TR77-369, Three volumes, Rome Air Development Center, November 1977.

[Milgram 93] Milgram P., Zhai S., and Grodski J. Applications of augmented reality for humanrobot communication. International Conference on Intelligent Robots and System (IROS), Yokohama, Japan, July 1993.

[Milgram 94] Milgram P., Kishino, F. A Taxonomy of Mixed Reality Visual Displays, Transactions on Information Systems, pages 1321-1329, 1994.

[Molineros 99] V. Raghavan Molineros J. and R. Sharma. Interactive evaluation of assembly sequences using augmented reality. IEEE Transactions on Robotics and Automation, 1999.

[Murakami 01] Murakami T., Contact water. SIGGRAPH, 2001.

N

[NetSize 06] Site officiel, http://www.netsize.com

[Nigay 94] Nigay L., Conception et modélisation logicielles des systèmes interactifs : application aux interfaces multimodales. PhD dissertation, University of Grenoble, France, 315 pages, 1994.

O

[Ohshima 98] Ohshima T., Satoh K., Yamamoto H., and Tamura H.. Ar2hockey : A case study of collaborative augmented reality. IEEE, pages 268–295, 1998.

P

[Perry 92] Perry D., Wolf A., Foundations for the Study of Software Architecture, ACM SIGSOFT Software Engineering Notes, pages 40-50, October 1992.

[Piekarski 02] Piekarski W. and Bruce T., ARQuake: The Outdoor Augmented Reality Gaming System, Communications of the ACM, vol. 45. No 1, pages 36-38, 2002.

R

[Renevier 01] Renevier P., Nigay L., Salembier P., Pasqualetti L. Systèmes mixtes mobiles et collaboratifs. Colloque sur la mobilité, Décembre 2002, LORIA, Nancy.

[Rekimoto 95] Rekimoto J. and Nagao K. The world through the computer : Computer augmented interaction with real world environments. Symposium on User Interface Software and Technology, pages 29–36, 1995.

S

[Schmidt 99] Schmidt D., Wrapper façade: A Structural Pattern for Encapsulating Functions within Classes, Report magazine, 1999.

[Schmidt 01] Schmidt D., Stal M., Rohnert H., Buschmann F., Pattern-Oriented Software Architecture, Volume 2: Patterns for Concurrent and Networked Objects, John Wiley & Sons, 2001.

[Schmidt 03] Schmidt D., Buschmann F. Patterns, Frameworks, and Middleware: Their Synergistic Relationships, Proceedings of the IEEE/ACM International Conference on Software Engineering, Portland, Oregon, May 2003.

[Struts 00] Site officiel, http://struts.apache.org

T

[Tastet 04] Tastet L. AGAP : un Atelier de Gestion et d'Applications de Patrons, Mémoire d'Ingénieur C.N.A.M, Spécialité Informatique, Grenoble, mars 2004.

[Taylor 92] Taylor, R., Paul, H., Cutting, C., Mittlestadt, B., Hanson, W., Kazanzides, P., Musits, B., Kim, Y., Kalvin, A., Haddad, B., Khoramabadi, D., Larose, D., "Augmentation of human precision in computer-integrated surgery", Innovation and Technology in Biology and Medicine, pages 450- 468, 1992.

[Thomas 99] H. Thomas, W. Piekarski, and B. Gunther. Using augmented reality to visualise architecture designs in an outdoor environment. International Journal of Design Computing Special Issue on Design Computing on the Net (DCNet), 1999.

[Thomas 00] Bruce Thomas, Ben Close, John Donoghue, John Squires, Phillip De Bondi, Michael Morris, and Wayne Piekarski. Arquake: An outdoor/indoor augmented reality first person application. pages 139–146, 2000.

U

[UIMS 92] The UIMS Tool Developers Workshop, A Metamodel for the Runtime Architecture of an Interactive System. SIGCHI Bulletin pages 32-37, 1992.

W

[Webster 96] Webster A., Feiner S., MacIntyre B., Massie W., and Krueger T. Augmented reality in architectural construction, inspection and renovation. Congress on Computing in Civil Engineering, pages 913–919, Anaheim, USA, June 1996.

[Wellner 91] Wellner P. The digitaldesk calculator : tangible manipulation on a desk top display. Symposium on User interface software and technology, pages 27–33. ACM Press, 1991.

[Wellner 93] Pierre Wellner. Interacting with paper on the digitaldesk. Communications of the ACM, pages 86–97, 1993.

Y

[Yvan 65] Yvan E. Sutherland. The ultimate display. In *IFIPS Congress*, volume 2, pages 506–508, New York, USA, May 1965.

[Yvan 68] Yvan E. Sutherland. A head-mounted three-dimensional display. In *AFIPS Conference*, volume 33, pages 757–764, 1968.

Chapitre VIII. Annexes

Annexe 1 : Termes du « Domaine de développement »

Termes de Technologie de développement :

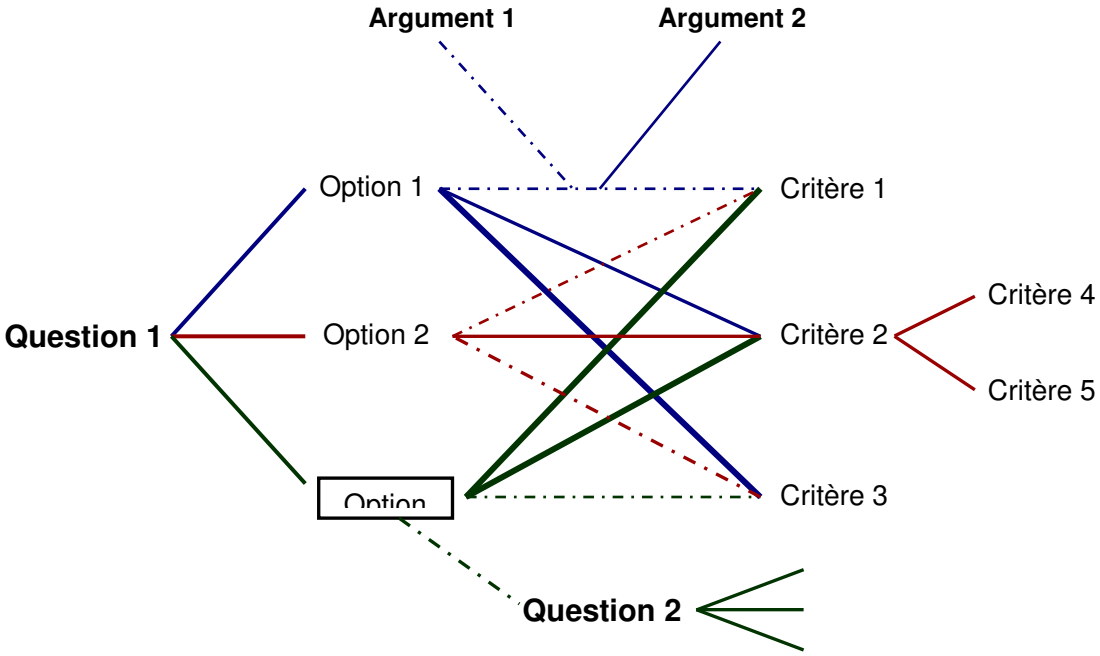
- Composant,
- Oriente Objet,
- Réutilisation.

Liste de termes du domaine :

- Activité,
- Analyse,
- Architecture,
- Architecture technique,
- Composant métier,
- Composant technique,
- Conception,
- Cycle de vie,
- Démarche,
- Déploiement,
- Étude préalable,
- Expression des besoins,
- Implantation,
- Intégration,
- Maquettage,
- Maquette,
- Modélisation métier,
- Objet métier,
- Objet technique,
- Phase,
- Processus,
- Processus métier,
- Produit,
- Recette,
- Spécification conceptuelle des besoins,
- Spécification organisationnelle des besoins.

Annexe 2 : Notation QOC

QOC (*Questions, Options, Critères*), est une notation semi formelle proposée par MacLean [MacLean 91]. Cette notation donne lieu à une représentation sous forme de diagramme, comme montré sur la figure suivant :



Ce diagramme peut se décomposer en trois colonnes, une par élément de la notation (questions, options, critères), et des liens entre les éléments de ces colonnes. A chaque question on associe plusieurs options (par exemple, choix de conception), et à ces différentes options on associe des critères qui favorisent (barre épaisse) ou défavorisent (barre en pointillé) les options. L'option retenue (l'option 3) est encadrée. Dans QOC, une option peut déboucher sur une nouvelle question (question 2), si bien qu'il apparaît des liens explicitent entre différents diagrammes QOC. En outre des arguments peuvent être rajoutés pour supporter ou non l'évaluation portée par les liens entre les options et les critères).

Annexe 3 : Termes et concepts

A

Activité : Tâche dans la description de l'élaboration d'un produit.

Agent : Un agent est une unité informatique spécialisée qui a un état et est capable d'introduction et la réaction aux événements.

Analyse : Activité consistant à mettre en évidence ce que doit faire un système sans tenir compte de la manière de le réaliser.

Application : Une application est l'union cohérente de tout l'ensemble d'instructions reconnues et exécutées au moyen d'un langage de programmation qui permet l'interaction entre l'usager et l'ordinateur. Cette communication est réalisée quand l'usager choisit entre les différentes fonctionnalités offertes par l'application.

Architecture : Structure globale, composantes logiques, et relations logiques du système. L'architecture logicielle concerne l'abstraction, la composition, la décomposition, le style et l'esthétique. Elle traite de la conception technique et de l'implémentation de structures logicielles de haut niveau.

Architecture technique : Description de l'environnement de production, de l'architecture réseau et matérielle et des contraintes d'exploitation (volumétrie, ordonnancement des batchs).

C

Composant : Une entité autonome spécifiquement conçue, packagée et documentée de façon à être réutilisable. Un composant dispose généralement d'une (ou plusieurs) interface(s) publique(s) stable(s).

Composant métier : Permet de matérialiser les connaissances d'un domaine ou d'un métier et de les rendre ainsi réutilisables dans l'ingénierie des systèmes d'information.

Cycle de vie : Union de toutes les étapes de la conception d'un logiciel.

D

Démarche: Une démarche est le processus grâce auquel s'effectue le travail de modélisation. Ce processus est décomposable en processus élémentaires et chaque processus possède des modèles / produits en entrée et des modèles / produit au résultat.

F

Framework : Un framework est défini comme un ensemble de classes qui coopèrent et permettent des conceptions réutilisables dans des catégories spécifiques de logiciels.

I

Infrastructure : Ensemble bien définies, fiables et publiquement accessibles de technologies qui agissent pour d'autres systèmes.

Interface : Un type utilisé pour décrire le comportement externe visible d'une classe, d'un objet ou d'une autre entité. Dans le cas d'une classe ou d'un objet, l'interface comporte les signatures des opérations (méthodes). Une même entité peut comporter plusieurs interfaces, en particulier dans le cas des composants.

L

Librairie : Ensemble de programmes relatifs à un sujet (orienté réutilisation de code).

M

Maquette : Ensemble logiciel ou papier qui a pour vocation de démontrer la faisabilité d'une application, d'en matérialiser certaines caractéristiques ou en illustrer certaines fonctionnalités d'un point de vue externe (celui de l'utilisateur final).

O

Objet métier : Structure modulaire spécifique à un système d'information donné. Il est une représentation d'un concept du monde réel relatif à un domaine ou à un métier spécifique.

OpenGL : (Open Graphics Library) est une spécification qui définit une API multi plate-forme pour la conception d'applications générant des images 3D (mais également 2D).

P

Processus métier : Séquence d'activités internes du domaine d'étude dont l'objectif est de fournir un résultat observable et mesurable pour un utilisateur individuel du métier.

R

Réalité augmentée : Caractérise tout système qui améliore la perception de l'opérateur vis à vis de l'environnement réel, généralement par superposition d'images de synthèse sur des images réelles ou vidéo.

Réalité virtuelle : Désigne tout système qui procure à l'opérateur humain la sensation d'immersion et la capacité d'interaction face à un environnement virtuel, c'est-à-dire basé sur un modèle de synthèse entièrement généré par ordinateur.

Ressource métier : Une entité produite ou utilisée par un ou plusieurs processus métier.

S

STRUTS : Es un cadre d'application Open Source pour développer des applications Web J2EE. Il utilise et étend l'API Servlet Java afin d'en d'encourager les développeurs à adopter l'architecture Modèle-Vue-Contrôleur version 2.

T

Tangible User Interfaces : Augmente le monde physique, intégrant des informations numériques avec des objets physiques quotidiens, Généralement, l'entrée physique contrôle la production graphique ou audio.

Toolkit(Boîte à outils) : Collection de composants pour des fonctionnalités communes.

V

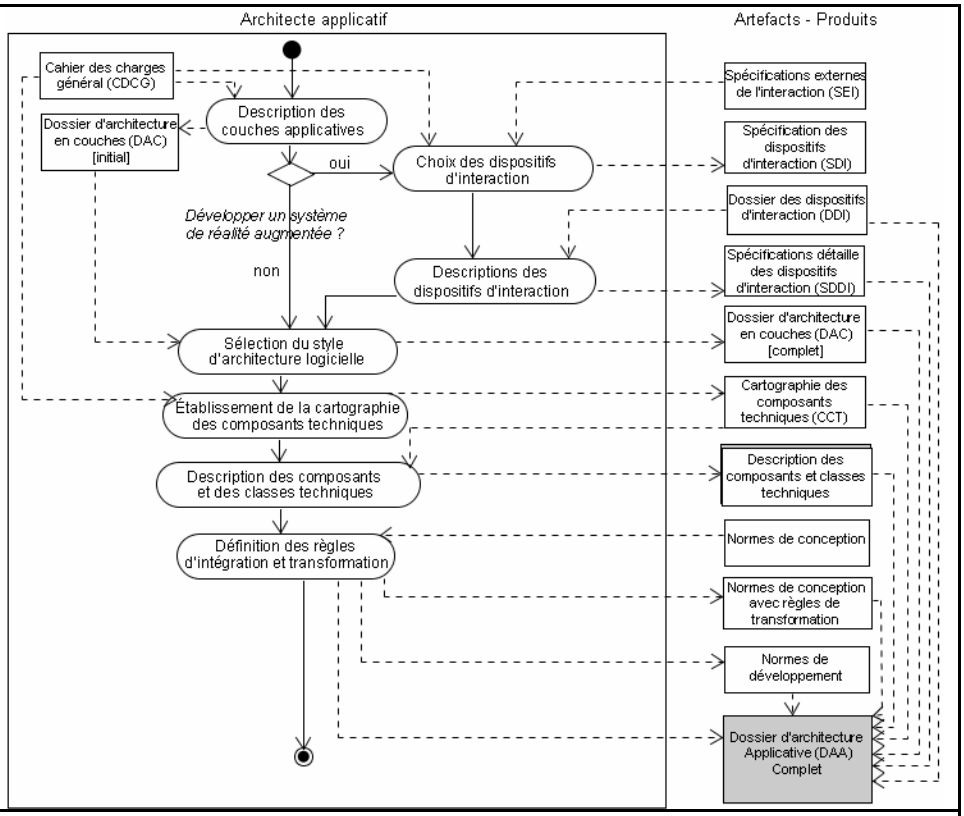
Virtualité augmentée : Consiste en enrichir l'interaction d'un utilisateur avec le monde numérique au moyen d'outils et d'actions du monde physique

Annexe 4 : Système de Patrons pour la Branche Technique en Systemes de Realite Augmentee

1. Patrons Processus

1.1. Patron d'architecture applicative

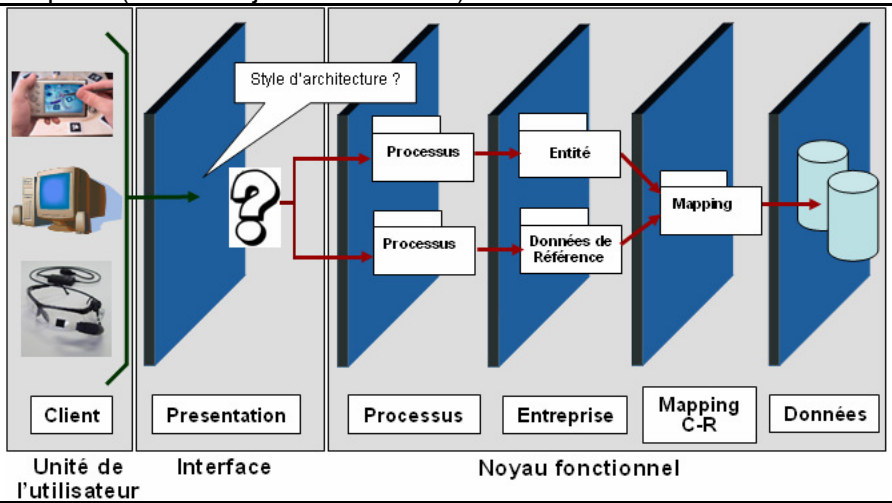
Patron : Phase d'architecture applicative	
Auteur (s)	J.Pérez
Domaine	Domaine de développement
Technologie	Technologie de développement
P-System	Système de Patrons pour les systèmes de Réalité Augmentée
Formalisme	P-SIGMA + étendu pour les systèmes de Réalité Augmentée
Identifiant	
Phase d'architecture applicative	
Participant (s)	
• « Architecte applicatif »	
Classification	
Terme (s) de domaine	{Phase ^ Processus ^ Architecture applicative ^ Composant technique ^ Conception}
Contexte	
Artefact(s) : { [REQ] Cahier Des Charges Général (CDCG) }	
Problème	
L'objectif de cette phase est de spécifier et d'analyser les composants techniques et ses supports logiciels pour un système de réalité augmentée à partir du cahier des charges général.	
Force (s)	
Force (s)	Dossier d'architecture applicative (DAA) : • Description des buts métier dans un contexte architectural, basés en couches applicatives, • Choix et description des dispositifs d'interaction, • Établissement de la cartographie des composants techniques, • Définition de règles de développement, intégration et transformation des composants techniques.
Technologie (s)	{ Réutilisation ^ Composants ^ Orienté objet }
Solution démarche	

Démarche formelle	
Démarche textuelle	Cette phase est réalisée en parallèle avec les exigences fonctionnelles, son but est de spécifier et de décrire sous une forme détaillée les composantes techniques à partir de la distribution des couches applicatives et de la sélection d'une architecture de logiciel. Cette phase décrit également les règles d'intégration et de transformation des composants techniques détaillés. Ce patron s'appuie sur les activités suivantes : • **Description des couches applicatives** : consiste en la décomposition et distribution des buts métier en plusieurs composants. • **Choix des dispositifs d'interaction** : consiste en la sélection des dispositifs à partir de cahier de charges général et l'étude de l'activité réelle de l'utilisateur. • **Description des dispositifs d'interaction** : consiste en détailler des spécifications techniques de chacun des dispositifs d'interaction qui a été choisi. • **Sélection du style d'architecture logicielle** : consiste à appliquer une architecture logicielle en fonction de la distribution de couches applicatives. • **Établissement de la cartographie des composants techniques** : consiste en la sélection des composantes et frameworks techniques sur la base des spécifications des composantes de l'activité antérieure. • **Description des composants et des classes techniques** : consiste à détailler les composants et les frameworks techniques, leurs interfaces et leurs classes spécialisables. • **Définition des règles d'intégration et transformation** : consiste à spécifier et / ou identifier les règles de conception et de transformation applicables dans la conception de l'application (par ex.

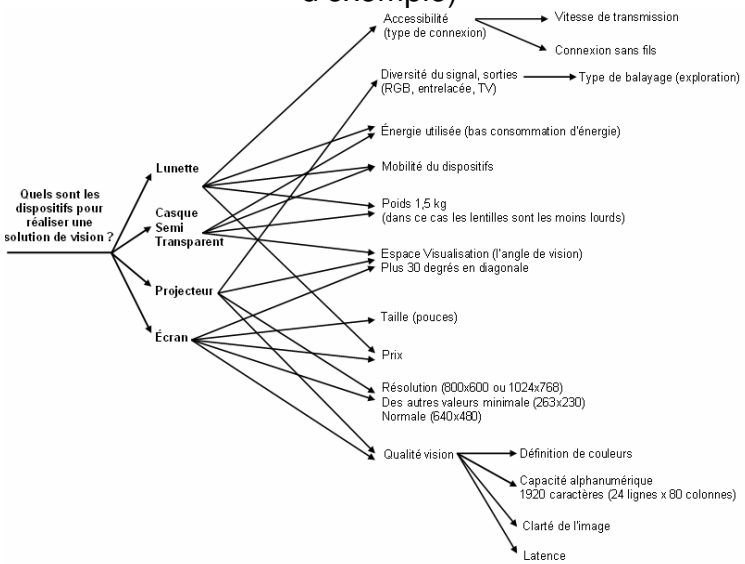
	la transformation du modèle d'analyse en modèle de conception).
Document (s)	{ Dossier d'architecture Applicative (DAA) }
Utilise	
{ Description des couches applicatives, Choix des dispositifs d'interaction, Description des dispositifs d'interaction. Sélection du style d'architecture logicielle, Établissement de la cartographie des composants techniques, Description des composants et des classes techniques, Définition des règles d'intégration et transformation}	
Requiert	
{ Phase d'étude préalable }	

1.2. Patron de description des couches applicatives

Patron : Description des couches applicatives	
Auteur (s)	J.Pérez
Domaine	Domaine de développement
Technologie	Technologie de développement
P-System	Système de Patrons pour les systèmes de Réalité Augmentée
Formalisme	P-SIGMA + étendu pour les systèmes de Réalité Augmentée
Identifiant	
Description des couches applicatives	
Participant (s)	
• « Architecte applicative »	
Classification	
Terme (s) de domaine	{Activité ^ Processus ^ Description des couches applicatives ^ Conception architecturale ^ Composant technique ^ Conception}
Contexte	
Artefact (s) : { [REQ] Cahier Des Charges Général (CDCG) }	
Problème	
On veut faire la décomposition et distribution des buts métier en plusieurs composants à partir du cahier des charges général (CDCG).	
Force (s)	
Force (s)	• Description de charge de travail au moyen des couches.
Technologie (s)	{ Réutilisation ^ Composants ^ Orienté objet }
Solution démarche	
Démarche textuelle	Cette activité définit un schéma d'organisation pour les systèmes de réalité augmentée en termes de composants. L'organisation fixe les directives générales de développement, et les structures qu'elle induit aident au maintien de l'intégrité du modèle. Pour garantir des tels buts, elle se base sur l'application de patrons de conception Façade, pour spécifier l'ensemble des couches applicatives de l'architecture logicielle. Une architecture multicouches permet la compréhension, gestion, réutilisation, facilite l'implémentation, la portabilité et la maintenance du logiciel. Chaque couche peut être modélisée, développe et testée individuellement. Cette notion correspond au patron de conception et d'architecture « Layers » [BUS 96]. La distribution en couches logicielles peut constituer des modèles de spécification à réutiliser par différentes applications de l'entreprise. Elles structurent la création de frameworks techniques qui constituent des

	mécanismes de conception pour la technologie concernée. Cependant, dans le cas des systèmes de RA, on doit choisir un style d'architecture.
Document (s)	{ Dossier d'architecture en couches [initial] (DAC) }
Cas d'application	
Description cas	La distribution en couches logicielles se basent sur l'application du principe d'un système interactif et du patron « Layer » [Bus 96], donc un système de réalité augmentée peut être structuré dans les points de vue : utilisateur, interface et noyau fonctionnel, au moyen des couches suivantes : **Couche client :** est relative aux composants (dispositifs d'interaction par exemple) qui permettant aux utilisateurs interagir avec l'application. **Couche de Présentation** : Cette couche permet de gérer la présentation, l'interaction des objets du monde réel et virtuel avec les couches inférieurs qui correspond aux systèmes classiques. **Couche processus** : cette couche est réalisée par un ensemble de classes indépendantes qui encapsulant la logique applicative des systèmes classiques qui se communiquent avec la couche de présentation. La communication doit être réalisée entre le sous-système d'application et les processus applicatives. **Couche entreprise** : représente l'ensemble des objets métier tels que les « filtres organisationnels » (représentation des acteurs du système), des objets « produit », les données de référence « Référentiel ». Cette couche comprend les entités qui seront souvent persistantes, donc « mappées » avec la base de données. **Couche « Mapping » objet relationnel** : permet de transformer des objets métier entité et données de référence en requêtes SQL et peut être prise en charge par un framework (Open Source, commercial ou « maison »). Il est souhaitable dans ce cas de respecter au mieux les standards (exemple : JDO pour les applications J2EE). **Couche données** : cette couche utilise les SGBDr et les technologies de bases de données pour sérialiser les objets métier de la couche Entreprise (via les objets relationnels).
Représentation de cas	
Utilise	
{Couches (layer), Façade }	

1.3. Patron Choix des dispositifs d'interaction

Patron : Choix des dispositifs d'interaction	
Auteur (s)	J.Pérez
Domaine	Domaine de développement
Technologie	Technologie de développement
P-System	Système de Patrons pour les systèmes de Réalité Augmentée
Formalisme	P-SIGMA + étendu pour les systèmes de Réalité Augmentée
Identifiant	
Choix des dispositifs d'interaction	
Participant (s)	
• « Architecte applicative, concepteur d'architecture logicielle »	
Classification	
Terme (s) de domaine	{Activité ^ Processus ^ Architecture logicielle ^ Conception}
Contexte	
Artefact(s) : { [REQ] Spécifications externes de l'interaction (SEI), [REQ] Dossier de spécifications des dispositifs d'interaction (SDI) }	
Problème	
On veut choisir les dispositifs d'interaction à partir des spécifications externes d'interaction (SEI).	
Force (s)	
Force (s)	• Dossier des dispositifs d'interaction (DDI).
Technologie (s)	{ Réutilisation ^ Composants ^ Orienté objet }
Solution démarche	
Démarche formelle	Critères de sélection pour les dispositifs de vision (a titre d'exemple) 
Démarche textuelle	La sélection d'un dispositif d'interaction s'appuie sur l'usage de la notation QOC (Questions, Options, Critères). QOC est une notation semi formelle proposée par MacLean [MacLean 91]. Par rapport à la sélection des dispositifs de vision on doit considère les critères de choix suivants : • Accessibilité, • Diversité du signal,

	• Energie utilisée, • Mobilité du dispositif, • Poids, • Espace de visualisation, • Taille, • Prix, • Résolution, • Qualité de la vision.
Document (s)	{ Dossier des dispositifs d'interaction (DDI) }
Cas d'application	
Description cas	Dans cadre de l'EDL, nous avons choisi un **casque semi transparent**, ce type de dispositif est de consommation d'énergie bas, garantit la mobilité de l'utilisateur et offre un ample espace de visualisation
Représentation de cas	
Requiert	
{Phase d'architecture applicative, }	

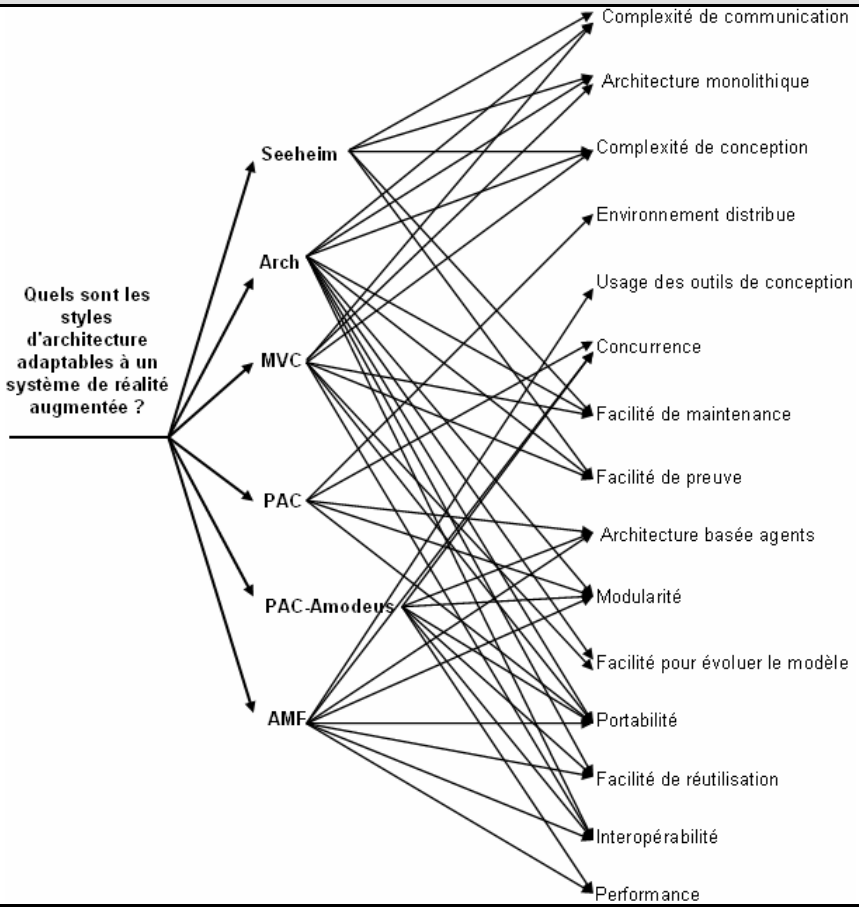
1.4. Patron Description des dispositifs d'interaction

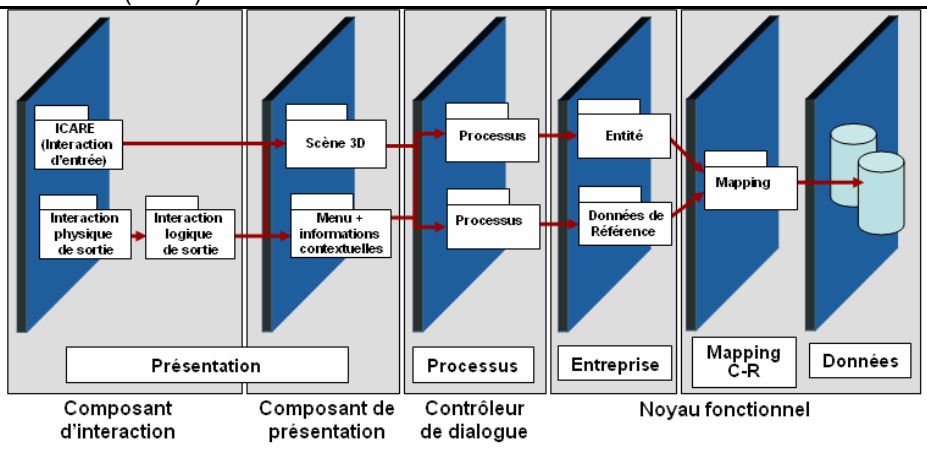
Patron : Description des dispositifs d'interaction	
Auteur (s)	J.Pérez
Domaine	Domaine de développement
Technologie	Technologie de développement
P-System	Système de Patrons pour les systèmes de Réalité Augmentée
Formalisme	P-SIGMA + étendu pour les systèmes de Réalité Augmentée
Identifiant	
Description des dispositifs d'interaction	
Participant (s)	
• « Architecte applicative, concepteur d'architecture logicielle »	
Classification	
Terme (s) de domaine	{Activité ^ Processus ^ Architecture logicielle ^ Conception}
Contexte	
Artefact(s) : { [REQ] Spécification des dispositifs d'interaction (SDI) }	
Problème	
On doit détailler les spécifications techniques de chacun des dispositifs d'interaction qui a été choisi dans l'activité précédente.	
Force (s)	
Force (s)	• Dossier de spécifications détaille des dispositifs d'interaction (SDDI).
Technologie (s)	{ Réutilisation ^ Composants ^ Orienté objet }
Solution démarche	
Démarche textuelle	Un modèle d'architecture définit un guide pour l'organisation du logiciel et permet de décomposer le système en sous-systèmes. Les spécifications des dispositifs d'interaction permet de donner touts les caractéristiques techniques des dispositifs choisi. Ces spécifications peuvent se rendre au moyen d'un tableau.

<table>
<tr><td>Document (s)</td><td colspan="3">{ Dossier d’architecture en couches [complet] (DAC) }</td></tr>
<tr><td colspan="4">Cas d’application</td></tr>
<tr><td>Description cas</td><td colspan="3">Dans le cas d’un EDL, les spécifications des dispositifs d'interaction choisis se présentent dans le tableau suivant :
<table>
<tr><th></th><th>Dispositif</th><th>Spécifications techniques</th></tr>
<tr><td rowspan="8">Unité mobile</td><td>PC Portable</td><td>Modèle SONY VAIO PCG-F809K, processeur : Pentium III - 850MHz, RAM : 128Mo, Système d'exploitation : Windows XP, Disque dur : 30Go, rotation à 4200 tours/min, 2 ports PCMCIA : 1er port : carte Ethernet 3COM 10/100Mbps, 2nd port : carte WiFi U.S. Robotics 802.11b - 11 Mbps. Entrée audio (microphone), sortie VGA, 2 ports USB, 1 port COM/Serial, dimensions : 32.4 x 26.6 x 5.4 cm, Poids : 3.2 Kg.</td></tr>
<tr><td>Casque semi-transparent (HMD)</td><td>Modèle SONY PC Glasstron PLM-S700E (écran à cristaux liquides 1.5 mégapixels à transparence réglable, résolution maximale : 800 x 600 à 85Hz, Entrée VGA classique, RGB et S-Video pour assurer le transfert des données vidéo. Oreillettes pour des notifications sonores et/ou des messages vocaux à l’utilisateur).</td></tr>
<tr><td>Capteur d'orientation</td><td>Modèle InterSense InterTrax (Capte les 3 angles de liberté (X, Y et Z), Intervalles : X : +/- 80°, Y : +/- 180°, Z : +/- 90°, Latence : 4ms, Surface à scratch pour la fixation, Ports : USB, COM/Serial, Poids : 39 grammes).</td></tr>
<tr><td>Capteur de localisation</td><td>En vue des coûts élevés d’un capteur de localisation l’on peut facilement le remplacer par un « Magicien d'Oz ».</td></tr>
<tr><td>Microphone</td><td>Microphone générique avec deux sorties d'audio. Il est fixé au casque et relié à l'ordinateur portable, assure la capture de la voix de l'utilisateur et rend ainsi possible le contrôle de l'application par commandes vocales.</td></tr>
<tr><td>PDA</td><td>Modèle Dell AXIM X30, utilisant Windows Mobile 2003. Il est doté du Standard Wi-Fi (802.11b) et d’un écran tactile avec stylet de résolution : 240 x 320 pixels en 16 bits. C'est cette dernière caractéristique et sa petite taille qui justifient le choix d'un tel dispositif. : les étapes demandant de positionner et/ou de sélectionner des objets (des dommages) demandent une précision que la commande vocale ne peut assurer. Le PDA transmet à l'ordinateur portable les déplacements réalisés par l'utilisateur.</td></tr>
<tr><td>Webcam</td><td>Modèle Philips ToUcam PRO PCVC740K ou similaire (Capteur CCD 1.2 mégapixels, enregistrement jusqu'à 60 images/seconde, résolution maximale en capture vidéo : 640 x 480 et en capture photo : 1280 x 960 en 24bits).</td></tr>
<tr><td>Poste fixe</td><td>Ordinateur fixe</td><td>Ses spécifications contemplent la technologie Wi-Fi (802.11b) 11 Mbps. Ce dispositif supportera la plupart de l'application et sera relié au PDA par liaison Wi-Fi.</td></tr>
</table></td></tr>
<tr><td>Représentation de cas</td><td colspan="3"></td></tr>
<tr><td colspan="4">Requiert</td></tr>
<tr><td colspan="4">{Phase d’architecture applicative }</td></tr>
</table>

1.5. Patron Sélection du style d’architecture logicielle

Patron : Sélection du style d’architecture logicielle	
Auteur (s)	J.Pérez
Domaine	Domaine de développement
Technologie	Technologie de développement
P-System	Système de Patrons pour les systèmes de Réalité Augmentée
Formalisme	P-SIGMA + étendu pour les systèmes de Réalité Augmentée
Identifiant	
Sélection du style d’architecture logicielle	

Participant (s)	
• « Architecte applicative, concepteur d'architecture logicielle »	
Classification	
Terme (s) de domaine	{Activité ^ Processus ^ Architecture logicielle ^ Conception}
Contexte	
Artefact(s) : { [REQ] Dossier d'architecture en couches [initial] (DCA) }	
Problème	
On veut choisir l'architecture logicielle à partir du dossier d'architecture en couches pinitial] (DAC).	
Force (s)	
Force (s)	• Dossier d'architecture en couches [complet] (DAC) qui permettra de définir l'organisation du logiciel.
Technologie (s)	{ Réutilisation ^ Composants ^ Orienté objet }
Solution démarche	
Démarche formelle	
Démarche textuelle	Un modèle d'architecture définit un guide pour l'organisation du logiciel et permet de décomposer le système en sous-systèmes. La sélection d'une architecture logicielle s'appuie sur l'usage de la notation QOC (Questions, Options, Critères). QOC est une notation semi formelle proposée par MacLean [MacLean 91]. Par rapport à la sélection des styles d'architecture on doit considère les critères de choix suivants : • Complexité de communication, • Type d'architecture (monolithique, basée agents),

	• Complexité de conception, • Environnement distribue, • Usage des outils de conception, • Concurrence, • Facilité de maintenance, • Modularité, • Facilite pour évoluer le modèle, • Portabilité, • Facilite de réutilisation, • Interopérabilité, • Performances.
Document (s)	{ Dossier d'architecture en couches [complet] (DAC) }
Cas d'application	
Description cas	Dans le cas d'un EDL, nous avons décidé d'adopter le style d'architecture ARCH conçu pour les applications interactives. Nous avons choisi Arch parce qu'il est une architecture avec bas complexité de conception et de communications des composants. Aussi que, Il est de maintenance et preuve facile. Le découpage en 5 niveaux de cette architecture permet la modularité de l'application et par conséquence la réutilisation des composants. La sélection d'un ensemble d'architectures se résume dans le Dossier d'Architecture en Couches (DAC).
Représentation de cas	
Utilise	
{Seeheim, Modèle Vue Contrôleur 2 (MVC 2), ARCH, PAC, PAC-Amodeus, AMF}	

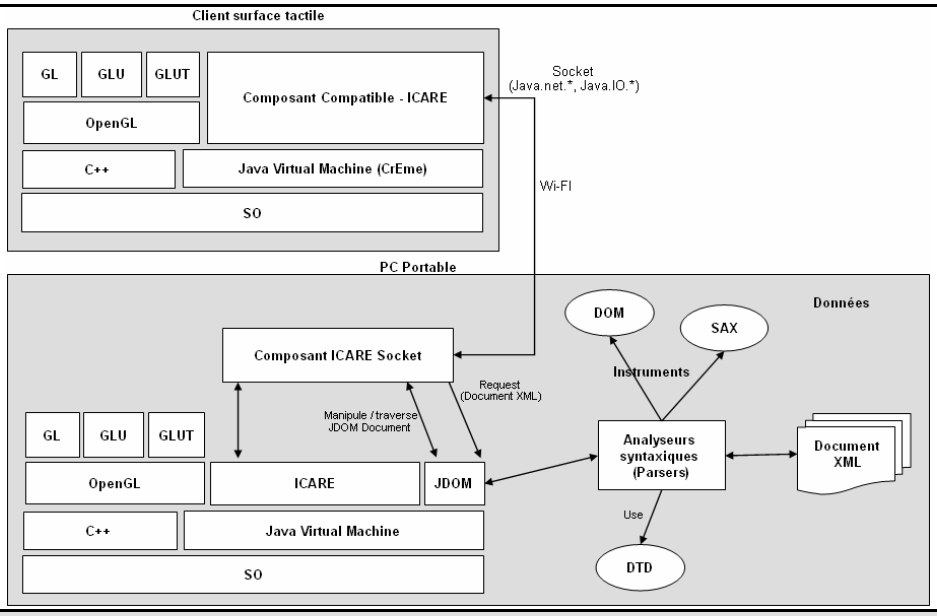
1.6. Patron Etablissement de la cartographie des composants techniques

Patron : Établissement de la cartographie des composants techniques	
Auteur (s)	J.Pérez
Domaine	Domaine de développement
Technologie	Technologie de développement
P-System	Système de Patrons pour les systèmes de Réalité Augmentée
Formalisme	P-SIGMA + étendu pour les systèmes de Réalité Augmentée
Identifiant	
Établissement de la cartographie des composants techniques	
Participant (s)	
• « Architecte applicative »	

<table>
<tr><th colspan="2">Classification</th></tr>
<tr><td>Terme (s) de domaine</td><td>{Activité ^ Processus ^ Support logiciel ^ Composant technique ^ Conception}</td></tr>
<tr><th colspan="2">Contexte</th></tr>
<tr><td colspan="2">Artefact (s) : { [REQ] Cahier Des Charges Général (CDCG), [REQ] Dossier d’architecture en couches (DAC) }</td></tr>
<tr><th colspan="2">Problème</th></tr>
<tr><td colspan="2">On doit définir les technologies de développement de systèmes de réalité augmentée à mettre en place dans l’application et décrire l’ensemble des interactions.</td></tr>
<tr><th colspan="2">Force (s)</th></tr>
<tr><td>Force (s)</td><td>• Cartographie de Composants Techniques (CCT).</td></tr>
<tr><td>Technologie (s)</td><td>{ Réutilisation ^ Composants ^ Orienté objet }</td></tr>
<tr><th colspan="2">Solution démarche</th></tr>
<tr><td>Démarche textuelle</td><td>Cette activité constitue le coeur de la phase d'architecture applicative car l'architecture doit définir les technologies à mettre en place compte tenu des besoins fonctionnels.</td></tr>
<tr><td>Document (s)</td><td>{ Cartographie de composants techniques (CCT) }</td></tr>
<tr><th colspan="2">Cas d’application</th></tr>
<tr><td>Description cas</td><td>Dans le cadre d’un EDL la cartographie des composants techniques peut s'appuyer sur différentes catégories de composants techniques et frameworks :

• frameworks oriente au composants : ARToolkit, AMIRE, ARVIKA, DWARF, DART, ICARE, Papier-Mâché, OpenTracker, VHD++, GLUT, OpenGL,
• frameworks oriente objets : Gasp, Coterie, Tinmith-evo5, Studierstube, MARE, BEACH,
• composants : Wcomp, MR Platform, ImageTclAR,
• middleware : DART,
• toolkits : ARToolkit, MR Platform, ImageTclAR,
• Outils d’authoring : AMIRE, DART.
• Composant pour le stockage des données : JDOM.

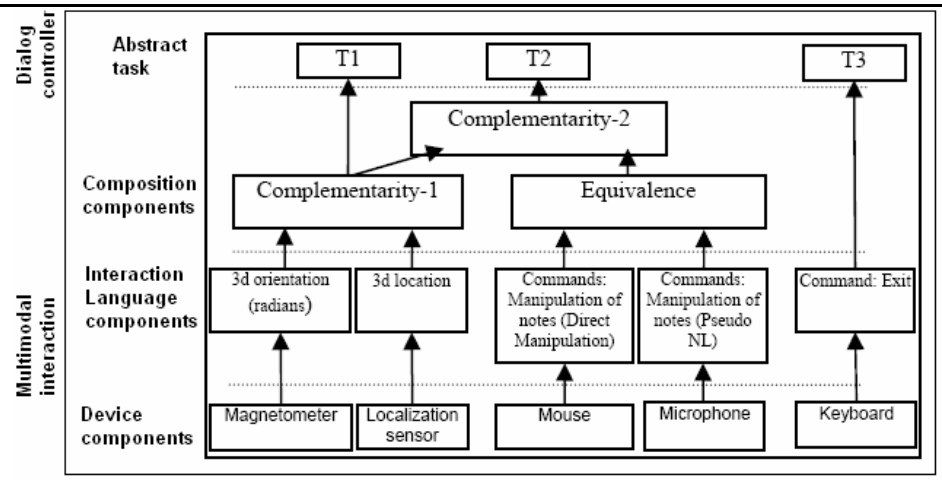
Une cartographie de composants techniques est souvent décrite par des schémas spécifiques du fait d'un manque dans la notation des diagrammes de composants UML.
Le schéma suivant illustre une cartographie type de composants techniques:</td></tr>
</table>

Représentation de cas	
Requiert	
{ Phase d'architecture applicative }	

1.7. Patron Description des composants et des classes techniques

Patron : Description des composants et des classes techniques	
Auteur (s)	J.Pérez
Domaine	Domaine de développement
Technologie	Technologie de développement
P-System	Système de Patrons pour les systèmes de Réalité Augmentée
Formalisme	P-SIGMA + étendu pour les systèmes de Réalité Augmentée
Identifiant	
Description des composants et des classes techniques	
Participant (s)	
• « Architecte applicative »	
Classification	
Terme (s) de domaine	{Activité ^ Processus ^ Support logiciel ^ Composant technique ^ Objet technique ^ Conception}
Contexte	
Artefact (s) : { [REQ] Cartographie des composants techniques (CCT) }	
Problème	
Décrire les composants techniques. Détailler en particulier leurs interfaces, les classes spécialisables, leur fonctionnement interne si nécessaire.	
Force (s)	
Force (s)	• Description des Composants et Classes Techniques (intégrées dans le DAA).
Technologie (s)	{ Réutilisation ^ Composants techniques ^ Orienté objet }
Solution démarche	
Démarche textuelle	L'activité consiste à : • Décrire le contenu du composant, l'interface du composant, les classes que le concepteur peut redéfinir, le fonctionnement interne du composant ou framework. • Décrire la spécialisation éventuelle du composant. Certains

	composants nécessitent un enrobage pour faciliter leur utilisation dans le cadre des développements. Cela est particulièrement d'éviter la redondance de code. La conception de ces classes d'enrobage se fera dans la phase de conception. • Déterminer l'utilisation du composant. L'architecte doit définir quel utilisation il doit faire du composant (d'origine ou spécialisé) dans le cadre du projet. Cette description peut se faire par des exemples concrets d'utilisation. Dans le cas d'un composant complexe ou d'un framework, une documentation doit être annexée au dossier d'architecture applicative (exemple : Description de framework des composants ICARE).
Document (s)	{ Description des composants et classes techniques (DCCT) }
Cas d'application	
Description cas	ICARE (Interaction Complémentarité Assignation Redondance Equivalence) aide à la construction d'applications interactives multimodales en proposant des composants réutilisables. Il utilise une approche à composants (JavaBeans) qui facilite la mise en œuvre de la combinaison de modalités. Les composants d'ICARE sont indépendants des modalités et peuvent donc être réutilise dans différentes applications. Ils remplissent les fonctions d'interaction physique et logique en entrée/sortie en fournissant au système via le contrôleur de dialogue une expression symbolique de la commande. Dans le cas des systèmes de réalité augmentée ce composant permet de utiliser diverses types d'interaction, telles comme : Commandes vocales, binaire (manipulation directe), orientation et localisation. Cela, permet de dire que dans las applications basées ICARE la perception de l'utilisateur est augmentée. On distingue deux types de composants ICARE : les composants élémentaires et des composition. **Composant élémentaires** : Les composant élémentaires sont l'association des composants Dispositifs et Systèmes Représentationnels qui permet la multi modalité. Le composant Utilisateur représente les propriétés de l'utilisateur qui peuvent concerner l'identité, les coordonnées ainsi que les données biométriques et sociales qui le caractérisent [Bouchet 04]. Le composant Environnement permet d'enregistrer des données sur l'environnement physique du contexte d'utilisation (température, niveau sonore, etc.). **Composants de composition** : On distingue trois sortes de composants de composition : Le composant Complémentarité traduit la propriété ergonomique de Complémentarité et représente la composition de plusieurs dispositifs physiques (niveau physique) ou de plusieurs systèmes représentationnels (niveau logique). Le composant Redondance traduit la propriété ergonomique de redondance, c'est-à-dire la possibilité d'utiliser simultanément deux modalités différentes pour interagir avec le système. Le composant Equivalence/Redondance traduit les propriétés ergonomiques d'équivalence et de redondance, c'est-à-dire la possibilité d'utiliser deux modalités différentes, de manière simultanée ou non, pour interagir avec le système. Finalement, les opérations de fusion de deux modalités se font au niveau de ces trois composants.

Représentation de cas	

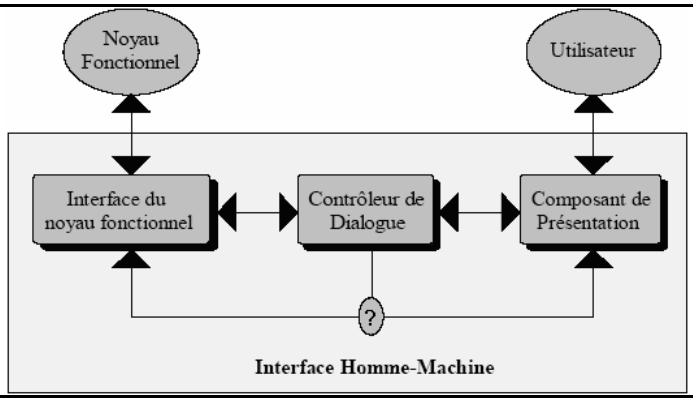
Requiert
{ Phase d'architecture applicative, Etablissement de la cartographie des composants techniques }

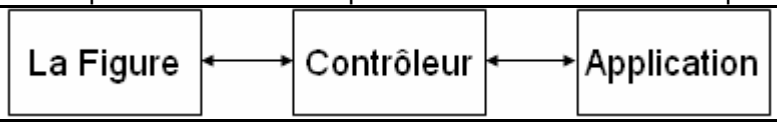
2. Patrons Produit

2.1. Patrons d'architecture

2.1.1. Patrons d'architecture Seeheim

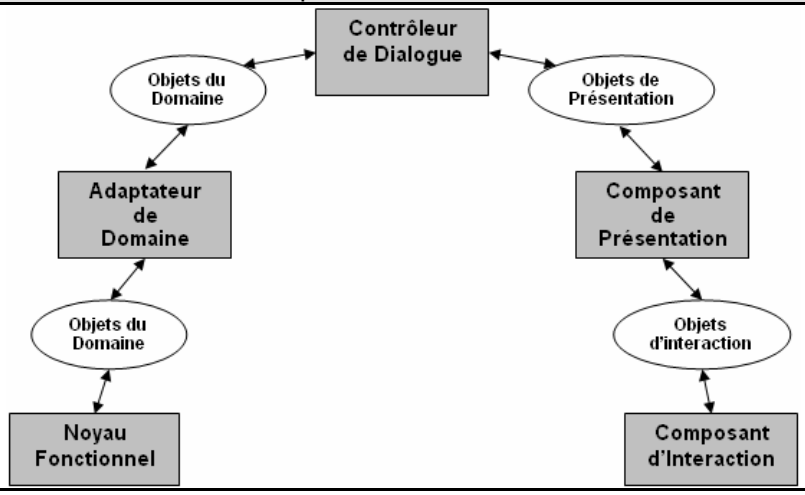
Patron : Seeheim	
Auteur (s)	J.Pérez
Domaine	Domaine de développement
Technologie	Technologie de développement
P-System	Système de Patrons pour les systèmes de Réalité Augmentée
Formalisme	P-SIGMA + étendu pour les systèmes de Réalité Augmentée
Identifiant	
Seeheim	
Participant (s)	
• « Architecte applicative »	
Classification	
Terme (s) de domaine	{Activité ^ Produit ^ Seeheim ^ Architecture logicielle ^ Conception ^ Réutilisation}
Contexte	
Contexte formel	Phase Conception Architecturel
Contexte textuel	Application Interactive a développer avec une séparation entre l'interface de l'utilisateur et l'application.
Problème	
Seeheim résout la construction d'applications interactives au moyen de la séparation rigoureuse entre l'interface de l'utilisateur et l'application.	
Force (s)	
Force (s)	• Considère qu'un système interactif est constitue de deux acteurs : l'application et l'usager. • Les problèmes de l'application et de 'l'interface peuvent être isolés et traite séparément. • Propose un processus de conception descendant et en séparant clairement les niveaux d'abstraction de l'application. • Cette architecture est adapte aux interfaces à base de menus, d'écrans de saisie ou pour des applications à interaction graphique. • L'interface peut être partagée par différentes applications.
Technologie (s)	{ Réutilisation ^ Composants ^ Orienté objet }
Solution produit	

Diagramme de classes	
Solution textuelle	Seeheim traite ce problème en séparant le code en trois domaines distincts : • **Le composant de présentation**, c'est la partie statique et visible de l'interface qui communique avec l'utilisateur et il est construit sur des systèmes de fenêtres et des boîtes d'outils. • **Le contrôleur de dialogue**, c'est la partie dynamique qui manie les évènements ou les messages qui se produisent comme conséquence des actions de l'utilisateur de l'interface et établit la communication entre le niveau de présentation et le niveau d'application. • **Le noyau fonctionnel (NF)**, c'est la partie de l'application que l'utilisateur contrôle à travers de l'interface. Cette partie contient en général une description des entités du domaine qui sont directement liées à l'interface ainsi qu'une description des procédures du noyau fonctionnel qui peuvent être invoquées par le contrôleur de dialogue. Une première classe de relations désigne la correspondance entre les concepts du domaine et les interacteurs constitutifs de la Présentation. *Un interacteur* est un instrument logiciel, c'est-à-dire une entité capable de restituer une abstraction auprès de l'utilisateur sur un dispositif de sortie (écran, haut-parleur, etc.) et de lui permettre de manipuler cette abstraction via les dispositifs d'entrée (clavier, souris, microphone, doigt, etc.). Une seconde classe de relations concerne l'enchaînement des appels de service entre le NF et la Présentation. L'enchaînement des appels doit être, en principe, conforme aux relations temporelles et structurelles du modèle de tâche produit très en amont dans le processus de conception du système interactif. Dans cette conformité on peut signaler la capacité du système à permettre la réalisation « simultanée » de plusieurs tâches. Le NF et la Présentation définissent deux perspectives d'un même domaine, mais chaque perspective a un rôle précis à remplir. Dans le NF, la modélisation du domaine est guidée par des considérations de calcul, alors que dans la Présentation, le modèle est conçu pour manipuler et rendre perceptible l'état du NF. Seeheim considère uniquement une architecture modulaire, sans prêter attention à la relation entre le modèle conceptuel et le modèle mental de l'utilisateur, c'est par cela que l'interface de l'utilisateur devrait être capable de représenter les objets de l'application de manière que l'utilisateur comprenne la relation entre la représentation et le monde réel.
Document (s)	{ }

Cas d'application	
Description cas	Un exemple simple pour montrer l'usage du modèle Seeheim est de considérer une application qui permet de visualiser et de configurer une pièce géométrique de trois dimensions : La présentation (appelée La Figure) correspond à l'image perçue par l'utilisateur, dans cette abstraction l'utilisateur peut éditer les visages et les sommets de la figure. Le contrôleur de dialogue permet de manipuler les évènements associés aux éléments dans la cape de présentation, au moyen de la communication directe entre les objets et le domaine fonctionnel de l'application. Le moyau fonctionnel (appelée Application) contient les procédures et les routines qui permettent de représenter la vue de figure géométrique par rapport aux paramètres reçus au moyen du contrôleur de dialogue. Dans cette cape nous pouvons trouver les fonctions telles que le calcul de l'aire, de périmètre, une définition de la figure, etc. La figure ci-après illustre les composants associés à cet exemple :
Représentation de cas	

2.1.2. Patrons d'architecture Arch

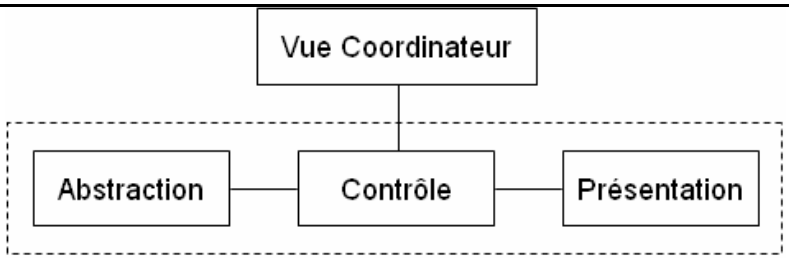
Patron : ARCH	
Auteur (s)	J.Pérez
Domaine	Domaine de développement
Technologie	Technologie de développement
P-System	Système de Patrons pour les systèmes de Réalité Augmentée
Formalisme	P-SIGMA + étendu pour les systèmes de Réalité Augmentée
Identifiant	
ARCH	
Participant (s)	
• « Architecte applicative »	
Classification	
Terme (s) de domaine	{Activité ^ Produit ^ ARCH ^ Architecture logicielle ^ Conception ^ Réutilisation}
Contexte	
Contexte formel	Phase Conception Architecturel
Contexte textuel	Application Interactive a développer avec une flexible IHM.
Problème	
ARCH résout la construction d'applications interactives au moyen de la séparation rigoureuse en niveaux d'abstraction, elle a été conçue dans le but de minimiser les effets des évolutions des outils d'interaction.	
Force (s)	
Force (s)	• Cette modèle permet varier l'importance relative à ses cinq

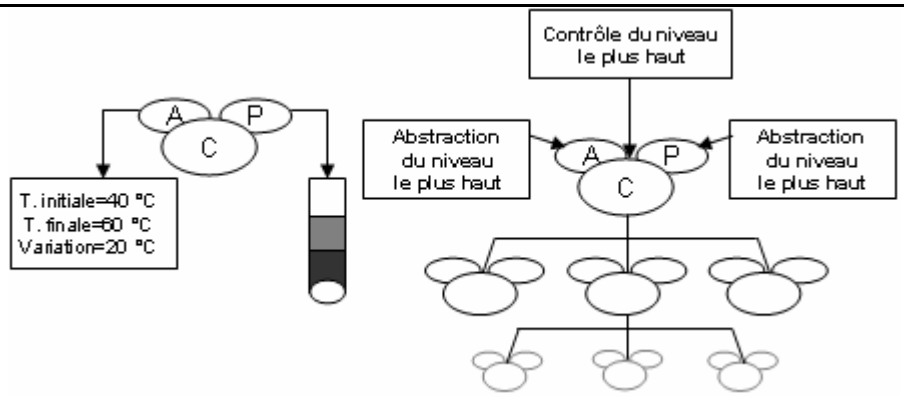
	composants et peut facilement être adapté aux contraintes d'un environnement particulier. • L'IHM d'un système regroupe tous les composants d'une instance d'ARCH à l'exception du noyau fonctionnel, le noyau fonctionnel ne doit avoir aucune connaissance des fonctionnalités relevant de l'interface utilisateur pour faciliter une conception itérative de l'interface et pour favoriser la réutilisation et la portabilité du logiciel.
Technologie (s)	{ Réutilisation ^ Composants ^ Orienté objet }
Solution produit	
Diagramme de classes	
Solution textuelle	ARCH offre une structure qui s'appuie sur les composants conceptuels du modèle Seeheim, elle classifie un système interactif de manière suivant : • **Le noyau fonctionnel (NF)**, implémente les fonctionnalités et les concepts du domaine indépendamment de leur présentation. Les structures de données manipulées par ces composants sont les objets du domaine. • **L'adaptateur de domaine (AD)**, joue un rôle de médiateur entre le NF et le CD. Les données échangées avec le NF sont les objets du domaine que le NF exporte ver l'utilisateur. Similairement, les données échangées avec le CD sont des objets conceptuels correspondant à une représentation mentale de l'utilisateur des objets du domaine. • **Le contrôleur de dialogue (CD)** est responsable de l'enchaînement des tâches. Il manipule à la fois les objets conceptuels et les objets de présentation nécessaires à l'interaction. • **Le composant de présentation (CP)** permet au CD de s'affranchir du fonctionnement de la boîte à outils du niveau interaction. Généralement assimilé à une boîte à outils graphiques abstraits, permet l'indépendance vis-à-vis des boîtes à outils graphiques du niveau du composant physique. • **Le composant d'interaction (CI)** représente les interacteurs logiciels (widget) et matériels. Il s'agit en générale d'une boîte à outils graphique (User Interface Toolkit) et des périphériques d'interaction.
Document (s)	{ }
Cas d'application	
Description cas	
Représentation	

de cas	

2.1.3. Patrons d'architecture Présentation Abstraction Contrôleur (PAC)

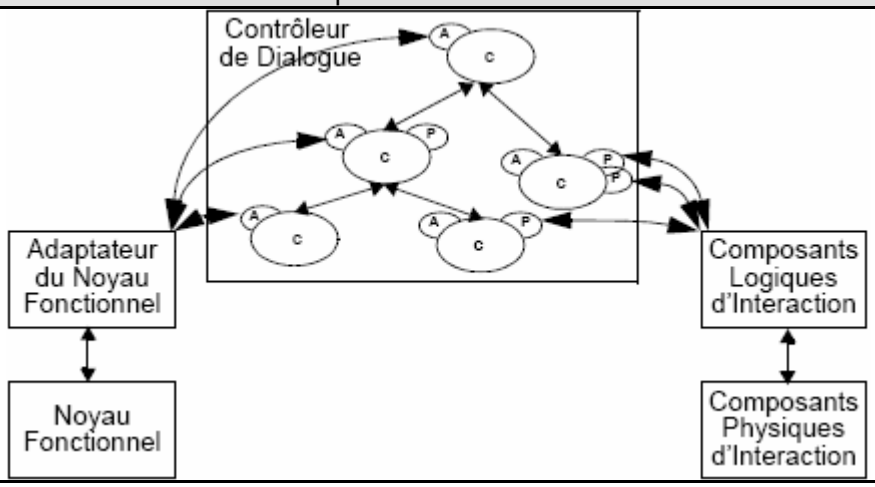
Patron : Présentation Abstraction Contrôleur (PAC)	
Auteur (s)	J.Pérez
Domaine	Domaine de développement
Technologie	Technologie de développement
P-System	Système de Patrons pour les systèmes de Réalité Augmentée
Formalisme	P-SIGMA + étendu pour les systèmes de Réalité Augmentée
Identifiant	
PAC	
Participant (s)	
• « Architecte applicative »	
Classification	
Terme (s) de domaine	{Activité ^ Produit ^ PAC ^ Architecture logicielle ^ Conception ^ Composants ^ Réutilisation}
Contexte	
Contexte formel	Phase Conception Architecturel
Contexte textuel	Application Interactive a développer avec l'aide d'agents
Problème	
Les systèmes interactifs peuvent souvent être vus comme un jeu d'agents que coopèrent entre ceux-ci. Agents spécialisés dans l'interaction homme - machine acceptent l'entrée de l'utilisateur et montrent des données. D'autres agents fournissent le modèle de données du système et offrent la fonctionnalité qui fonctionne sur ces données. Additionnel, les agents sont responsables de diverses tâches telles que le maniement des erreurs ou la communication avec d'autres systèmes. Dans une architecture d'agents qui coopèrent, chaque agent est spécialisé pour faire une tâche spécifique, et tous les agents joints proportionnent la fonctionnalité de système. Cette architecture capture aussi tant une décomposition horizontale comme verticale.	
Force (s)	
Force (s)	• Les agents sont responsables de son propre état et données. Cependant, ils doivent coopérer effectivement pour proportionner les fonctionnalités du système. Ils ont besoin d'un mécanisme pour l'échange des données, de messages et d'évènements. • Les agents interactifs proportionnent ses propres interfaces d'utilisateur, puisque ses interactions un homme - machine diffèrent souvent longuement. • Les systèmes se développent avec le temps. Son aspect de présentation est enclin aux échanges requis en réponse à nouveaux dispositifs d'interaction.
Technologie (s)	{ Réutilisation ^ Composants ^ Orienté objet }
Solution produit	
Diagramme de classes	

	
Solution textuelle	Le modèle de Présentation Abstraction Contrôleur (PAC) [Coutaz 87] est un modèle multi-agent qui repose sur deux principes directeurs : le concept d'agents réactifs a facettes et l'organisation hiérarchique de ces agents. *(Un agent* [Feber 89] est une entité informatique, réelle ou abstraite, indépendante à durée de vie limitée. Un *agent réactif* est alors défini comme un agent dont le comportement est la conséquence de ses observations et de ses interactions avec l'utilisateur). Structurer une application interactive comme une hiérarchie pareille à un arbre d'agents PAC. Devrait y avoir un agent de haut niveau, quelques agents de niveau intermédiaire, et encore plus les agents de niveau inférieur. Chaque agent est responsable d'un aspect spécifique de la fonctionnalité de l'application, et consiste en trois composants : une présentation, une abstraction, et un contrôle. L'arbre hiérarchique réfléchit la dépendance transitive entre les agents. Chaque agent est dépendant de tous les agents de niveau plus haut qui se trouvent dans son arbre hiérarchique. Le composant de présentation de l'agent proportionne la conduite visible de l'agent PAC. Son composant d'abstraction, il fournit, le modèle de données qui est la base de l'agent, et proportionne la fonctionnalité requises par manipuler ces données. Son composant de contrôle unit la présentation et le composant d'abstraction, et proportionne la fonctionnalité qui permet à l'agent de communiquer avec d'autres agents PAC. Le niveau plus haut proportionne le coeur fonctionnel du système. Les agents de PAC de niveau inférieur représentent des concepts sémantiques autonomes sur lesquels les utilisateurs du système peuvent mettre en action. Les agents de PAC de niveau intermédiaire représentent des combinaisons de, ou des relations entre, les agents de niveau inférieur. Par exemple, un agent de niveau intermédiaire peut manier quelque vue des mêmes donnes.
Document (s)	{ }
Cas d'application	
Description cas	L'abstraction de un thermomètre pour représenter la notation de température dans un système de contrôle industriel nous permet de illustrer l'usage de l'agent PAC. La Présentation de cet agent sait dessiner une forme de thermomètre et reçoit les événements qui traduisent les actions de l'utilisateur. Ce composant pourrait transmettre une valeur liée au formalisme graphique, et notamment une longueur de segment de l'a image visualisée qui représente le niveau de mercure sur l'écran.

	L'Abstraction, avec son vecteur d'état « température initiale, température finale et variation de température », constitue un modèle abstrait de l'état du thermomètre. Ce composant déduit une température finale, calcule la variation et transmet le ou les résultats pertinents au composant de Contrôle. Le Contrôle effectue le pont entre les deux facettes fonctionnelles en gérant la correspondance des phénomènes abstraits et concrets. Ce composant traduit la longueur du segment (en degrés Celsius ou en degrés Fahrenheit) en fonction du protocole convenue avec l'Abstraction.
Représentation de cas	

2.1.4. Patrons d'architecture PAC-Amodeus

Patron : PAC-Amodeus	
Auteur (s)	J.Pérez
Domaine	Domaine de développement
Technologie	Technologie de développement
P-System	Système de Patrons pour les systèmes de Réalité Augmentée
Formalisme	P-SIGMA + étendu pour les systèmes de Réalité Augmentée
Identifiant	
PAC	
Participant (s)	
• « Architecte applicative »	
Classification	
Terme (s) de domaine	{Activité ^ Produit ^ PAC-Amodeus ^ Architecture logicielle ^ Conception ^ Composants ^ Réutilisation}
Contexte	
Contexte formel	Phase Conception Architecturel
Contexte textuel	Application Interactive a développer avec l'aide d'agents et affinement du contrôleur de dialogue.
Problème	
Pac-Amodeus résout la construction d'applications interactives au moyen de la séparation rigoureuse en niveaux d'abstraction. Elle propose l'affinement du contrôleur de dialogue en termes d'agents PAC que coopèrent entre ceux-ci., elle a été conçue dans le but de maximiser la performance des applications.	
Force (s)	

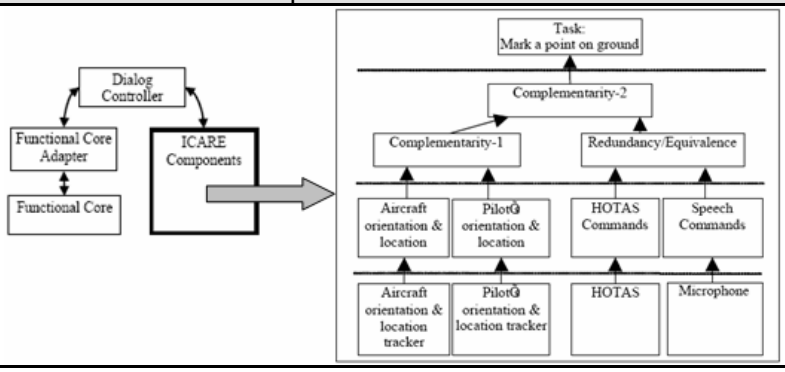
Force (s)	• Utilise la décomposition fonctionnelle du modèle de référence ARCH. • Affine dans différents niveaux d'abstraction le contrôleur de dialogue. • Conserve la modularité, encapsulation/indépendance et le parallélisme des modèles basées en agents.
Technologie (s)	{ Réutilisation ^ Composants ^ Orienté objet }
Solution produit	
Diagramme de classes	
Solution textuelle	Le modèle PAC-Amodeous [Nigay 94] est l'intégration du modèle multi-agent PAC au modèle fonctionnel Arch. Dans le modèle PAC-Amodeus, le Contrôleur de Dialogue d'Arch est organisé en une hiérarchie d'agents PAC. La facette Abstraction de chaque agent est en relation avec un objet du domaine (objet conceptuel) situé dans le Noyau Fonctionnel. De même, la facette Présentation de chaque agent communique avec le Composant Physique d'Interaction. Les messages échanges entre les facettes Contrôle permettent d'articuler l'activité de l'utilisateur et de réguler les interactions. Dans PAC-Amodeus, la structuration en trois facettes d'un agent PAC n'est pas obligatoire. Ainsi un agent peut ne pas avoir de facette Présentation. Un agent peut aussi implémenter plusieurs facettes Présentation dans de cas de vues multiples. Un agent situe a un nœud de la hiérarchie n'implémentant que la facette Contrôle et/ou la facette Abstraction est un agent ciment : cet agent résulte d'une factorisation de compétences pour offrir un niveau supplémentaire d'abstraction et accroître la modularité du code. Finalement, PAC-Amodeus constitue un cadre conceptuel adapte a l'interaction multimodale en entrée et en sortie.
Document (s)	{ }
Cas d'application	
Description cas	MATIS (Multimodal Airline Travel Information System) est un système d'information sur les transports aériens. Dans s'architecture conceptuelle le Noyau Fonctionnel est une base de données sur les transports aériens accessible par des requêtes SQL. Dans cet exemple, l'ANF joue le rôle de traducteur entre le

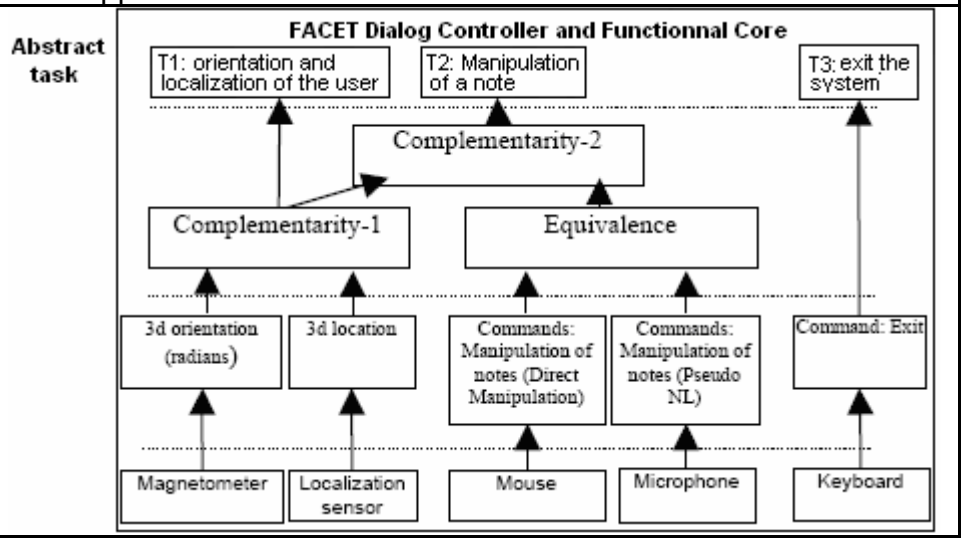
	formalisme SQL et la structure textuelle utilisée par le CD. De l'autre cote de l'arche, le CTP abstrait les informations spécifiées par l'utilisateur dans le formalisme du CD et vice versa. En l'occurrence, il est scindé en deux parties : l'une est dédiée a la présentation graphique et a l'interaction par manipulation directe sur les objets graphiques, l'autre est dédiée a l'analyse des phases en langage naturel. Le Composant Bas Niveau d'Interaction désigne la plate-forme d'accueil logicielle et matérielle. Ce niveau regroupe les services d'acquisition, d'estampille et de répartition des événements, mais aussi les objets d'interaction des boîtes à outils et les systèmes de reconnaissance spécialisés.
Représentation de cas	

2.2. Patrons de solutions technique

2.2.1. Patron ICARE

Patron : ICARE	
Auteur (s)	J.Pérez
Domaine	Domaine de développement
Technologie	Technologie de développement
P-System	Système de Patrons pour les systèmes de Réalité Augmentée
Formalisme	P-SIGMA + étendu pour les systèmes de Réalité Augmentée
Identifiant	
ICARE	
Participant (s)	
• « Architecte applicative »	
Classification	
Terme (s) de domaine	{Activité ^ Produit ^ ICARE ^ Architecture logicielle ^ Conception ^ Composants ^ Réutilisation}
Contexte	
Contexte formel	Phase Conception Architecturel
Contexte textuel	Application Interactive a développer avec une flexible dans l'usage des dispositifs d'entrée – sortie.
Problème	
ICARE (Interaction Complémentarité Assignation Redondance Equivalence) résout la construction d'applications interactives au moyen de la conception d'interfaces multimodal (un système est multimodal s'il dispose d'au moins deux modalités pour le traitement des données de entrée ou sortie).	
Force (s)	
Force (s)	• Les composants d'ICARE sont indépendants des modalités et peuvent donc être réutilisés dans différentes applications. • Les composants d'ICARE permettent de définir des modalités d'interaction et combiner les données des modalités définies.

	• Génération automatique du code par un système interactif.
Technologie (s)	{ Réutilisation ^ Composants ^ Orienté objet }
Solution produit	
Diagramme de classes	
Solution textuelle	ICARE (Interaction Complémentarité Assignation Redondance Equivalence) est une plateforme qui propose un ensemble de composants permettant de concevoir une interface multimodale. Les composants d'ICARE sont indépendants des modalités et peuvent donc être réutilise dans différentes applications. On distingue deux types de composants : les composants élémentaires et des composition. **Composant élémentaires** : Les composant élémentaires sont l'association des composants Dispositifs et Systèmes Représentationnels qui permet la multi modalité. Le composant Utilisateur représente les propriétés de l'utilisateur qui peuvent concerner l'identité, les coordonnées ainsi que les données biométriques et sociales qui le caractérisent [Bouchet 03]. Le composant Environnement permet d'enregistrer des données sur l'environnement physique du contexte d'utilisation (température, niveau sonore, etc.). **Composants de composition** : On distingue trois sortes de composants de composition : Le composant Complémentarité traduit la propriété ergonomique de Complémentarité et représente la composition de plusieurs dispositifs physiques (niveau physique) ou de plusieurs systèmes représentationnels (niveau logique). Le composant Redondance traduit la propriété ergonomique de redondance, c'est-à-dire la possibilité d'utiliser simultanément deux modalités différentes pour interagir avec le système. Le composant Equivalence/Redondance traduit les propriétés ergonomiques d'équivalence et de redondance, c'est-à-dire la possibilité d'utiliser deux modalités différentes, de manière simultanée ou non, pour interagir avec le système. Finalement, les opérations de fusion de deux modalités se font au niveau de ces trois composants.
Document (s)	{ }
Cas d'application	
Description cas	L'application Mémo développé par J. Bouche [Bouche 03] est un exemple de ce type de système. Elle permet à un utilisateur de découvrir, lire, poser ou supprimer des mémos (post-its) directement dans l'environnement physique dans lequel il évolue, en utilisant l'aide d'une souris et/ou des commandes vocales.

	L'utilisateur voit au travers d'un casque semi-transparent le monde réel et des mémos numériques déposés précédemment à des endroits du monde réel. On dit dans ce cas que la perception de l'utilisateur est augmentée. La figure suivant permet de illustrer le schéma d'interaction de cette application.
Représentation de cas	

MEMOIRE D'Master 2 Matis

UFR Informatique et Mathématiques appliquées

Spécifications techniques de la démarche Symphony dans le cadre des systèmes de réalité augmentée

Jorge Luis PEREZ-MEDINA **Grenoble, le 30 Août 2006**

Résumé

Les systèmes de Réalité Augmentée suscitent aujourd'hui l'intérêt de domaines très divers, par exemple : ergonomie, interaction Homme-Machine, génie logiciel, etc. Leur conception devient alors un processus complexe visant à coordonner des nouveaux contextes, des sources d'information, des applications préexistantes et en évolution. Cette complexité rend nécessaire la définition d'un processus adapté qui minimise la difficulté de la conception et maximise la réutilisation et la sélection des technologies existantes.

La difficulté de conception des systèmes de réalité augmentée, en particulier, la sélection des dispositifs d'interaction diverses et de styles d'architecture logiciel rend nécessaire la formalisation de méthodes, modèles et techniques, proches du modèle de raisonnement du développeur.

Dans le cadre de notre mémoire la solution que nous proposons pour développer les systèmes de réalité augmentée est de proposer une démarche de développement, en particulier pour les parties techniques. Nos propositions se basent sur une démarche de développement de système d'information basée sur la réutilisation de composants.

Symphony est une méthode pour la conception de l'architecture logicielle qui intègre des règles de conception et des patrons de conception, Symphony dont l'objectif est de pallier les manques technologiques. En conséquence, Symphony se présente sous la forme d'un guide méthodologique offrant une solution basée sur l'utilisation de composants dès les phases amont du processus. Cette démarche est centrée sur des patrons, formalisée par un ensemble de patrons orientés vers le « processus » (c'est-à-dire de savoir-faire) combinés à des patrons orientés vers les « produits » (c'est-à-dire de savoir) réutilisables dans de nombreuses applications.

Mots clés : Conception du logiciel, Réalité augmentée, Architectures logicielles, Système de patrons.